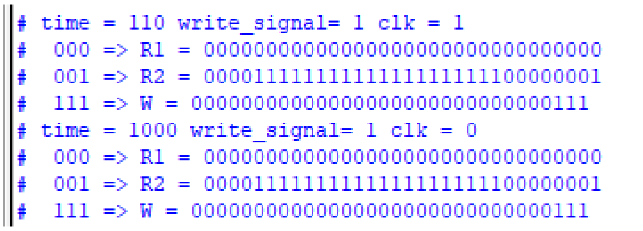
CSE 331 Final Project – MiniMIPS Design

Muhammed Bedir ULUCAY

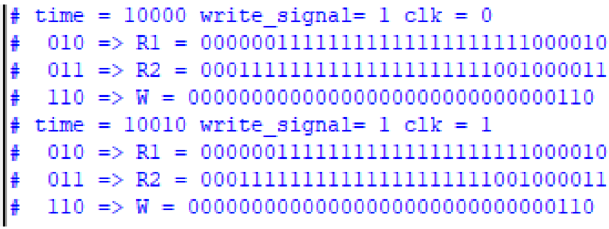
1901042697

I try to do everything in the homework, there is no missing part that I can notice.

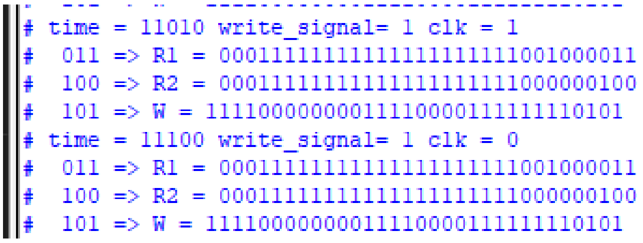
Mips Register :



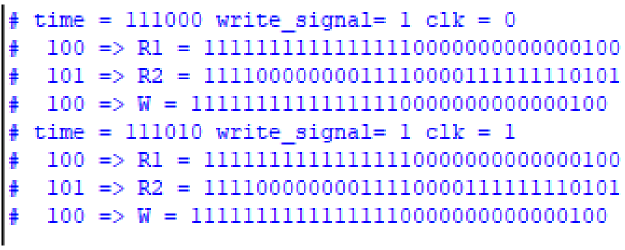
```
# time = 110 write_signal= 1 clk = 1
#  000 => R1 = 00000000000000000000000000000000
#  001 => R2 = 00001111111111111111111100000001
#  111 => W = 00000000000000000000000000000111
# time = 1000 write_signal= 1 clk = 0
#  000 => R1 = 00000000000000000000000000000000
#  001 => R2 = 00001111111111111111111100000001
#  111 => W = 00000000000000000000000000000111
```



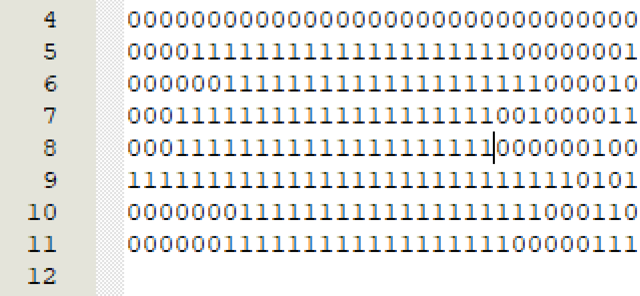
```
# time = 10000 write_signal= 1 clk = 0
#  010 => R1 = 00000011111111111111111111000010
#  011 => R2 = 00011111111111111111111001000011
#  110 => W = 00000000000000000000000000000110
# time = 10010 write_signal= 1 clk = 1
#  010 => R1 = 00000011111111111111111111000010
#  011 => R2 = 00011111111111111111111001000011
#  110 => W = 00000000000000000000000000000110
```



```
# time = 11010 write_signal= 1 clk = 1
#  011 => R1 = 00011111111111111111111001000011
#  100 => R2 = 00011111111111111111111000000100
#  101 => W = 11110000000011110000111111110101
# time = 11100 write_signal= 1 clk = 0
#  011 => R1 = 00011111111111111111111001000011
#  100 => R2 = 00011111111111111111111000000100
#  101 => W = 11110000000011110000111111110101
```

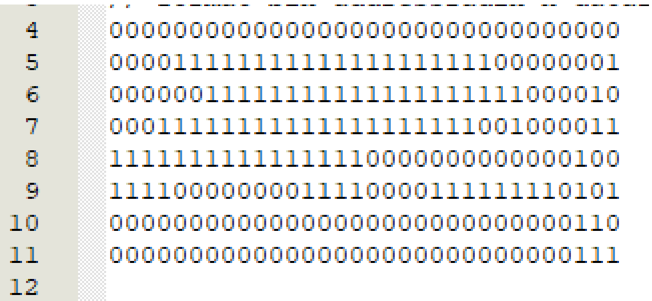


```
# time = 111000 write_signal= 1 clk = 0
#  100 => R1 = 11111111111111110000000000000100
#  101 => R2 = 11110000000011110000111111110101
#  100 => W = 11111111111111110000000000000100
# time = 111010 write_signal= 1 clk = 1
#  100 => R1 = 11111111111111110000000000000100
#  101 => R2 = 11110000000011110000111111110101
#  100 => W = 11111111111111110000000000000100
```



```
 4   00000000000000000000000000000000
 5   00001111111111111111111100000001
 6   00000011111111111111111111000010
 7   00011111111111111111111001000011
 8   00011111111111111111111000000100
 9   11111111111111111111111111110101
10   00000001111111111111111111000110
11   00000011111111111111111100000111
12
```

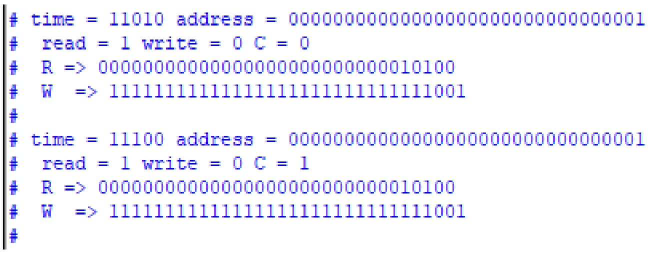
Before



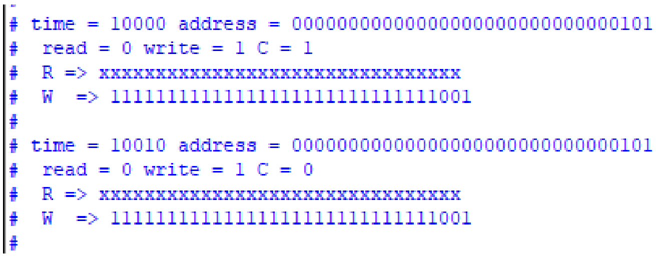
```
 4   00000000000000000000000000000000
 5   00001111111111111111111100000001
 6   00000011111111111111111111000010
 7   00011111111111111111111001000011
 8   11111111111111110000000000000100
 9   11110000000011110000111111110101
10   00000000000000000000000000000110
11   00000000000000000000000000000111
12
```

After

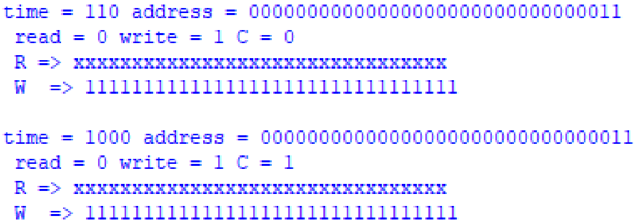
Mips Memory :



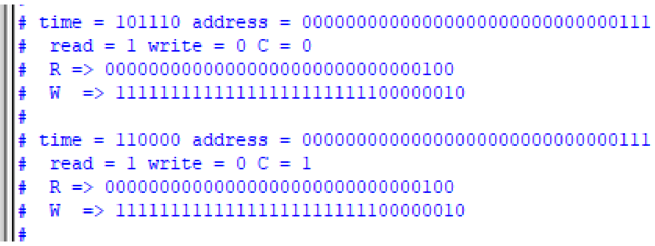
```
# time = 11010 address = 00000000000000000000000000000001
#  read = 1 write = 0 C = 0
#  R => 00000000000000000000000000010100
#  W  => 11111111111111111111111111111001
#
# time = 11100 address = 00000000000000000000000000000001
#  read = 1 write = 0 C = 1
#  R => 00000000000000000000000000010100
#  W  => 11111111111111111111111111111001
#
```



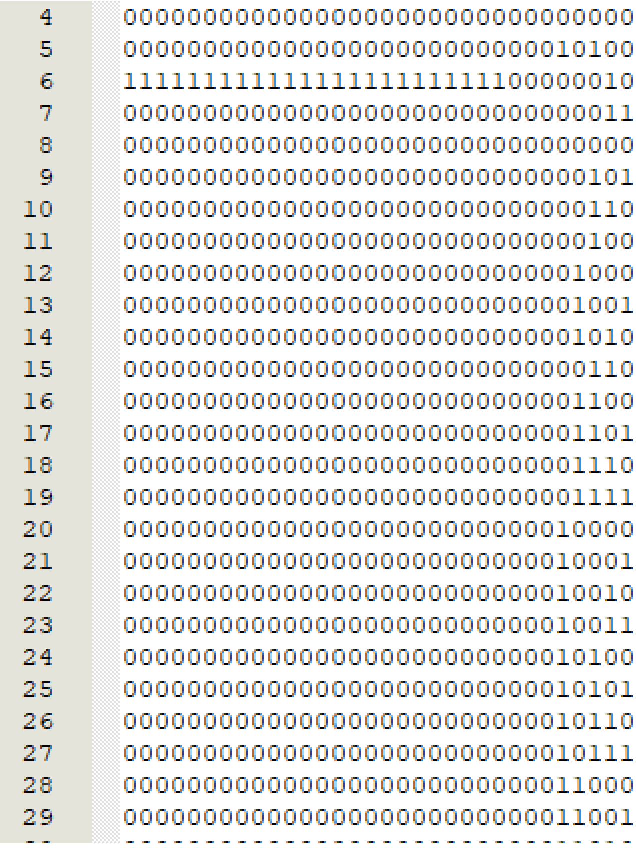
```
# time = 10000 address = 00000000000000000000000000000101
#  read = 0 write = 1 C = 1
#  R => xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#  W  => 11111111111111111111111111111001
#
# time = 10010 address = 00000000000000000000000000000101
#  read = 0 write = 1 C = 0
#  R => xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#  W  => 11111111111111111111111111111001
#
```



```
time = 110 address = 00000000000000000000000000000011
 read = 0 write = 1 C = 0
 R => xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 W  => 11111111111111111111111111111111

time = 1000 address = 00000000000000000000000000000011
 read = 0 write = 1 C = 1
 R => xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 W  => 11111111111111111111111111111111
```

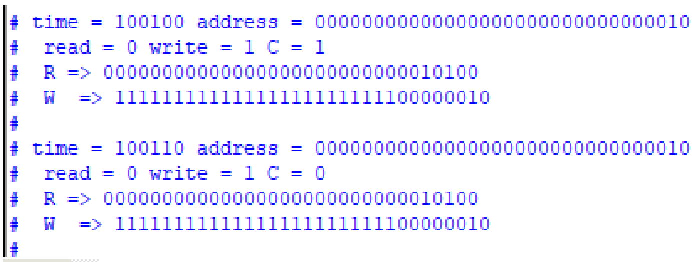


```
# time = 101110 address = 00000000000000000000000000000111
#  read = 1 write = 0 C = 0
#  R => 00000000000000000000000000000100
#  W  => 11111111111111111111111110000010
#
# time = 110000 address = 00000000000000000000000000000111
#  read = 1 write = 0 C = 1
#  R => 00000000000000000000000000000100
#  W  => 11111111111111111111111110000010
#
```

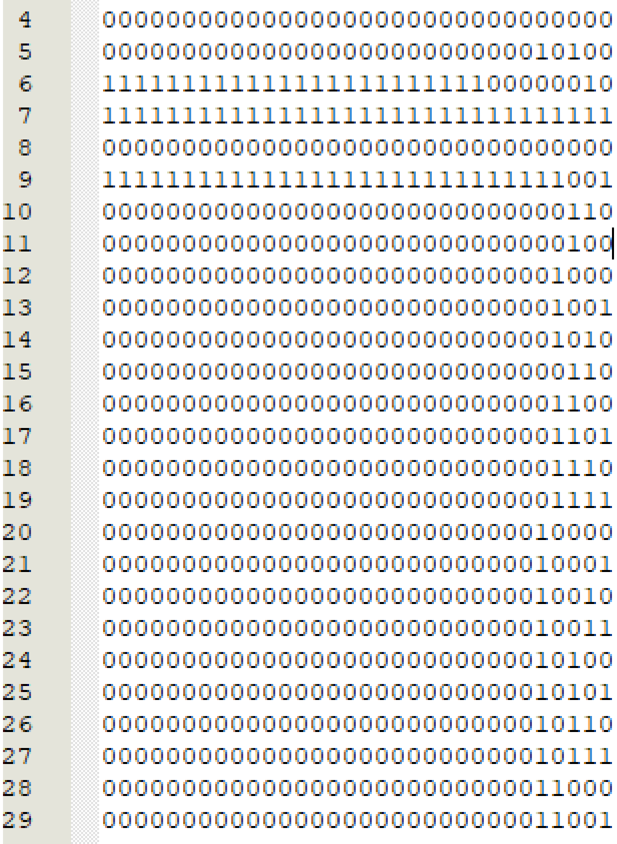


```
 4   00000000000000000000000000000000
 5   00000000000000000000000000010100
 6   11111111111111111111111100000010
 7   00000000000000000000000000000011
 8   00000000000000000000000000000000
 9   00000000000000000000000000000101
10   00000000000000000000000000000110
11   00000000000000000000000000000100
12   00000000000000000000000000001000
13   00000000000000000000000000001001
14   00000000000000000000000000001010
15   00000000000000000000000000000110
16   00000000000000000000000000001100
17   00000000000000000000000000001101
18   00000000000000000000000000001110
19   00000000000000000000000000001111
20   00000000000000000000000000010000
21   00000000000000000000000000010001
22   00000000000000000000000000010010
23   00000000000000000000000000010011
24   00000000000000000000000000010100
25   00000000000000000000000000010101
26   00000000000000000000000000010110
27   00000000000000000000000000010111
28   00000000000000000000000000011000
29   00000000000000000000000000011001
```

Before



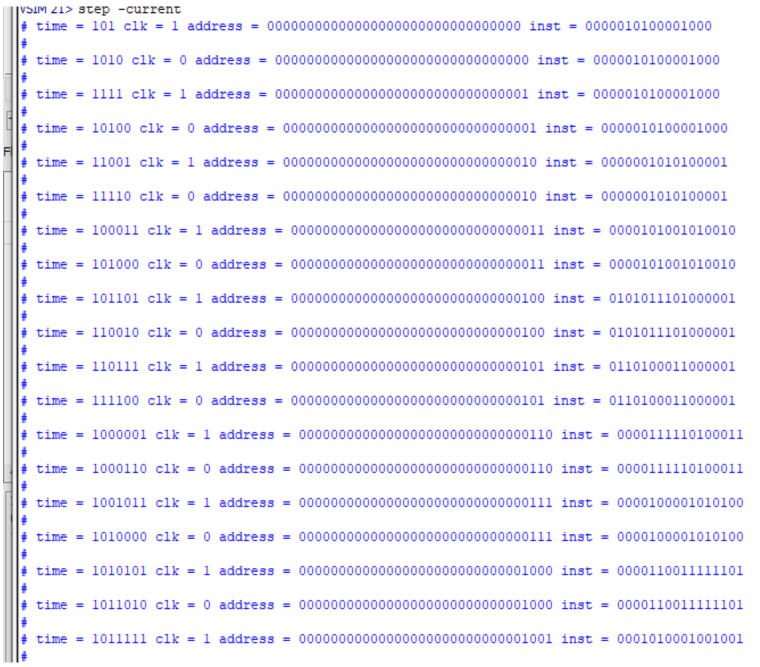
```
# time = 100100 address = 00000000000000000000000000000010
#  read = 0 write = 1 C = 1
#  R => 00000000000000000000000000010100
#  W  => 11111111111111111111111100000010
#
# time = 100110 address = 00000000000000000000000000000010
#  read = 0 write = 1 C = 0
#  R => 00000000000000000000000000010100
#  W  => 11111111111111111111111100000010
#
```



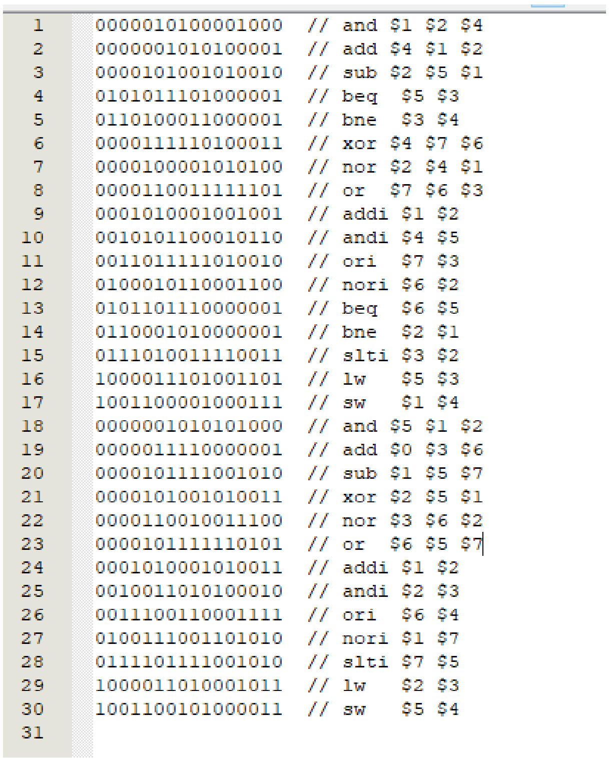
```
 4   00000000000000000000000000000000
 5   00000000000000000000000000010100
 6   11111111111111111111111100000010
 7   11111111111111111111111111111111
 8   00000000000000000000000000000000
 9   11111111111111111111111111111001
10   00000000000000000000000000000110
11   00000000000000000000000000000100
12   00000000000000000000000000001000
13   00000000000000000000000000001001
14   00000000000000000000000000001010
15   00000000000000000000000000000110
16   00000000000000000000000000001100
17   00000000000000000000000000001101
18   00000000000000000000000000001110
19   00000000000000000000000000001111
20   00000000000000000000000000010000
21   00000000000000000000000000010001
22   00000000000000000000000000010010
23   00000000000000000000000000010011
24   00000000000000000000000000010100
25   00000000000000000000000000010101
26   00000000000000000000000000010110
27   00000000000000000000000000010111
28   00000000000000000000000000011000
29   00000000000000000000000000011001
```

After

Instruction Memory:

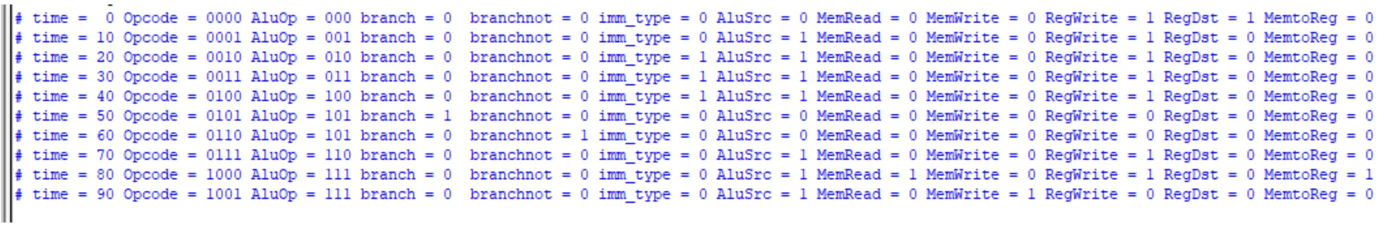


```
VSIM 21> step -current
# time = 101 clk = 1 address = 00000000000000000000000000000000 inst = 0000010100001000
#
# time = 1010 clk = 0 address = 00000000000000000000000000000000 inst = 0000010100001000
#
# time = 1111 clk = 1 address = 00000000000000000000000000000001 inst = 0000010100001000
#
# time = 10100 clk = 0 address = 00000000000000000000000000000001 inst = 0000010100001000
#
# time = 11001 clk = 1 address = 00000000000000000000000000000010 inst = 0000001010100001
#
# time = 11110 clk = 0 address = 00000000000000000000000000000010 inst = 0000001010100001
#
# time = 100011 clk = 1 address = 00000000000000000000000000000011 inst = 0001010010100010
#
# time = 101000 clk = 0 address = 00000000000000000000000000000011 inst = 0001010010100010
#
# time = 101101 clk = 1 address = 00000000000000000000000000000100 inst = 0101011101000001
#
# time = 110010 clk = 0 address = 00000000000000000000000000000100 inst = 0101011101000001
#
# time = 110111 clk = 1 address = 00000000000000000000000000000101 inst = 0110100011000001
#
# time = 111100 clk = 0 address = 00000000000000000000000000000101 inst = 0110100011000001
#
# time = 1000001 clk = 1 address = 00000000000000000000000000000110 inst = 0000111110100011
#
# time = 1000110 clk = 0 address = 00000000000000000000000000000110 inst = 0000111110100011
#
# time = 1001011 clk = 1 address = 00000000000000000000000000000111 inst = 0001000001010100
#
# time = 1010000 clk = 0 address = 00000000000000000000000000000111 inst = 0001000001010100
#
# time = 1010101 clk = 1 address = 00000000000000000000000000001000 inst = 0000110011111101
#
# time = 1011010 clk = 0 address = 00000000000000000000000000001000 inst = 0000110011111101
#
# time = 1011111 clk = 1 address = 00000000000000000000000000001001 inst = 0001010001001001
#
```



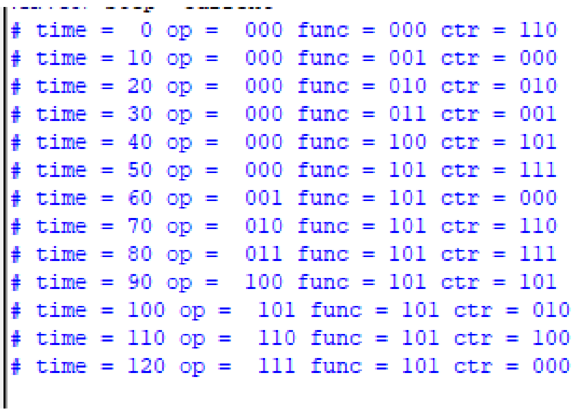
```
0000010100001000   // and $1 $2 $4
0000001010100001   // add $4 $1 $2
0001010010100010   // sub $2 $5 $1
0101011101000001   // beq  $5 $3
0110100011000001   // bne  $3 $4
0000111110100011   // xor $4 $7 $6
0001000001010100   // nor $2 $4 $1
0000110011111101   // or  $7 $6 $3
0001010001001001   // addi $1 $2
0010101100010110   // andi $4 $5
0011011111010010   // ori  $7 $3
0100010110001100   // nori $6 $2
0101101110000001   // beq  $6 $5
0110001010000001   // bne  $2 $1
0111010011110011   // slti $3 $2
1000011101001101   // lw   $5 $3
1001100001000111   // sw   $1 $4
0000001010101000   // and $5 $1 $2
0000011110000001   // add $0 $3 $6
0000101111001010   // sub $1 $5 $7
0000101001010011   // xor $2 $5 $1
0000110010011100   // nor $3 $6 $2
0000101111110101   // or  $6 $5 $7
0001010001010011   // addi $1 $2
0010011010100010   // andi $2 $3
0011100110001111   // ori  $6 $4
0100111001101010   // nori $1 $7
0111101111001010   // slti $7 $5
1000011010001011   // lw   $2 $3
1001100101000011   // sw   $5 $4
```

Main Control:



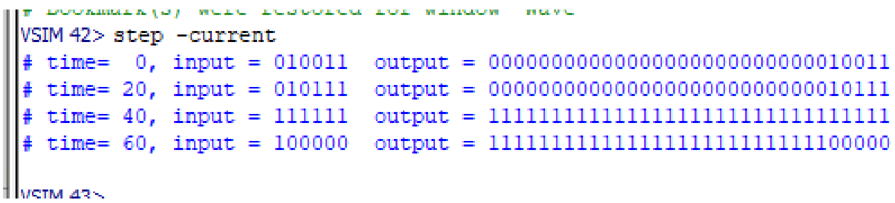
```
# time =  0 Opcode = 0000 AluOp = 000 branch = 0  branchnot = 0 imm_type = 0 AluSrc = 0 MemRead = 0 MemWrite = 0 RegWrite = 1 RegDst = 1 MemtoReg = 0
# time = 10 Opcode = 0001 AluOp = 001 branch = 0  branchnot = 0 imm_type = 0 AluSrc = 1 MemRead = 0 MemWrite = 0 RegWrite = 1 RegDst = 0 MemtoReg = 0
# time = 20 Opcode = 0010 AluOp = 010 branch = 0  branchnot = 0 imm_type = 1 AluSrc = 1 MemRead = 0 MemWrite = 0 RegWrite = 1 RegDst = 0 MemtoReg = 0
# time = 30 Opcode = 0011 AluOp = 011 branch = 0  branchnot = 0 imm_type = 1 AluSrc = 1 MemRead = 0 MemWrite = 0 RegWrite = 1 RegDst = 0 MemtoReg = 0
# time = 40 Opcode = 0100 AluOp = 100 branch = 0  branchnot = 0 imm_type = 1 AluSrc = 1 MemRead = 0 MemWrite = 0 RegWrite = 1 RegDst = 0 MemtoReg = 0
# time = 50 Opcode = 0101 AluOp = 101 branch = 1  branchnot = 0 imm_type = 0 AluSrc = 0 MemRead = 0 MemWrite = 0 RegWrite = 0 RegDst = 0 MemtoReg = 0
# time = 60 Opcode = 0110 AluOp = 101 branch = 0  branchnot = 1 imm_type = 0 AluSrc = 0 MemRead = 0 MemWrite = 0 RegWrite = 0 RegDst = 0 MemtoReg = 0
# time = 70 Opcode = 0111 AluOp = 110 branch = 0  branchnot = 0 imm_type = 0 AluSrc = 1 MemRead = 0 MemWrite = 0 RegWrite = 1 RegDst = 0 MemtoReg = 0
# time = 80 Opcode = 1000 AluOp = 111 branch = 0  branchnot = 0 imm_type = 0 AluSrc = 1 MemRead = 1 MemWrite = 0 RegWrite = 1 RegDst = 0 MemtoReg = 1
# time = 90 Opcode = 1001 AluOp = 111 branch = 0  branchnot = 0 imm_type = 0 AluSrc = 1 MemRead = 0 MemWrite = 1 RegWrite = 0 RegDst = 0 MemtoReg = 0
```

Alu Control:



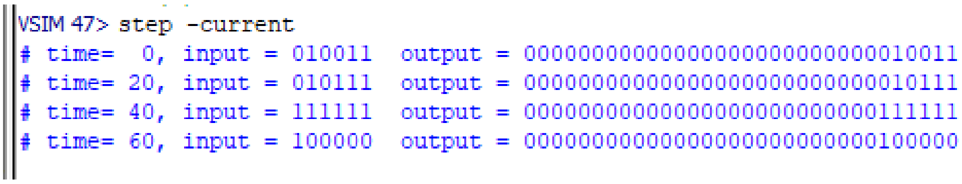
```
# time =   0 op =  000 func = 000 ctr = 110
# time =  10 op =  000 func = 001 ctr = 000
# time =  20 op =  000 func = 010 ctr = 010
# time =  30 op =  000 func = 011 ctr = 001
# time =  40 op =  000 func = 100 ctr = 101
# time =  50 op =  000 func = 101 ctr = 111
# time =  60 op =  001 func = 101 ctr = 000
# time =  70 op =  010 func = 101 ctr = 110
# time =  80 op =  011 func = 101 ctr = 111
# time =  90 op =  100 func = 101 ctr = 101
# time = 100 op =  101 func = 101 ctr = 010
# time = 110 op =  110 func = 101 ctr = 100
# time = 120 op =  111 func = 101 ctr = 000
```

Sign Extend :



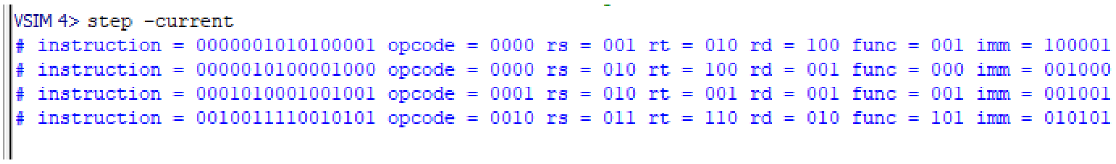
```
VSIM 42> step -current
# time=  0, input = 010011  output = 00000000000000000000000000010011
# time= 20, input = 010111  output = 00000000000000000000000000010111
# time= 40, input = 111111  output = 11111111111111111111111111111111
# time= 60, input = 100000  output = 11111111111111111111111111100000

VSIM 43>
```

Zero Extend :



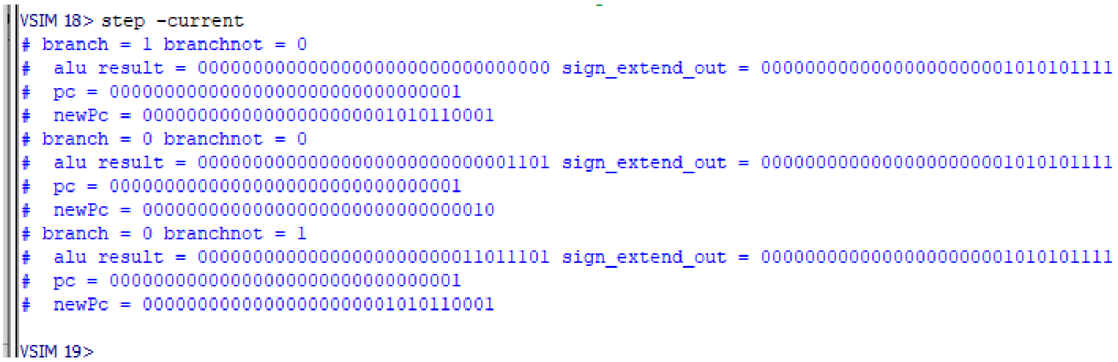
```
VSIM 47> step -current
# time=  0, input = 010011  output = 00000000000000000000000000010011
# time= 20, input = 010111  output = 00000000000000000000000000010111
# time= 40, input = 111111  output = 00000000000000000000000000111111
# time= 60, input = 100000  output = 00000000000000000000000000100000
```

Instruction Decoder:



```
VSIM 4> step -current
# instruction = 0000001010100001 opcode = 0000 rs = 001 rt = 010 rd = 100 func = 001 imm = 100001
# instruction = 0000010100001000 opcode = 0000 rs = 010 rt = 100 rd = 001 func = 000 imm = 001000
# instruction = 0001010001001001 opcode = 0001 rs = 010 rt = 001 rd = 001 func = 001 imm = 001001
# instruction = 0010011110010101 opcode = 0010 rs = 011 rt = 110 rd = 010 func = 101 imm = 010101
```

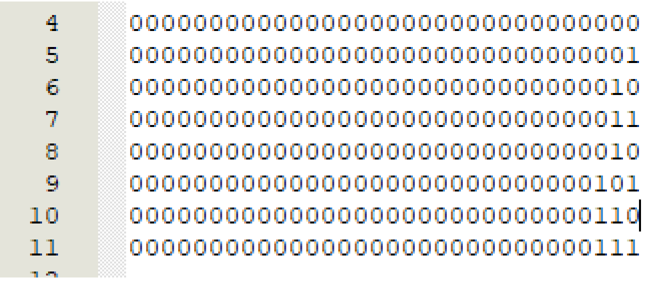
Branch :



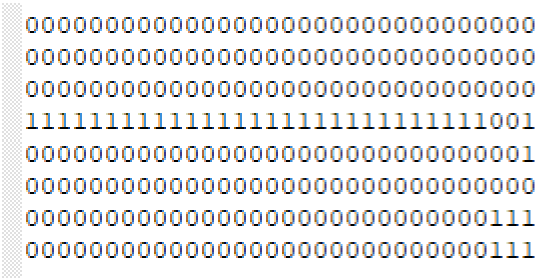
```
VSIM 18> step -current
# branch = 1 branchnot = 0
#  alu result = 00000000000000000000000000000000 sign_extend_out = 00000000000000000000001010101111
#  pc = 00000000000000000000000000000001
#  newPc = 00000000000000000000001010110001
# branch = 0 branchnot = 0
#  alu result = 00000000000000000000000000001101 sign_extend_out = 00000000000000000000001010101111
#  pc = 00000000000000000000000000000001
#  newPc = 00000000000000000000000000000010
# branch = 0 branchnot = 1
#  alu result = 00000000000000000000000011011101 sign_extend_out = 00000000000000000000001010101111
#  pc = 00000000000000000000000000000001
#  newPc = 00000000000000000000001010110001

VSIM 19>
```

Results:

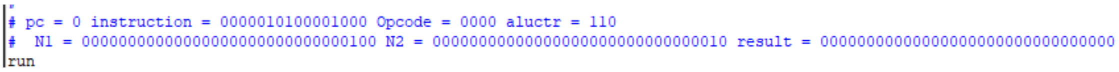
And - Xor – Nor - Or :



```
 4   00000000000000000000000000000000
 5   00000000000000000000000000000001
 6   00000000000000000000000000000010
 7   00000000000000000000000000000011
 8   00000000000000000000000000000010
 9   00000000000000000000000000000101
10   00000000000000000000000000000110
11   00000000000000000000000000000111
```

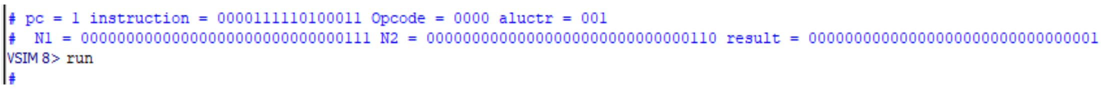


```
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
11111111111111111111111111111001
00000000000000000000000000000001
00000000000000000000000000000000
00000000000000000000000000000111
00000000000000000000000000000111
```



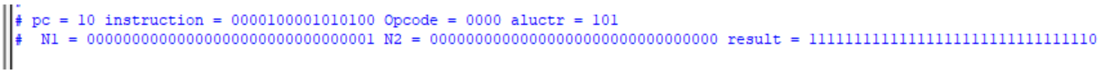
```
# pc = 0 instruction = 0000010100001000 Opcode = 0000 aluctr = 110
#  N1 = 00000000000000000000000000000100 N2 = 00000000000000000000000000000010 result = 00000000000000000000000000000000
run
```

// and $1 $2 $4



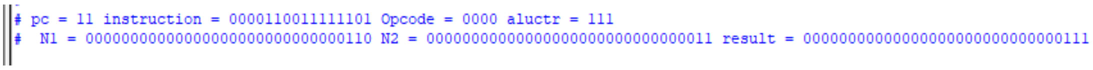
```
# pc = 1 instruction = 0000111110100011 Opcode = 0000 aluctr = 001
#  N1 = 00000000000000000000000000000111 N2 = 00000000000000000000000000000110 result = 00000000000000000000000000000001
VSIM 8> run
#
```

// xor $4 $7 $6



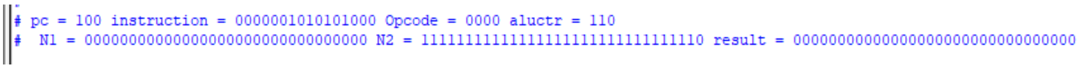
```
# pc = 10 instruction = 0000100001010100 Opcode = 0000 aluctr = 101
#  N1 = 00000000000000000000000000000001 N2 = 00000000000000000000000000000000 result = 11111111111111111111111111111110
```

// or $7 $6 $3



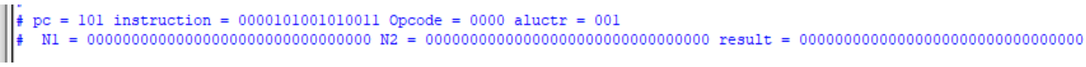
```
# pc = 11 instruction = 0000110011111101 Opcode = 0000 aluctr = 111
#  N1 = 00000000000000000000000000000110 N2 = 00000000000000000000000000000011 result = 00000000000000000000000000000111
```

// nor $2 $4 $1



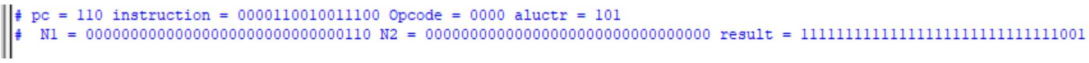
```
# pc = 100 instruction = 0000001010101000 Opcode = 0000 aluctr = 110
#  N1 = 00000000000000000000000000000000 N2 = 11111111111111111111111111111110 result = 00000000000000000000000000000000
```

// and $5 $1 $2



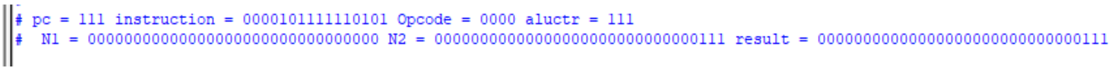
```
# pc = 101 instruction = 0000101001010011 Opcode = 0000 aluctr = 001
#  N1 = 00000000000000000000000000000000 N2 = 00000000000000000000000000000000 result = 00000000000000000000000000000000
```

// xor $2 $5 $1



```
# pc = 110 instruction = 0000110010011100 Opcode = 0000 aluctr = 101
#  N1 = 00000000000000000000000000000110 N2 = 00000000000000000000000000000000 result = 11111111111111111111111111111001
```

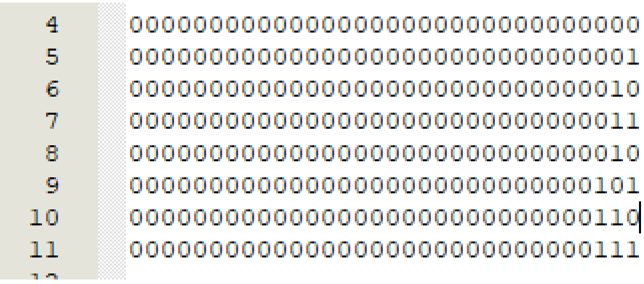
// nor $3 $6 $2



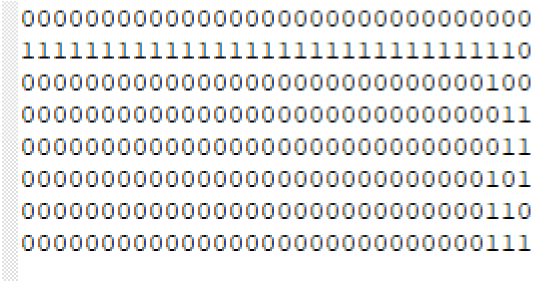
```
# pc = 111 instruction = 0000101111110101 Opcode = 0000 aluctr = 111
#  N1 = 00000000000000000000000000000000 N2 = 00000000000000000000000000000111 result = 00000000000000000000000000000111
```

// or $6 $5 $7

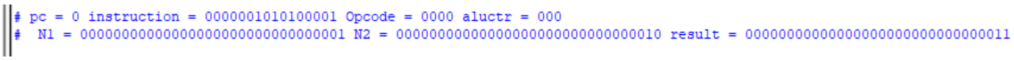
Add – Sub :



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

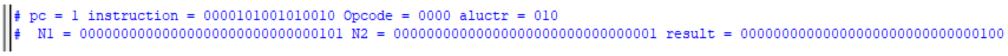


```
00000000000000000000000000000000
11111111111111111111111111111110
00000000000000000000000000000100
00000000000000000000000000000011
00000000000000000000000000000011
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```



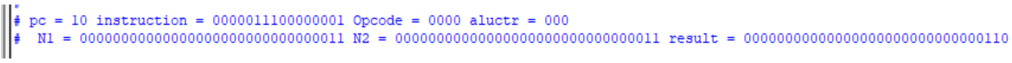
```
# pc = 0 instruction = 0000001010100001 Opcode = 0000 aluctr = 000
#  N1 = 00000000000000000000000000000001 N2 = 00000000000000000000000000000010 result = 00000000000000000000000000000011
```

// add $4 $1 $2



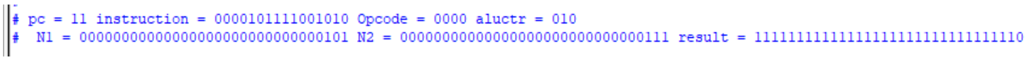
```
# pc = 1 instruction = 0000101001010010 Opcode = 0000 aluctr = 010
#  N1 = 00000000000000000000000000000101 N2 = 00000000000000000000000000000001 result = 00000000000000000000000000000100
```

// sub $2 $5 $1



```
# pc = 10 instruction = 0000011100000001 Opcode = 0000 aluctr = 000
#  N1 = 00000000000000000000000000000011 N2 = 00000000000000000000000000000011 result = 00000000000000000000000000000110
```

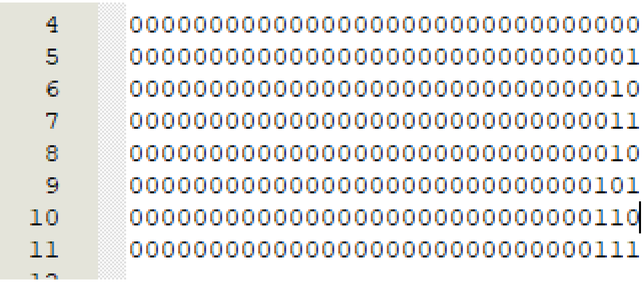
// add $0 $3 $4



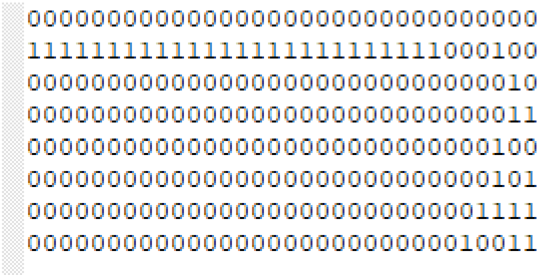
```
# pc = 11 instruction = 0000101111001010 Opcode = 0000 aluctr = 010
#  N1 = 00000000000000000000000000000101 N2 = 00000000000000000000000000000111 result = 11111111111111111111111111111110
```

// sub $1 $5 $7

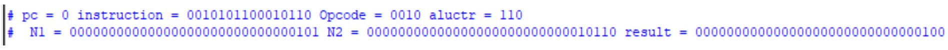
Andi – Nori - Ori :



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

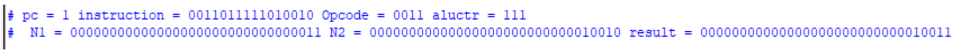


```
00000000000000000000000000000000
11111111111111111111111111000100
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000001111
00000000000000000000000000010011
```



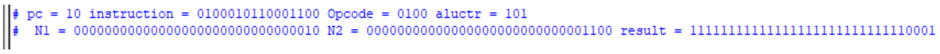
```
# pc = 0 instruction = 0010101100010110 Opcode = 0010 aluctr = 110
#  N1 = 00000000000000000000000000000101 N2 = 00000000000000000000000000010110 result = 00000000000000000000000000000100
```

// andi $4 $5 imm



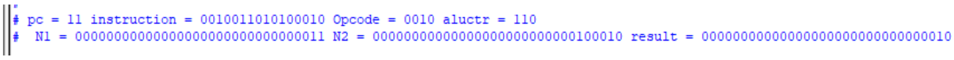
```
# pc = 1 instruction = 0011011111010010 Opcode = 0011 aluctr = 111
#  N1 = 00000000000000000000000000000011 N2 = 00000000000000000000000000010010 result = 00000000000000000000000000010011
```

// ori $7 $3 imm



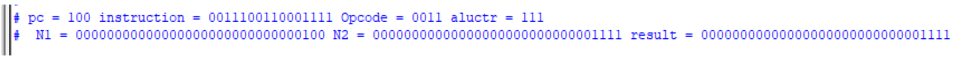
```
# pc = 10 instruction = 0100010110001100 Opcode = 0100 aluctr = 101
#  N1 = 00000000000000000000000000000010 N2 = 00000000000000000000000000001100 result = 11111111111111111111111111110001
```

// nori $6 $2 imm



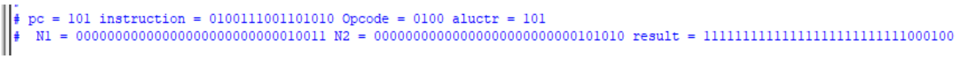
```
# pc = 11 instruction = 0010011010100010 Opcode = 0010 aluctr = 110
#  N1 = 00000000000000000000000000000011 N2 = 00000000000000000000000000100010 result = 00000000000000000000000000000010
```

// andi $2 $3 imm



```
# pc = 100 instruction = 0011100110001111 Opcode = 0011 aluctr = 111
#  N1 = 00000000000000000000000000000100 N2 = 00000000000000000000000000001111 result = 00000000000000000000000000001111
```

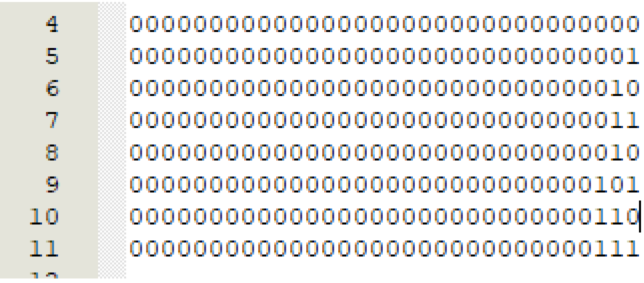
// ori $6 $4 imm



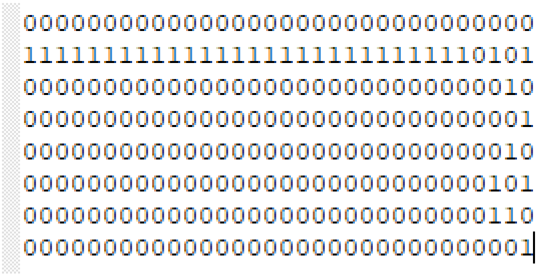
```
# pc = 101 instruction = 0100111001101010 Opcode = 0100 aluctr = 101
#  N1 = 00000000000000000000000000010011 N2 = 00000000000000000000000000101010 result = 11111111111111111111111111000100
```

// nori $1 $7 imm

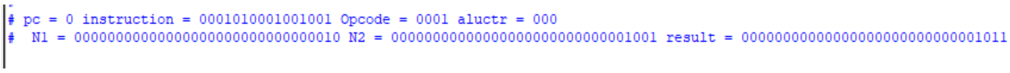
Addi - Slti :



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

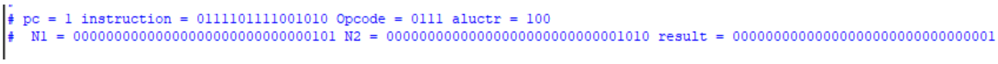


```
00000000000000000000000000000000
11111111111111111111111111110101
00000000000000000000000000000010
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000001
```



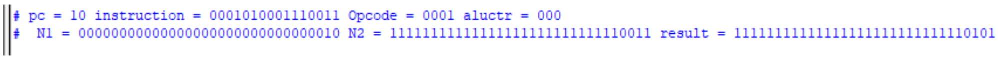
```
# pc = 0 instruction = 0001010001001001 Opcode = 0001 aluctr = 000
#  N1 = 00000000000000000000000000000010 N2 = 00000000000000000000000000001001 result = 00000000000000000000000000001011
```

// addi $1 $2



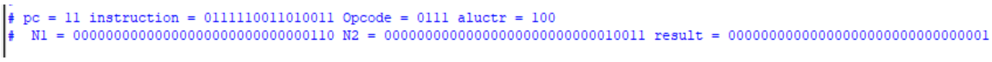
```
# pc = 1 instruction = 0111101111001010 Opcode = 0111 aluctr = 100
#  N1 = 00000000000000000000000000000101 N2 = 00000000000000000000000000001010 result = 00000000000000000000000000000001
```

// slti $7 $5



```
# pc = 10 instruction = 0001010001110011 Opcode = 0001 aluctr = 000
#  N1 = 00000000000000000000000000000010 N2 = 11111111111111111111111111110011 result = 11111111111111111111111111110101
```

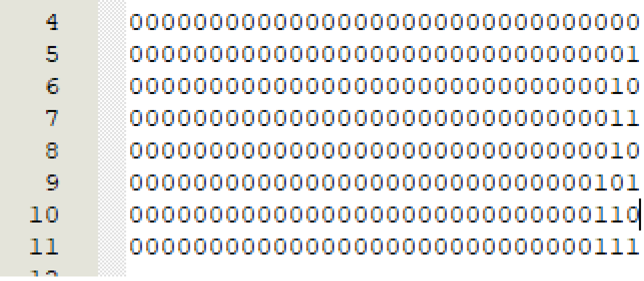
// addi $1 $2



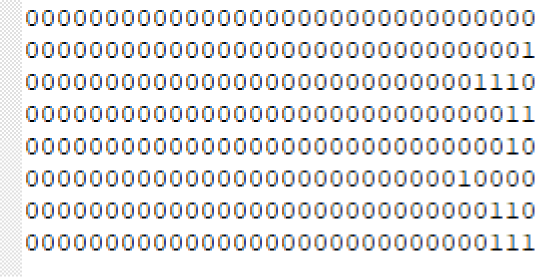
```
# pc = 11 instruction = 0111110011010011 Opcode = 0111 aluctr = 100
#  N1 = 00000000000000000000000000000110 N2 = 00000000000000000000000000010011 result = 00000000000000000000000000000001
```

// slti $3 $4

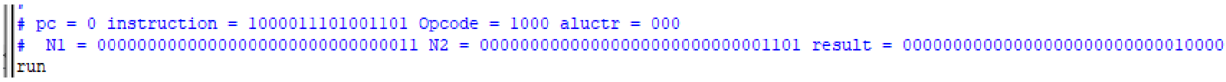
Lw – Sw :



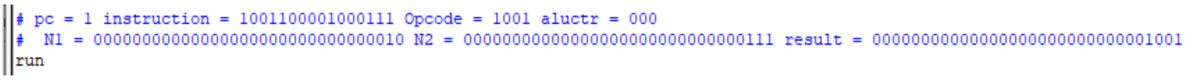
```
 4   00000000000000000000000000000000
 5   00000000000000000000000000000001
 6   00000000000000000000000000000010
 7   00000000000000000000000000000011
 8   00000000000000000000000000000010
 9   00000000000000000000000000000101
10   00000000000000000000000000000110
11   00000000000000000000000000000111
```



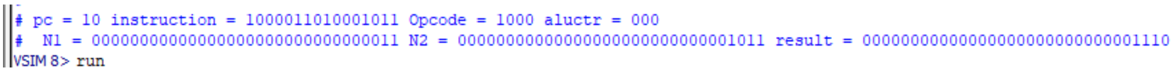
```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000001110
00000000000000000000000000000011
00000000000000000000000000000010
00000000000000000000000000010000
00000000000000000000000000000110
00000000000000000000000000000111
```



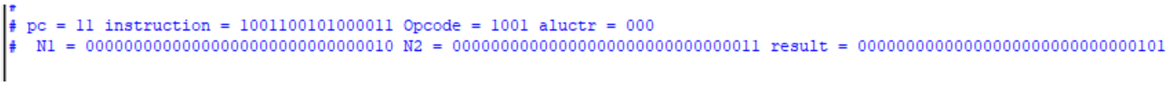
```
# pc = 0 instruction = 1000011101001101 Opcode = 1000 aluctr = 000
#  N1 = 00000000000000000000000000000011 N2 = 00000000000000000000000000001101 result = 00000000000000000000000000010000
run
```



```
# pc = 1 instruction = 1001100001000111 Opcode = 1001 aluctr = 000
#  N1 = 00000000000000000000000000000010 N2 = 00000000000000000000000000000111 result = 00000000000000000000000000001001
run
```

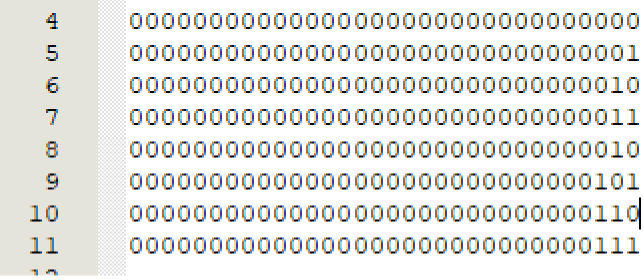


```
# pc = 10 instruction = 1000011010001011 Opcode = 1000 aluctr = 000
#  N1 = 00000000000000000000000000000011 N2 = 00000000000000000000000000001011 result = 00000000000000000000000000001110
VSIM 8> run
```

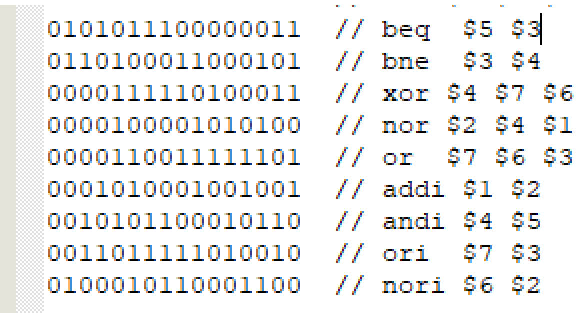


```
# pc = 11 instruction = 1001100101000011 Opcode = 1001 aluctr = 000
#  N1 = 00000000000000000000000000000010 N2 = 00000000000000000000000000000011 result = 00000000000000000000000000000101
```

Beq - Bne :

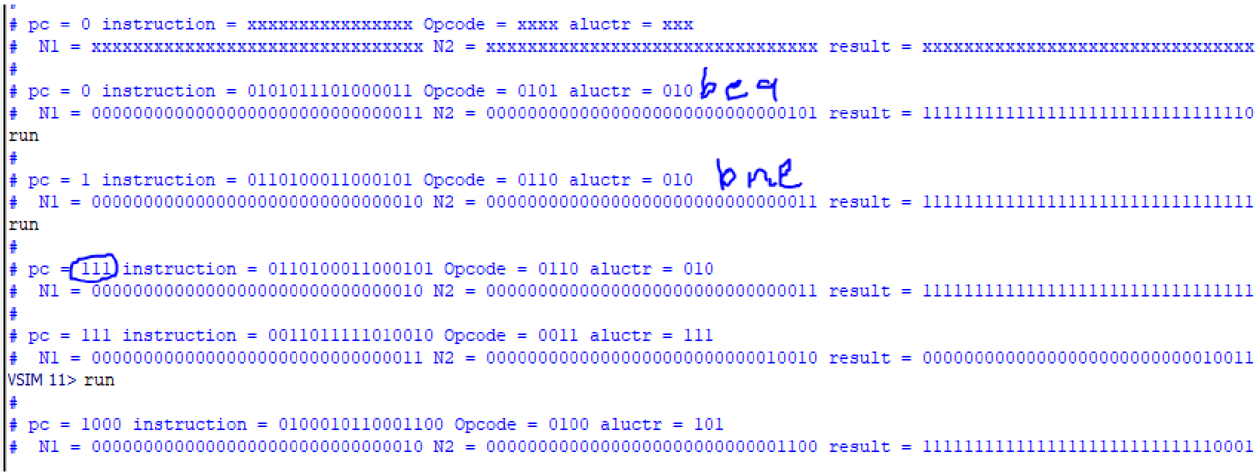


```
 4   00000000000000000000000000000000
 5   00000000000000000000000000000001
 6   00000000000000000000000000000010
 7   00000000000000000000000000000011
 8   00000000000000000000000000000010
 9   00000000000000000000000000000101
10   00000000000000000000000000000110
11   00000000000000000000000000000111
```

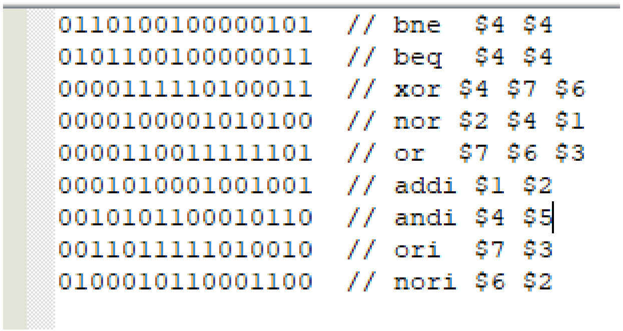


```
0101011100000011  // beq  $5 $3
0110100011000101  // bne  $3 $4
0000111110100011  // xor $4 $7 $6
0000100001010100  // nor $2 $4 $1
0000110011111101  // or  $7 $6 $3
0001010001001001  // addi $1 $2
0010101100010110  // andi $4 $5
0011011111010010  // ori  $7 $3
0100010110001100  // nori $6 $2
```

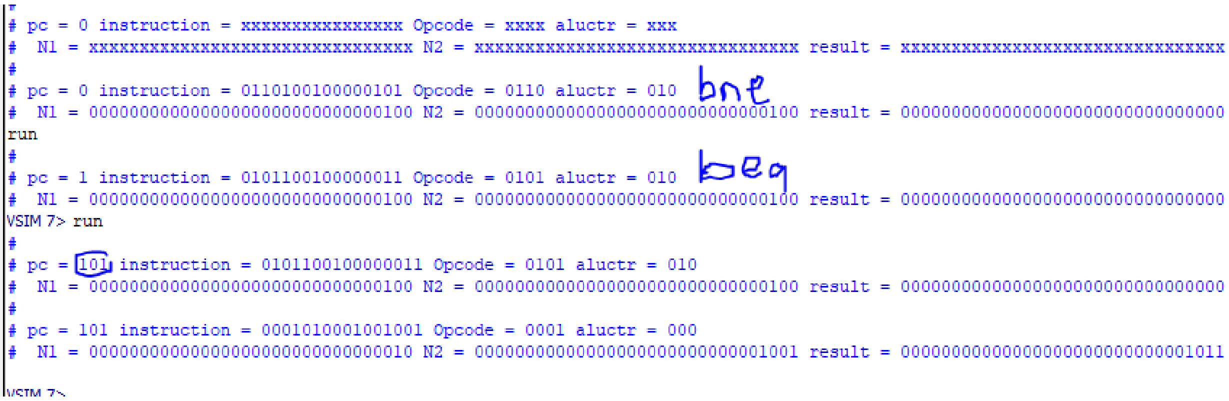
imm = 000011



```
#
# pc = 0 instruction = xxxxxxxxxxxxxxxx Opcode = xxxx aluctr = xxx
#  N1 = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx N2 = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx result = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#
# pc = 0 instruction = 0101011101000011 Opcode = 0101 aluctr = 010 beq
#  N1 = 00000000000000000000000000000011 N2 = 00000000000000000000000000000101 result = 11111111111111111111111111111110
run
#
# pc = 1 instruction = 0110100011000101 Opcode = 0110 aluctr = 010  bne
#  N1 = 00000000000000000000000000000010 N2 = 00000000000000000000000000000011 result = 11111111111111111111111111111111
run
#
# pc = 111 instruction = 0110100011000101 Opcode = 0110 aluctr = 010
#  N1 = 00000000000000000000000000000010 N2 = 00000000000000000000000000000011 result = 11111111111111111111111111111111
#
# pc = 111 instruction = 0011011111010010 Opcode = 0011 aluctr = 111
#  N1 = 00000000000000000000000000000011 N2 = 00000000000000000000000000010010 result = 00000000000000000000000000010011
VSIM 11> run
#
# pc = 1000 instruction = 0100010110001100 Opcode = 0100 aluctr = 101
#  N1 = 00000000000000000000000000000010 N2 = 00000000000000000000000000001100 result = 11111111111111111111111111110001
```



```
0110100100000101  // bne  $4 $4
0101100100000011  // beq  $4 $4
0000111110100011  // xor $4 $7 $6
0000100001010100  // nor $2 $4 $1
0000110011111101  // or  $7 $6 $3
0001010001001001  // addi $1 $2
0010101100010110  // andi $4 $5
0011011111010010  // ori  $7 $3
0100010110001100  // nori $6 $2
```



```
#
# pc = 0 instruction = xxxxxxxxxxxxxxxx Opcode = xxxx aluctr = xxx
#  N1 = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx N2 = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx result = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#
# pc = 0 instruction = 0110100100000101 Opcode = 0110 aluctr = 010  bne
#  N1 = 00000000000000000000000000000100 N2 = 00000000000000000000000000000100 result = 00000000000000000000000000000000
run
#
# pc = 1 instruction = 0101100100000011 Opcode = 0101 aluctr = 010  beq
#  N1 = 00000000000000000000000000000100 N2 = 00000000000000000000000000000100 result = 00000000000000000000000000000000
VSIM 7> run
#
# pc = 101 instruction = 0101100100000011 Opcode = 0101 aluctr = 010
#  N1 = 00000000000000000000000000000100 N2 = 00000000000000000000000000000100 result = 00000000000000000000000000000000
#
# pc = 101 instruction = 0001010001001001 Opcode = 0001 aluctr = 000
#  N1 = 00000000000000000000000000000010 N2 = 00000000000000000000000000001001 result = 00000000000000000000000000001011

VSIM 7>
```