CSE 331/503

HMW4 REPORT:

-Serdil Anıl Ünlü

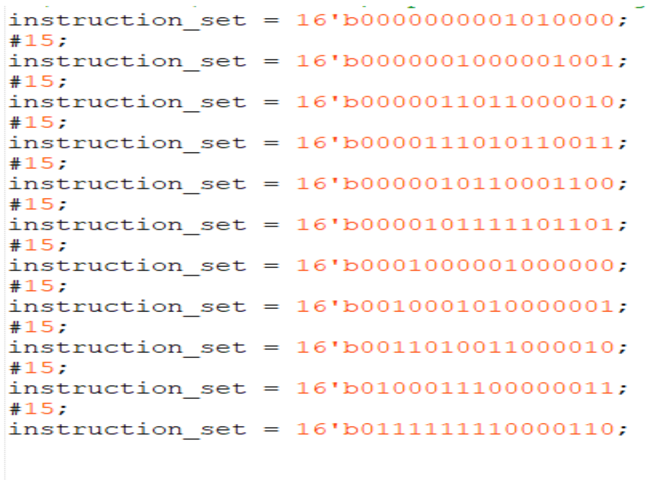
-1801042672

In the homework, I did not do the BEQ, BNE, LW and SW commands in the MiniMips table. Other commands work without any problems.

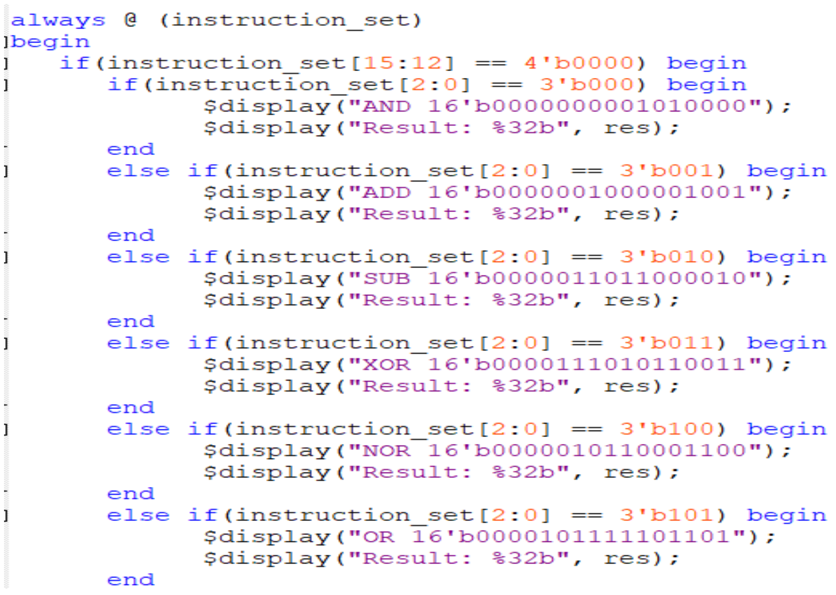
After I did my operations on the ALU on the processor I made, I connected it directly to the result and printed the results without any problems.

While printing the test results, I write the name of the transaction, specify what type of transaction I will write, and print the result of the operation on a bottom line.

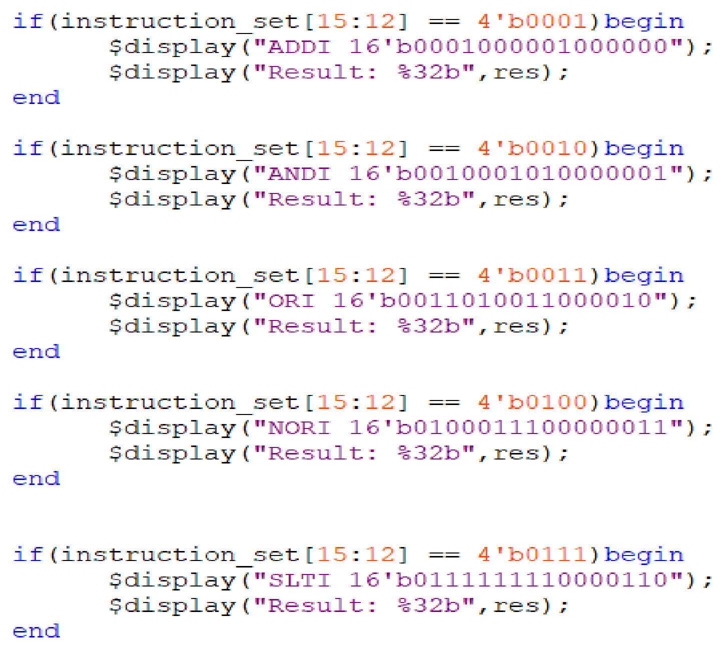
I tested all the processes twice and added the test results to the report with the codes in the testbench.



```
instruction_set = 16'b0000000001010000;
#15;
instruction_set = 16'b0000001000001001;
#15;
instruction_set = 16'b0000011011000010;
#15;
instruction_set = 16'b0000111010110011;
#15;
instruction_set = 16'b0000010110001100;
#15;
instruction_set = 16'b0000101111101101;
#15;
instruction_set = 16'b0001000001000000;
#15;
instruction_set = 16'b0010001010000001;
#15;
instruction_set = 16'b0011010011000010;
#15;
instruction_set = 16'b0100011100000011;
#15;
instruction_set = 16'b0111111110000110;
```

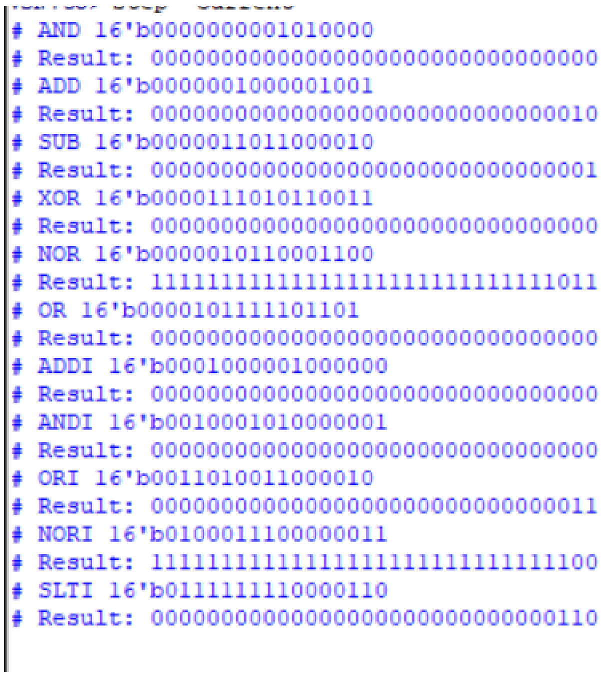


```
always @ (instruction_set)
begin
   if(instruction_set[15:12] == 4'b0000) begin
      if(instruction_set[2:0] == 3'b000) begin
            $display("AND 16'b0000000001010000");
            $display("Result: %32b", res);
      end
      else if(instruction_set[2:0] == 3'b001) begin
            $display("ADD 16'b0000001000001001");
            $display("Result: %32b", res);
      end
      else if(instruction_set[2:0] == 3'b010) begin
            $display("SUB 16'b0000011011000010");
            $display("Result: %32b", res);
      end
      else if(instruction_set[2:0] == 3'b011) begin
            $display("XOR 16'b0000111010110011");
            $display("Result: %32b", res);
      end
      else if(instruction_set[2:0] == 3'b100) begin
            $display("NOR 16'b0000010110001100");
            $display("Result: %32b", res);
      end
      else if(instruction_set[2:0] == 3'b101) begin
            $display("OR 16'b0000101111101101");
            $display("Result: %32b", res);
      end
```



```
if(instruction_set[15:12] == 4'b0001)begin
      $display("ADDI 16'b0001000001000000");
      $display("Result: %32b",res);
end

if(instruction_set[15:12] == 4'b0010)begin
      $display("ANDI 16'b0010001010000001");
      $display("Result: %32b",res);
end

if(instruction_set[15:12] == 4'b0011)begin
      $display("ORI 16'b0011010011000010");
      $display("Result: %32b",res);
end

if(instruction_set[15:12] == 4'b0100)begin
      $display("NORI 16'b0100011100000011");
      $display("Result: %32b",res);
end


if(instruction_set[15:12] == 4'b0111)begin
      $display("SLTI 16'b0111111110000110");
      $display("Result: %32b",res);
end
```

The test result below is the result of the above testbench code.

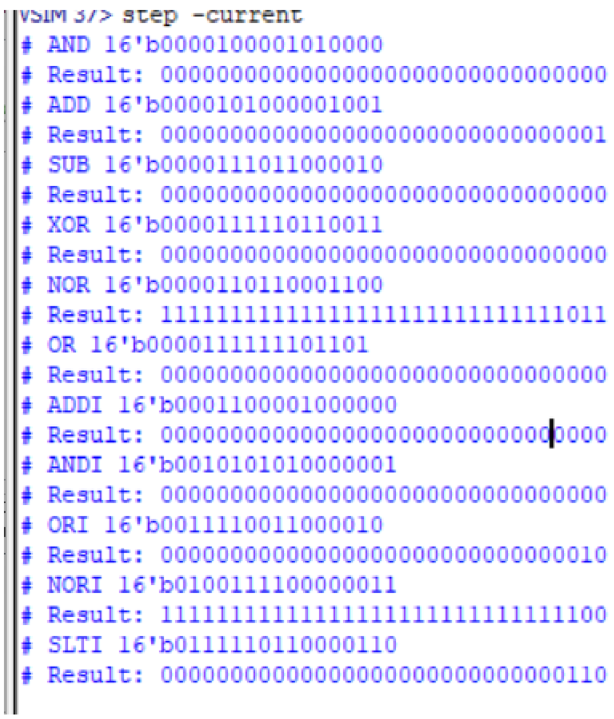


```
# AND 16'b0000000001010000
# Result: 00000000000000000000000000000000
# ADD 16'b0000001000001001
# Result: 00000000000000000000000000000010
# SUB 16'b0000011011000010
# Result: 00000000000000000000000000000001
# XOR 16'b0000111010110011
# Result: 00000000000000000000000000000000
# NOR 16'b0000010110001100
# Result: 11111111111111111111111111111011
# OR 16'b0000101111101101
# Result: 00000000000000000000000000000000
# ADDI 16'b0001000001000000
# Result: 00000000000000000000000000000000
# ANDI 16'b0010001010000001
# Result: 00000000000000000000000000000000
# ORI 16'b0011010011000010
# Result: 00000000000000000000000000000011
# NORI 16'b0100011100000011
# Result: 11111111111111111111111111111100
# SLTI 16'b0111111110000110
# Result: 00000000000000000000000000000110
```

```
instruction_set = 16'b0000100001010000; //AND
#15;
instruction_set = 16'b0000101000001001; //ADD
#15;
instruction_set = 16'b0000111011000010; //SUB
#15;
instruction_set = 16'b0000111110110011; //XOR
#15;
instruction_set = 16'b0000110110001100; //NOR
#15;
instruction_set = 16'b0000111111101101; //OR
#15;
instruction_set = 16'b0001100001000000; //ADDI
#15;
instruction_set = 16'b0010101010000001; //ANDI
#15;
instruction_set = 16'b0011110011000010; //ORI
#15;
instruction_set = 16'b0100111100000011; //NORI
#15;
instruction_set = 16'b0111110110000110; //SLTI
```

```
if(instruction_set[15:12] == 4'b0000) begin
   if(instruction_set[2:0] == 3'b000) begin
         $display("AND 16'b0000100001010000");
         $display("Result: %32b", res);
   end
   else if(instruction_set[2:0] == 3'b001) begin
         $display("ADD 16'b0000101000001001");
         $display("Result: %32b", res);
   end
   else if(instruction_set[2:0] == 3'b010) begin
         $display("SUB 16'b0000111011000010");
         $display("Result: %32b", res);
   end
   else if(instruction_set[2:0] == 3'b011) begin
         $display("XOR 16'b0000111110110011");
         $display("Result: %32b", res);
   end
   else if(instruction_set[2:0] == 3'b100) begin
         $display("NOR 16'b0000110110001100");
         $display("Result: %32b", res);
   end
   else if(instruction_set[2:0] == 3'b101) begin
         $display("OR 16'b0000111111101101");
         $display("Result: %32b", res);
   end

 if(instruction_set[15:12] == 4'b0001)begin
       $display("ADDI 16'b0001100001000000");
       $display("Result: %32b",res);
 end

 if(instruction_set[15:12] == 4'b0010)begin
       $display("ANDI 16'b0010101010000001");
       $display("Result: %32b",res);
 end

 if(instruction_set[15:12] == 4'b0011)begin
       $display("ORI 16'b0011110011000010");
       $display("Result: %32b",res);
 end

 if(instruction_set[15:12] == 4'b0100)begin
       $display("NORI 16'b0100111100000011");
       $display("Result: %32b",res);
 end


 if(instruction_set[15:12] == 4'b0111)begin
       $display("SLTI 16'b0111110110000110");
       $display("Result: %32b",res);
 end
```

The result of the 2nd test below is the result of the above testbench code.



```
VSIM 37> step -current
# AND 16'b0000100001010000
# Result: 00000000000000000000000000000000
# ADD 16'b0000101000001001
# Result: 00000000000000000000000000000001
# SUB 16'b0000111011000010
# Result: 00000000000000000000000000000000
# XOR 16'b0000111110110011
# Result: 00000000000000000000000000000000
# NOR 16'b0000110110001100
# Result: 11111111111111111111111111111011
# OR 16'b0000111111101101
# Result: 00000000000000000000000000000000
# ADDI 16'b0001100001000000
# Result: 00000000000000000000000000000000
# ANDI 16'b0010101010000001
# Result: 00000000000000000000000000000000
# ORI 16'b0011110011000010
# Result: 00000000000000000000000000000010
# NORI 16'b0100111100000011
# Result: 11111111111111111111111111111100
# SLTI 16'b0111110110000110
# Result: 00000000000000000000000000000110
```