1) AND

Tests:

```
initial begin
a=5; b=3;
#`DELAY;
a=10; b=8;
#`DELAY;
a=300; b=15;
#`DELAY;
a=429400000; b=67296;
#`DELAY;
end
```

Results:

```
VSIM 47> step -current
# time =  0, a =          5, b=          3, result =          1
# time = 20, a =         10, b=          8, result =          8
# time = 40, a =        300, b=         15, result =         12
# time = 60, a = 429400000, b=      67296, result =       1728
```

2) ADD

Tests:

```
initial begin
a=15; b=20; carry_in=0;
#`DELAY;
a=10; b=8; carry_in=1;
#`DELAY;
a=300; b=15; carry_in=1;
#`DELAY;
a=4294900000; b=67296; carry_in=1;
#`DELAY;
end
```

Results:

```
VSIM 45> step -current
# time =  0, a =         15, b=         20, carry_in=0, sum =         35, carry_out=0
# time = 20, a =         10, b=          8, carry_in=1, sum =         19, carry_out=0
# time = 40, a =        300, b=         15, carry_in=1, sum =        316, carry_out=0
# time = 60, a =4294900000, b=      67296, carry_in=1, sum =          1, carry_out=1
```

3)Sub

Tests:

```
initial begin
a=21; b=20;
#`DELAY;
a=10; b=8;
#`DELAY;
a=300; b=15;
#`DELAY;
a=45; b=4294967260;
#`DELAY;
end
```

Results:

```
VSIM 42> step -current
# time =  0, a =        21, b=        20, result =         1, carry_out=1
# time = 20, a =        10, b=         8, result =         2, carry_out=1
# time = 40, a =       300, b=        15, result =       285, carry_out=1
# time = 60, a =        45, b=4294967260, result =        81, carry_out=0
```

4) XOR

Tests:

```
initial begin
a=5; b=3;
#`DELAY;
a=10; b=8;
#`DELAY;
a=300; b=15;
#`DELAY;
a=429400000; b=67296;
#`DELAY;
end
```

Results:

```
VSIM 75> step -current
# time =  0, a =00000000000000000000000000000101, b=00000000000000000000000000000011,
#  result =00000000000000000000000000000110
# time = 20, a =00000000000000000000000000001010, b=00000000000000000000000000001000,
#  result =00000000000000000000000000000010
# time = 40, a =00000000000000000000000100101100, b=00000000000000000000000000001111,
#  result =00000000000000000000000100100011
# time = 60, a =00011001100110000001111111000000, b=00000000000000010000011011100000,
#  result =00011001100110010001100100100000
```

5) NOR

Test:

```
initial begin
a=5; b=3;
#`DELAY;
a=10; b=8;
#`DELAY;
a=300; b=15;
#`DELAY;
a=429400000; b=67296;
#`DELAY;
end
```

Result:

```
VSIM 61> step -current
# time =  0, a =00000000000000000000000000000101, b=00000000000000000000000000000011,
#  result =11111111111111111111111111111000
# time = 20, a =00000000000000000000000000001010, b=00000000000000000000000000001000,
#  result =11111111111111111111111111110101
# time = 40, a =00000000000000000000000100101100, b=00000000000000000000000000001111,
#  result =11111111111111111111111011010000
# time = 60, a =00011001100110000001111111000000, b=00000000000000010000011011100000,
#  result =11100110011001101110000000011111
```

6) OR

Test:

```
initial begin
a=5; b=3;
#`DELAY;
a=10; b=8;
#`DELAY;
a=300; b=15;
#`DELAY;
a=429400000; b=67296;
#`DELAY;
end
```

Result:

```
VSIM 54> step -current
# time =  0, a =          5, b=          3, result =          7
# time = 20, a =         10, b=          8, result =         10
# time = 40, a =        300, b=         15, result =        303
# time = 60, a = 429400000, b=      67296, result = 429465568
```

7)BEQ

Test:

```
0101_010_010_000001
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0101_000_001_000101
0110_010_010_000010
```

beq $2,$2,1 => Statement is true so new PC must be PC+1 +4➔due to shifting 000001=>000100

beq,$0,$1,5 => Statement is false, new PC =PC+1

bne $2,$2 ,2

Result: Test Passed.

```
counter=      18,instruction=0100010011001100,in1=      2,in2=        12, result= 4294967281, nextCounter=      19
counter=      19,instruction=0100000001000101,in1=      1,in2=         5, result= 4294967290, nextCounter=      20
counter=      20,instruction=0101010010000001,in1=      2,in2=         2, result=          0, nextCounter=      25
counter=      25,instruction=0101000001000101,in1=      1,in2=4294967290, result=          7, nextCounter=      26
counter=      26,instruction=0110010010000010,in1=      2,in2=         2, result=          0, nextCounter=      27
```

8)BNE

Test:

```
0101_000_001_000101
0110_010_010_000010
0110_000_001_000010
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0111_010_010_001100
```

beq $0,$1,5 =>statement is false PC<=PC+1

bne $2,$2,2=>statement is false PC<=PC+1

bne $0,$1,2=>statement is true PC<=PC+1+8➔due to shifting :000010=>001000

slti $2,$2,12

Result: Test Passed.

```
counter=      25,instruction=0101000001000101,in1=      1,in2=4294967290, result=          7, nextCounter=      26
counter=      26,instruction=0110010010000010,in1=      2,in2=         2, result=          0, nextCounter=      27
counter=      27,instruction=0110000001000010,in1=      1,in2=4294967290, result=          7, nextCounter=      36
counter=      36,instruction=0111010010001100,in1=      2,in2=        12, result=          1, nextCounter=      37
```

9)SLT

Test:

```
initial begin
a=21; b=20;
#`DELAY;
a=5; b=8;
#`DELAY;
a=300; b=15;
#`DELAY;
a=45; b=-999999;
#`DELAY;
end
```

Result:

```
VSIM 47> step -current
# time =  0, a =        21, b=        20, result =         0
# time = 20, a =         5, b=         8, result =         1
# time = 40, a =       300, b=        15, result =         0
# time = 60, a =        45, b=4293967297, result =         0
```

10)LW

Test: $0=1 $1=2 $3=8

```
1000_000_011_001100 → lw $3,mem[$0+12]
1000_000_001_000101 → lw $1,mem[$0+5]
```

Result: Test Passed.

```
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
```

→ MEMORY

Before:

```
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000010
00000000000000000000000000001000
00000000000000000000000000010000
00000000000000000000000000100000
00000000000000000000000001000000
00000000000000000000000010000000
```

After:

```
// format=bin addressradix=h datar
00000000000000000000000000000001
00000000000000000000000000000110
00000000000000000000000000000001
00000000000000000000000000001101
00000000000000000000000000010000
00000000000000000000000000100000
11111111111111111111111111111110
00000000000000000000000010000000
```

11)SW

Test: $0=1

```
1001_000_110_001101
1001_000_100_000100
```

sw $6,mem[$0+13]

sw $4,mem[$0+4]

Result: Test Passed.

```
00000000000000000000000000000001
00000000000000000000000000000110
00000000000000000000000000000001
00000000000000000000000000001101
00000000000000000000000000010000
00000000000000000000000000100000
11111111111111111111111111111110
00000000000000000000000010000000
```

Registers

Before:

```
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
00000000000000000000000000001110
00000000000000000000000000001111
```

After:

```
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000010000
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
11111111111111111111111111111110
00000000000000000000000000001111
```

12) ALU

Tests:

```
initial begin
    alu_control=3'b000;
    readData1=32'd5;
    data2=32'd5;
    #`DELAY;
    alu_control=3'b001;
    readData1=32'd5;
    data2=32'd5;
    #`DELAY;
    alu_control=3'b010;
    readData1=32'd5;
    data2=32'd5;
    #`DELAY;
    alu_control=3'b011;
    readData1=32'd5;
    data2=32'd5;
    #`DELAY;
    alu_control=3'b100;
    readData1=32'd5;
    data2=32'd5;
    #`DELAY;
    alu_control=3'b101;
    readData1=32'd5;
    data2=32'd5;
    #`DELAY;
    alu_control=3'b110;
    readData1=32'd5;
    data2=32'd5;
    #`DELAY;
    alu_control=3'b111;
    readData1=32'd5;
    data2=32'd6;
    #`DELAY;
end
```

Results:

```
VSIM 50> step -current
# time =  0, alu_control=000 , readDatal =         5, data2=         5, alu_res=         5 , zero= 0
# time = 20, alu_control=001 , readDatal =         5, data2=         5, alu_res=        10 , zero= 1
# time = 40, alu_control=010 , readDatal =         5, data2=         5, alu_res=         0 , zero= 1
# time = 60, alu_control=011 , readDatal =         5, data2=         5, alu_res=         0 , zero= 0
# time = 80, alu_control=100 , readDatal =         5, data2=         5, alu_res=4294967290 , zero= 0
# time = 100, alu_control=101 , readDatal =         5, data2=         5, alu_res=         5 , zero= 1
# time = 120, alu_control=110 , readDatal =         5, data2=         5, alu_res=         0 , zero= 1
# time = 140, alu_control=111 , readDatal =         5, data2=         6, alu_res=4294967295 , zero= 1
# A time value could not be extracted from the current line
```

13) ALU CONTROL

			AluOP			func			Out			
			A	B	C	D	E	F	2	1	0	
		and	0	0	0	0	0	0	0	0	0	
		add	0	0	0	0	0	1	0	0	1	
		sub	0	0	0	0	1	0	0	1	0	
		xor	0	0	0	0	1	1	0	1	1	
		nor	0	0	0	1	0	0	1	0	0	
		or	0	0	0	1	0	1	1	0	1	
sw	lw	addi	0	0	1	x	x	x	0	0	1	
		andi	0	1	0	x	x	x	0	0	0	
		ori	0	1	1	x	x	x	1	0	1	
		nori	1	0	0	x	x	x	1	0	0	
		beq	1	0	1	x	x	x	0	1	0	sub
		slti	1	1	0	x	x	x	1	1	0	slt
		bne	1	1	1	x	x	x	1	1	1	

out[2]= BC + AC' + B'C'D
out[1]= AC + AB + A'B'C'E
out[0]= A'C + BC + A'B'F

Test:

```
AluOP=3'b000;func=3'b000;
#`DELAY;
AluOP=3'b000;func=3'b011;
#`DELAY;
AluOP=3'b000;func=3'b101;
#`DELAY;
AluOP=3'b001;func=3'b001;
#`DELAY;
AluOP=3'b001;func=3'b000;
#`DELAY;
AluOP=3'b101;func=3'b001;
#`DELAY;
AluOP=3'b111;func=3'b010;
#`DELAY;
AluOP=3'b101;func=3'b111;
#`DELAY;
```

Result:

```
# time =  0, AluOP=000,func=0,out=000
# time = 20, AluOP=000,func=3,out=011
# time = 40, AluOP=000,func=5,out=101
# time = 60, AluOP=001,func=1,out=001
# time = 80, AluOP=001,func=0,out=001
# time = 100, AluOP=101,func=1,out=010
# time = 120, AluOP=111,func=2,out=111
# time = 140, AluOP=101,func=7,out=010
```

14) Instruction Memory

Test:

```
initial
   begin
   #`DELAY;
      address=32'd1;
      #`DELAY;
      address=32'd2;
      #`DELAY;
      address=32'd3;
      #`DELAY;
      address=32'd3;
      #`DELAY;
      address=32'd4;
      #`DELAY;
      $stop;
   end
```

Instructions:

1	0000_010_011_110_000
2	0000_000_001_011_000
3	0000_010_011_110_001
4	0000_000_001_011_001
5	0000_010_011_110_010

Result: address is initially zero , so prints first instruction too.

```
#    Time: 0 ps  Iteration: 0  Instance: /ins
# time =  0, instruction=0000010011110000
# time = 20, instruction=0000000001011000
# time = 40, instruction=0000010011110001
# time = 60, instruction=0000000001011001
# time = 100, instruction=0000010011110010
# Break in Module instruction_memory_test at
```

15)CONTROL UNIT

OPCODE											
A	B	C	D	regDst	branch	memRead	memToReg	AluOP	memWrite	AluSrc	regWrite
0	0	0	0	1	0	0	0	000	0	0	1
0	0	0	1	0	0	0	0	001	0	1	1
0	0	1	0	0	0	0	0	010	0	1	1
0	0	1	1	0	0	0	0	011	0	1	1
0	1	0	0	0	0	0	0	100	0	1	1
0	1	0	1	x	1	0	x	101	0	0	0
0	1	1	0	x	1	0	x	111	0	0	0
0	1	1	1	0	0	0	0	110	0	1	1
1	0	0	0	0	0	1	1	001	0	1	1
1	0	0	1	x	0	0	x	001	1	1	0
1	0	1	0	x	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x	x

regDst= A'B'C'D'

branch= BC'D + BCD'

memRead=AD'

memToReg= A

memWrite= AD

AluSrc= A + B'D + B'C + CD + BC'D'

reqWrite=A'B' + C'D' + CD

AluOP<2>= B

AluOP<1>= C

AluOP<0>= A + B'D + C'D + BCD'

Test:

```
instruction=4'b0000;
#`DELAY;
instruction=4'b0001;
#`DELAY;
instruction=4'b0010;
#`DELAY;
instruction=4'b0011;
#`DELAY;
instruction=4'b0100;
#`DELAY;
```

```
instruction=4'b0101;
#`DELAY;
instruction=4'b0110;
#`DELAY;
instruction=4'b0111;
#`DELAY;
instruction=4'b1000;
#`DELAY;
instruction=4'b1001;
#`DELAY;
$stop;
```

Result:

```
# time =  0, instruction=0000,regDst=1,branch=0,memRead=0,memToReg=0,AluOP=000,memWrite=0,AluSrc= 0,regWrite=1
# time = 20, instruction=0001,regDst=0,branch=0,memRead=0,memToReg=0,AluOP=001,memWrite=0,AluSrc= 1,regWrite=1
# time = 40, instruction=0010,regDst=0,branch=0,memRead=0,memToReg=0,AluOP=010,memWrite=0,AluSrc= 1,regWrite=1
# time = 60, instruction=0011,regDst=0,branch=0,memRead=0,memToReg=0,AluOP=011,memWrite=0,AluSrc= 1,regWrite=1
# time = 80, instruction=0100,regDst=0,branch=0,memRead=0,memToReg=0,AluOP=100,memWrite=0,AluSrc= 1,regWrite=1
# time = 100, instruction=0101,regDst=0,branch=1,memRead=0,memToReg=0,AluOP=101,memWrite=0,AluSrc= 0,regWrite=0
# time = 120, instruction=0110,regDst=0,branch=1,memRead=0,memToReg=0,AluOP=111,memWrite=0,AluSrc= 0,regWrite=0
# time = 140, instruction=0111,regDst=0,branch=0,memRead=0,memToReg=0,AluOP=110,memWrite=0,AluSrc= 1,regWrite=1
# time = 160, instruction=1000,regDst=0,branch=0,memRead=1,memToReg=1,AluOP=001,memWrite=0,AluSrc= 1,regWrite=1
# time = 180, instruction=1001,regDst=0,branch=0,memRead=0,memToReg=1,AluOP=001,memWrite=1,AluSrc= 1,regWrite=0
# Break in Module control_unit_test at C:/Users/BARAN SOLMAZ/Desktop/Solmaz Baran 1801042601/control_unit_test.v
```

16) Register File

Test:

```
initial
   begin
      read_reg1=3'd1; read_reg2=3'd2;
      #`DELAY;
      reg_write=1'b1; write_data=32'd123;write_reg=3'b101;
      #`DELAY;
      $writememb("out_reg.mem",fatb.data);
      $stop;
   end
```

Result:

Before:

```
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000010
00000000000000000000000000001000
00000000000000000000000000010000
00000000000000000000000000100000
00000000000000000000000001000000
00000000000000000000000010000000
```

After:

```
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000010
00000000000000000000000000001000
00000000000000000000000000010000
00000000000000000000000001111011
00000000000000000000000001000000
00000000000000000000000010000000
```

17) Data Memory

Test:

```
initial
   begin
      memWrite=1'd1; address=32'd2;write_data=32'd123;
      #`DELAY;
      memWrite=1'b0; memRead=1'b1;
      #`DELAY;
      $writememb("out_memmory.mem",fatb.data);
      $stop;
   end
```

Result:

```
VSIM 107> step -current
# time =  0, adres=        2, write_data=      123,memWrite=1,memRead=0, read_data=        x
# time = 20, adres=        2, write_data=      123,memWrite=0,memRead=1, read_data=      123
# Break in Module data_memory_test at C:/Users/BARAN SOLMAZ/Desktop/Solmaz Baran 1801042601/data
```

Before:

```
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000011
```

After:

```
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000001111011
00000000000000000000000000000011
```

18)Sign Extend

Test:

```
initial begin
a=6'b000101;
#`DELAY;
a=6'b001010;
#`DELAY;
a=6'b110010;
#`DELAY;

end
```

Result:

```
# time =  0, a =                    000101, result =00000000000000000000000000000101
# time = 20, a =                    001010, result =00000000000000000000000000001010
# time = 40, a =                    110010, result =11111111111111111111111111110010
```

19)MiniMips

Tests: instructions.mem

Results: output.txt / modelsim->transcript

output.txt is a copy of modelsim->transcript.