Name (Last, First): _____

This exam consists of 5 questions on 9 pages; be sure you have the entire exam before starting. The point value of each question is indicated at its beginning; the entire exam has 100 points. Individual parts of a multi-part question are generally assigned approximately the same point value: exceptions are noted.

You are allowed to have one page of notes (front and back). You may not use any other materials. You may not share notes with classmates during the exam.

Be concise and clearly indicate your answer. Presentation and simplicity of your answers may affect your grade. Answer each question in the space following the question. If you find it necessary to continue an answer elsewhere, indicate clearly the location of its continuation and label its continuation with the question number and subpart if appropriate.

You should read through all the questions first, then pace yourself.

If you need to ask a question during the exam, please raise your hand and I will come to your desk. Do not come to me.

The questions begin on the next page.

Problem	Possible	Score
1	20	
2	20	
3	20	
4	20	
5	20	
Total	100	

1. (_____/20 points)

Short Questions

(a) (5 points) Give the 8 bit two's complement representation of -21. Show your work.

(b) (5 points) What is the machine code representation of the MIPS assembly following instruction?

```
lw $t0, 12($S1)
```

Hint ($t0 is register 8, $s0 is register 17, the opcode for lw is 35).

(c) (10 points) Consider a boolean function that has three inputs: a, b, c, and one output r. The output r is true if at least two of the inputs are 1. Give the sum of products equation for this function. Show your work by deriving the truth table and then giving the boolean equation. You do not need to minimize the equation.

2. (_____/20 points)

MIPS on MIPS

As we found in Project 1, it is useful to write MIPS assembly code that can analyze MIPS machine code. For this problem you will write a MIPS function called `isMemEqualNeg`, that takes one argument, register $a0 and returns a value in register $v0. This function expects a MIPS instruction word in $a0. It will return the value 1 (true) in register $v0 if and only if all the following conditions are true:

- The instruction word in $a0 is a lw or sw instruction.
- The two operands in the instruction word are equal. That is they all specify the same register number.
- The 16 bit immediate field is negative.

Otherwise the function returns the value 0 in register $v0.

(Note: the opcode for lw is 35 and the opcode for sw is 43.)

Here are some examples:

- `lw $t0, -4($t0)$` returns 1
- `sw $s1, -20($s1)` returns 1
- `lw $t1, 4($t1$)$` returns 0
- `add $t1, $t1, $s0` returns 0

3. (_____/20 points)

Basic Blocks

Consider the following MIPS assembly language function that sums the four arguments a passed to it in `$a0, $a1, $a2, $a3`. It returns the sum in `$v0`.

```
foo:
    add $v0, $a0, $a1
    add $v0, $v0, $a2
    add $v0, $v0, $a3
    jr $ra
```

If we gave this MIPS code as input to your basic block finder it would identify one basic block with a length of 3. How can you modify this code so that your basic block finder identifies 3 basic blocks of each of length 1. You can add instructions, but the new code should not modify any additional registers. That is, the new function foo should work the same way as the original, but it should contain 3 basic blocks each of length 1. Give your modified code here:

4. (_____/20 points)

Verilog to Schematic

For each of the following Verilog modules, draw a corresponding schematic representation using gates and logic components such as muxes, adders, and registers.

(a) Simple

```
module foo(input a, b, c,
           ouput r)

    assign r = (~(a | b)) & c;
endmodule
```

(a) Harder

```
module goo(input [31:0] a, b, c
           input s,
           ouput [31:0] r)

    assign r = (s ? a + b : a + c);

endmodule
```

5. (_____/20 points)

Verilog - LED Animation

Consider the following implementation of the LED Animation finite state machine. Currently this implementation animates a pair of LED lights by moving the pair to a new position (currently to the right) on each clock cycle. Now consider a new input to the module called "direction". This is a 1-bit input that will be connected to one of the switches on the Spartan-3E board. The purpose of this switch is to change the direction of the animation. When direction = 0, the animation will move to the right as normal. However, when direction = 1, the animation should move in the opposite direction (to the left). Make modifications to the module below to support the new direction input. Explain your answer.

```
module animFSM(input clk, input reset, input direction
               output [7:0] ledout);

    reg [7:0] state, nextstate;
    parameter S0 = 8'b1100_0000;
    parameter S1 = 8'b0110_0000;
    parameter S2 = 8'b0011_0000;
    parameter S3 = 8'b0001_1000;
    parameter S4 = 8'b0000_1100;
    parameter S5 = 8'b0000_0110;
    parameter S6 = 8'b0000_0011;
    parameter S7 = 8'b1000_0001;

    // state register
    always @ (posedge clk, posedge reset)
        if (reset) state <= S0;
        else state <= nextstate;

    // next state logic
    always @ (*)
        case (state)
            S0: nextstate = S1;
            S1: nextstate = S2;
            S2: nextstate = S3;
            S3: nextstate = S4;
            S4: nextstate = S5;
            S5: nextstate = S6;
            S6: nextstate = S7;
            S7: nextstate = S0;
            default: nextstate = S0;
        endcase
    // output logic
    assign ledout = state;
endmodule
```

Continue your answer here.

Continue your answers here if necessary.