1. The IEEE single-precision floating-point format is composed of a 1-bit sign, an 8-bit biased exponent with a bias of 127 and a 23-bit fraction. Consider the following two operands $X$ and $Y$ in this format, which are specified in hex:

$$X = 42018000 \quad Y = C1240000$$

(a) (16 points) Determine the decimal value of each of the operands.

(b) (9 points) Determine the IEEE single-precision floating-point representation (specified in hex) for the sum $X + Y$. (Perform the addition in decimal and then convert the result into the IEEE format.)

2. The following machine language VeSPA program is placed into the file v.out. (Note that some instruction encodings are provided on the following page.)

58 00 00 19
28 40 00 00
30 83 FF FF
08 C4 08 00
FB 00 00 00

(a) (15 points) Give the corresponding assembly language program.

(b) (15 points) Give the 32-bit contents (specified in hex) of the PC and registers r0 through r3, inclusive, and the values of the condition code bits C, V, Z and N after each instruction has been executed by behavioral.v.

3. On a certain computer, instructions in Class A have a CPI of 2, instructions in Class B have a CPI of 3 and instructions in Class C have a CPI of 4. Consider the following two code sequences: Code sequence 1 contains 3 instructions in Class A, 4 instructions in Class B and 3 instructions in Class C. Code sequence 2 contains 10 instructions in Class A, 2 instructions in Class B and 8 instructions in Class C.

(a) (4 points) Determine the number of instructions executed in each sequence.

(b) (8 points) Determine the number of CPU clock cycles required for each sequence.

(c) (8 points) Determine the CPI for each sequence.
4. (25 points) A file called calc.v contains the following:

```verilog
module calc(a, b, r);

input [10:0] a, b;
output [12:0] r;
wire [11:0] e;
wire [3:0] f;
assign e = a + b + 1;
assign f = {1'b0, e[11], e[10]} + 3'b111;
assign r = {f[2:0], e[9:0]};
endmodule
```

Another file called tbcalc.v contains the following:

```verilog
module tbcalc; // calc testbench

reg [10:0] x, y;
wire [12:0] z;
integer zval;

// instantiate the calc module
calc cl(x, y, z);

// perform a set of simulations and print the results
initial begin
    x = 1022;
    y = 1023;
    repeat (3) begin
        #10 zval = -z[12]*4096 + z[11:0];
        $display($time, " x = %d , y = %d , z = %d", x, y, zval);
        x = x + 1;
    end
end
endmodule
```

If the command:

```
verilog tbcalc.v calc.v
```

is executed, give the output that will be produced by the $display statement. Be sure to clearly show how you obtained your answer by including all of the intermediate calculations.
ADD* – Addition
*This instruction sets the condition code bits.

Assembly code notation
a) ADD rdst, rs1, rs2
b) ADD rdst, rs1, #immed16

Instruction encoding
a) 31 27 26 22 21 17 16 15 11 10 0
   00001 rdst rs1 0 rs2 000 0000 0000

   b) 31 27 26 22 21 17 16 15 0
      00001 rdst rs1 1 immed16

HLT – Halt
Assembly code notation
HLT

Instruction encoding
31 27 26 0
1111 000 0000 0000 0000 0000 0000 0000

LDI – Load immediate
Assembly code notation
LDI rdst, #value

Instruction encoding
31 27 26 22 21 0
01011 rdst immed22

NOT – Bit-wise logical complement
Assembly code notation
a) NOT rdst, rs1

Instruction encoding
31 27 26 22 21 17 16 0
00101 rdst rs1 0 0000 0000 0000 0000

XOR – Bit-wise logical exclusive-OR
Assembly code notation
a) XOR rdst, rs1, rs2
b) XOR rdst, rs1, #immed16

Instruction encoding
a) 31 27 26 22 21 17 16 15 11 10 0
   00110 rdst rs1 0 rs2 000 0000 0000

   b) 31 27 26 22 21 17 16 15 0
      00110 rdst rs1 1 immed16