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 in this format, which are specified in hex:

BE000000 42100000

(a) (10 points) Determine the decimal value of each of the operands.

(b) (5 points) Determine the IEEE single-precision floating-point representation (specified in hex) for the product of these two operands.

2. A file called q.v contains the following:

```verilog
module half_adder(a, b, s, cout);
  input a, b;
  output s, cout;
  assign s = a^b;
  assign cout = a&b;
endmodule

module full_adder(a, b, cin, s, cout);
  input a, b, cin;
  output s, cout;
  assign s = a^b^cin;
  assign cout = (a&b) | (b&cin) | (a&cin);
endmodule

module q(x, y, z);
  input [1:0] x, y;
  output [3:0] z;
  wire t1, t2, t3;
  assign z[0] = x[0]&y[0];
  half_adder hal(x[1]&y[0], x[0]&y[1], z[1], t1);
  full_adder fa1(x[1]&y[0], t1, x[1]&y[1], z[2], t2);
  full_adder fa2(x[1]&y[0], t2, x[1]&y[1], z[3], t3);
endmodule
```
Another file called tbq.v contains the following:

```verilog
module tbq; // testbench for module q

reg [1:0] x, y;
integer xval;
wire [3:0] z;
integer zval;
integer i, j;

q q1(x, y, z);

initial begin
  for (i = 0; i < 4; i = i + 1) begin
    x = i;
    xval = -x[1]*2 + x[0];
    for (j = 0; j < 4; j = j + 1) begin
      y = j;
      $display($time, "%d %d %d", xval, y, zval);
    end
  end
endmodule
```

(a) (15 points) If the command:

```
verilog q.v tbq.v
```

is executed, give the output that will be produced by the `$display` statement. Be sure to clearly show how you obtained your answer!

(b) (5 points) What mathematical function is performed by module q? Be as precise as you can in terms of the number of bits in the operands and the result and whether the quantities are signed or unsigned.

3. Consider the following portion of a VeSPA assembly language program:

```
and r1, r2, r3 ; instruction 1
or r4, r1, r5 ; instruction 2
xor r6, r1, r4 ; instruction 3
add r3, r8, r9 ; instruction 4
sub r6, r3, r1 ; instruction 5
```

(a) (5 points) Identify the flow dependences that exist amongst these instructions. In each case, specify the register that is involved.

(b) (5 points) Suppose that instruction 1 is in the instruction fetch (IF) stage of the 5-stage pipelined VeSPA implementation during cycle 1. Determine which registers are being read from and written to the register file during cycle 5.
4. 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 80 00 32
10 c5 00 3c
28 46 00 00
30 02 10 00
F8 00 00 00

(a) (7 points) Give the corresponding assembly language program. (Specify any immediate values in decimal format.)

(b) (8 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.

5. (10 points) Consider the following parameter values: memory is byte-addressable, the virtual address is 30 bits, the page size is 8K bytes and each page table entry is 32 bits. Determine the number of bits in the virtual page number field, the number of bits in the page offset field, the number of page table entries and the total size (in Kbytes) of the page table.

6. (15 points) Here is a series of address references given as word addresses: 4, 5, 12, 10, 4, 20, 21, 13, 2, 5, 20, 18, 2, 4. Show the hits and misses and the final cache contents for a 2-ways set-associative cache with one-word blocks and a total size of 16 words. Assume that LRU replacement cache is used.

7. A program has a branch statement. When the program is executed, the outcomes of the branch are as follows (T = taken, N = not taken): T, N, T, T, N, N, T, N, T, T. List the predictions and give the prediction accuracy (expressed as a percentage) for each of the following four branch predictors:

   (a) (2 points) Always taken
   (b) (2 points) Always not taken
   (c) (4 points) 1-bit predictor, initialized to predict taken
   (d) (7 points) 2-bit predictor, initialized to weakly predict taken
**SUB** – Subtraction

*This instruction sets the condition code bits.*

**Assembly code notation**

a) SUB rdst, rs1, rs2
b) SUB rdst, rs1, #immed16

**Instruction encoding**

<table>
<thead>
<tr>
<th>31 ... 27</th>
<th>26 ... 22</th>
<th>21 ... 17</th>
<th>16</th>
<th>15 ... 11</th>
<th>10 ... 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00010</td>
<td>rdst</td>
<td>rs1</td>
<td>0</td>
<td>rs2</td>
<td>0 000000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31 ... 27</th>
<th>26 ... 22</th>
<th>21 ... 17</th>
<th>16</th>
<th>15 ... 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00010</td>
<td>rdst</td>
<td>rs1</td>
<td>1</td>
<td>immed16</td>
</tr>
</tbody>
</table>

**HLT** – Halt

**Assembly code notation**

HLT

**Instruction encoding**

<table>
<thead>
<tr>
<th>31 ... 27</th>
<th>26 ... 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111</td>
<td>0000000000000000</td>
</tr>
</tbody>
</table>

**LDI** – Load immediate

**Assembly code notation**

LDI rdst, #value

**Instruction encoding**

<table>
<thead>
<tr>
<th>31 ... 27</th>
<th>26 ... 22</th>
<th>21 ... 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>01011</td>
<td>rdst</td>
<td>immed22</td>
</tr>
</tbody>
</table>

**NOT** – Bit-wise logical complement

**Assembly code notation**

a) NOT rdst, rs1

**Instruction encoding**

<table>
<thead>
<tr>
<th>31 ... 27</th>
<th>26 ... 22</th>
<th>21 ... 17</th>
<th>16 ... 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00101</td>
<td>rdst</td>
<td>rs1</td>
<td>0 0000000000000000</td>
</tr>
</tbody>
</table>

**XOR** – Bit-wise logical exclusive-OR

**Assembly code notation**

a) XOR rdst, rs1, rs2
b) XOR rdst, rs1, #immed16

**Instruction encoding**

<table>
<thead>
<tr>
<th>31 ... 27</th>
<th>26 ... 22</th>
<th>21 ... 17</th>
<th>16</th>
<th>15 ... 11</th>
<th>10 ... 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00110</td>
<td>rdst</td>
<td>rs1</td>
<td>0</td>
<td>rs2</td>
<td>0 000000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31 ... 27</th>
<th>26 ... 22</th>
<th>21 ... 17</th>
<th>16</th>
<th>15 ... 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00110</td>
<td>rdst</td>
<td>rs1</td>
<td>1</td>
<td>immed16</td>
</tr>
</tbody>
</table>