1. (15 points)

(This problem is important well beyond the points we assign to it, because it continues our gentle ramp up of SimpleScalar)

For this problem you will be implementing a 3-bit branch predictor. A 3-bit predictor has eight states and it is a natural extension of the 2-bit predictor (H&P page 83).

Edit `bpred.h` and `bpred.c` to support the 3-bit prediction scheme. For each entry in the prediction buffer, you MUST initialize the state of the branch predictor to a “very weak not taken”.

a. Using sim-outorder, run the 3-bit predictor with sim-outorder's default configuration. Use the three benchmarks gcc, mcf, and crafty. Fast forward 100M instructions and run for the next 50M instructions for all the benchmarks. Similarly, run the built-in perfect, bimod and 2 lev prediction schemes. Compare and comment on your results of the four policies in terms of IPC and miss rate at the L1 and the L2 cache.

b. Include a print-out of the changes to `bpred.c` and `bpred.h` with your changes highlighted. Please document your changes.

2. (15 points)

Consider the following piece of code.

```
loop:
    ldf X(r1),f1
    addf f1,f3, f0    ; f0 = f1 + f3
    ldf X(r2),f1
    mulf f1, f0, f2
    addi r1,4,r1
    stf f2,X(r1)
    slt r1,r3,r4
    BNQZ r4,loop
```

Assume that the above code is being run on MIPS R10K like out-of-order processors. Construct and fill out the following tables for each cycle of the program's execution.
For how many cycles does this program run?

Assumptions:
1) The processor can dispatch, issue, complete and retire one instruction at a time.
2) Floating Point multiplication operations take 4 cycles to execute, while floating point add takes 2 cycles (this is only the number of execute cycles and does not include the “complete” cycle).
3) All load operations take 2 cycles and store operations take 2 cycles to execute (this is only the number of execute cycles and does not include the “complete” cycle).
3) Integer operations take 1 cycle to execute (this is only the number of execute cycles and does not include the “complete” cycle).
4) Functional units are fully pipelined.
5) Branch prediction is perfect and the above loop is repeated twice (but remember that branch also has to go through the execution pipeline to verify the correctness of prediction).
6) There are only 16 physical registers and they can be used to store floating point and integer values.
7) There are only 8 ROB slots available and 5 reservation stations.

The tables' status at the end of cycle 2 (note the column for Retire (R) also):

### ROB

<table>
<thead>
<tr>
<th>ht</th>
<th>#</th>
<th>Inst</th>
<th>T</th>
<th>Told</th>
<th>S</th>
<th>X</th>
<th>C</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>1</td>
<td>ldf X(r1),f1</td>
<td>PR#9</td>
<td>PR#2</td>
<td>c2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>2</td>
<td>addf f1,f3,f0</td>
<td>PR#10</td>
<td>PR#1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Reservation Stations

<table>
<thead>
<tr>
<th>#</th>
<th>FU</th>
<th>Busy</th>
<th>op</th>
<th>T</th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ALU</td>
<td>N</td>
<td></td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>L/S1</td>
<td>Y</td>
<td>ldf</td>
<td>PR#9</td>
<td>-</td>
<td>PR#5+</td>
</tr>
<tr>
<td>3</td>
<td>L/S2</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>FP1</td>
<td>Y</td>
<td>addf</td>
<td>PR#10</td>
<td>PR#9</td>
<td>PR#4+</td>
</tr>
<tr>
<td>5</td>
<td>FP2</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Map Table

| f0 | PR#10 |
f1  PR#9
f2  PR#3+
f3  PR#4+
r1  PR#5+
r2  PR#6+
r3  PR#7+
r4  PR#8+

Free List
PR#11,PR#12,PR#13,PR#14,PR#15,PR#16