EECS 470 Fall ‘07
Homework 2
Due Wed. Sept 26th at the start of lecture. No late homeworks.

Name: _______________________________ unique name: ________________

You are to turn in this sheet as a cover page for your assignment. The rest of the assignment should be stapled to this page. This is an individual assignment, all of the work should be your own. Assignments that are unstapled, lack a cover sheet, or are difficult to read will lose at least 50% of the possible points and we may not grade them at all. This assignment is worth a bit less than 2% of your grade in the class and is graded out of 30 points. Remember you may drop one homework assignment.

ILP scheduling. (based on code taken from H&P).
Consider the following code sequence (destination register specific last):

```
Loop: LD 0(R1),F2
I0:  ADDD F0, F2, F4  
I1:  DIVD F4, F0, F6
I2:  LD 8(R1), F8
I3:  MULTD F6, F8, F10
I4:  ADDD F4, F0, F8
I5:  SD 0(R1), F8  
I6:  SD 8(R1), F10
I7:  ADDI R1, #16, R1
I8:  SUB R4, R1, R20
I9:  BNZ R20, Loop
```

1. Determine how long the loop takes to complete on an in-order machine where an instruction may not issue (start execution) until the preceding instruction is writing back its result, whether they are dependant or not. (Ignore instruction fetch). Create a table showing when each instruction finishes decode, execute, and writeback. Report the total number of cycles per iteration. We have filled in the first few rows of the table below: [4]

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Decode (D)</th>
<th>Execute (X)</th>
<th>Writeback (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop: LD 0(R1),F2 (2)</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>I0: ADDD F0, F2, F4 (2)</td>
<td>4</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

2. Now, consider the execution of this code on a machine with scoreboard scheduling. Create a table showing when each instruction completes the dispatch, issue, execute, and writeback stages, like the table shown in lecture, for a single iteration. Each time there is a stall, indicate the reason for the stall (structural hazard on FU, WAW, WAR, or RAW dependency on a register). Assume that the instruction after a branch can issue when the branch reaches the writeback stage. Report the number of cycles it takes to complete an iteration (count the number of cycles from dispatch of Loop in the first iteration to dispatch of Loop in the second). Here is the state of the table after 5 cycles [15]

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Dispatch (D)</th>
<th>Issue (S)</th>
<th>Execute (X)</th>
<th>Writeback (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop: LD 0(R1),F2 (2)</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>I0: ADDD F0, F2, F4 (2)</td>
<td>2</td>
<td>5 (RAW F2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I1: DIVD F4,F0,F6 (10)</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2: LD 8(R1), F8 (2)</td>
<td>5 (FU hazard)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Now, draw a dependence graph, where the vertices correspond to instructions and the edges correspond to true control and data dependences among the instructions. (Because this graph shows the data flow of an instruction sequence, it is sometimes called a dataflow graph). Annotate each instruction with its latency [7]

4. Now, imagine an ideal out-of-order processor with infinite issue width, an unbounded number of functional units and an ideal register renaming implementation (i.e., the renaming hardware eliminates all false dependencies). Using your dependence graph, figure out how fast such an ideal machine can complete the instruction sequence. How many of each functional unit are necessary to achieve this execution time? [4]