Midterm Examination I
In-Class
Wednesday, February 16, 2005
Weight: 25%

1:15 minutes.

CLOSED BOOK, etc., but one cheat sheet allowed (two-sided 8.5x11 page).

The exam is two-sided and has EIGHT pages, including two blank pages at the end.

Plan your time carefully, since some problems are longer than others.

NAME: __________________________________________________________

ID# ______________________________________________________________

<table>
<thead>
<tr>
<th>Problem Number</th>
<th>Maximum Points</th>
<th>Actual Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1: Data Parallel Programming (12 points)

(a) Using the data parallel pseudo-code of Hillis & Steele, show code to compute the number of positive elements of a floating-point array A[] of N elements.

(b) What is the order of the code’s runtime on an ideal data parallel machine (e.g., CM-1/2)?

(c) What factors might make the actual runtime on practical machine different from your answer to part (b)?
Problem 2: Communication in Message-Passing (12 points)

Consider a square grid computation like the SPLASH-2 Benchmark Ocean.

\[
\text{for } i = 1 \text{ to } T \\
\quad \text{for } j = 2 \text{ to } W-1 \\
\quad \quad \text{for } k = 2 \text{ to } W-1 \\
\quad \quad \quad \text{Ocean}[j,k] = 0.2*(\text{Ocean}[j,k]+\text{Ocean}[j-1,k]+\text{Ocean}[j+1,k] \\
\quad \quad \quad \quad +\text{Ocean}[j,k-1]+\text{Ocean}[j,k+1])
\]

Consider parallelizing this code under message-passing on \( P \) processors, where (i) in each iteration you can read the “old” or “new” values of Ocean elements, (ii) you can assume all parameters are divisible (e.g., use \( W/P \) without worrying about integer round-off), and (iii) \( P \ll W^2 \). State any other assumptions you use.

(a) What work would you assign to each processor to minimize communication?

(b) As a function of \( P, T, \) and \( W \), how much communication is needed to complete the computation of part (a)?
Problem 3: Coherence (12 points)

(a) What is the potential benefit of the MESI “E” (Exclusive) state? Beyond what is need to implement an MSI snooping protocol, what additional bus or memory mechanisms are needed to implement a MESI snooping protocol?

(b) Provide an example memory reference sequence in which an update protocol outperforms an invalidation protocol. List same sequence twice and annotated with “bad” events (e.g., cache misses) for each protocol. We give you the first two memory references of the sequence.

<table>
<thead>
<tr>
<th>Update Protocol</th>
<th>Invalidation Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: write 100</td>
<td>P1: write 100</td>
</tr>
<tr>
<td>P2: read 100</td>
<td>P2: read 100</td>
</tr>
</tbody>
</table>
Problem 4: Sequential Consistency (12 points)

Consider the following example. Assume memory variables A and B are initially 0 and rij means processor Pi’s register j.

<table>
<thead>
<tr>
<th>Processor P1</th>
<th>Processor P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 3;</td>
<td>r21 = B;</td>
</tr>
<tr>
<td>B = 4;</td>
<td>r22 = A;</td>
</tr>
</tbody>
</table>

(a) What values are left in registers r11 and r22 at the end of each possible sequentially consistent execution? Why?

(b) Give values left in registers r21 and r22 for an execution that is not sequentially consistent. Explain why it is not sequentially consistent.

(c) Give an example of plausible hardware that could fail to be sequentially consistent for the example in your answer to part (b). Explain why it could fail.
Problem 5: Tread-Level Speculation (12 points)

Consider Stanford Hydra executing a sequential program with thread-level speculation. Assume that thread 1 executes the first iteration of a loop, while thread 2 executes the second iteration. For each of the following four cases, describe (i) what behavior is correct and (ii) how Hydra achieves correct behavior.

(a) Thread 2 executes \texttt{load r2, A} before thread 1 executes \texttt{store r1, A}.

(b) Thread 2 executes \texttt{store r2, A} before thread 1 executes \texttt{load r1, A}.

(c) Thread 2 executes \texttt{store r2, A} before thread 1 executes \texttt{store r1, A}.

(d) Thread 2 executes \texttt{load r2, A} before thread 1 executes \texttt{load r1, A}.
Scratch Sheet 1 of 2 (in case you need additional space for some of your answers)