Optimization sample problems

1. Given the 3-address code for a basic block:

<table>
<thead>
<tr>
<th>Num</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ld a, T1</td>
</tr>
<tr>
<td>2</td>
<td>Ld b, T2</td>
</tr>
<tr>
<td>3</td>
<td>Ld c, T3</td>
</tr>
<tr>
<td>4</td>
<td>Ld d, T4</td>
</tr>
<tr>
<td>5</td>
<td>+ T1, T2, T5</td>
</tr>
<tr>
<td>6</td>
<td>+ T5, T3, T6</td>
</tr>
<tr>
<td>7</td>
<td>+ T6, T4, T7</td>
</tr>
<tr>
<td>8</td>
<td>ST T7, a</td>
</tr>
</tbody>
</table>

a. Show the register lifetimes of each virtual register (T1, T2, …) in the basic block above. The lifetime of T1 is given. See the answer on the other solution sheet.

b. How many registers are needed to allocate this basic block with no spills?
5 – the maximum number of live values at any time, as can be seen from the solution to 1a.

c. Register allocate the basic block using 3 physical registers. When a register needs to be spilled, pick the one whose next use would be furthest away. If there is a tie, spill the lowest numbered register.

<table>
<thead>
<tr>
<th></th>
<th>r1</th>
<th>r2</th>
<th>r3</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ld a</td>
<td>r1</td>
<td>T1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ld b</td>
<td>r2</td>
<td>T1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ld c</td>
<td>r3</td>
<td>T1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>st</td>
<td>r3</td>
<td>T1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ld d</td>
<td>r3</td>
<td>T1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ld</td>
<td>r3</td>
<td>T1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>+ r1</td>
<td>r1</td>
<td>T1</td>
<td>T1 and T2 are not used again, and so r1 and r2 are freed. T5 is allocated into r1.</td>
</tr>
<tr>
<td>6</td>
<td>+ r1</td>
<td>r2</td>
<td>T6</td>
<td>The spilled value of T3 is reloaded into r2. It is not used again, so r3 is free after register allocating the statement</td>
</tr>
<tr>
<td>7</td>
<td>+ r1</td>
<td>r3</td>
<td>T7</td>
<td>T6 is not used after this statement, and so r1 is free to store the result (T7). T4 is also not used again, freeing r3.</td>
</tr>
<tr>
<td>8</td>
<td>st r1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. Create a precedence graph for the basic block above. You do not need to fill in delays. Label the nodes on the precedence graph with the statement number above.
3. Given the precedence graph:

Let each Ld operation take two cycles, and each + operation take a single cycle.
1. Label each node with its delay
2. Schedule the operations using the algorithm in the notes. Show the schedule by putting instructions in the cycle they are scheduled. **In the case of a tie, schedule the lowest numbered node first.**

**Note:** Even if instruction 5 did not exist, instruction 3 could not have been scheduled at cycle 3 because it needed to wait until cycle 4 the load of b to finish.
4. Give the loops:

\[
\begin{align*}
d\text{i} & = 1, n \\
\text{do } & a(i) = b(i) + c(i) \\
\text{end do} \\
\text{do } & d(i) = a(i) + c(i) \\
\text{end do}
\end{align*}
\]

(a) What transformation could help the performance of these two loop nests? 
\emph{Fusion}

(b) Why does it help the performance?
\emph{Improves the cache performance by allowing cached values of \textit{a} to be reused.}

5. Given the loop nest:

\[
\begin{align*}
d\text{i} & = 1, n \\
\text{do } & a(i) = b(i) + c(i+1) \\
\text{c(i)} & = a(i-1) + b(i) \\
\text{end do}
\end{align*}
\]

(a) Fill in the table below with the information about each pair of references listed. Only fill in the last four columns when a dependence exists.

<table>
<thead>
<tr>
<th>Pair of references</th>
<th>Dependent? (Yes or No)</th>
<th>Kind of dependence</th>
<th>Distance of the dependence</th>
<th>Direction of the dependence</th>
<th>Loop Carried?</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(i), a(i-1)</td>
<td>Yes</td>
<td>True</td>
<td>1</td>
<td>“1” or “&lt;”</td>
<td>yes</td>
</tr>
<tr>
<td>b(i), b(i)</td>
<td>No (both references are reads.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c(i+1), c(i)</td>
<td>Yes</td>
<td>Anti</td>
<td>1</td>
<td>“1” or “&lt;”</td>
<td>yes</td>
</tr>
</tbody>
</table>

(b) Fill in the iteration space diagram (below) for one of the dependences in the table below. See the answer on the other solution sheet.

\[
\begin{array}{cccccccc}
I & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
O & O & O & O & O & O & O & O
\end{array}
\]

(c) Can the loop be parallelized? Why or why not? \emph{No – it has loop carried dependences.}
6. Given the loop nest:

do $i_1 = 1, n$
  do $i_2 = 1, n$
    $a(i_1 + 2, i_2 - 1) = a(i_1, i_2)$
  end do
end do

(a) What kind of dependence exists on the “$a$” references? True or flow.

(b) What is the distance and direction of the dependence? Distance = (2, -1). Direction = (1, -1) or (<, >)

(c) Show the instances of the dependence on the iteration space diagram below. See the answer on the other solution sheet.

(d) Can the dependence be eliminated? Why or why not? No, it is a true dependence and is inherent to the computation.

\[
\begin{array}{ccccccc}
I_2 = 5 & O & O & O & O & O & O \\
i_2 = 4 & O & O & O & O & O & O \\
i_2 = 3 & O & O & O & O & O & O \\
i_2 = 2 & O & O & O & O & O & O \\
i_2 = 1 & O & O & O & O & O & O \\
I_1 = 1 & 2 & 3 & 4 & 5
\end{array}
\]