Full Name: ________________________________

MIT ID: __________________________________

Athena ID: __________________________________

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points</td>
<td>29</td>
<td>20</td>
<td>31</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Consider a simple machine with one ALU and one memory unit. The machine can execute an ALU operation and a memory operation at the same clock. A memory load operation takes two clocks (i.e. its result will be available in two clocks) and all other operations (including a memory store operation) complete in one clock.

Consider the following code fragment:

\begin{verbatim}
L1:       mov 0(%rsp), %r10
L2:       mov %r10, 4(%rsp)
L3:       mov 8(%rsp), %r11
L4:       add %r8, %r11
L5:       add %r10, %r11
L6:       mov %r11, 12(%rsp)
\end{verbatim}

where \texttt{mov src, dst} copies a value from \texttt{src} to \texttt{dst} and \texttt{add src, dst} means \texttt{dst += src}.

(a) (5 points) A dependence DAG represents the constraints on the possible schedules of the instructions in a basic block. The nodes in a dependence DAG represent machine instructions and its edges represent dependencies between the instructions. Each edge is labeled with latency \( \ell \) indicating that the destination node must be issued no earlier than \( \ell \) clocks after the source node is issued. Draw the dependence DAG for the above code fragment.

\begin{center}
\begin{tikzpicture}
  \node (L1) at (0,0) {L1};
  \node (L2) at (2,0) {L2};
  \node (L3) at (4,0) {L3};
  \node (L4) at (6,0) {L4};
  \node (L5) at (8,0) {L5};
  \node (L6) at (10,0) {L6};

  \draw[->] (L1) -- node[above] {2} (L2);
  \draw[->] (L3) -- node[above] {2} (L4);
  \draw[->] (L4) -- node[above] {1} (L5);
  \draw[->] (L5) -- node[above] {1} (L6);

  \draw[->] (L1) -- (L3);
  \draw[->] (L5) -- (L3);
\end{tikzpicture}
\end{center}
(b) (8 points) How many clocks does it take to execute the code fragment without any instruction scheduling? Fill the following table with the instruction labels (L1, L2, \ldots L6) that describes what instructions are executing in the memory unit and the ALU at each clock. Leave blank any clocks that are not needed.

<table>
<thead>
<tr>
<th></th>
<th>Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Unit</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>L1</td>
</tr>
<tr>
<td></td>
<td>L1</td>
</tr>
<tr>
<td></td>
<td>L2</td>
</tr>
<tr>
<td></td>
<td>L3</td>
</tr>
<tr>
<td></td>
<td>L3</td>
</tr>
<tr>
<td></td>
<td>L6</td>
</tr>
<tr>
<td>ALU</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>L4</td>
</tr>
<tr>
<td></td>
<td>L5</td>
</tr>
</tbody>
</table>

8 clocks

(c) (8 points) How are the instructions in the code fragment reordered when we apply the list-scheduling algorithm with the heuristics for selection presented in Lecture 13? Write your answer by listing the instruction labels (L1, L2, \ldots L6) in the new order.

L3
L1
L4
L5
(L2)
(L6)

(d) (8 points) How many clocks does it take to execute the reordered code fragment after applying the list-scheduling algorithm? Fill the following table with the instruction labels (L1, L2, \ldots L6) that describes what instructions are executing in the memory unit and the ALU at each clock. Leave blank any clocks that are not needed.

<table>
<thead>
<tr>
<th></th>
<th>Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Unit</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>L3</td>
</tr>
<tr>
<td></td>
<td>L3</td>
</tr>
<tr>
<td></td>
<td>L1</td>
</tr>
<tr>
<td></td>
<td>L1</td>
</tr>
<tr>
<td></td>
<td>L2</td>
</tr>
<tr>
<td></td>
<td>L6</td>
</tr>
<tr>
<td>ALU</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>L4</td>
</tr>
<tr>
<td></td>
<td>L5</td>
</tr>
</tbody>
</table>

6 clocks
2. (a) Consider the following control flow graph with the assumption that all variables are live on exit from the loop:

![Control Flow Graph]

i. (5 points) Which statements are loop-invariant? 1, 2

ii. (5 points) Which statements can we move out of the loop? none
(b) Consider the following control flow graph with the assumption that all variables are live on exit from the loop.

(1) $A = B + C$
(2) $E = 2$
(3) $E = 3$
(4) $D = A + 1$
(5) $F = E + 2$

i. (5 points) Which statements are loop-invariant? 1, 2, 3, 4

ii. (5 points) Which statements can we move out of the loop? 1, 4
3. Consider the following control flow graph:

(a) (8 points) Identify all the webs in the graph. Leave blank any rows or cells that are not needed.
(b) (8 points) Draw the interference graph for the webs you identified in (a). Label each node with its web number and variable.

\[
\begin{align*}
1, & \quad x_2 \\
2, & \quad x_4 \\
3, & \quad x_7 \\
4, & \quad x_8 \\
5, & \quad x_6 \\
6, & \quad x_9
\end{align*}
\]

(c) (5 points) How many colors are required to color the nodes of the graph such that no two adjacent nodes share the same color? 4

(d) (5 points) The target machine has less registers than required to allocate a register to every web. Which web will be chosen to be spilled first when we adopt the interference-degree heuristic?

(1, x_2), (2, x_4) or (4, x_8)

(e) (5 points) Which web will be chosen to be spilled first when we adopt the spill-cost heuristic?

(5, x_6)
4. (a) 

```plaintext
FOR I = 1 TO N
  FOR J = 1 TO N
  }
}
```

i. (3 points) What is the distance vector between the source and sink of a dependence in the iteration space of the loop nest?

\[
\begin{pmatrix}
1 \\
0
\end{pmatrix}
\]

ii. (3 points) What kinds of dependencies exist in the loop nest?

true

iii. (3 points) Which loop(s) can be fully parallelized?

the inner loop

(b) 

```plaintext
FOR I = 1 TO N
  FOR J = 1 TO N
    A = A + A;
  }
}
```

i. (4 points) What is the distance vector between the source and sink of a dependence in the iteration space of the loop nest?

\[
\begin{pmatrix}
0 \\
1
\end{pmatrix}, \quad \begin{pmatrix}
1 \\
-1 \cdot N
\end{pmatrix} = \begin{pmatrix}
* \\
*
\end{pmatrix}
\]

ii. (4 points) What kinds of dependencies exist in the loop nest?

true, anti, output

iii. (3 points) Which loop(s) can be fully parallelized?

none