1. Consider the characteristics of two different architectures

<table>
<thead>
<tr>
<th>Architecture</th>
<th>L1 cache</th>
<th>L2 cache</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>size</td>
<td>access time</td>
<td>size</td>
</tr>
<tr>
<td>1</td>
<td>32KB</td>
<td>2</td>
<td>2MB</td>
</tr>
<tr>
<td>2</td>
<td>64KB</td>
<td>3</td>
<td>1MB</td>
</tr>
</tbody>
</table>

where access time is measured in cycles and both architectures have the same cycle time. Suppose that an application has a data set of 1MB and the load/store access window ranges from 16KB to 1MB. For architecture 1, 75% of these accesses are in L1 cache, 23% are in L2 cache, and 2% are in memory. For architecture 2, 93% of these accesses are in L1 cache, 2% are in L2 cache, and 5% are in memory. How many cycles are used to access data on average on these architectures for this application? Which architecture is faster for this application?

2. Consider the following code with four assignment instructions:

```
x = a+1
y = 2*x
z = a+b
x = z+1
```

(a) Indicate the flow dependences (RAW) and anti dependences (WAR) between the statements.

(b) Suppose a pipelined CPU is used. Assume that each operation has a throughput of 1 cycle (i.e. a new operation can start every cycle) and operations have a latency of 3 cycles. When you take the dependences into account and assume that instructions are executed strictly in order, how many cycles does it take to execute this code?

(c) Suppose a pipelined CPU is used, instructions can be executed out of order (dynamic scheduling), and each operation has a throughput of 1 cycle and a latency of 3 cycles. Assuming that the dynamic scheduler is optimal, what is the minimum number of cycles required to execute this code?
3. What is a cache conflict miss? Is it possible to completely eliminate the possibility of cache conflict misses? If so, what kind of cache will be needed?

4. How can the TLB be a limiting factor for the performance of your application? In other words, when do you expect to see an increase in the TLB miss ratio per memory access?

5. Consider the following program fragment

\[
\begin{align*}
\text{DO } & \text{I = 1, N} \\
& B(I) = T(I) \times X(I) \\
\text{ENDDO} \\
\text{DO } & \text{I = 2, N} \\
& A(I) = B(I) - B(I-1) \\
\text{ENDDO}
\end{align*}
\]

(a) Apply loop peeling to the first loop and fuse the loops. Explain why the fused loops produce the same output of the code when you run the code.

(b) What is the potential benefit of fusing these loops?

(c) Suppose that array \texttt{B} is no longer used after these loops. Can you eliminate array \texttt{B} from the fused loops and replace the \texttt{B(I)} and \texttt{B(I-1)} accesses with something else without having to recompute \texttt{T(I)*X(I)}? That is, ensure that the number of multiplications is 1 per loop iteration.

(d) Suppose the body of the second loop is changed to \texttt{A(I) = B(I+1) - B(I-1)}, e.g. when using a different finite differencing stencil. Now there is a dependence that prevents loop fusion, since after fusion \texttt{B(I+1)} is not computed until one iteration later. Is it still possible to safely fuse the loops in some way by manually rewriting the code?