QUESTION #1

The attached cpp source code file (the one introduced in the classroom before) is a multi-threaded application, which spawns two (child) threads after it is started. Each of the two child threads executes a for loop, which performs printf system call in each iteration of the for loop. While the two child threads execute their for loop, the main (the parent) thread waits for the two threads to finish.

Q1: Which level(s) of tightly-coupled parallel computers (fine-grain, medium-grain, and coarse grain) may be applied to the program? Technically justify your response.

Q2: Which of the Flynn’s parallel model (SIMD, MISD, and MIMD) may be applied? Technically justify your response.

QUESTION #2

Most of the application programs consist of loop structures (e.g., for loops and while loops). The term “the size of a loop structure” refers to the number of instructions in a loop structure (i.e., the instruction count from the beginning to the end of a loop structure – see the figure below). Suppose that programmers do not apply any compile-time code optimization techniques, such as loop-unrolling, what is the impact of “the size of a loop structure” to the performance (execution speed) improvement for a tightly-coupled multi-processor system? Technically justify your response.

```
main:
    li   $t0, 100
    ...  

LOOP_BEGIN:
    ...  
    sub  $t0, $t0, 1
    bnez $t0, LOOP_BEGIN
    ...  

jr   $31
```

---

“the size of a loop structure” means the number of instructions in this area.
QUESTION #3

Suppose that there is a data structure which consists of \( N^2 \) elements. At the beginning, each element in the data structure is initialized by a constant “x” (Figure 1). If we need to update the contents of the data structure as shown in the figure below (Figure 2) using a SIMD computer, answer the following questions:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

Figure 1

\[
\begin{array}{c|c|c|c|c|}
    x & x^2 & \cdots & x^{(N-1)} & x^N \\
\hline
    x^{(N+1)} & x^{(N+2)} & \cdots & x^{(2N-1)} & x^{2N} \\
\hline
    \vdots & \vdots & \ddots & \vdots & \vdots \\
\hline
    \vdots & \vdots & \ddots & \vdots & \vdots \\
\hline
    \vdots & \vdots & \ddots & \vdots & \vdots \\
\hline
    \vdots & \vdots & \ddots & \vdots & \vdots \\
\hline
    \vdots & \vdots & \ddots & \vdots & \vdots \\
\hline
\end{array}
\]

Figure 2

Questions:
(a) What is the best achievable algorithm complexity?
(b) How many processors are necessary to achieve the best algorithm complexity?
(c) Describe how you got the solutions for the above two question.

Assumptions:
(a) One multiplication of two numbers (integers) actually requires a processor \( O(\log_2 x) \) physical operations, but assume that a multiplication can be performed as a 1-shot activity.
(b) One addition of two numbers (integers) actually requires a processor \( O(\log_2 x) \) physical operations, but for assume that an addition can be performed as a 1-shot activity.
(c) Assume that \( x^i \) (for \( i > 2 \)) can not be performed as a 1-shot activity (e.g., \( x^3 \) requires two multiplications).