1. A matrix multiplication program can spend 10% of its execution time in reading inputs from a disk, 10% of its execution time in parsing and creating arrays in the memory and 80% of time in the computation kernel that performs the real matrix multiplication.

   A. Assume that using a quad-core processor with a parallel algorithm can accelerate 50% of the computation kernel by 4x and the rest of 50% in the computation kernel by 2x. Also assume that using an SSD can improve the performance of reading data by 100x. What’s the effective speedup if the system adopts both the quad-core processor and the SSD?

   B. Assume that using a GPU can improve the computation kernel by 50x. Comparing with A., which one would provide better performance?
2. Another common performance figure is MFLOPS (millions of floating-point operations per second), defined as

\[ \text{MFLOPS} = \left( \frac{\text{No. FP operations}}{\text{execution time}} \right) \times 10^{-6}. \]

Now, assume you have two processors on two machines:

- **P1** – clock rate 4 GHz, average CPI of 0.9
- **P2** – clock rate 3 GHz, average CPI of 0.75.

If the compiled binary on an application requires \(5 \times 10^9\) instructions where 40% of them are floating point operations to execute on P1 and requires \(1 \times 10^9\) instructions where 50% of them are floating point operations to execute on P2, please answer the following:

A. Which processor will provide better performance for this application?

B. If we use this application as the benchmark to calculate the MFLOPS of both processors, please list the MFLOPS you calculated on these machines. Comparing with your answer in A, do you think MFLOPS is a good metric for the performance of these two machines? Why or why not?
3. The following table lists the time spent in each module of a processor:

<table>
<thead>
<tr>
<th>Module</th>
<th>Instruction Fetch</th>
<th>Instruction Decode</th>
<th>Execution</th>
<th>Data memory</th>
<th>Write back</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay</td>
<td>300 ps</td>
<td>250 ps</td>
<td>200 ps</td>
<td>350 ps</td>
<td>200 ps</td>
</tr>
</tbody>
</table>

Please answer the following questions:

A. If the processor uses a single-cycle design and supports MIPS ISA, what’s the cycle time of the processor?

B. Assume adding a pipeline register would incur 50 ps overhead, what’s the cycle time of a “5-stage pipeline processor” by using each of the above module as a pipeline stage?
C. Assume an application contains 20% load/store instructions, 20% branch instructions and 60% of ALU instructions. If we execute this program on the pipelined processor, the processor needs to stall until the execution module resolves the outcome of a branch instruction before the processor can fetch the next instruction. In addition, 20% of the instructions depend on a value generated by previous instruction – force these instruction to stall for an average of 2 cycles. Please evaluate the speedup of the pipelined processor when executing this program.
4. To figure out the portion of an application that contains the most potential for performance improvement, it’s important to figure out the execution time in each part of the application. There are quite a few tools that can help us figure this out. For example, on a Linux/UNIX machine, you may use gprof.

On a Mac, you may use instruments:
```
        -t
        
        " /Applications/Xcode.app/Contents/Applications/Instruments.app/Contents/Resources/templates/TimeProfiler.tracetemplate -D log_file_name
        application_executable [arguments]"
```


Now, download the code from http://cseweb.ucsd.edu/classes/su16/cse141-a/homework/arraySort.cpp and compile the code using a C++ without any optimization.

A. If you pass “0” as the argument of the compiled program, what’s the total execution time? How much time did the program spend in `sort` and `calculate_sum`, respectively.

B. If you pass “1” as the argument of the compiled program, what’s the total execution time? How much time did the program spend in `sort` and `calculate_sum`, respectively.