**Lab 1 – Parallel Computer Architectures**

**1a –** 2.8Ghz  
**1b –** 1.75Ghz  
**1c –** 1.25Ghz

**2a – out-of-order:** a [paradigm](https://en.wikipedia.org/wiki/Paradigm) used in most high-performance [central processing units](https://en.wikipedia.org/wiki/Central_processing_unit) to make use of [instruction cycles](https://en.wikipedia.org/wiki/Instruction_cycle) that would otherwise be wasted. In this paradigm, a processor executes [instructions](https://en.wikipedia.org/wiki/Instruction_(computing)) in an order governed by the availability of input data and execution units, rather than by their original order in a program In doing so, the processor can avoid being idle while waiting for the preceding instruction to complete and can, in the meantime, process the next instructions that are able to run immediately and independently. For example, I-1 and I-2 are the two instructions where I-1 comes first then I-2. In the out-of-order execution, a processor can execute I-2 instruction before I-1 instruction has been completed. This flexibility will improve the performance of the processor since it allows execution with less waiting time.

**2b – superscalar:** a [CPU](https://en.wikipedia.org/wiki/Central_processing_unit) that implements a form of [parallelism](https://en.wikipedia.org/wiki/Parallel_computer) called [instruction-level parallelism](https://en.wikipedia.org/wiki/Instruction-level_parallelism) within a single processor. In contrast to a [scalar processor](https://en.wikipedia.org/wiki/Scalar_processor) that can execute at most one single instruction per clock cycle, a superscalar processor can execute more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to different [execution units](https://en.wikipedia.org/wiki/Execution_unit) on the processor. Example: Intel i960CA

**2c- speculative execution**: an [optimization](https://en.wikipedia.org/wiki/Optimization_(computer_science)) technique where a [computer system](https://en.wikipedia.org/wiki/Computer_system) performs some task that may not be needed. Work is done before it is known whether it is actually needed, so as to prevent a delay that would have to be incurred by doing the work after it is known that it is needed. If it turns out the work was not needed after all, most changes made by the work are reverted and the results are ignored.

**2d - loop unrolling:** a [loop transformation](https://en.wikipedia.org/wiki/Loop_transformation) technique that attempts to optimize a program's execution speed at the expense of its [binary](https://en.wikipedia.org/wiki/Binary_file) size, which is an approach known as [space–time tradeoff](https://en.wikipedia.org/wiki/Space%E2%80%93time_tradeoff). The transformation can be undertaken manually by the programmer or by an [optimizing compiler](https://en.wikipedia.org/wiki/Optimizing_compiler). On modern processors, loop unrolling is often counterproductive, as the increased code size can cause more cache misses;