ELL 782 Major Exam

2020-21 Spring Semester

- Each question is for 10 marks
- Note that Turnitin will be used to check for plagiarism (copying). Turnitin finds matches between exam papers and between a paper and a source on the web. If there is a match of more than 5 words, then the student will get a 0 in the major exam.
- Thoroughly prove and justify all your answers. I am looking for creativity and original thinking.

1. Propose a method to do the following.
    a. Create a profiler (Google for what it is if you don't know) to find all the loads that are hard to predict.
    b. A method to annotate "hard to predict loads" by the compiler. This might need changes to the ISA. What kind of changes?
    c. A method in hardware to detect such annotated load instructions, and a method to efficiently process this information.
2. List some effective strategies that can make an OOO processor efficiently process the operations of a linked list and a binary tree. Do not paraphrase things written in the book. Suggest a few bespoke schemes, and fully justify why they work. Cite any relevant papers. You need to focus on all aspects.
    a. How to identify if a linked list or a tree is being processed? "Processed" refers to insert, traverse, and delete operations.
    b. What changes need to be done at the level of the compiler and the ISA?
    c. What changes need to be done to the pipeline?
    d. How and why will they work?
3. Should the LSQ store physical addresses or virtual addresses? What are the trade-offs? Explain in detail.
4. For an arbitrary model we map the parallel execution to an equivalent sequential execution. However, the latter may not be *legal*. Then what is the advantage of doing so? Explain your answer in great detail.
5. Prove that if we have a *data race* in a program, then it is possible to construct a sequentially consistent execution that also has a data race. Write the proof in your own words and your own commentary. Why is this result so important? I claim that it is one of the biggest results in computer architecture. What makes it so great?