Comparative Architectures Supervision Problem (Set 1)

Wei Song 31/01/2017

The following problems are mostly open-ended. Diagrams or charts are not needed in the answers. Therefore it is highly recommended to type the answers. Please keep your answers organised and tidy. Most of the questions are taken from the supervision notes from the lecturer. If you are not sure about the answers, you could look up from the Hennessy and Patterson book or directly from the Internet. If you could not find the answer or would like a discussion during the supervision session, please label it in your answer.

Lecture 1. Introduction

Q1: Why does power consumption now constrain processor design much more than it has historically?

Q2: Why might MIPS (millions of instructions per second) be a poor measure of performance?

Q3: (Hard) how might recent advances in die stacking help to improve microprocessor performance and reduce costs?

Lecture 2. Fundamentals of Computer Design

Q4: (Hard) discuss the pros and cons of architectures with a 64-bit word size versus those with a 32-bit size. What applications are likely to benefit most?

Q5: (Hard) If you were a processor architect targeting embedded applications where memory is a scarce resource, how might you design a RISC-like instruction set that will achieve efficient use of memory?

Lecture 3 & 4. Advanced Pipelining

Q6: What is the architecture requirement for an anti-dependence (WAR) to cause a data hazard?

Q7: What parameters determine the optimal pipeline length for a microprocessor?

Q8: How does the use of multiple parallel pipelines (eg. Integer + floating + load/store) make it more difficult for us to provide support for precise exceptions?

Q9: (Hard) discuss the pros and cons of different address formats, from zero-address to three-address format (“specifying an instruction’s operands” in Lecture 2). (hint, what is the effect on structure and data hazards?)

Q10: Why are 2-bit saturating counters often used to predict a branch's direction in preference to single bit schemes?

Q11: Why might a branch target buffer provide a poor prediction of procedure return addresses and what hardware solution may be employed to improve the accuracy of such predictions?

Q12: (Hard) why might a single conditional move instruction be supported rather than the conditional (or predicated) execution of every instruction in the instruction set? (hint, think about pipeline control)