2010 Paper 9 Question 10

Optimising Compilers

A CPU has the following two features:

1. Every arithmetic instruction (as well as the special "compare" instruction) sets condition codes based on the value of the result, and subsequent conditional instructions can test these.

2. If you execute an instruction that attempts to load from or store to a memory address that is not a multiple of the word-size, the CPU generates an interrupt.

You are concerned with decompilation from the target instructions of this machine back into a high-level programming language.

(*a*) (*i*) What issues are raised by the condition codes, and what effect would they have on a naïve decompilation? [5 marks]

(*ii*) What optimisation techniques could you use to improve matters, and what (if any) are the limitations of your approach? [5 marks]

(*b*) Direct decompilation would need to prefix every memory access with a test to see whether the address being used was properly aligned. This would lead to bulky code that was hard to read and had poor performance. It is expected that, in the program that is being decompiled, unaligned memory accesses are very rare. You are expected to decompile into portable code in the high-level language that may not rely on any treatment of unaligned addresses on the fresh computer you compile for. Again, what optimisation techniques could you apply? [10 marks]