Preface

I love virtual machines (VMs) and I have done for a long time. If that makes me "sad" or an "anorak", so be it. I love them because they are so much fun, as well as being so useful. They have an element of original sin (writing *assembly* programs and being in control of an *entire machine*), while still being able to claim that one is being a respectable member of the community (being structured, modular, high-level, object-oriented, and so on). They also allow one to design machines of one's own, unencumbered by the restrictions of a particular processor (at least, until one starts optimising it for some physical processor or other).

I have been building virtual machines, on and off, since 1980 or thereabouts. It has always been something of a hobby for me; it has also turned out to be a technique of great power and applicability. I hope to continue working on them, perhaps on some of the ideas outlined in the last chapter (I certainly want to do some more work with register-based VMs and concurrency).

I originally wanted to write the book from a purely semantic viewpoint. I wanted to start with a formal semantics of some language, then show how a virtual machine satisfied the semantics; finally, I would have liked to have shown how to derive an implementation. Unfortunately, there was insufficient time to do all of this (although some parts—the semantics of ALEX and a part proof of correctness—were done but omitted). There wasn't enough time to do all the necessary work and, in addition, Stärk *et al.* had published their book on Java [47] which does everything I had wanted to do (they do it with Java; I had wanted to define *ad hoc* languages).

I hope to have made it clear that I believe there to be a considerable amount of work left to be done with virtual machines. The entire last chapter is about this. As I have tried to make clear, some of the ideas included in that chapter are intended to make readers think, *even if* they consider the ideas stupid!

A word or two is in order concerning the instruction sets of the various virtual machines that appear from Chapter Four onwards. The instructions for the stack machines in Chapter Four seem relatively uncontroversial. The instructions in the chapter on register machines (Chapter Seven) might seem to be open to a little more questioning.

First, why not restrict the instruction set to those instructions required to implement ALEX? This is because I wanted to show (if such a demonstration were really required) that it is possible to define a larger instruction set so that more than one language can be supported.

Next, most of the jump and arithmetic instructions seem sensible enough but there are some strange cases, the jump branching to the address on the top of the stack is one case in point; all these stack indexing operations constitute another case. I decided to add these "exotic" instructions partly because, strange as they might appear to some, they are useful. Somewhere or other, I encountered a virtual machine that employed a jump instruction similar to the one just mentioned (I also tried one out in one of the Harrison Machine's implementations—it was quite useful), so I included it. Similarly, a lot of time is spent in accessing variables on the stack, so I added instructions that would make such accesses quite easy to compile; I was also aware that things like process control blocks and closures might be on stacks. I decided to add these instructions to build up a good repertoire, a repertoire that is *not* restricted to the instructions required to implement ALEX or one of the extensions described in Chapter Five.

I do admit, though, that the mnemonics for many of the operations could have been chosen with more care. (I was actually thinking that an assembler could macro these names out.) One reason for this is that I defined the register machine in about a day (the first ALEX machine was designed in about forty-five minutes!). Another (clearly) is that I am not terribly good at creating mnemonics. I thought I'd better point these matters out before someone else does.

I have made every effort to ensure that this text is free of errors. Undoubtedly, they still lurk waiting to be revealed in their full horror and to show that my proof-reading is not perfect. Should errors be found, I apologise for them in advance.

Acknowledgements

Beverley Ford first thought of this book when looking through some notes I had made on abstract machines. I would like to thank her and her staff at Springer, especially Catherine Drury, for making the process of writing this book as smooth as possible.

My brother Adam should be thanked for creating the line drawings that appear as some of the figures (I actually managed to do the rest myself). I would also like to thank all those other people who helped in various ways while I was writing this book (they know who they are).

Iain Craig
Market Square
Atherstone
14 June, 2005