Executing Prolog at Ten Times DEC-10 Speed

WFC (28 November 83)

Because of the large corpus of existing applications software, a decent Prolog system *must* include facilities to enable compatibility with the existing DEC-10 implementation. The main problem is maintaining the 'recorded'-style database without degrading the rest of system performance. Furthermore, a sysytem should provide *full* garbage collection, which means doing better than the current DEC-10 system. Providing a GC method that does not severely impact performance is the subject of another note.

Having such facilities causes overheads that serve to reduce performance by unknown factors (estimated from between 10% to 40%). I shall ignore the effects of such facilities on the performance estimates derived below, because the facilities are not used when the performance of the DEC-10 Prolog is measured. These facilities are not discussed further: suffice it to say that Prologs not offering these facilities should not be seriously considered.

The ZIP Machine

The ZIP Machine is the virtual machine specification for a Prolog machine. Its simulator has been packaged as a Prolog system, and is called Prolog-X. The simulator emulates byte codes taken from an instruction stream. A version of Prolog-X written in PASCAL, running on a VAX-780, emulates about 5000 bytecodes per second. The same system running on an ICL 2980 emulates about 10000 bytecodes per second. Translating Prolog-X into C would provide a system at least 5 times faster. However, this is a relatively minor speedup compared with what is required -- ten times DEC-10 Prolog.

How fast will ZIP go?

I shall estimate the speed required of a ZIP machine in the following way. I refuse to use the informal term LIPS (Logic Inferences per Second), because it is not defined: many people have different conceptions about what it really means. As no other proposal has been made, I shall put forward one based on the only documented evidence of DEC-10 Prolog performance: the benchmarks listed in (Warren, 1977). We choose the naive reverse benchmark: although it does not require the full spectrum of Prolog capabilities, it is simple to perform and to measure, and it has a high concentration of procedure calls.

I shall define 1 drev as the speed of processor required to compute the naive reverse of a list 30 elements in length in 53.7 milliseconds.

The unit *drev* stands for "DEC-10 reverse": the DEC-10 Prolog system computes a naive reverse of a 30 element list in 53.7 milliseconds, so we rate a DEC-10 as having a performance of 1 drev. Thus, a processor that can compute at a rate of 10 drev will reverse a 30-element list in 5.37 milliseconds or 5370 microseconds.

How fast must the components of a ZIP machine go in order to go at a rate of 10 drev? The ZIP compiler emits 36 instructions for the naive reverse program. When a 30-element list is reversed, 6648 instructions are executed. Thus, to perform at a rate of 10 drev, a ZIP instruction must be executed every 808 nanoseconds on average.

Of these 6648 instructions, 525 are procedure calls, the most expensive instruction to execute. The remaining instruction are relatively cheap, and consist of fewer than 10 register transfers each. The single instruction executed most often (2775 times) consists of only 3 transfers (2 of them to main memory). 1395 of the instruction cause only one register transfer -- easily done in one minor ALU cycle. The procedure calls are the real problem -- new activation records need to be set up (8 memory transfers), the procedure must be located (about 10 memory transfers for naive reverse), and other new contexts need to be established (about 8 register transfers and 5 memory transfers).

There is a difference between transfers to internal CPU registers, which can be done on one ALU minor cycle, and transfers to main memory. Main memory transfers depend on the comparatively slow main memory which has a wide and unpredictable variance in cycle time. Let us assume that an average cycle time of 500 ns is possible for main memory transfers.

Pure worst-case guesswork based on the above figures leads me to suggest that with relatively modest hardware resembling a suitably microcoded HLH Orion fitted with a faster main memory, the 30-element reverse could be executed in about 21,175 microseconds, which represents a rate of only under 2 drev. This figure has been derived in the absence of more detailed ZIP simulator results (my instrumented simulator is on a VAX in Oxford, and not on SERCnet!), and should underestimate performance, but should be good to within 30%. I have crudely assumed that 1395 of the instructions are executed in 500 ns each, 4728 in 1000 ns each and 525 in 30000 ns each. The procedure call is the killer. If the time could be reduced to 10000 ns (quite feasible), then this would increase the speed to 5 drev.

Increasing the speed to 10 drev must be accomplished by providing faster hardware. The memory cycle time is the problem. Even if the CPU could be quadrupled in speed (by using ECL, say), I would predict roughly the same performance. To achieve 10 drev, we must either redesign the way that Prolog systems in general use memory, or we must obtain much faster memory technology. The problem will be exacerbated because improvements I wish to make in the ZIP design will result in a system which requires even more memory accesses for a given computation.