PATTERSON81

Patterson D.A.; An Experiment in High Level Language Microprogramming and Verification; CACM 24, 10 (October 1981) pp. 699-709.

This paper discusses the language STRUM, an experiment in its use for writing microprograms, and verification of the programs.

The author states "Microprogramming has the classic reliability and maintainability problems of software, and has inherited size and speed efficiency as practically the only measure of success from hardware." He also mentions that various researchers have indicated that for reasonable efficiency it is necessary to use microassemblers.

STRUM was developed to demonstrate that the situation could be improved, to make microprogramming easier, and clearer. Included in the system is an optimizing compiler, and a verifier. The result is that high level language structured programming and verification can be applied to microprogramming.

To test STRUM, it was used to write an emulation of the HP-2115 computer on the Burroughs D-machine, a horizontally microprogrammed computer. This emulation was then compared with another, which was written in the usual manner.

The STRUM language itself is derived from ALGOL, but was designed to give efficient object code, and hence was not designed to be machine independent. The requirement for verification forced the language to have a clean, axiomatic design. The programs are structured in modules (called macros) and procedures. A good variety of control structures are included. The goto statement was not included, and was not missed. The types correspond to the various objects available, such as registers, flags, and words of memory.

Despite the design criterion, the language seemed to turn out fairly machine independent. The only strongly machine dependent section of the language, were the declarations, which were tied to the types of objects in the machine, and the numeric operations, which are again dependent on the machine architecture.

Originally the project did not plan to include any optimization, but instead to have the compiler output the microassembler statements, which could then be optimized by hand. This decision was a result of the prevailing view in the literature which indicated that the optimizations were not of great practical value.

Near the end of the project, it was decided to add some simple optimizations. These were simple peep-hole optimizations to improve the if statement and to combine some statements. In addition there were some cases where chained branches can be combined.

The two implementations of the the HP-2115 emulator were similar in size. The hand written version used 946 microinstructions. The unoptimized version of the STRUM emulator used 1124 instructions, and the optimized version used 940. Surprisingly the STRUM versions both ran faster. The author suggested that this may have been due to the difference in view between the programming in the small of the hand microprogrammer compared to the programming in the large which is used by the high level language programmer. The optimizations resulted in improvements in both time and space usage. The code produced by the optimizer was very efficient, but also hard to understand. This resulted in some proposed additional hand optimizations having to be changed.

The author concludes that with the current trends in microprogrammed machines that the high level language system will be very useful. Some examples are trends to using RAM for control store, the tendency of the system to become more complex (pdp-11 256 microinstructions, vax-11 9000 microinstructions), and the increasing frequency of application programmers writing the microinstructions.

The last section of the paper discusses the formal verification of the HP-2115 emulator written with STRUM. Although a concise specification language (ISP) was used the verification conditions produced were very large. A number of modifications were made to the system, before it could digest the large conditions. The formal verification system produced a large quantity of output, but the author claims that it was in fact not a large problem to finish. The end result was that ten errors were found in the program, eleven in the specification, and eleven in the assertions.

The final conclusion from the verification experiment, was that a number of widely held generalizations about verification were not supported. He states that it is a useful tool if properly used. Finally he ends by mentioning others working on similar topics.