GANDF: Status III

Richard L. Ford
June 28, 1993
Open Software Foundation
Research Institute

Recent Progress on GANDF is reported: performance and correctness have been improved, and the DEC Alpha port has been mostly completed.

1. Introduction

GANDF is an experimental ANDF[1] translator being implemented at the Open Software Foundation (OSF) Research Institute (RI), based on the back-end technology of the Gnu C Compiler(GCC), produced by the Free Software Foundation, along with some support routines from the DRA technology.

This third[2]in a series of papers describing the GANDF experiment is just a brief update on progress relative to the last report. There are three main areas of progress--bug fixing, performance improvements, and completion of the DEC Alpha port.

1. ANDF is an architecture- and language- neutral distribution format being developed by OSF and other collaborators around the world. It is based on the TDF technology provided by the Defence Research Agency (DRA) of the UK Ministry of Defense.
2. The others appeared in OSF RI ANDF Collected Papers, Volumes 1 and 2.

2. Bug Fixing

As of our last report GANDF targeted to a number of architectures was passing the Plum-Hall test suites and the SPEC-Int and Dhrystone benchmarks. However, when applied to a more complex application like the Informix Wingz spreadsheet, it was failing.

As of our last report GANDF was not able to generate any debug output. When we went to debug the incorrect code for complex applications like Wingz it quickly became clear that this was a serious drawback. We have therefore added to GANDF the capability to produce debugging information of the same sort as GCC, so that GANDF output can be debugged whenever there is gdb or another debugger that will accept GCC debugger output. Our current debugging implementation allows:

- Procedure and Statement Breakpoints
- Single Stepping
- Showing of source lines
- Showing values of variables.

It is only incomplete in that variables are dumped in the data type that the variable has in the ANDF, and not in the original source language. It should not be too hard to correct this deficiency.

With the aid of this debugging capability, we were able to find the bad code in even complex applications. As a result, Wingz compiled by GANDF now executes successfully.

3. Performance Improvements

In our last report we reported that code compiled using GANDF was running about 20% slower than code compiled by GCC. We speculated that much of this was due to two factors:

- We were not yet putting out GCC loop marks, so loop optimization was not being done.
- We were treating all routines as if they had a variable number of arguments, thus incurring extra overhead in saving registers on routine entry.

GANDF has now been fixed to handle both of the problems. The effect of these vary depending on the benchmark, but the net effect is that now the GANDF produced code is usually within 5% of the GCC code, and in several cases in the 0-2% range. Most of the performance improvement came from the loop marks, but for some tests avoiding the variable number of arguments overhead improved performance by as much as 15%.

4. DEC Alpha Port

The DEC Alpha port is now essentially complete. This port involved additional complications that were not present in the others, since it has a 64 bit word-size.

One problem was that the DRA code used to read in the ANDF mapped the integer varieties into either 8, 16, or 32 bit integers. I could have extended this by just adding an additional 64 bit integer type, but this would have lead to some rather *ad hoc* code in places. Instead, I made use of GCC tables of integer data types available on the target architecture, in order to find a more general solution to the data type mapping problem.

Another problem had to do with compile-time evaluation of constant expressions. The problem here was that parts of the existing constant folding routines were assuming a 32 bit word-size. Also, I wanted to avoid having to have different versions of this code for 32 and 64 bit targets. In addition, GCC is designed to be able to be a cross-compiler, and I wanted GANDF to be able to run on a 32 bit machine while producing code for a 64 bit machine. I solved this problem by making use of the GNU Multiple Precision Package to do the constant folding. This allows me to do infinite precision arithmetic. Then, when necessary, the results are checked or truncated to make them fit the required constraints. I should mention that I did not actually built a 32 bit to 64 bit cross-compiler, and I’m not sure the Alpha GCC port can handle that, but at least the constant folding part of GANDF does not prevent such a cross-compiler from being built. One of the scenarios for use of ANDF is that a network installation server will produce executables for a variety of targets. This would imply that such an installation server would need to have a number of ANDF cross-installers.

5. Conclusions

We have made significant progress toward one of GANDF's goals, namely to show that ANDF installers can be effectively produced by interfacing ANDF with existing compiler back ends.

- We have shown that code produced by GANDF can be within 5% in performance of the back end on which it is based.
- We have shown that GANDF is reaching a robustness that allows it to be used for complex applications.
- We have shown that GANDF can be used to install ANDF on 64 bit targets.

GANDF has influenced the evolution of ANDF. We raised the issue of the impossibility of implementing procedures with multiple return types while maintaining ABI compatibility, which resulted in a change to the ANDF specification to avoid this problem.

There still is more work ahead for the GANDF installer project:

- We presently only implement a "C" subset of ANDF. More work is needed to implement full ANDF.
- GANDF will be ported to allow targets to Windows NT, and perhaps to DOS (using the djgcc port of GCC).
- Some parts need to be rewritten so that all of it can be freely distributed.
- Work is planned to adapt GANDF for massively parallel machines.

Even though there is more work to be done, we believe that these results are already sufficiently positive to encourage system vendors to adopt this approach toward producing ANDF installers.