DATE: April 8, 1982

TO: R D & E People

FROM: Louis Gross

SUBJECT: Goals of the Retargetable Back End Project

REFERENCE: None

KEYWORDS: LANGUAGES

ABSTRACT

This document contains a short statement of the goals of a proposed new project to produce a retargetable back end for Prime translators. It is being distributed to solicit comments from Engineering people. The goals have been agreed upon by the project's members: Debby Minard, David Spector, Scott Turner, and Louis Gross. After considering the comments received, this PE-TI will be replaced with a more detailed Requirements document.

Comments should be sent to the author by X.MAIL to LOU, or mail stop 10B-17, or telephoned to X4052.

What Prime Needs in a Back End

The main requirements for a compiler back end are:

- High quality object code -- it is important that a program run on our computers faster than it does on similarly priced computers of our competitors. This depends both on the speed of our hardware and on the quality of code produced by our compilers.

- Fast compilations -- a user debugging a program cares most about the speed with which the next compilation happens.

- Maintainability -- for most of its lifetime, a compiler is not being built, but being enhanced and having bugs fixed. There should be hardly any bugs, and the cost of enhancements and bug fixes should be minimized. It would be worth a lot of development effort to minimize maintenance effort.

- Easy Retargetability -- Prime programs are going to have to run on lots of different machine architectures. The list will certainly include V-mode, I-mode, the NSP instruction set, and various micros (e.g., M68000 now, but micro-architectures are evolving rapidly). We don't know what they all are today; it is clear that we will have to produce new code generators, and that it will be worth a lot of effort now to develop the tools to minimize the effort in retargeting later.

All of the above requirements are important: to the extent that we do not meet any of them, our costs will increase or our competitive position will be worse.

Why Our Current Compilers Won't Do

The TSI back end in current use at Prime fails to meet the above requirements:

- The code it produces is not of very high quality -- the VAX Fortran compiler seems to produce much more impressive object code, especially for loops.

- It runs slowly. This is partly because the TSI back end (also front end) was written for maximum rehostability. In order to run on machines with very small address spaces, it does software storage management instead of using our large virtual address space. While it has been possible to improve its compile speed some by twiddling with it, we could do much better with a basic design that takes advantage of the strengths of Prime architecture.

- It is very difficult to maintain -- there are a lot of bugs being reported, and many of our programmers (who could be doing development work) are tied up in fixing them.

- It is not easily retargetable -- producing a not very good code generator for I-mode took about a year. The experience among our competitors with the TSI back end has not been favorable: the most successful compilers that used the TSI front end (e.g., Digital's PL/1 compiler for the VAX) have used back ends created in-house.

The Goal of the Retargetable Back End Project

The goal of the Retargetable Back End Project is to produce a compiler back end that satisfies the needs described above. The new compiler back end will:

1 be easily retargetable -- perhaps on the order of four months by one experienced programmer to produce a very good code generator for a new architecture.

2 be easily maintainable: bugs are few and easy to fix, improvements (as compiler technology improves) and extensions (as new languages are added) are easy to make.

3 be usable on output from the current TSI front end, probably through an interface program that translates from TSI intermediate language. Note that the intermediate language actually used by the Retargetable Back End will be one chosen to facilitate the kinds of manipulations that the back end will be doing, but we will have to compile output of the current TSI front end until we have new front ends. While the interface program is in use, compilers using the new back end will run a little slower and produce code of not quite the same quality that we can eventually get.

4 be able to produce code for the intermediate language generated for languages currently handled by the TSI front end: PL/1, PL/1G, SPL, Fortran 77, Pascal, Cobol, RPG.

5 be easily extendable to produce code for the intermediate language generated by a front end for languages we are likely to have in the future: Ada, Modula-2 or Mesa. It will not be easily extendable to languages like APL, LISP, Smalltalk, or SNOBOL.

6 initially (in about 2.5 years with four to six programmers) generate code for I-mode (perhaps, the new NSP-mode -- the hardware schedule should determine which).

7 be very fast when run for compile speed, but still produce decent code.

8 produce great code when run for code quality, but not be so slow as to make people unhappy,