Does anybody out there want to write HALF of a compiler?

Andrew S. Tanenbaum
Ed Keizer
Hans van Staveren

Vakgroep Informatica
Vrije Universiteit
Postbus 7161
1007 MC Amsterdam, The Netherlands

University researchers love nothing better than to invent a new programming language. The trouble is, unless the language is only intended for mental exercise, the next step is writing a compiler for the new language, which is a lot of work. Typically one makes a prototype compiler that runs slowly and produces awful code for one particular CPU that happens to be available. Once the project succeeds, there is often a need for a better quality compiler or compilers for other machines. This step is even more work and even less fun. Over the past few years we have developed a set of tools that greatly eases the job of producing good quality portable compilers quickly. The purpose of this paper is to discuss the system, tell what its current status is, and encourage other university researchers to take advantage of it.

The system, called the Amsterdam Compiler Kit (ACK) has been described in some detail in [1-4]. We will just give a brief overview here. The tool kit has 8 components:

- The preprocessor (does macro expansion, conditional compilation, etc.)
- The front ends (translate from source code to our intermediate code, EM)
- The peephole optimizer (does local optimization on the EM code)
- The global optimizer (does data flow analysis and global optimization on EM)
- The back end (translates from EM to target assembly code)
- The target machine optimizer (does machine-dependent peephole optimization)
- The assembler-linker (assembles target program to binary with libraries)
- The utility package (test programs, libraries, documentation, etc.)

The first seven parts are cascaded in the manner of UNIX* filters, with source programs coming in the front and binary programs coming out the back. The nice thing about the tool kit is that it uses the same intermediate code for all languages and machines, which means that once one has developed the driving tables for the back end and assembler for some set of machines, e.g., PDP-11, VAX, 8086/8088, and 68000, by writing one new front end, the new language automatically has good quality compilers for all four machines. Also, the peephole and global optimizers are both language- and machine-independent, which means that they can be used with any language and machine combination.

Basically, EM is an UNCOL, which is an old idea. What we have done is work out all the details to produce a highly efficient and useable system. The state of the system as of April 1984 is as follows:

*UNIX is a Trademark of Bell Laboratories.

PDP-11/44 with Version 7 UNIX.
VAX-11/750 with Berkeley 4.1 UNIX
68000 systems from Altos, Bleasdale and Philips with UNIX System III

It probably can be easily booted to other systems. We intend to try to squeeze it onto an IBM PC-XT in the future.

A UNIX tar tape containing all the programs and tables that are marked "finished" above, all the documentation and instructions for installing everything (a total of 6 megabytes!) is now being licensed to universities holding a UNIX source license from Western Electric. We hope that researchers involved in programming language design and compiler construction will find this useful for their work. In particular, we hope that other people will make new front ends and tables for the back end and assembler, thus increasing the mass and utility of the whole kit.

Our experience is that making a new front end is the hardest part, typically taking about 6-12 man-months. Any standard compiler writing tools, such as Yacc can be used. The front end writer's job is considerably eased by the design of the EM code, which is a simple stack machine, with no registers, condition codes etc. (Of course the back end has to worry about these things, but back end tables for the most popular machines are already finished or in progress.) A back end table for a new machine is typically about 2 man-month's work. A table for the universal assembler should take a good programmer less than a week. (The record is the 6502 assembler, which was done in less than a day.) The quality of the object code produced is roughly comparable to that of pcc on the PDP-11 and cc on the VAX (which is actually pcc in sheep's clothing).

For information about how to obtain the tool kit, contact us. (Inquiries from companies are also welcome.)

Acknowledgments

We would like to thank Henri Bal, Ceriel Jacobs, and Johan Stevenson for their contributions to the project.

References

[1] Tanenbaum, A.S., Van Staveren, H., Keizer, E.G., and Stevenson, J.W.: "A Practical Tool Kit for Making Portable Compilers," CACM, vol. 26, pp. 654-660, Sept. 1983.

[2] Tanenbaum, A.S., van Staveren, H., Keizer, E.G., and Stevenson, J.W.: "Description of a Machine Architecture for Use with Block Structured Languages," Report IR-81, Wiskundig Seminarium, Vrije Universiteit, 80 pp., Aug. 1983.

[3] Tanenbaum, A.S., van Staveren, H. and Stevenson, J.W.: "Using Peephole Optimization on Intermediate Code," ACM Trans. Prog. Lang. and Syst., vol. 4, pp. 21-36, Jan. 1982.

[4] Tanenbaum, A.S.: "Implications of Structured Programming for Machine Architecture," CACM, vol. 21, pp. 237-246, March 1978.

Preprocessor: finished (standard UNIX C preprocessor)

Front ends:
- Pascal: finished (2-byte integers)
- C: finished (2- or 4-byte integers)
- Basic: in final test phase
- Plain: in progress (in collaboration with Prof. A. I. Wasserman, UCSF)
- Algol 68: in progress (being done by Charles Lindsey, Manchester)
- DAS: (Delft Ada Subset- being done by Jan van Katwijk at TH Delft)
- Various other languages are currently under discussion

Peephole optimizer: finished
Global optimizer: almost finished

Back end: Program is finished (new version with more checking in progress)
- DEC PDP-11 table: finished
- DEC VAX table: finished
- Intel 8080 table: in progress
- Intel 8086/8088 table: finished
- Mostek 6502 table: in final test phase
- Motorola 6809 table: in progress
- Motorola 68000 table: finished
- National Semiconductor 16032 table: to be started shortly
- Zilog Z80 table: in progress
- Zilog Z8000 table: in final test phase

Target optimizer: Scheduled to be started in May 1984

Universal Assembler/Linker: Program is finished
- Intel 8080 table: finished
- Intel 8086/8088 table: finished
- Mostek 6502 table: finished
- Motorola 6800 table: finished
- Motorola 6809 table: finished
- Motorola 68000 table: finished
- National Semiconductor 16032 table: in final test phase
- Signetics 2650: finished
- Zilog Z80 table: finished
- Zilog Z8000 table: finished

EM Interpreters:
- DEC PDP-11: finished
- Motorola 68000: in progress
- Zilog Z80: finished

Miscellaneous:
- EM assembler (translates EM to binary for interpretation): finished
- EM, Pascal, and C test programs: finished
- Various libraries and utility programs are also provided

The tool kit is highly modular, for example, the assemblers can be used separately as general cross-assemblers without any front end, etc. Except for the Pascal front end, which is in Pascal, the tool kit is written almost entirely in Yacc and C. It is known to run on the follow systems: