Keith D. Cooper

Department of Computer Science
Rice University
Houston, Texas, USA

Dr. Cooper received his Ph.D. in 1983 from Rice University's Department of Mathematical Sciences. He worked with Ken Kennedy at Rice from 1983 to 1990 as a research associate; in 1990, he was appointed to the position of Associate Professor in Computer Science at Rice and was awarded tenure in 1992.

Teaching

In his role as a teacher, Dr. Cooper's primary responsibility has been to teach the course sequence in compilation and optimization (COMP 412, COMP 512, and COMP 612). In the 1992–1993 academic year, he taught the junior-level software engineering course, COMP 310, as a voluntary overload. In the spring of 2000, he will teach COMP 210—the first course in the Computer Science curriculum.

COMP 412 Dr. Cooper's primary undergraduate teaching assignment has been the compiler construction course. He has taught it each year since the 1989-1990 academic year. The course is an elective, senior-level course that typically draws thirty-five to forty students. Over the ten years that he has taught the course, he has changed the philosophy, the emphasis, and the specific subjects covered. (The original course largely followed Aho, Sethi, and Ullman's text, with a series of laboratory exercises where students built a scanner & parser, a context-sensitive analyzer & intermediate code generator, and a code generator for a simple RISC machine.)

The material taught in the course today is chosen to provide the student with a broad perspective on the issues that arise in designing and implementing a compiler for a modern computer. Less time is devoted to front-end issues, such as scanning and parsing, than in previous versions of the course. That time is spent, instead, on instruction selection, register allocation, and instruction scheduling. The course sketches the implementation of object-oriented languages (a subject missing from most textbooks) and introduces students to the rudimentary ideas underlying code optimization.

The goal of the laboratory exercises is to elucidate complex principles, rather than to have the students create mediocre solutions to large problems. In the first lab, students build a local register allocator. For many students, it is the first time that they have been asked to approximate the solution to a tough problem, rather than to prove it NP-Complete. The second lab is still a scanner & parser. In coming years, it may be replaced with a lab where students add several new linguistic constructs to an already large and complex parser—for example, one of the constructs should introduce significant ambiguity into the source language. The final lab has the students build an instruction scheduler. This reuses the simple assembly code from the first lab, but introduces the complications of multiple function units and different operational latencies.

Dr. Cooper's lecture notes have been available online since 1991; they are currently available at `http://www.cs.rice.edu/~keith/412/` . They have been used, in part or in whole, in undergraduate courses at many colleges and universities, including the University of Utah, Michigan Technological University, the University of Houston, the University of Massachusetts, The University of Maryland, and Rutgers University. Judging by comments, corrections, and questions received from colleagues, they have been used for course preparation in many more colleges and universities. One European university replicated the entire Web site to provide students with a resource other than the textbook. Requests for electronic copies (on floppy disk) and paper copies of the materials have come from

several foreign universities. Dr. Cooper is currently writing a textbook with his colleague, Dr. Linda Torczon. The book is tentatively titled "Engineering a Compiler." The draft version will be used at Rice in the Fall of 1999; other universities have expressed interest in testing the text in the Spring semester of 2000.

COMP 512: Since joining the faculty, Dr. Cooper has regularly taught COMP 512, an elective graduate course focused on topics in compiler-based analysis and optimization. The course routinely draws thirteen to seventeen students—a mix of undergraduate and graduate students. The course has a standard lecture format; the students read an assigned research paper as preparation for each lecture. To round out the students' experience, they have a laboratory exercise in which they implement a recently-published optimization inside the research compiler infrastructure built and used in Dr. Cooper's research group. The students design, implement, and test a new pass for the compiler. Finally, they write a report (five to ten pages) detailing their experience. The content of the course changes from year to year, incorporating recently published material. The most recent lectures are available at `http://www.cs.rice.edu/~keith/512/` .

COMP 310: During the 1992-93 academic year, Dr. Cooper taught COMP 310, a laboratory course in team programming and the design of software systems. The course was focused on a set of three-week laboratory exercises that were assigned to teams of three students. Each student leads at least one team during the semester. Students never work with the same team twice. When Dr. Cooper taught the course, the lectures focused equally on the design of software systems and non-technical issues such as giving and taking constructive criticism, managing team meetings, and assessing the work of others.

Other Activities Dr. Cooper has supervised a number of students in independent study courses. These have ranged from reading courses on specific advanced topics through implementation projects in compiler construction. These students worked directly with Dr. Cooper, his staff, and his graduate students on problems of current research interest. In the spring of 1999, for example, he had one student implementing analysis passes for a joint Stanford-Harvard compiler project and a second student doing a reading course in advanced code optimization techniques, using Robert Morgan's new book "Building an Optimizing Compiler."

Dr. Cooper has offered a graduate reading seminar, COMP 612, several times. Initially, the seminar covered a broad selection of recent work in the general area of analysis and optimization. In recent years, the seminar has focussed on a single problem. For example, in the 1997-1998 academic year, COMP 612 covered transformations aimed at reducing the size of compiled code.

Finally, in the spring of 2000, Dr. Cooper will teach the introductory Computer Science course. It presents a disciplined, data-driven approach to writing programs in Scheme. Developed by Felleisen and Cartwright, it uses a draft version of a new text by Felleisen *et al.* The course averages ninety to one hundred students each semester.

Research

Dr. Cooper's research has focused on four general areas in compilation: interprocedural analysis and optimization, classical scalar analysis and optimization, code generation, and unusual applications of compiler technology.

Interprocedural Analysis and Optimization Dr. Cooper has worked on problems in interprocedural analysis and optimization for fifteen years. In the early years, most of this work involved developing new algorithms for interprocedural data-flow analysis.

He developed (with K. Kennedy and L. Torczon) the fastest known algorithms for several interprocedural data-flow problems. They designed and implemented the $\mathcal{R}^n$ programming environment – the first practical system for efficient interprocedural analysis. The algorithms and ideas developed in $\mathcal{R}^n$ have been used to build practical commercial systems that perform analysis and optimization of whole programs.

In recent years, the focus has shifted toward using interprocedural analysis to support code-improving transformations. He published (with M. Hall and L. Torczon) an experimental study of how well commercial optimizing compilers handled code resulting from inline substitution. This study inspired some of the work in classical optimization and code generation described below. His work on register promotion (with J. Lu) uses the result of interprocedural points-to analysis to improve pointer-based references. His work on compiler-controlled memory (with T.J. Harvey) and on active code compression (with N. McIntosh) involved both interprocedural analysis and interprocedural transformations.

Classical Scalar Analysis and Optimization In the area of single-procedure data-flow analysis, Dr. Cooper developed (with P. Briggs, T.J. Harvey, and L.T. Simpson) improved techniques for constructing and dismantling static single assignment form. He worked on techniques for analyzing data-reuse patterns across multiple loop nests (with N. McIntosh and K. Kennedy); the results are used to improve compiler placement of advisory prefetch instructions. In a more theoretical vein, he developed (with C. Click) a framework for combining disjoint analyses into a single, unified analysis.

Dr. Cooper has worked on a number of problems that involve transforming code to improve its performance. These include work on algebraic reassociation of expressions, (with P. Briggs), algorithms for regional and global value numbering (with P. Briggs and L.T. Simpson), a technique for register promotion that rewrites unambiguous pointer-based variable accesses as scalars (with J. Lu), and a new algorithm for operator strength reduction (with L.T. Simpson and C. Vick). In recent years, he has worked on problems that arise in the context of embedded systems. This work has included development of algorithms for managing a small compiler-controlled memory (with T.J. Harvey), on code compression using procedure abstraction and cross-jumping (with N. McIntosh), and on using a genetic algorithm to discover program-specific optimization sequences that result in more compact code (with P. Schielke and D. Subramanian).

Code Generation Dr. Cooper has worked on problems in code generation since the late 1980's. He published (with P. Briggs and L. Torczon) a series of papers that detailed improvements to Chaitin's scheme for graph-coloring register allocation. More recent work includes a paper on how to implement the interference graph construction (with T.J. Harvey and L. Torczon), a technique for passive live-range splitting (with L.T. Simpson), and a proposal for Compiler-Controlled Memory on DSP-chips (with T.J. Harvey). This last paper also explores coloring spill memory as a way of reducing the data-space requirements of programs.

With P. Schielke and D. Subramanian, he has been working on problems in instruction scheduling. They have investigated the quality of schedules produced by list scheduling for a large number of basic blocks and are developing a characterization of what properties make a block hard for list scheduling to handle. They have developed a family of scheduling algorithms based on iterative repair, and worked on algorithms for cross-block scheduling with limited code growth for size constrained environments.

Application of Compiler Technology Recently, Dr. Cooper has worked with J. Bennett and L. Torczon on the problem of applying classical compiler-based optimization techniques to problems that

arise in the derivation of digital circuits from specifications written in VHDL. Their system takes designs derived from VHDL by commercial synthesis tools, uses transformations such as value numbering, peephole optimization, and replication to rewrite the designs, and feeds them back into the commercial tool set. The goals of the project are to improve the quality of designs produced from VHDL programs, to broaden the set of VHDL programs that produce reasonably efficient hardware designs, and to modify the designs in ways that improve their place-and-route times. This work is still preliminary—it should produce experimental results in the next twelve months.

Funding Dr. Cooper has attracted funding from DARPA, from DOE, from NSF, and from industry. As of October 1, 1999 the primary funding sources for the group will be:

1. "*Code optimization for embedded systems*" (with D. Subramanian and L. Torczon), $1.1M from DARPA (7/97–10/00);
2. "*Optimizing* VHDL *intermediate forms*" (with J. Bennett and L. Torczon), $751,000 from DARPA (9/97–10/00);
3. "Compiling for IA-64" (with L. Torczon), $180,000 as part of the Los Alamos Computer Science Institute (DOE) (10/99 to 9/00);
4. "*Grid Application Development Software*" (with 11 other principal investigators at 8 institutions). Dr. Cooper's subproject is budgeted at $500,000 for three years.

Service

In his role as a member of the Rice community, Dr. Cooper has been a faculty associate of Brown College since 1983, an outstanding associate since 1984, and an Undergraduate Divisional Advisor for Engineering students since the inception of that program in 1988. He has served on several University Committees, including the Engineering Computer Planning Committee, the University Committee on Computing, the University Task Force on Aids, the University Research Council, the Committee of Marshalls, the President's *Ad Hoc* Security Review Committee, the President's *Ad Hoc* Committee on Access, the Search Committee for the Dean of Engineering, the Committee on Public Lectures, and the Parking Study Advisory Committee. He was deeply involved in the design and construction of Duncan Hall, the home of Rice's Computational Engineering community; this role led to his appointment, *ex officio,* to the Duncan Hall Operations Committee. It also led to his most unusual speaking engagement – an invited talk at the annual meeting of the Marek Brothers Family of Companies (one of the largest drywall companies in Texas). He is currently a member of the faculty search committee for the Electrical and Computer Engineering Department and chair of the Computer Science Department's Industrial Affiliates Committee.

As a member of his research community, Dr. Cooper has served on several program committees and tutorial committees for major conferences. He was the founding vice-president of the Sun Microsystems Users Group, Incorporated (a for-profit California corporation). He served on the Policy Board of the Concurrent Supercomputing Consortium – the board that managed the installation and operation of the Intel Touchstone Delta machine (at the time, the world's fastest computer). Most recently, he was Program Chair for the 1998 ACM SIGPLAN Symposium on Programming Language Design and Implementation (PLDI).