Real Compilers, Tools, and Systems

Reinhard Wilhelm
Universität des Saarlandes
wilhelm@cs.uni-sb.de

July 16, 1997

“Standard” Structure

source(text)	
↓	
lexical analysis(7)	finite automata
↓	
tokenized-program	
↓	
syntax analysis(8)	pushdown automata
↓	
syntax-tree	
↓	
semantic-analysis(9)	attribute grammar evaluators
↓	
decorated syntax-tree	
↓	
optimizations(10)	abstract interpretation + transformations
↓	
intermediate rep.	
↓	
code-generation(11, 12)	tree automata + dynamic programming + $\cdots$
↓	
machine-program	

Issues

- How are compilers really produced?
- Which tools are used?
- Goals:
  - Target code efficiency,
  - Compiler speed,
  - Portability, retargetability
- Compiler systems,
- Where does the development go?

Tools, Generators, Systems

Lexing: lex, flex, . . .

Parsing: yacc, bison, . . .

Semantic analysis:

- Attibute grammar systems:
  - GAG (Karlsruhe, Paderborn),
  - OPTRAN (UdS), incl. tree pattern matching, transformation, attribute reevaluation,
  - Cornell Program Synthesizer Generator (commercial), incl. reevaluation,
  - FNC–2 (INRIA)
  - ELI (Boulder, Paderborn)
- Natural semantics based systems
  - CENTAUR (INRIA, CWI),
  - RML (Linkoeping),
  - TwoBig (UdS).

Static program analysis: Systems based on Abstract Interpretation

- Program Analyzer Generator (PAG) (UdS),
- Spare (Madison),
- Sharlit.

Code generators: Code selector generators with (mostly) on the fly register allocator and/or scheduler:

- BEG (Karlsruhe),
- `lburg` (Bell Labs, U. Arizona).

Tool kits: Set of (generator) tools with defined interfaces,

- ELI (Boulder, Paderborn),
- Cocktail (Karlsuhe) (commercial).

Compiler Systems: Predefined or definable data structures and interfaces,

- CoSy (Compare Consortium),
- SUIF (Stanford University),
- National Compiler Infrastructure (US).

A Real Compiler, `lcc`

- Production C compiler,
- Authors: Christopher Fraser, David Hanson,
- Documented in: "A Retargetable C Compiler: Design and Implemantation, Benjamin Cummings 1995,
- Available with diskette,
- Designed and implemented by the authors,
- Goal: A C compiler for teaching purposes,
  portability, retargetability,
  simple design, moderate optimizations (only local not global), fast execution,
- Respects calling conventions for the targets,
- Not using some tools to "avoid the dependence on them", seen as a problem when the design started.

Compiler Structure and Interfaces

1. Imported preprocessor,

2. Lexing with `lex`–generated scanner,

3. Parsing with a recursive descent parser, combined with construction and decoration of abstract syntax trees, type checking, introduction of conversions.

4. Conversion to dags, eliminating common subexpressions,

5. Control flow representation in a "code list", the interface between front end and back end,

6. Instruction selection done by an `lburg` generated code selector, selected code as annotations on the dags in the code list,

7. Three pass register allocator using a simple heuristics for spills,

8. Generated assembly code is assembled and linked by UNIX tools.