Compiler Frameworks

A (non-exhaustive) survey

with a focus on open-source frameworks

LCC (Little C Compiler)
- Drag-on-book style C compiler implementation in C
- Very small (20K Loc), well documented, well tested, widely used
- Open source: http://www.cs.princeton.edu/software/lcc
- Textbook: A retargetable C compiler [Fraser, Hanson 1995] contains complete source code
- One-pass compiler, fast
- C frontend (hand-crafted scanner and recursive descent parser) with own C preprocessor
- Low-level IR
  - Basic-block graph containing DAGs of quadruples
  - No AST
- Interface to IBURG code generator generator
  - Example code generators for MIPS, SPARC, Alpha, x86 processors
  - Tree pattern matching + dynamic programming
- Few optimizations only
  - local common subexpr. elimination, constant folding
- Good choice for source-to-target compiling if a prototype is needed soon...

GCC 4.x
- GNU Compiler Collection (earlier: GNU C Compiler)
- Compilers for C, C++, Fortran, Java, Objective-C, Ada ...
  - sometimes with own extensions, e.g. Gnu-C
- Open-source, developed since 1985
- Very large
- 3 IR formats (all language independent)
  - GENERIC: tree representation for whole function (also statements)
  - GIMPLE (simple version of GENERIC for optimizations) based on trees but expressions in quadruple form.
  - High-level, low-level and SSA low-level form
- RTL (Register Transfer Language, low-level, Lisp-like) (the traditional GCC-IR)
  - only word-sized data types, block explicit; statement scope
- Many optimizations
- Many target architectures
- Version 4.x (since ~2004) has strong support for retargetable code generation
  - Machine description in .md file
  - Reservation tables for instruction scheduler generation
- Good choice if one has the time to get into the framework

Open64 / ORC Open Research Compiler
- Based on SGI Pro-64 Compiler for MIPS processor, written in C++, went open source in 2000
- Several tracks of development (Open64, ORC, ...) for Intel Itanium (IA-64) and x86 (IA-32) processors.
  - Also retargeted to x86-64, Ceva DSP, Tensilica, XScale, ARM ...
  - "simple to retarget" (?)
- Languages: C, C++, Fortran95 (uses GCC as frontend), OpenMP and UPC (for parallel programming)
- Industrial strength, with contributions from Intel, Pathscale, ...
- Open source: www.open64.net, ipf-orc.sourceforge.net
- 6-layer IR:
  - WHIRL (VH, H, M, L, VL) – 5 levels of abstraction
  - All levels semantically equivalent
  - Each level a lower level subset of the higher form
  - and target-specific very low-level CGIR
  - Good choice for source-to-target compiling if a prototype is needed soon...

Open64 / ORC Open Research Compiler
- Multi-level IR
  - translation by lowering
  - ☺ Analysis / Optimization engines can work on the most appropriate level of abstraction
  - ☺ Clean separation of compiler phases
  - ☺ Framework gets larger and slower

- Many optimizations, many third-party contributed components
Traditional Compiler Structure

- Traditional compiler model: sequential process
- Improvement: Pipelining (by files/modules, classes, functions)
- More modern compiler model with shared symbol table and IR

A CoSy Compiler with Repository-Architecture

- "Engines" (compiler tasks, phases)
- Transformation
- Parser
- Optimizer
- Codegen
- "Blackboard architecture"

Composite Engines in CoSy

- Built from simple engines or from other composite engines by combining engines in interaction schemes (Loop, Pipeline, Fork, Parallel, Speculative, ...)
- Described in EDL (Engine Description Language)
- View defined by the joint effect of constituent engines
- A compiler is nothing more than a large composite engine

ENGINE CLASS compiler (IN u: mirUNIT) {
  PIPELINE
  frontend (u)
  optimizer (u)
  backend (u)
}

A CoSy Compiler

- Modular compiler building block
- Performs a well-defined task
- Focus on algorithms, not compiler configuration
- Parameters are handles on the underlying common IR repository
- Execution may be in a separate process or as subroutine call - the engine writer does not know!
- View of an engine class: the part of the common IR repository that it can access (scope set by access rights: read, write, create)
- Examples: Analyzers, Lowerers, Optimizers, Translators, Support

Composite Engines in CoSy
Example for CoSy EDL (Engine Description Language)

- Component classes (engine class)
- Component instances (engines)
- Basic components are implemented in C
- Interaction schemes (cf. skeletons) form complex connectors
  - SEQUENTIAL
  - PIPELINE
  - DATAPARALLEL
  - SPECULATIVE
- EDL can embed automatically
  - Single-call-components into pipes
  - p<> means a stream of p-items
  - EDL can map their protocols to each other (p vs p<>)

ENGINE CLASS optimizer (procedure p)
{ControlFlowAnalyser cfa;
 CommonSubExprEliminator cse;
 LoopVariableSimplifier lvs;
 PIPELINE cfa(p); cse(p); lvs(p);}

ENGINE CLASS compiler (file f)
{... Token token;
 Module m;
 PIPELINE // lexer takes file, delivers token stream:
 lexer( IN f, OUT token<> );
 // Parser delivers a module
 parser( IN token<>, OUT m );
 sema( m );
 decompose( m, p<> );
 // here comes a stream of procedures
 // from the module
 optimizer( p<> );
 backend( p<> );}

Evaluation of CoSy

- The outer call layers of the compiler are generated from view description specifications
  - Adapter, coordination, communication, encapsulation
  - Sequential and parallel implementation can be exchanged
  - There is also a non-commercial prototype

- Access layer to the repository must be efficient (solved by generation of macros)
- Because of views, a CoSy-compiler is very simply extensible
  - That's why it is expensive
  - Reconfiguration of a compiler within an hour

More frameworks...

- LLVM (Univ. of Illinois at Urbana Champaign)
  - llvml.org
  - "Low-level virtual machine", IR
  - compiles to several target platforms: x86, Itanium, ARM, Alpha, SPARC
  - Open source

- Cetus
  - http://cobweb.ecn.purdue.edu/ParaMount/Cetus/
  - C/C++ source-to-source compiler written in Java.
  - Open source

- Tools and generators
  - TXL source-to-source transformation system
  - ANTLR frontend generator

More frameworks...

- Some influential frameworks of the 1990s:
  - SUIF Stanford university intermediate format, suif.stanford.edu
  - Trimaran (for instruction-level parallel processors)
    www.trimaran.org
  - Polaris (Fortran) UIUC
  - GMD Toolbox / Cocolab Cocktail™ compiler generation tool suite
  - and many others ...

- And many more for the embedded domain ...

The End (?)

"Now this is not the end.
It is not even the beginning of the end.
But it is, perhaps, the end of the beginning."
- W. Churchill

- Do you like compiler technology? Learn more?
  - TDDC86 Compiler optimizations and code generation 6hp
  - TDDC18 Component-based software 4.5hp
  - Thesis project at PELAB, 30 hp