DF00100
Advanced Compiler Construction

Organizational issues
www.ida.liu.se/~chrke/courses/ACC

Course moments (total: 9 hp)
- Lectures and exam
  2 lecture blocks (week 7 + week 9, Monday-Thursday)
- See course web page for schedule, contents
- Written/oral exam 17 March 2010 14:00-18:00, 4.5hp
- Labs, 3 hp (could be done in groups of 2)
  - LLVM open-source compiler framework, llvm.org
  - Lab part 1: IR and program analysis, 1.5hp
  - Lab part 2: Code generation, 1.5hp
- Presentation 15 March 2010 09:15–…, 1.5hp
  - Of a recent compiler research paper
  - Opposition on another presentation
- Written summary with your own words, ca. 2 pages

Why Another Compiler Course? (1)
- Focus of traditional compiler courses (e.g., TDDB44, TDDD16):
  - Understand concepts of programming languages
  - Syntax, semantics
  - Good application of formal languages and automata theory
  - Lexing, parsing
  - Toy languages and toy target architectures
  - Front-end, Parser generators, symbol table, AST, syntax-driven translation, quadruples, simple code generation
  - Technology well-established since 1970s
- Current compiler technology R&D has a different focus:
  - Rate of language introduction is low
  - Few students will be hired to write industrial frontends
  - Rate of architectural change and variety is high
  - Embedded pr., uC, DSP, NP, superscalar, VLIW/EPIC, SIMD, SMP, Cluster, Multicore, NoC, reconfigurable, FPGA, ...
  - A new computer architecture does not sell without a (~C) compiler
  - Optimizing compilers vs. Manual low-level coding and tuning
  - High requirements on code
  - Execution time, Realtime constr., Code size, Energy consumption
  - Hot issues: Automatic program optimization, HQ code generation
  - Necessary for this: Static analysis of programs
  - Also hot, but not covered here: Static analysis for correctness and security properties

Contents
- Advanced Intermediate Representation Design
  - Multi-Level IRs
  - Static Single Assignment (SSA) Form
- Static Analysis of Programs
  - Control Flow Analysis
  - Data Flow Analysis
  - Abstract Interpretation
  - Points-to Analysis
  - Dependence Analysis
- Target-independent / High-Level Optimizations
  - Loop Optimizations e.g. for Data Locality, Loop Parallelization, ...
  - Optimized Code Generation
  - Instruction Selection, Instruction Scheduling, Register Allocation, Predication, ...
  - Code Generation for embedded, DSP, and parallel target architectures
- Autotuning and Other Issues (as time permits)
Literature

- No single book covers the course contents completely.
  - Combine different book chapters and papers
- List on course homepage
- In the library

Prerequisites

- A first course in compiler construction
  - TDDD16, TDDB44 or similar
  - or read the Dragon book in advance
- A course in computer architecture
  - Processor structure, pipelining, assembler language...
  - or read Hennessy/Patterson: Computer Architecture
- Background in discrete maths, data structures and algorithms
  - Graphs, trees; depth-first search; connected components; backtracking, dynamic programming, branch-and-bound....
- Some recapitulation material available on course homepage