DF00100
Advanced Compiler Construction

Organizational issues

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

Christoph Kessler, IDA,
Linköpings universitet, 2014.

Staff 2014

- Lectures / Presentation session / Examination
  - Christoph Kessler, IDA, Linköping University
    christoph.kessler \at liu.se
  - Welf Löwe, Linné-universitetet, Växjö,
    welf.lowe \at vxu.se (guest lecturer / examiner)
  - Sudipta Chattopadhyay, IDA, Linköping University
    sudipta.chattopadhyay \at liu.se (guest lecturer)
- Lessons
  - Erik Hansson, IDA, eriha \at ida.liu.se
- Labs (LLVM)
  - Erik Hansson
- Course administrator
  - Anne Moe, IDA, anne.moe \at liu.se

Course moments (total: 9 hp)

- **Lectures and exam**
  - 2 lecture blocks (week 36 + week 37, Monday-Friday/Thursday)
  - See course web page for schedule, contents
  - Written/oral exam __ oct 2014 afternoon, 4.5hp
  - Mandatory presence 50% of the lectures + lessons for admission to presentation and exam
- **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** 29 sep 2014 09:15-… (whole day), 1.5hp
  - of a recent compiler research paper
  - Opposition on another presentation
  - Written summary with your own words, ca. 2 pages

Lessons and Labs

Lessons:

- Theory exercises, good as preparation for the written exam
- To get out most of the lessons for yourself:
  - Prepare your solutions ahead of time
  - Present your solution in class

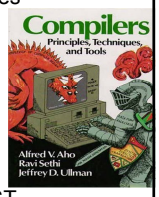
Labs:

- Lab introduction tomorrow (Tuesday) at 11:00
- Mission critical, attendance is highly recommended

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



Why Another Compiler Course? (2)

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., DSP, NP, superscalar, VLIW/EPIC, SIMD, GPU, SMP, Cluster, Multicore, MPSoC, reconfigurable, FPGA, Memory hierarchy…
  - A new computer architecture does not sell without a (~C) compiler
  - Optimizing compilers vs. Manual low-level coding and tuning
- High requirements on code
  - Performance, Realtime constraints, Code size, Energy efficiency
- Hot issues: Automatic program optimization, vectorization and parallelization, high-quality target code generation; (run-time adaptation)
  - Necessary for this: Static analysis of programs
- Also hot, but not covered here: Static analysis for correctness and security

Contents

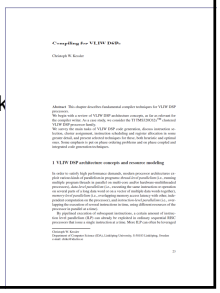
- **Advanced Intermediate Representation Design**
  - Multi-Level IRs
  - Static Single Assignment (SSA) Form
- **Static Analysis of Programs**
  - Control Flow Analysis
  - Data Flow Analysis
  - Abstract Interpretation
  - Points-to Analysis
  - Dependence Analysis
  - WCET Analysis
- **Target-independent / 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

Literature (cont.)

- C. Kessler: Compiling for VLIW DSPs.
  Book chapter, in S. Bhattacharyya, E. Deprettere, R. Leupers and J. Takala, eds., *Handbook of Signal Processing Systems*, Springer, 2010; also: 2nd edition, 2013. Ca. 41 pages.
  - Preprint handed out
  - Mandatory course literature for the code generation part
  - TekNat-Library has the complete book



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 repetition material available on course homepage