TDDC86: Compiler Optimizations and Code Generation

Organizational issues
www.ida.liu.se/~TDDC86

Staff
- Lectures / Seminars / Examination
  - Christoph Kessler, IDA, Linköpings universitet
  - chrke@ida.liu.se
- Lessons
  - Mattias Eriksson, IDA, Linköpings universitet
  - mater@ida.liu.se
- Course administrator
  - Gunilla Mellheden
  - gunme@ida.liu.se

Course structure (total: 6 hp)
- Lectures and exam, 3 hp
  - 13 lectures (including today)
  - + possibly a guest lecture
  - See course homepage for schedule, contents
  - Written exam, 23 Oct 2009, 14:00-18:00
- Project, 3 hp
  - Either: Compiler framework experiment (in groups of 2)
  - Or: Compiler research paper study (alone)
  - Presentation in the seminars 7/10, 9/10, 12/10, 13/10
  - Opposition - -
  - Presence in seminars is mandatory
  - Written summary

Why Another Compiler Course? (1)
- Focus of traditional compiler courses
  (e.g., TDDD44, 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, SoC, MPSoC, 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 IPs, Static Single Assignment (SSA) Form
- Static Analysis of Programs
  - Control Flow Analysis
  - Data Flow 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
- A course in computer architecture
  - Processor structure, pipelining, assembler language
  - If not: 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

Project Assignments

For more details and resources, see the course homepage, www.ida.liu.se/~TDDC86

Project Variant 1:
Compiler Framework Experiment

- Done in groups of two, preferably at home
  - On native (your laptop) or simulated processor architecture
- Choose a free compiler framework, e.g. GCC, LLVM, Open64, ...
- Choose (at least) two compiler optimizations / transformations
  - E.g., software pipelining, loop unrolling, loop blocking, automatic vectorization, predication, etc.
- Choose (at least) 3 simple but nontrivial test programs
  - The usual suspects: matrix-matrix-multiply, sorting, ...
  - Or, use existing benchmark material and/or share with others
- Predict execution time and experiment with the chosen optimizations
  - On/off, ordering; with different problem sizes; ...
  - Display the measurements in diagrams (e.g., with gnuplot)
- Try to explain the observations
- Conclusions: When / how to apply for this platform?

Project Variant 2:
Research Paper Study

- Done alone
- Select a recent research paper that is related to our course contents
  - Int. conference or journal paper, peer-reviewed
  - Resources: See course homepage
  - If you cannot find a suitable paper on your own, ask me to assign one to you.
- Read critically and analyze the paper carefully,
- Check also related work
  - Follow up the paper references and citation information (e.g. in the ACM DL about later papers that refer to yours)
  - And read the paper you are opponent for
- Presentation, Opposition, Summary

Project: Deadlines and Schedule

- Decide for a project and mail the following information to me:
  - For a Compiler framework project:
    - Who; Which compiler, transformations, platform, test programs
  - For a Research paper study project:
    - Which paper (full citation data or author+title+DOI)
    - By 25 sept 2009 for approval.
    - Find an opponent / project for opposition
    - Mail opponent name to me by 25 sept 2009
    - Do the project
    - Deadline: 5 oct 2009
    - Prepare presentation
    - Send the slides to me by 5 oct 2009, cc to your opponent
    - Prepare opposition - in particular: 3 questions, mail me by 6 oct 2009
    - Presentations with opposition: 7/10, 9/10, 12/10, (13/10)
    - Written summary of your project (~3 pages – no copying!) by 16 oct 2009

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
- A course in computer architecture
  - Processor structure, pipelining, assembler language
  - If not: 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

Project Assignments:

For more details and resources, see the course homepage, www.ida.liu.se/~TDDC86

Project Variant 1:
Compiler Framework Experiment

- Done in groups of two, preferably at home
  - On native (your laptop) or simulated processor architecture
- Choose a free compiler framework, e.g. GCC, LLVM, Open64, ...
- Choose (at least) two compiler optimizations / transformations
  - E.g., software pipelining, loop unrolling, loop blocking, automatic vectorization, predication, etc.
- Choose (at least) 3 simple but nontrivial test programs
  - The usual suspects: matrix-matrix-multiply, sorting, ...
  - Or, use existing benchmark material and/or share with others
- Predict execution time and experiment with the chosen optimizations
  - On/off, ordering; with different problem sizes; ...
  - Display the measurements in diagrams (e.g., with gnuplot)
- Try to explain the observations
- Conclusions: When / how to apply for this platform?

Project Variant 2:
Research Paper Study

- Done alone
- Select a recent research paper that is related to our course contents
  - Int. conference or journal paper, peer-reviewed
  - Resources: See course homepage
  - If you cannot find a suitable paper on your own, ask me to assign one to you.
- Read critically and analyze the paper carefully,
- Check also related work
  - Follow up the paper references and citation information (e.g. in the ACM DL about later papers that refer to yours)
  - And read the paper you are opponent for
- Presentation, Opposition, Summary

Project: Deadlines and Schedule

- Decide for a project and mail the following information to me:
  - For a Compiler framework project:
    - Who; Which compiler, transformations, platform, test programs
  - For a Research paper study project:
    - Which paper (full citation data or author+title+DOI)
    - By 25 sept 2009 for approval.
    - Find an opponent / project for opposition
    - Mail opponent name to me by 25 sept 2009
    - Do the project
    - Deadline: 5 oct 2009
    - Prepare presentation
    - Send the slides to me by 5 oct 2009, cc to your opponent
    - Prepare opposition - in particular: 3 questions, mail me by 6 oct 2009
    - Presentations with opposition: 7/10, 9/10, 12/10, (13/10)
    - Written summary of your project (~3 pages – no copying!) by 16 oct 2009