DF00100

Advanced Compiler Construction

2021

Organization, Motivation, Overview

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

Christoph Kessler, IDA,
Linköping University

Staff 2021

- Lectures / Presentation session / Examination
  - Christoph Kessler, IDA, Linköping University
    christoph.kessler \at liu.se
  - Welf Löwe, Linnaeus-University, Växjö,
    welf.lowe \at lnu.se   (guest lecturer / guest examiner)
  - Martin Sjölund, IDA, Linköping University
    martin.sjolund \at liu.se   (guest lecturer)
  - Possibly 1-2 further guest lectures to be confirmed/scheduled
- Lessons
  - Christoph Kessler
  - August Ernstsson, IDA,   august.ernstsson \at liu.se
- Labs (LLVM)
  - August Ernstsson
- Course administrator
  - Anne Moe, IDA, anne.moe \at liu.se

Course moments (total: 9 hp)

- **Lectures and exam**
  - 2 lecture blocks (week 6 + week 7)
  - See course web page for schedule, contents
  - Written/oral exam 31 March 2021 09:00-13:00, 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** 24 March 2021 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 exam

■ To get out most of the lessons for yourself:

◆ Prepare your solutions ahead of time

◆ Present your solution in class

Labs:

■ Lab introduction today (Tuesday) at 13:15

■ Mission critical, attendance is highly recommended

Why Another Compiler Course? (1)

Focus of traditional compiler courses
(e.g., TDDB44, TDDD55):

■ 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
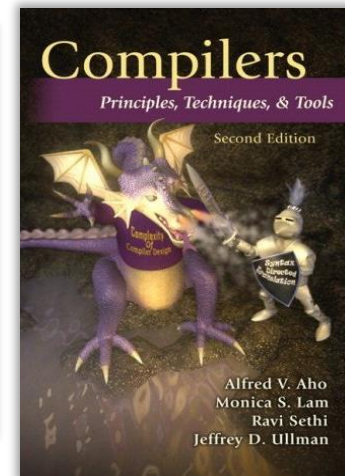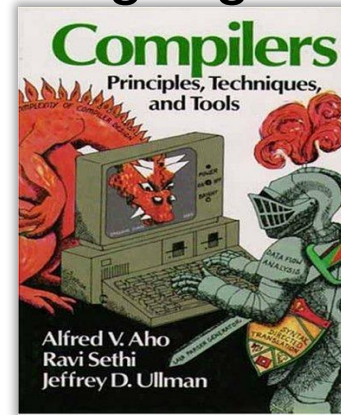
Why Another Compiler Course? (2)

Current compiler technology R&D has a different focus:

■ Rate of programming language introduction is rather low
- ◆ Mostly, DSLs (usually compile to C/C++)
- ◆ Few students will be hired to write industrial frontends

■ Rate of architectural change and variety is high
- ◆ Processor architecture: DSP, superscalar, VLIW/EPIC, SIMD, GPU, ML accelerators, SMP, NUMA, Cluster, Multicore, MPSoC, reconfigurable, FPGA
- ◆ Memory architecture: memory hierarchy, HBM, prefetching, device memory, local memory, memory banks, in-memory-computing, …
- ◆ 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
- ◆ Performance portability, utilization of accelerators
- ◆ Moore's Law is slowing down → more performance growth must come from the software – most conveniently from an optimizing compiler

■ Hot issues: Automatic program optimization, vectorization and parallelization, accelerator use, high-quality target code generation; run-time adaptivity; coupling with libraries
- ◆ Required for this: Static analysis of programs, multi-level internal representations, graph algorithms, combinatorial optimization, architecture modeling

■ 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 (not 2021)
- **High-Level Optimizations**
  - Loop Optimizations e.g. for Data Locality; Loop Parallelization; …
  - Task Fusion, Resource Allocation, Mapping, Scheduling, DVFS
- **Optimized Target Code Generation**
  - Instruction Selection, Instruction Scheduling, Register Allocation, …
- Code generation and optimization for special target architectures
  - DSP, clustered VLIW, SIMD, parallel, GPU, ...
- **Adaptivity, Autotuning and Other Issues** (as time permits)
- Guest lectures e.g. about **DSL compilation**, e.g. Modelica compiler
- Student presentations about selected recent compiler research papers
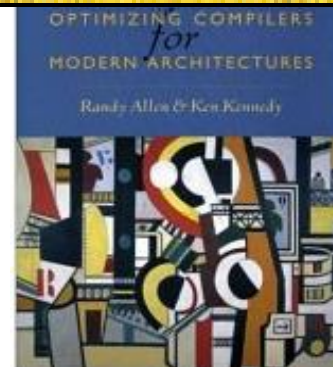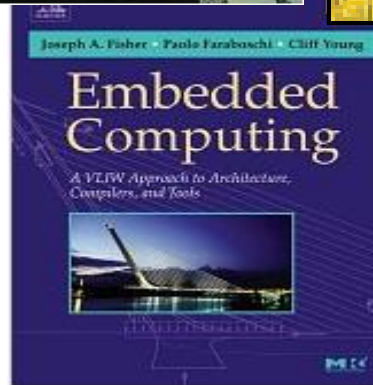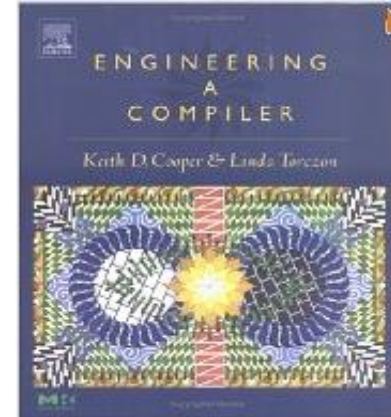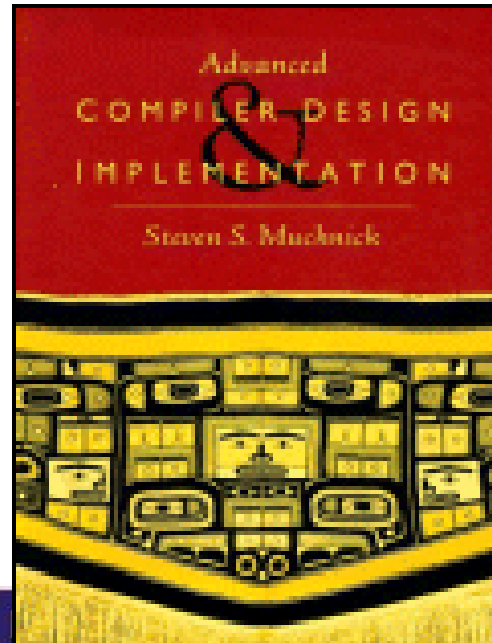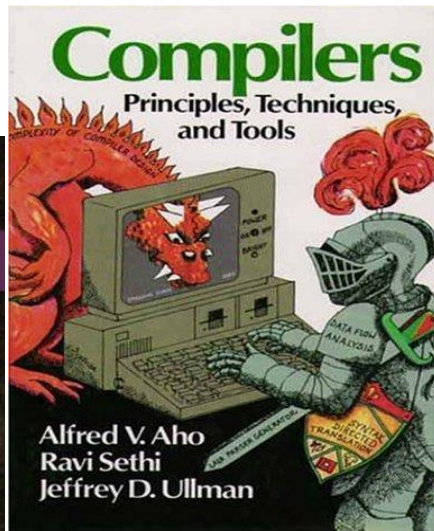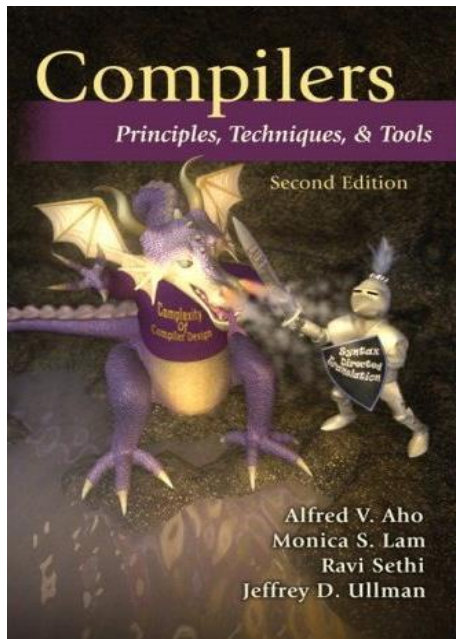- Labs: LLVM

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*,
3rd Edition, Springer, 2019. Ca. 41 pages.

◆ Preprint handed out

◆ Mandatory course literature
   for the code generation part

◆ TekNat-Library has the complete book
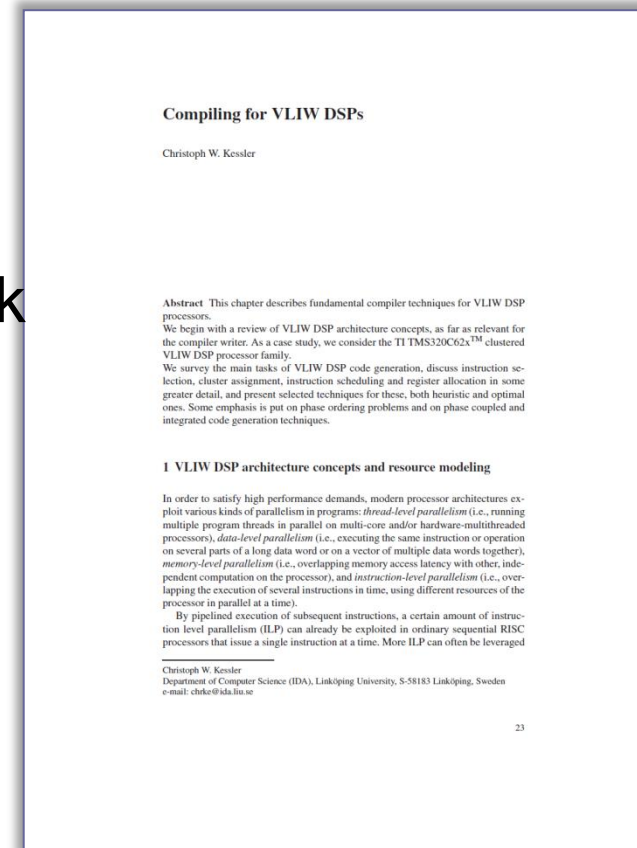
Prerequisites

- A first course in compiler construction
  - TDDB44, TDDD55 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*
  - or watch Onur Mutlu's 2020 Computer Architecture lectures videos on youtube:
    https://www.youtube.com/watch?reload=9&v=c3mPdZA-Fmc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN
- Background in discrete maths, data structures and algorithms
  - Graphs, trees; depth-first search; connected components; backtracking, dynamic programming, branch-and-bound,…
  - Integer linear programming
- Some repetition material available on course homepage