UBC Department of Electrical and Computer Engineering
EECE 571 T: Optimizing Compilers (2024W)
Course Syllabus

Description: As technology scaling reaches its limits computer systems performance has begun to rely upon a combination of domain specific languages and specialized hardware. Bridging the gap between software languages and high-performance hardware is the task of the compiler. This course introduces modern optimizing compiler design and implementation and provides students with experience making changes to an open-source compiler.

Contact Information
Instructor: Prof. Tor Aamodt (aamodt@ece.ubc.ca)

Course Structure
Lectures: Monday 0900-1200 (SWNG 405)
Office hours: by appointment

References (not required)
Aho, Sethi, Ullman and Lam, Compilers: Principles, Techniques, and Tools, 2nd edition, Pearson
Muchnick, Advanced Compiler Design and Implementation. Morgan Kaufmann

Course Content: Topics covered will include compiler intermediate representations, control flow analysis, data flow analysis, code optimization, instruction scheduling, register allocation, static single assignment, inter procedural analysis, alias analysis, just in time compilation, automated parallelization. Experience implementing compiler optimizations in the open source LLVM compiler (llvm.org).

Learning Objectives: By the end of the course students will have both a theoretical knowledge of optimizing compiler technology an practical ability to implement novel compiler optimizations in widely used open source compiler infrastructure.

Course Activities and Assessment: Your mark is based upon assessment of compiler assignments, paper reports based upon assigned readings a project and a final exam. The weight of each component on your final grade will be:

Assignments:	20% (5 assignments × 4% each)
Paper readings:	10% (5 paper readings x 2% each)
Project:	35%
Exam:	25 %

Webpages: EECE 571T will make use of http://canvas.ubc.ca and possibly Piazza.

Slides: The lectures slides are not a complete record of the course. You should take notes while attending lectures. Updates to slides may be posted after lectures.

Assignments: The course emphasizes practical skills in developing compiler optimizations. To enable a gradual introduction, early assignments will use the educational BRIL intermediate representation (https://capra.cs.cornell.edu/bril/intro.html) while later assignments will introduce students to LLVM.

Exams: The final exam date and time is TBD.

Academic Integrity: Students are to work individually. Use of code written by anyone, but an authorized lab partner is forbidden. Code from lectures or assigned textbooks can be used if the source is cited. Use of AI permitted provided that use is documented and cited in submitted code.

Final Exam: Final exam is expected to take place in person. For the final exam you can bring two 8.5x11" hand written aid sheets (or use both sides of one sheet).

Course Schedule

NOTE: timing will vary depending upon pace of lectures [Version 1; 5 Jan 2025]

		Lecture topics	**Assignments**
Week 1	1/6/25	Introduction, Intermediate Representations, Intro to BRIL	
Week 2	1/13/25	Control flow analysis	hw1
Week 3	1/20/25	Data flow analysis	hw2
Week 4	1/27/25	Local optimizations, redundancy optimizations	hw3
Week 5	2/3/25	Dead code elimination, loop optimizations	hw4
Week 6	2/10/25	Register allocation, intro to LLVM	hw5
Week 7	2/17/25	*midterm break*	
Week 8	2/24/25	Instruction scheduling	
Week 9	3/3/25	Locality optimizations	paper #1
Week 10	3/10/25	Parallelization	paper #2
Week 11	3/17/25	*Interprocedural Optimizations*	paper #3
Week 12	3/24/25	Alias Analysis	paper #4
Week 13	3/31/25	Dynamic Compilation	paper #5
Week 14	4/7/25	*Project Presentations*	