[Note: These are the notes I use for the lectures. To use these as lecture notes, you need to ignore some irrelevant parts. They should work as a summary of the lecture topics, though.]

1 Intro, lecture 1

1.1 Welcome

Course: Compiler Construction, DD2488, 9hp
Teachers:
  Lecturer: Torbjörn Granlund, (overdue) PhD student TCS, D83, GNU hacker
  Assistant: Pedro Gomes, (final year) PhD student TCS

1.2 Course Registration

Anybody not reg'd in LADOK?
List your names on my list here...then talk to your student counsellor ASAP.

Anybody lacks a CSC/Nada account? Talk to the service desk (you need to be reg'd first).

When reg'd in LADOK, you need to make yourself active in rapp (else I cannot report results...).

1.3 Course Organisation

Teaching: 8 Lectures on theory (Torbjörn), project start session in 4 weeks (Pedro), then project workshops (Torbjörn, Pedro, Douglas).

Examination: Project report, oral project presentation, exam on theory (dat TBD).

4	We	23 Jan	13-15	Lecture 1	D2
5	We	30 Jan	13-15	Lecture 2	D2
6	We	06 Feb	13-15	Lecture 3	D3
7	We	13 Feb	13-15	Lecture 4	D2
8	We	20 Feb	13-15	Lecture 5	D3
9	We	29 Feb	13-15		

Check 'kurspm' for more info

We will have some semi-compulsory workshops starting about 10 days after the project launch. Explain how these will work!

1.4 Course overview

(10 minutes)

Project (PRO1):

1. Write a compiler in Java for a subset of Java ("Minijava")
2. We follow Appel's book, with slight modifications (no nested comments, if-else, etc)
3. Possible targets: JVM, Sparc, MIPS, X86-64 (not X86-32)
4. Point system for project grading (with cap!), graded A-E
5. Tested using "Tigris" (cf "Kattis")
6. Project groups + mutual feedback groups (explain!)
7. 3 test programs by each project, with minimum size
8. BIG project, start soon!

Proj changes from prev years: Very similar to last years, changes after student feedback.

Suggest that students run qemu (except if just JVM backend). You own qemu installs, or images I provide.

2 A compiler's organisation

(65 minutes)

Lex	break source text into words (jenecomprendepaslejava)
Parse	analyse phrase structure according to language grammar, build syntax tree
Semantic analysis	bind variable use to decl, check typing, assign “meaning”
Frame Layout	determine location of incoming params, place local vars, place outgoing params
Translate	syntax trees $\rightarrow$ source-lang indep IR trees
Canonicalise	move out side-effects, clean up various things
Insn selection	match IR nodes into insns, assuming ∞ reg model
Control flow anal	gen graph for all possible insn sequences
Data flow anal	gen graph for information flow, determining the *live range* of scalars
Register allocation	match pseudo regs into physical regs, handle partial failures with *spilling*
Code emission	print out generates sequence of insns

(These notes lack the details from the lecture of these 11 steps. See course book for details.)