Introduction

EE599-201/EE699-201, Spring 2021

Hank Dietz

`http://aggregate.org/hankd/`

UK UNIVERSITY OF KENTUCKY

What Is A Compiler?

- CoBOL notion of collecting code fragments
- ForTran assignment statements
- Program Understanding AI: understand the meaning and translate into another language
- Translates a program into…
  - Another program?
  - Hardware?
  - Both?
- Are interpreters compilers?

Optimizing Compilers

- What does optimizing mean?
  - To make optimal?
  - To *probably* improve in some aspect
  - To automatically parallelize, if that helps
- A compiler applies **correctness-preserving transformations** to improve performance

This Course

- You will learn how to write a simple compiler
- You will learn how to write an assembler
- You will write (modify) compilers to perform
  - Analysis & optimization
  - Parallel code scheduling
  - Logic circuit design & optimization
- You will learn about HW/SW codesign

Textbook

- The text is… *there really isn’t one.*
- To get started, we’ll use my old course notes:
  `http://aggregate.org/EE380/notes.pdf`
  but that’s just the basics…
- Lots of additional materials at the course URL and presented in class

Grading & Such

- About 40%: 2-3 exams
- About 60%: 4 projects
  - Basic-block optimizer
  - Basic-block parallelizer
  - Control-flow optimizer/parallelizer
  - Hardware compilation
- I try not to curve much, but do adjust %

SCARY (TEAM?) PROJECTS!

- You need to be comfortable with C or C++
- All these are modifying code, not from scratch:
  - Basic-block optimizer
  - Basic-block parallelizer
  - Control-flow optimizer/parallelizer
  - Hardware compilation
- Everything can be done in 30 pages of code

Why This Is So Cool

- Consider this:

  `A = B * C; D = C * B; B = B * C; E = B * C;`

- That’s really the same as:

  `A = B * C; D = A; B′ = A; E = A * C;`

- And if `B = 2; C = B + B;` came before it:

  `A = 8; D = 8; B′ = 8; E = 32;`

Why This Is So Cool

- Consider this:

```
int f() {
  int r = 0;
  for (int i=0; i<1000000; ++i) ++r;
  return(r);
}
```

- With a little loop optimization this becomes:

```
int f() { return(1000000); }
```

Why This Is So Cool

- Consider this:

```
int:8 a, b, c;
a = (c * c) ^ 70;
a = ((a >> 1) & 1);
a = b + (c * b) + a;
a = a + ~(b * (c + 1));
```

- That causes about 206,669 gate operations
- Optimizing at the bit (gate) level, it’s just:

```
a = 0;
```

Course Content

- Introduction

- Simple compilation and assembly
  - Target model issues, superoptimizers
  - Assembler with forward reference resolution
  - Simple compiler
  - Peephole optimizations, constant folding, Sethi-Ullman numbering

Course Content

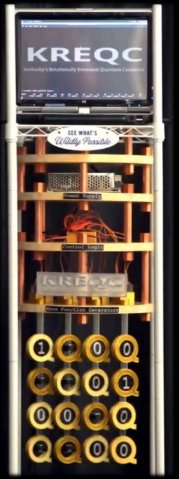
- Analysis and transformation
  - Value numbering, linear nested regions, static single assignment (SSA)
  - Common subexpression elimination (CSE) with value forwarding
  - Parallelization transformations, pipelining
  - Loop analysis and transformations
  - Interprocedural analysis and recursion

Course Content

- Silicon compilation & high-level logic synthesis
  - Transformation from word to bit level
  - Bit-level optimization & transformation
  - Normal form transformations
  - State machines
- Hardware acceleration and hardware/software codesign

Me (and why I'm biased)

- Hank Dietz, ECE Professor and James F. Hardymon Chair in Networking
- Built world's 1st Linux PC cluster supercomputer
- I have a lot of cool toys…



UNIVERSITY OF KENTUCKY

My bias about compilers

- PhD: **The Refined-Language Approach To Compiling For Parallel Supercomputers**
  `http://aggregate.org/REFINED/thesis.pdf`
- **Purdue Compiler Construction Tool Set**
  `http://www.polhode.com/pccts.html`
  `http://antlr.org/`
- C to low level, not FORTRAN to FORTRAN