# Topic 1:  Introduction

COS 320

Compiling Techniques

Princeton University
Spring 2015

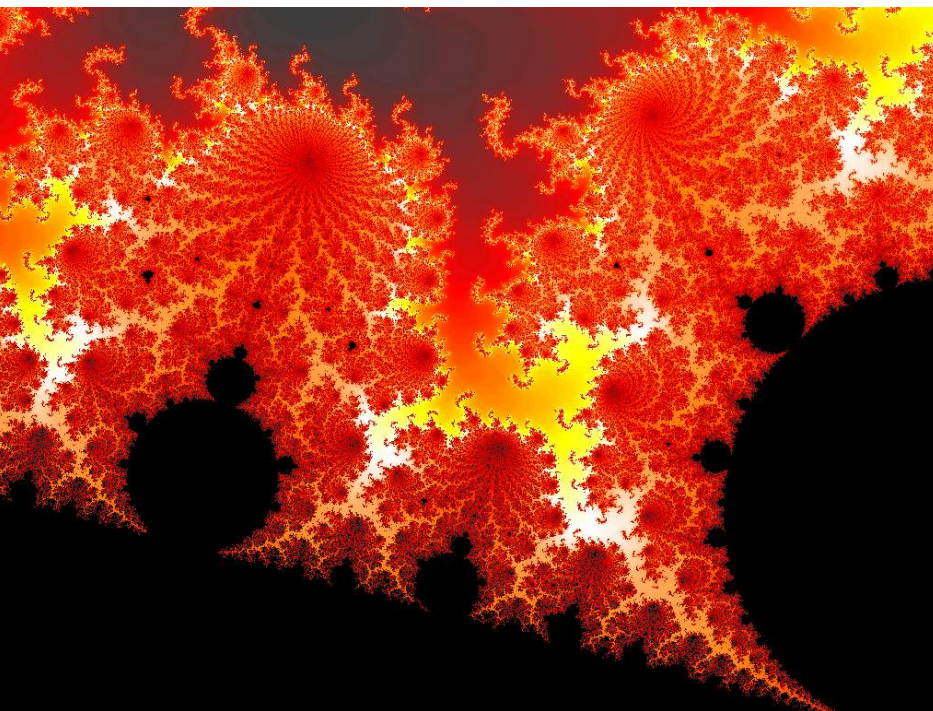Prof. David August

1

# The Usual Suspects

Me:   Prof. David August, 221 CS Building
        august@, 258-2085
        Office Hours:  Tu/Th after class and by appointment

TA:    Heejin Ahn, 226 CS Building
        heejin@
        Office Hours: MF 3-4PM and by appointment

        Fanglu Liu, 317 CS Building
        fanglul@
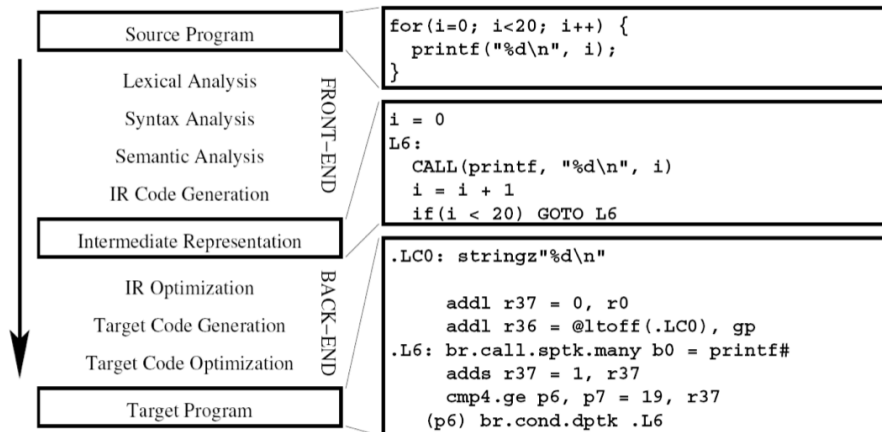        Office Hours: MF 9-10AM and by appointment

2

# What is a Compiler?

- A *compiler* is a program that takes a program written in a *source language* and translates it into a functionally equivalent program in a *target language*.

- Source Languages: C, C++, Swift, FORTRAN,...

- Target Languages: x86 Assembly, Arm, Assembly, C,...

- Compiler can also:
  - Report errors in source
  - Warn of potential problems in source
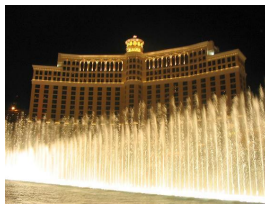  - Optimize program

# What is a Compiler?

```
                                    for(i=0; i<20; i++) {
  Source Program                       printf("%d\n", i);
                                    }
  Lexical Analysis
  Syntax Analysis      FRONT-END    i = 0
  Semantic Analysis                 L6:
                                       CALL(printf, "%d\n", i)
  IR Code Generation                   i = i + 1
                                       if(i < 20) GOTO L6
  Intermediate Representation
                                    .LC0: stringz"%d\n"
  IR Optimization      BACK-END
                                          addl r37 = 0, r0
  Target Code Generation                  addl r36 = @ltoff(.LC0), gp
                                    .L6:  br.call.sptk.many b0 = printf#
  Target Code Optimization                adds r37 = 1, r37
                                          cmp4.ge p6, p7 = 19, r37
  Target Program                          (p6) br.cond.dptk .L6
```

# Why Learn About Compilers?

**Compiler technology everywhere.**

- C++ → Assembly
- Assembly → Machine Code
- Microcode → microcode binary
- Interpreters: Perl, Python, Java, ...
- JITs: Android Dalvik VM, Java VM, ...

Bellagio, Las Vegas

- Publishing: Latex → PDF → Print on Paper
- Hardware Design: HW Description → Circuit/FPGA
- SPAM → /dev/null
- Automation: Water Fountain DL → Water Display
- Next Revolution in Processors

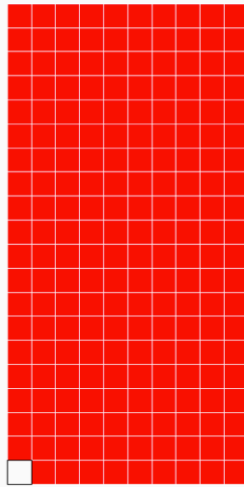**Almost all code goes through a compiler.**

**Linux**

- C = 2,558,100 lines
- x86 assembly = 12,164 lines

**99.5% of Linux source goes through a compiler!**

**Compilers teach us about:**

- Programming Languages
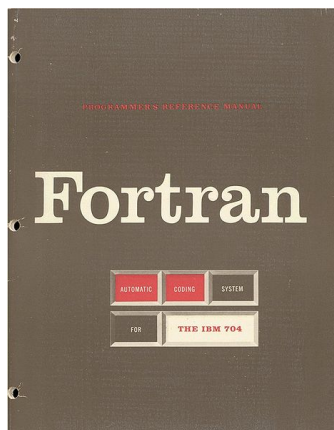- Computer Architectures

7

```
sum = 0;
for(i = 0; i < 1000000; i++)
  {
    sum = sum + big_array[i];
  }
```

```
sum = 0;
for(i = 0; i < 250000; i+=4)
  {
    sum = sum + big_array[i];
    sum = sum + big_array[i+1];
    sum = sum + big_array[i+2];
    sum = sum + big_array[i+3];
  }
```

8

- IBM developed the first FORTRAN compiler in 1957

- Took 18 person-years of effort

- You will be able to do it in less than a week!

**Fortran**

AUTOMATIC    CODING    SYSTEM

FOR    THE IBM 704

9

## Why Learn About Compilers?
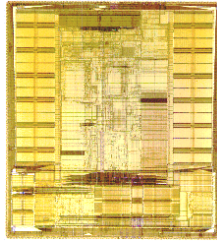## Hardware Design
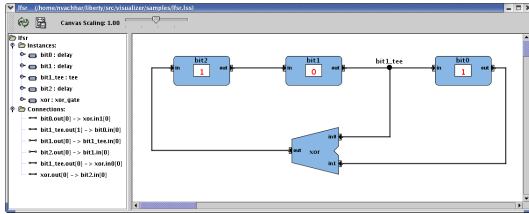


```verilog
module toplevel(clock,reset);
  input clock;
  input reset;

  reg flop1;
  reg flop2;

  always @ (posedge reset or posedge clock)
    if (reset)
      begin
        flop1 <= 0;
        flop2 <= 1;
      end
    else
      begin
        flop1 <= flop2;
        flop2 <= flop1;
      end
endmodule
```
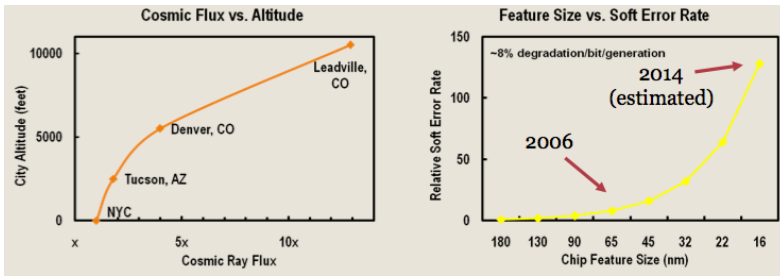
10

## Why Learn About Compilers?
## Computer Architecture



### Princeton Research on Fault Tolerance wins CGO Test of Time Award

*February 2, 2015*

Every year, the International Symposium on Code Generation and Optimization (CGO) recognizes the paper appearing 10 years earlier that is judged to have had the most impact on the field over the intervening decade.  This year at CGO 2015, the paper entitled "SWIFT: Software Implemented Fault Tolerance" by George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, and David I. August won the award.  The paper originally appeared at CGO 2005 and also won the best paper award that year at the conference.  Congratulations to Princeton's Liberty Research Group for winning this prestigious award!
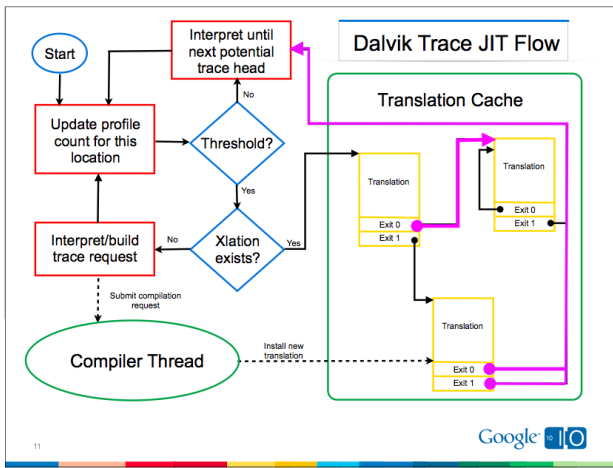
11

## Why Learn About Compilers?

Your chosen field of computer architecture effectively dead?
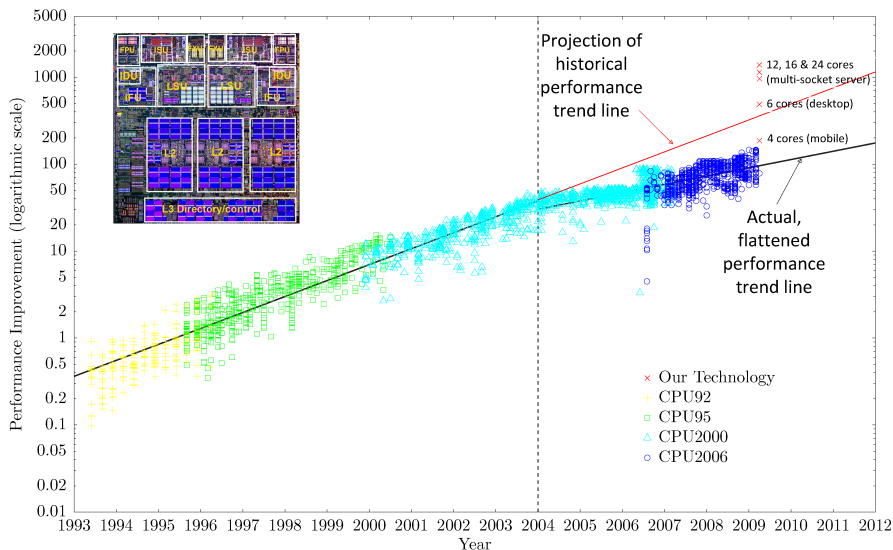


12

# Why Learn About Compilers?



Dalvik Trace JIT Flow

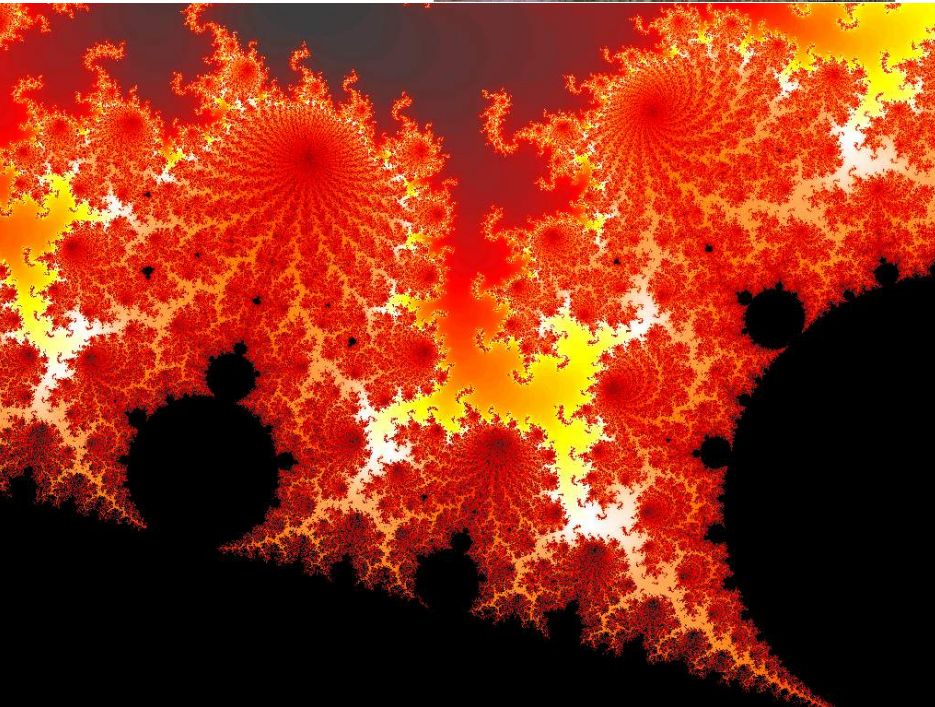13

# Why Take 320 Seriously?



Ge Wang

ChucK : Strongly-timed, Concurrent, and On-the-fly Audio Programming Language

Welcome to ChucK!

14

# Grading

| | |
|---|---|
| Assignments | 50% |
| Exams | 50% |
| Quizzes | Extra Credit |
| Participation | Extra Credit |

# Project

Build an optimizing compiler
- Front end
  - Lexer
  - Parser
  - Type Checker
  - Code Generator
- Back End Optimization
  - Superblock formation
  - Profiling
  - ILP Optimization

# Special Project

- Something else

# Exams

- Exams cover concepts presented in the lecture material, homework assignments, and/or required readings
- One double sided 8.5x11 page of notes allowed

## Midterm Exam

- Thursday before break
- In class

## Final Exam

- The final exam will be cumulative, three hours in length
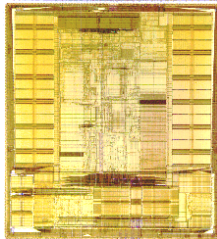- Time/Place determined by the Registrar

# Pick a number 1,2,3
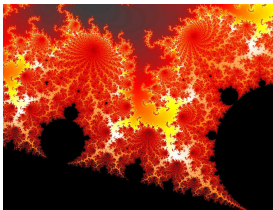


If the random number is the picture of a "processor", then we have a quiz.

# Quizzes

- Chance quiz at the beginning of each Tuesday class
- Not intended as a scare tactic – liberally graded
- Helps assess progress of class
- Just one question usually

# Participation

<span style="color:red">Negatives</span>
- Class disruptions (snoring, email, reading a book, etc.)
- Mistreatment of TAs

<span style="color:green">Positives</span>
- Contribute questions and comments to class
- Participate in discussions
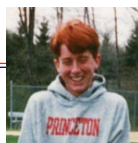- Feedback
- Stop by office hours to introduce yourself

# Reading

- Optional: Jeffrey D. Ullman, Elements of ML Programming, 2nd Edition, Prentice Hall.

- Required: Andrew W. Appel, Modern Compiler Implementation in ML. Cambridge University Press.

- CHECK ERRATA ON BOOK WEB SITES!

- Course Web Page – Off of CS page
  - Lecture Notes
  - Project Assignments
  - Course Announcements
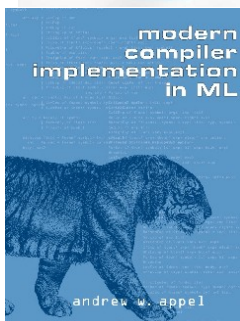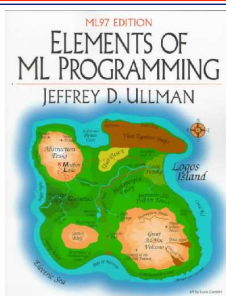
# Who Am I?

At Princeton (Computer Science, 1999-Present):
- Professor
- Compiler and computer architecture research
- Liberty Research Group

Education (Ph.D. in 2000):
- Ph.D. Electrical Engineering from University of Illinois
- Thesis Topic: Predication
- The IMPACT Compiler Research Group

# Who Am I?

Professional Experience:

- Intel (Oregon) – P6 multiprocessor validation
- Hewlett-Packard (San Jose, CA) – research compiler
- Intel (Santa Clara, CA) – IA-64 design
- Startups inspired by compiler technology
- Consulting for Intel, Lucent, Google, etc.

# Our Pledge to You

- Quick response to questions and issues
- Reasonable late policy
  - Up to 3 days late for any single assignment without penalty
  - Up to 7 days late total across all assignments
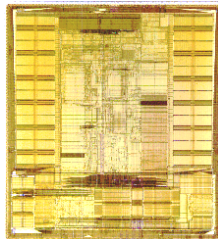  - Contact me prior to deadline for special circumstances
- Fast turn-around on grading

**END OF ADMINISTRATIVE STUFF**
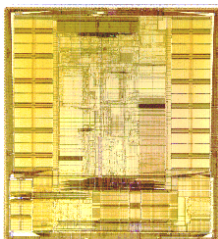
# It's Tuesday: Pick a number 1,2,3
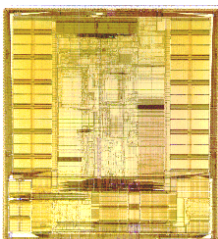
Front:

1. Full name and Email Address <u>above the red line</u>
2. Major/UG or G/Year (immediately below the red line)
3. Area (G: Research Area/UG: Interests)
4. Briefly describe any ML experience.
5. Briefly describe any C/C++ experience.
6. Briefly describe any compiler experience.
7. In which programming languages are you fluent?

Back:

1. Why do processors have registers?
2. What is an instruction cache?
3. Can one always convert an NFA to a DFA? (yes, no, or wha?)

28