Programming Victory

Computational problem solving ubiquitous

HW + SW for fast, reliable, cheap systems

Skilled workforce of ~ 18.5 million (2014)
If you have a hammer...

“We become what we behold. We shape our tools and then our tools shape us.”

Father John Culkin and Marshall McLuhan
Programming Hammers

Integers
Trees
Hashing
Caching
Programming Hammers

Integers       Modularity
Trees          Testing
Hashing        Proof (?)
Caching        Community
Programming Hammers

Integers*

Trees

Hashing

Caching

Modularity

Community

Towards Optimization-Safe Systems: Analyzing the Impact of Undefined Behavior

Xi Wang, Nickolai Zeldovich, M. Frans Kaashoek, and Armando Solar-Lezama

MIT CSAIL

Abstract

This paper studies an emerging class of software bugs called optimization-unstable code: code that is unexpectedly discarded by compiler optimizations due to undefined behavior in the program. Unstable code is present in many systems, including the Linux kernel and the PostgreSQL database. The consequences of unstable code range from incorrect functionality to missing security checks. To reason about unstable code, this paper proposes a novel model which views unstable code in terms of...
Programming Hammers

Integers* $\Rightarrow$ Modularity
Missing and Misused

Not everything is a nail. What tools are we missing?

Tools require skill. Make user friendlier?
Representative Examples

Floating Point

3D Printing

Robotics

\[-b + \sqrt{b^2 - 4ac} \]

\[2a\]
These Tools Matter

FP: research, global policy, markets

3DP: means of production, medicine

Robo: elderly care, integrated workforce
Goal: Democratize

Today only a few experts can effectively build these systems. This quarter we want to study what about the abstractions in these domains prevents a more diverse group of programmers from working in such spaces and what we can do about it.
599Z

Lofty goals, friendly discussions.

~ 1.0 paper / meeting

https://homes.cs.washington.edu/~ztatlock/599z-17sp/

2 small exercises (FP, 3DP)

1 large project (related to your research!)
Floating Point
Floating Point’s Wild Success

Accuracy
for basic ops

\[ [x \ast y]_F = \text{Round}( [x \ast y]_R ) \]

Compute in $\mathbb{F}$
Round to $\mathbb{F}$
Compute in $\mathbb{R}$

Flexibility
vast range: $10^{-324}$ to $10^{308}$

Performance
cheap GFLOPS
Floating Point’s Wild Success
Floating Point’s Wild Success

Often floating point is close to real arithmetic

But not always!
WELCOME TO
THE SECRET
ROBOT INTERNET

Prove you are human:

0.1 + 0.2 = ?

0.30000000000000004
Rounding Error in Quadratic

\[ \frac{-b + \sqrt{b^2 - 4ac}}{2a} \]
Rounding Error in Quadratic

\[- \frac{b + \sqrt{b^2 - 4ac}}{2a}\]

What is rounding error?

exact \[ [e]_R \]

computed \[ [e]_F \]

7 ULPs
Rounding Error in Quadratic

\[-b + \sqrt{b^2 - 4ac}\]

\[\frac{\text{What is rounding error?}}{2a}\]

What is rounding error?

exact \[\lceil e \rceil_R\]

computed \[\lceil e \rceil_F\]

7 ULPs

log(ULPs) estimates # of incorrect bits
Rounding Error in Quadratic

\[-b + \sqrt{b^2 - 4ac}\]

ULPs provide nice error measure:
- accounts for distribution of \( \mathbb{F} \)
- fast to compute

\[\text{ulps}(f_1, f_2) \approx |((\text{uint64}) f_1) - ((\text{uint64}) f_2)|\]

\[\log(\text{ULPs}) \text{ estimates } \# \text{ of incorrect bits}\]
Rounding Error in Quadratic

\[-b + \sqrt{b^2 - 4ac} \over 2a\]

\[\text{log(ULPs)}\]

\[b\]
Rounding Error in Quadratic

\[-b + \frac{\sqrt{b^2 - 4ac}}{2a}\]
Rounding Error in Quadratic

\[-b + \sqrt{b^2 - 4ac} \over 2a\] 

Overflow

If $b$ is large, $\lceil b^2 \rceil_F$ overflows and the whole expression returns $\infty$. 

\[\text{Overflow}\]
Rounding Error in Quadratic

\[ \frac{-b + \sqrt{b^2 - 4ac}}{2a} \]

\[ \begin{cases} \frac{c}{b} - \frac{b}{a} & \text{if } b \in A \end{cases} \]

Pretty Accurate
Rounding Error in Quadratic

\[ \frac{-b + \sqrt{b^2 - 4ac}}{2a} \]

\[ \Rightarrow \begin{cases} \frac{c}{b} - \frac{b}{a} & \text{if } b \in A \\ \frac{-b + \sqrt{b^2 - 4ac}}{2a} & \text{if } b \in B \end{cases} \]

**Catastrophic Cancellation**

If \( b \) is large, but \( a \) and \( c \) are small, \( b \approx \sqrt{b^2 - 4ac} \) and the difference is rounded off.
Rounding Error in Quadratic

\[ \frac{-b + \sqrt{b^2 - 4ac}}{2a} \]

\[ \begin{cases} \frac{c}{b} - \frac{b}{a} & \text{if } b \in A \\ \frac{-b + \sqrt{b^2 - 4ac}}{2a} & \text{if } b \in B \\ \frac{2c}{-b - \sqrt{b^2 - 4ac}} & \text{if } b \in C \end{cases} \]

Overflow again

\[ \log(ULPs) \]
Rounding Error in Quadratic

\[-b + \sqrt{b^2 - 4ac} \over 2a\]

\[\begin{aligned}
&\frac{c}{b} - \frac{b}{a} \\
&-\frac{b + \sqrt{b^2 - 4ac}}{2a} \\
&\frac{2c}{b - \sqrt{b^2 - 4ac}} \\
&-\frac{c}{b}
\end{aligned}\]

if \(b \in A\)

if \(b \in B\)

if \(b \in C\)

if \(b \in D\)
Rounding Error in Sculpture

Blake Courter
@bcourter
Rounding Error in Sculpture

Blake Courter
@bcourter

Rounding error
Rounding Error Impact

Numerous articles retracted [Altman 99, 03]

Financial regulations [Euro 98]

Market distortions [McCullough 99, Quinn 83]

How bad is it? No one knows, but it’s not getting any better.

-- Bill Kahan (approx)
**Options Today**

- **Futz**
  - Easy
  - Fast
  - Unreliable

- **Big Float**
  - Easy
  - More Reliable
  - Really Slow

- **Analyze**
  - Reliable
  - Fast
  - Difficult*
3D Printing
3D Printing: Industrial Origins

30 years of active development:
- focus on rapid prototyping
- diverse tech: SLS, SL, OJ, FFF
- sophisticated tooling + control
3D Printing: Industrial Origins

+ Reliable
+ Quality
3D Printing: Industrial Origins

+ Reliable
+ Quality
- Price
3D Printing: Industrial Origins

+ Reliable
+ Quality
- Price
- Expertise
3D Printing: Industrial Origins

+ Reliable
+ Quality
- Price
- Expertise
- Patented
3D Printing: Industrial Origins

+ Reliable
+ Quality
- Price
- Expertise

 $$$
 $$$
 $$$

Patented
3D Printing: Desktop Market

Rapidly improving space:

- open source: RepRap, Marlin
- commercial: MakerBot, Ultimaker
- prototyping, final parts (?)
3D Printing: Desktop Market

-printed :)
3D Printing: Desktop Market

+ Price
3D Printing: Desktop Market

+ Price
- Expertise
- Reliability
- Quality
Industrial Desktop: Reliability / Quality vs. Affordability

Incarnate

Desktop
Idea

3D Printing Workflow

Physical Part
1. Design

2. Slice

Idea

CAD

module snowman(scale, armAng) {
  rs = [scale, scale / 1.6, scale / chopBase(0.65 * rs[0]) {
    sphere(r = rs[0]);
    translate([0, 0, 0.85 * (rs[0] + armAng)];
    sphere(r = rs[1]);
    translate([0, 0, 0.85 * (rs[1] + armAng)];
    sphere(r = rs[2]);
    translate([0, 0, 0.8] + hat(scale));
    scale, armAng);
  } arm(scale, armAng);
}

STL

G1 X97.097 Y100.000 F6000
G1 X0.000 Y2400
G0 Z0
G1 X97.239 Y99.101 Z0.0136 F412
G1 X97.651 Y98.294 Z0.0272
G1 X98.264 Y99.561 Z0.0408
G1 X99.063 Y97.239 Z0.0544
G1 X100.000 Y97.097 Z0.0680
G1 X100.097 Y97.239 Z0.0816
G1 X101.706 Y97.651 Z0.0952
G1 X102.349 Y98.294 Z0.1088
G1 X102.761 Y99.103 Z0.1223
G1 X102.903 Y100.000 Z0.1359
G1 X102.761 Y100.097 Z0.1495

GCODE
1. Design

2. Slice

3. Print

Idea → 1. Design → CAD → STL → 2. Slice → Simplify3D → GCODE → 3. Print → Physical Part
1. Design

2. Slice

3. Print

4. OK?

Idea

Physical Part

CAD

STL

GCODE

SolidWorks

Simplify3D

SAILFISH

Marlin
1. Design

2. Slice

3. Print

4. OK?

Idea

iterate

Physical Part

1. Design

2. Slice

3. Print

4. OK?

GCODE

SAILFISH

Marlin
Do you want to build a snowman?

1. Design
2. Slice
3. Print
4. Check
Next Meeting

Design: “worse is better” and hierarchy

Dreamsongs

Worse Is Better
Richard P. Gabriel

The concept known as "worse is better" holds that in software making (and perhaps in life) it is better to start with a minimal creation and grow it as needed. Christopher Alexander called this "piecement growth." This is the story of the evolution of that concept.

This is the crux of the essay: The Rise of Worse is Better.

From 1984 until 1994 I had a Lisp company called "Lucid, Inc." In 1989 it was not doing well. The Lisp business was not going well, partly because the AI companies were floundering around. A small group of AI companies were starting to blame Lisp and its implementations for the failure. In early 1989, I was sitting on the Lucid porch with some of the hackers, and somebody thought people believed C and Unix were better than Lisp. I jokingly answered, "Oh, better." We laughed over it for a while as I tried to make up an argument for why it could be good.

A few months later, in Summer 1989, a small Lisp conference called LISP'89 ("Lisp the Practical Applications of Lisp") invited me to give a keynote, probably since I was running a Lisp company. I agreed, and while writing about what to talk about, I gravitated toward the explanation of the worse-is-better ideas we joked about as applied to Lisp. At Lucid we had a plan for how we would do Lisp ever to survive business realities as we saw them, and so I wrote "Lisp: Good News, Bad News, How to Win Big." [Link] (slightly abridged version, with details on the development and delivery of the Lisp applications).

I gave the talk in March 1990 at Cambridge University. I had never been to Cambridge, and I was quite nervous about speaking at a university level. There were about 500 people in the auditorium, and before my talk they played the Notting Hillbillies over the sound system and heard the group before. Then, indeed, the audience was not released in the US appropriate because I had decided to use a very colloquial American-style of writing. Notting Hillbillies played a style of music heavily influenced by traditional American music, as well as folk music from other countries. The use of these styles was appropriate because the audience had just heard them play the same music before. I did not want to use a more formal style because it would have been inappropriate in the context. Instead, I decided to use a more colloquial style because it would have been more appropriate.

...