Optimizations

Lecture 21

How to Get Rid of a Mouse

Optimal mouse removal technique

April 2, 2018
Linking Wrap up

- To support multiple compilation units, the compiler must link units together to produce a single binary
  - If main.c uses sqrt() defined in math.c, compiler must know about sqrt() when compiling the whole program

- Sometimes we refer to linking as link-time (cf. compile-time...)

Compiler Construction
Linking wrap up

Lots of tables used in Linking:

- Import table: what things does *this* compilation unit need that comes from someone else?
- Export table: what things does *this* compilation unit provide that someone else can use?
- Relocation table: where in *this* compilation unit are symbols referenced?
Big Link Example
Big Link Example (Answers)

```
<table>
<thead>
<tr>
<th>Relocatable object files</th>
<th>Executable object file</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Imports M</td>
<td>Imports X</td>
</tr>
<tr>
<td>M</td>
<td>Exports M</td>
</tr>
<tr>
<td>M</td>
<td>Relocation</td>
</tr>
<tr>
<td>Exports X</td>
<td>Code</td>
</tr>
<tr>
<td>Relocation</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>r1 := &amp;M (2300)</td>
<td></td>
</tr>
<tr>
<td>call M</td>
<td>call M (2300)</td>
</tr>
<tr>
<td>Data</td>
<td>Code</td>
</tr>
<tr>
<td>X:</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>Y:</td>
<td></td>
</tr>
</tbody>
</table>
```

---

Compiler Construction
Static vs. Dynamic Linking

- **Static linking**: copy all library (imported) code into one big binary
  - Benefit: portability
  - Drawback: code size, waste

- **Dynamic linking**: lazily load library code at runtime from one shared system copy
  - Benefit: smaller code, modular programs
  - Drawback: portability, security
Optimization Overview

- Maximize the value of some *objective function*
Optimization Overview

- Maximize the value of some *objective function*
- Execution time
  - Does the resulting program execute fast enough?
Optimization Overview

- Maximize the value of some *objective function*
- Execution time
  - Does the resulting program execute fast enough?
- Compilation time
  - Does the compiler produce code fast enough?
Optimization Overview

- Maximize the value of some *objective function*
- Execution time
  - Does the resulting program execute fast enough?
- Compilation time
  - Does the compiler produce code fast enough?
- Code size
  - Can the resulting code fit in a low-capacity embedded device?
Optimization Overview

- Maximize the value of some *objective function*
- Execution time
  - Does the resulting program execute fast enough?
- Compilation time
  - Does the compiler produce code fast enough?
- Code size
  - Can the resulting code fit in a low-capacity embedded device?
- Power consumption
  - Will my code make datacenter owners happy?
Optimization Overview

- Maximize the value of some *objective function*
- Execution time
  - Does the resulting program execute fast enough?
- Compilation time
  - Does the compiler produce code fast enough?
- Code size
  - Can the resulting code fit in a low-capacity embedded device?
- Power consumption
  - Will my code make datacenter owners happy?

**Remember** Correctness must never be implicated by optimizations
Pareto Optimization

- Optimization involves tradeoffs
PA6 Optimizations

- Your objective is to minimize execution time or cache misses

(this bike is not optimized for speed)
Optimizations to consider

- Things we have covered
  - Constant folding/propagation
  - Liveness analysis/dead code elimination
  - Register allocation

- Things to consider for PA6
  - Unboxing
  - Copy propagation
  - Function inlining
  - Code motion/hoisting
  - Loop invariants and unrolling
  - Common subexpression elimination
  - Static Single Assignment (SSA)
  - Peephole optimizations

- Others to think about (but not for PA6)
  - Whole-program and Link-time optimizations
  - Optimizations performed by hardware
Unboxing

$8000 Robot Unboxing! - YouTube
https://www.youtube.com/watch?v=OzrnNHfJ3SP_k
May 22, 2015 - Uploaded by Unbox Therapy
Wanna see more with Nao the robot? FOLLOW ME IN THESE PLACES FOR UPDATES Twitter - http://twitter ...

Surprise Eggs unboxing toys - YouTube
https://www.youtube.com/user/UnboxingSurpriseEgg
http://www.youtube.com/subscription_center?add_user=UnboxingSurpriseEgg Stop Motion Play Doh videos, Kinder surprise eggs, Hello Kitty, Disney princess ...

Unboxing LEGO Dimensions! - YouTube
https://www.youtube.com/watch?v=4xccV7lo6u8
Oct 4, 2015 - Uploaded by Unbox Therapy
Sponsored by WB Games Get the game here: http://bit.ly/Gel-LEGODimensions FOLLOW ME IN THESE ...

The World's Largest Unboxing Video - YouTube
https://www.youtube.com/watch?v=w9Ia0oOFyZY
Feb 5, 2014 - Uploaded by Unbox Therapy
Click here to subscribe - http://bit.ly/SubUnbox FOLLOW ME IN THESE PLACES Twitter - http://twitter.com ...

Unboxing the 2015 Grammy - YouTube
https://www.youtube.com/watch?v=fMCwii6MZcY
Apr 29, 2015 - Uploaded by alyankovic
And only he would do an unboxing video for his Grammy. He probably ... Weird Al is now doing parodies of ...

Unboxing Anki Overdrive - YouTube
https://www.youtube.com/watch?v=P9R039SYow
Nov 28, 2015 - Uploaded by Unbox Therapy
also do a unboxing and review of the parrot bebop 2 with ... a sickness that makes you not be able to unbox ...

iPad Pro UNBOXING!!! - YouTube
https://www.youtube.com/watch?v=6KQ9U55Wd7s
Check out the new iPad Pro! What do you think? (sorry had to
Boxed types

Why doesn’t this work?

class Derp {
    public static void main (String[] args) {
        HashMap <String, int> Herp = new HashMap<String, int>();
    }
}
Boxed types

That’s right, primitives aren’t Objects

```java
class Derp {
    public static void main (String[] args) {
        HashMap <String, Integer> Herp = new HashMap<String, Integer>();
    }
}
```
Cool Objects

- Every Object is boxed
  - Produces waste
  - In Cool ASM, adding 2 boxed Ints takes 11 operations
Cool Objects

Every Object is boxed

- Produces waste
- In Cool ASM, adding 2 boxed Ints takes 11 operations
  - 3 memory references to unbox each operand
  - 1 actual addition
  - 1 constructor call for result
  - 3 more memory references for storing result in new object
Cool Objects

- Every Object is boxed
  - Produces waste
  - In Cool ASM, adding 2 boxed Ints takes 11 operations
    - 3 memory references to unbox each operand
    - 1 actual addition
    - 1 constructor call for result
    - 3 more memory references for storing result in new object
  - That is just awful
Static Heuristic

- If there aren’t any calls to `(new Int).abort()`...
  - Should be pretty safe to keep integers unboxed.
  - Nothing can inherit from Int, Boolean, or String.
Static Heuristic

▶ If there aren’t any calls to (new Int).abort()...
  ▶ Should be pretty safe to keep integers unboxed.
  ▶ Nothing can inherit from Int, Boolean, or String.
▶ Also need to be careful of situations like case where you might be given an integer
Wastefulness of Boxing

Assume \( x=5, \ y=6, \ z=0; \)
\[ z <- x + y \]
\[ z <- z + 5 \]

What operations must happen?
Wastefulness of Boxing

Assume \( x=5 \), \( y=6 \), \( z=0 \);

\[
\begin{align*}
z &\leftarrow x + y \\
z &\leftarrow z + 5
\end{align*}
\]

What operations must happen?

1. unbox \( x \) in to register
2. unbox \( y \) in to register
3. Add results into a register
4. Invoke integer constructor (for \( z \))
5. move result value into new integer’s field
“Peephole” Heuristic

- We can just add TAC instructions to represent boxing and unboxing patterns
  - box_int
  - unbox_int
“Peephole” Heuristic

Assume \( x = 5, \ y = 6, \ z = 0; \)
\[ z <- x + y; \]
\[ z <- z + 5 \]

What operations must happen?

1. \( t1 = \text{unbox}(x) \)
2. \( t2 = \text{unbox}(y) \)
3. \( t3 = + t1 \ t2 \)
4. \( z = \text{box}(t3) \)
5. \( t5 = \text{unbox}(z) \)
6. \( t6 = \text{box}(5) \)
7. \( t7 = \text{unbox}(t6) \)
8. \( t8 = + t5 \ t7 \)
9. \( z = \text{box}(t8) \)
“Peephole” Heuristic

Assume \( x = 5, \ y = 6, \ z = 0; \)
\[
z \leftarrow x + y
\]
\[
z \leftarrow z + 5
\]

What operations must happen?

1. \( t1 = \text{unbox}(x) \)
2. \( t2 = \text{unbox}(y) \)
3. \( t3 = + \ t1 \ t2 \)
4. \( z = \text{box}(t3) \leftarrow \)
5. \( t5 = \text{unbox}(z) \leftarrow \)
6. \( t6 = \text{box}(5) \leftarrow \)
7. \( t7 = \text{unbox}(t6) \leftarrow \)
8. \( t8 = + \ t5 \ t7 \)
9. \( z = \text{box}(t8) \)
More Aggressive Approaches?

- You could assume all booleans and integers are unboxed
- Only box them when you absolutely need an object (like dispatch)
- You will need special checking code to handle odd cases (integers passed as parameters)
  - Remember your type tag? Maybe you could use it for something...
  - Also, what about data sizes and the target architecture?
Dataflow Analysis: Reminder

- Dataflow analysis is a collection of techniques for compile-time reasoning about the run-time flow of values.

- Local dataflow analysis: one basic block.

- Global dataflow analysis: entire CFG of a whole method.

- You have already seen specific examples of dataflow analysis: dead code elimination, register allocation, constant propagation.
General Dataflow Analysis

Basic idea

- Setting up and solving systems of equations that relate information at various points in a program
  - this is an iterative process

- Desired result is usually *meet over all paths* solution
  - “What is true on every path from the entry?”
  - “Can this happen on any path from the entry?”
Iterative Algorithms

- First, compute some local information within individual basic blocks
- Then, propagate local information along control flow edges
  - $\text{IN}(B)$: some property on entry to basic block $B$
  - $\text{OUT}(B)$: some property on exit from basic block $B$
  - Need to iterate until no changes

```plaintext
while change do
  change = false
  for each basic block
    apply equations updating $\text{IN}$ or $\text{OUT}$
    if $\text{IN}/\text{OUT}$ changes, set change to true
end
```