Final Review Examples and Compiler in Practice

Wei Le

2014.11
Object Layout in Memory

The first 3 words of Cool objects contain header information:

<table>
<thead>
<tr>
<th>Class Tag</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Size</td>
<td>4</td>
</tr>
<tr>
<td>Dispatch Ptr</td>
<td>8</td>
</tr>
<tr>
<td>Attribute 1</td>
<td>12</td>
</tr>
<tr>
<td>Attribute 2</td>
<td>16</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
## Object Layout in Memory

<table>
<thead>
<tr>
<th>Offset</th>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Atag</td>
<td>Btag</td>
<td>Ctag</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Dispatch Table

<table>
<thead>
<tr>
<th>Offset</th>
<th>0</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>fA</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>fB</td>
<td>g</td>
</tr>
<tr>
<td>C</td>
<td>fA</td>
<td>h</td>
</tr>
</tbody>
</table>

- The dispatch table for class A has only 1 method
- The tables for B and C extend the table for A to the right
- Because methods can be overridden, the method for f is not the same in every class, but is always at the same offset
Global Constant Propagation

\[ \begin{align*}
X & := 3 \\
B & > 0 \\
Y & := Z + W \\
X & := 4 \\
A & := 2 \times X \\
Y & := 0 \\
X & = * 
\end{align*} \]
Global Constant Propagation

\[ X := 3 \]
\[ B > 0 \]
\[ Y := Z + W \]
\[ X := 4 \]
\[ A := 2 \times X \]
\[ Y := 0 \]
\[ X = \ast \]
\[ X = 3 \]

Diagram:

- \( X := 3 \)
- \( B > 0 \)
- \( Y := Z + W \)
- \( X := 4 \)
- \( A := 2 \times X \)
- \( Y := 0 \)
- \( X = \ast \)
- \( X = 3 \)
Global Constant Propagation

\[ X := 3 \]
\[ B > 0 \]
\[ Y := Z + W \]
\[ X := 4 \]
\[ A := 2 \times X \]
\[ Y := 0 \]
\[ X = * \]
\[ X = 3 \]
\[ X = 3 \]
Global Constant Propagation

Variables and Equations:
- \( X := 3 \)
- \( Y := Z + W \)
- \( X := 4 \)
- \( B > 0 \)
- \( Y := 0 \)
- \( A := 2 \times X \)
- \( X = * \)
- \( X = 3 \)
- \( X = 3 \)
- \( X = 3 \)
Global Constant Propagation

- $X := 3$
- $B > 0$
- $Y := Z + W$
- $X := 4$
- $A := 2 \times X$
- $Y := 0$
- $X = *$
- $X = 3$
- $X = 3$
- $X = 3$
- $X = 4$
Global Constant Propagation

- $B > 0$
- $X := 3$
- $Y := Z + W$
- $X := 4$
- $A := 2 \times X$
- $Y := 0$
- $X = *$
- $X = 3$

Diagram:

- $X = 3$ to $X := 4$
- $X = 3$ to $Y := 0$
- $X = 3$ to $Y := Z + W$
- $X = 3$ to $X := 3$
- $X = *$ to $A := 2 \times X$
- $X = *$ to $X := 3$
- $X = 3$ to $X = *$
- $X = 3$ to $X := 3$
Global Constant Propagation

\begin{align*}
X &:= 3 \\
Y &:= Z + W \\
X &:= 4 \\
A &:= 2 \times X \\
Y &:= 0 \\
B &> 0
\end{align*}
Compiler in Practice

- Architecture types
- Source languages
- Intermediate code
- Code generation and optimization
The Sun SPARC Compilers

- C, C++, Fortune 77, Pascal
- C, C++, ANSI standard AT&T specification of the language
- Fortune 77: DEC and Cray compatibility
- Pascal: ANSI standard with Apollo compatibility

History
- Berkley 4.2 BSD unix software distribution
- Developed by Sun since 1982
- Global optimization 1984
- Inter-procedural and parallelization 1989
Other Compilers

- the IBM XL compilers for the Power and Power PC architecture: PL/8, C, Fortran 77, Pascal, and C++
- Digital Equipments compilers for Alpha architecture (GEM compiler)
- The Intel Reference Compilers for the Intel 386 Architecture Family: C, C++, Fortran 77, Fortran 90
## Optimizations for C

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Sun SPARC</th>
<th>IBM XL</th>
<th>DEC GEM</th>
<th>Intel 386 family</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant propagation of kind</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>dead-code elimination</td>
<td>almost all</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>loop-invariant code motion</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>strength reduction of height</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>reduction of area computation</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>loop unrolling factor</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>none</td>
</tr>
<tr>
<td>rolled loop</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>register allocation</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>instruction scheduling</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>stack frame eliminated</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>tail call optimized</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
GCC compiler fuzzers – generate random C programs (C99)

- Csmith
- ccg
- icfuzz (Intel internal tools)
GCC bugs

<table>
<thead>
<tr>
<th></th>
<th>crashes</th>
<th>timeouts</th>
<th>wrong code</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCC 2.7.2.3</td>
<td>13866</td>
<td>68</td>
<td>239</td>
</tr>
<tr>
<td>GCC r202341</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Clang r190166</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

Here's an example of a program that, when compiled by GCC 2.7.2.3 at -O2, causes the compiler to run for more than three minutes (in fact, it runs for more than 60 minutes — at which point I got bored and killed it):

```c
1 int x0;
2 int x1() {
3     for (;;) {
4         x0 = 0;
5         for (; x0;) {
6             for (; x0;)
7                 int x2 = 1;
8                 if (x2) return 0;
9             }
10         }
11     }
```