The Quest for Cheaper Variable Tracking in GCC

Alexandre Oliva

aoliva@redhat.com

http://identi.ca/lxoliva/

http://people.redhat.com/~aoliva/

GNU Tools Cauldron, 2012

Copyright 2012 Red Hat, Inc. This work is licensed under the Creative Commons CC BY-SA 3.0 Unported License.
Brain Dump Header

- Variable Tracking
- Micro Operations
- Dataflow Analysis
- Location Output
- Emitting Location Notes
- Multiple Locations
- Dataflow Confluence Operation

```
/dev# /sbin/dump brain
Warning: brain is in use, the dump may be inconsistent.
SIGPIPE Error: stdout: nobody is listening.
```
Variable Tracking

- VTA: Annotate gimple-reg ASSIGNs & PHIs
  - Automatic non-addressable variables
    \[ T_v = \text{min}; \quad // \ i \Rightarrow \text{min}; \]
    \[ (\text{var\_location } i (\text{reg } \#)) \]

- VT: Annotate REGs and MEMs with EXPRs
  - Global and automatic addressable variables
    \[ (\text{set } (\text{reg } \# [T]) (\text{mem } (?) [\text{min}])) \]
Micro Operations

- Preprocess insns into atomic operations:
  
  \[
  \text{(set (reg 2 [q]) (reg 1 [p]))}
  \]

  - USE (reg 1 [p])
  - SET (reg 2 [q])

<table>
<thead>
<tr>
<th>Var</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>(reg 4 [q])</td>
<td>(reg 1 [p])</td>
</tr>
<tr>
<td>q</td>
<td>(reg 2 [x])</td>
<td>(reg 2 [q])</td>
</tr>
<tr>
<td>x</td>
<td>(reg 2 [x])</td>
<td></td>
</tr>
</tbody>
</table>
Micro Operations (2)

- Other EXPR-based micro operations:
  - COPY adds a LOC to EXPR
  - USE_NO_VAR unbinds LOC
  - CLOBBERER resets EXPR’s locs too

- Call-related micro operations:
  - CALL clobbers MEMs and some REGs
  - ADJUST notes stack pointer changes
Micro Operations (3)

- VALUE-based micro operations:
  
  - VAL_USE (value 10) (reg 3)
  
  - VAL_SET (value 10) (mem (value 10))

  10 (reg 3), (mem (value 10))

- var_location i (reg 3))

- VAL_LOC i (value 10)

  i (value 10)
Dataflow Analysis

- Bind incoming arguments at ENTRY_BLOCK

- For each pending block, until convergence:
  - Combine confluent sets
    - Union of EXPR-based locations
    - Intersection of VALUE bindings
    - Canonicalization of equivalent VALUE
  - Process micro operations
# Dataflow Confluence

<table>
<thead>
<tr>
<th>p</th>
<th>(reg 1 [p])</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>(mem (value 10))</td>
</tr>
<tr>
<td>11</td>
<td>(reg 3)</td>
</tr>
<tr>
<td>i</td>
<td>(value 10)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>q</th>
<th>(reg 2 [q])</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>(value 13)</td>
</tr>
<tr>
<td>13</td>
<td>(value 10)</td>
</tr>
<tr>
<td>i</td>
<td>(value 13)</td>
</tr>
</tbody>
</table>

\[\downarrow\]
### Dataflow Canonicalization

<table>
<thead>
<tr>
<th>10</th>
<th>(value 13)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>(reg 5), (value 10), (value 15)</td>
</tr>
<tr>
<td>i</td>
<td>(reg 7), (value 13)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10</th>
<th>(reg 5), (reg 7), (value 13), (value 15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>(value 10)</td>
</tr>
<tr>
<td>15</td>
<td>(value 10)</td>
</tr>
<tr>
<td>i</td>
<td>(value 10)</td>
</tr>
</tbody>
</table>
Location Output

• For each block:
  – Compare current and incoming sets
    * Mark different LOCs/lists as changes
  – **Emit location notes** for changes
  – For each micro operation in the block:
    * Process it, marking changes as such
    * **Emit location notes** for changes
Emitting Location Notes

• Back-propagate changed VALUEs

• For each changed DECL:
  – Try each LOC, resolving VALUEs recursively
  – Add backlinks to used VALUEs
  – Assume tentative NO_LOC upon cycle
  – Back-notify upon LOC for tentative NO_LOC
  – Confirm remaining tentative NO_LOCs
Multiple Locations

- Variable live at e.g. both REG and MEM
- Currently **not** handled in VT or later passes
- Use PARALLELs with per-LOC expansions?
- Use RTL sharing to avoid explosion?
- Handling cycles needs different approach!
- Discard useless and too-large locations?
- Detect sub-permanence and equivalences?
Dataflow Confluence Operation

- Detect more equivalences for intersection
- Use equivalences from sets and cselib table
  - Regression: cselib locs removed from sets
- Revamp intersections:
  - Combine all sets at once (vs one at a time)?
  - Pull REG and CONST (vs DECL/VALUE)?
- Separate tables for VALUEs and DECLs?
Any Takers?

Thoughts?

aoliva@redhat.com

http://identi.ca/lxoliva/

Thank you!