redhat

The Quest for Cheaper Variable Tracking in GCC

Alexandre Oliva

aoliva@redhat.com

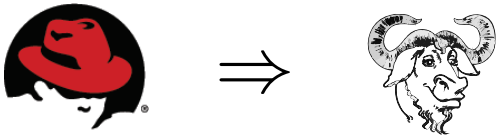
http://identi.ca/lxoliva/

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

GNU Tools Cauldron, 2012

Brain Dump Header

- Variable Tracking



- Micro Operations
- Dataflow Analysis
- Location Output

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

- Emitting Location Notes
- Multiple Locations
- Dataflow Confluence Operation

Variable Tracking

- VTA: Annotate gimple-reg ASSIGNs & PHIs
  - Automatic non-addressable variables

    T_v = min; // **i** ⇒ min;

    (var_location **i** (reg #))
- VT: Annotate REGs and MEMs with EXPRs
  - Global and automatic addressable variables

    (set (reg # [**T**]) (mem (?) [**min**]))

Micro Operations

- Preprocess insns into atomic operations:

  (set (reg 2 [**q**]) (reg 1 [**p**]))

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

Var	Before	After
p		(reg 1 [**p**])
q	(reg 4 [q])	(reg 2 [**q**])
x	(reg 2 [x])	

Micro Operations (2)

- Other EXPR-based micro operations:
  - COPY adds a LOC to EXPR
  - USE_NO_VAR unbinds LOC
  - CLOBBER 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:

  (set (mem (reg 3)) (reg 3))

  - 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

p	**(reg 1 [p])**
10	(mem (value 10))
11	(reg 3)
i	**(value 10)**

q	**(reg 2 [q])**
10	(value 13)
13	**(value 10)**
i	(value **13**)

p	(reg 1 [p])
q	(reg 2 [q])
i	(value 10)

Dataflow Canonicalization

10	(value 13)
13	**(reg 5)**, (value 10), (value **15**)
i	**(reg 7)**, (value **13**)

⇓ ⇓

10	**(reg 5)**, **(reg 7)**, (value 13), (value 15)
13	(value 10)
15	(value 10)
i	(value **10**)

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!