15-745
Register Allocation 2

David Koes
Feb 1, 2005

Review: Graph Coloring RA

- Build interference graph
- Simplify (degree < k)
- Coalesce (Briggs/Chaitin)
- Mark potential spills
- Select colors
- If stuck, spill

Example (F03 Midterm)

Build Interference Graph

Reaching Definitions
Live
Live Ranges

Build Interference Graph

Live(entry) = \(\{A\}\)
\(B = \text{global}\)
\(A \geq B?\)

\(C = A + B\)
\(C = B - A\)

\(A = C + 1\)
\(A \geq 0?\)

return \(A\)
Live(exit) = \(\{A\}\)

Build Interference Graph

Live(entry) = \(\{A\}\)
\(B = \text{global}\)
\(A \geq B?\)

\(C = A + B\)
\(C = B - A\)

\(A = C + 1\)
\(A \geq 0?\)

return \(A\)
Live(exit) = \(\{A\}\)

Reaching Definitions
\(\{A_0, B_0\}\) \(\{A_0, B_0\}\)
\(\{A_0, B_0\}\) \(\{A_0, B_0\}\)
\(\{C_0, C_1\}\) \(\{A_0, B_0, C_0\}\)

\(\{A_1, B_0, C_0, C_1\}\) \(\{C_1\}\)
\(\{A_1, B_0, C_0, C_1\}\) \(\{A, C\}\)
\(\{A_1, C_0, 1\}\)

\(\{A_1, B_0, C_0, C_1\}\) \(\{A_1\}\)
Build Interference Graph

**Live Ranges**

<table>
<thead>
<tr>
<th>Live(entry) = {A}</th>
</tr>
</thead>
<tbody>
<tr>
<td>B = global</td>
</tr>
<tr>
<td>A &gt; B?</td>
</tr>
<tr>
<td>(A₀, B₀)</td>
</tr>
<tr>
<td>(A₁, B₁)</td>
</tr>
<tr>
<td>(A₂, B₂)</td>
</tr>
<tr>
<td>(A₃, B₃)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C = A + B</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A₀, C₀)</td>
</tr>
<tr>
<td>(A₁, C₁)</td>
</tr>
<tr>
<td>(A₂, C₂)</td>
</tr>
<tr>
<td>(A₃, C₃)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A = C + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &gt; 0?</td>
</tr>
<tr>
<td>return A</td>
</tr>
</tbody>
</table>

| Live(exit) = {A} |

Trivial to color with 2 registers

Reducing Register Pressure

- Recall: Split pseudo-registers into live ranges to create an interference graph that is easier to color
  - Eliminate interference in a variable’s “dead” zones.
  - Increase flexibility in allocation: can allocate same variable to different registers

Insight

- Split a live range into smaller regions (by paying a small cost) to create an interference graph that is easier to color
  - Eliminate interference in a variable’s “nearly dead” zones.
    - **Cost**: Memory loads and stores
      - Load and store at boundaries of regions with no activity
      - # active live ranges at a program point can be > # registers
  - Can allocate same variable to different registers
    - **Cost**: Register operations
      - a register copy between regions of different assignments
      - # active live ranges cannot be > # registers

Examples

**Example 1:**

```
FOR i = 0 TO 10
  FOR j = 0 TO 10000
    A = A + ...
      (does not use B)
  FOR j = 0 TO 10000
    B = B + ...
      (does not use A)
```

**Example 2:**

```
a = b + c
b = a + b
```
Live-Range Splitting

- When do we apply live range splitting?
- Which live range to split?
- Where should the live range be split?
- How to apply live-range splitting with coloring?

One Algorithm

- **Observation:** Spilling is absolutely necessary if
  - number of live ranges active at a program point > \( n \)

- **Apply live-range splitting before coloring**
  - Identify a point where number of live ranges > \( n \)
  - For each live range active around that point
    - find the outermost “block construct” that does not access the variable
  - Choose a live range with the largest inactive region
  - Split the inactive region from the live range