Stutter Step Mode Execution

Introduction
- Collect Error Data in ARGOS
  - Enhance the error detection coverage
- Software Error Detection
  - Assigned signature analysis
  - Watchdog task
  - Assertions
  - Stutter step mode execution
- Primary Concern: Coverage
- No Error Recovery

Stutter Step Mode
- Definition
  - Repeat calculation and compare results
- Catch Transient Errors
  - Data transformations
- Size of Pieces to Be Repeated
- Change Code in Different Levels
- Machine State Changes
  - Initial data for second execution

Compiling Steps
- Change Intermediate Code in SUIF
  - Select group of instructions to be repeated
  - Duplicate with temp variables
  - Switch Off Some Optimizations

Simple ALU Instructions
- Single Error, Any Error
- Make Initial A' Different From A

Outline
- Introduction
- Stutter Step Mode
- Compiler Integration
- Instruction Types
- Alternatives
- Research Direction
Stutter Step Mode Execution

Other Cases

<table>
<thead>
<tr>
<th>A &lt;- A op. B</th>
<th>A' &lt;- A op. B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &lt;- A op. B</td>
<td>cmp A, A'</td>
</tr>
<tr>
<td></td>
<td>bne err_handler</td>
</tr>
</tbody>
</table>

- **Compares**
  - Affecting condition codes
  - Machine dependent

Reduce Overhead

<table>
<thead>
<tr>
<th>C &lt;- A op. B</th>
<th>C' &lt;- A op. B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &lt;- C op. D</td>
<td>A' &lt;- C op. D</td>
</tr>
<tr>
<td></td>
<td>cmp A, A'</td>
</tr>
<tr>
<td></td>
<td>bne err_handler</td>
</tr>
</tbody>
</table>

- **A Series of Computations**
  - A basic block
  - Compare the live variables at the end
  - An expression

Liveness

<table>
<thead>
<tr>
<th>C &lt;- C op. B</th>
<th>C' &lt;- C op. B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &lt;- C op. D</td>
<td>A' &lt;- C' op. D</td>
</tr>
<tr>
<td></td>
<td>cmp A, A'</td>
</tr>
<tr>
<td></td>
<td>bne err_handler</td>
</tr>
</tbody>
</table>

- **At Entry Point**
  - Temporary variables
  - Register Pressure
- **At Exit Point**
  - Compare checksum of live variables

Flaws!

- **Logical Operations**
  - A <- B op. C; D <- A and x..x0
  - y..y1 and x..x0 = y..y0 and x..x0
- **Integer Division**
  - A <- B op. C; D <- A / 5
  - A <- 10 + 7; D <- (19..15) / 5

Load Instruction

<table>
<thead>
<tr>
<th>A &lt;- mem[addr]</th>
</tr>
</thead>
<tbody>
<tr>
<td>A' &lt;- mem[addr]</td>
</tr>
<tr>
<td>cmp A, A'</td>
</tr>
<tr>
<td>bne err_handler</td>
</tr>
</tbody>
</table>

- **Can Load Twice**
- **Cache**
  - Second load will come from cache
- **ECC**
  - Checks upto the boundary of the CPU

Store Instruction

<table>
<thead>
<tr>
<th>mem[addr] &lt;- A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A' &lt;- mem[addr]</td>
</tr>
<tr>
<td>cmp A, A'</td>
</tr>
<tr>
<td>bne err_handler</td>
</tr>
</tbody>
</table>

- **No Point in Storing Twice**
- **Write Checked Data**
  - Last instruction of a piece
- **Load After Store**
  - One extra memory reference per store
  - Cache
Stutter Step Mode Execution

Memory Operations
- RISC vs. CISC
- Data Transfer, Not Transform
- Load and Store in Large Pieces
  - Modify and use
  - Memory disambiguation
  - Keep checksum of the stored data
- Two Data Segment
  - Address calculation

Control Transfer Instructions
- Not Applicable
- Control Flow Checking
  - Signature analysis
  - Assign signature
  - Watch-dog task

Problems
- Register Spill in Back-End
  - Produces extra load/store
  - Worst with added temporary variables
- Register Allocation
  - Need even usage
- SUIF Has Bugs!

Limits
- Capability
  - Data transformation
  - Some instruction types
  - Swap (atomic operation)
- Applicability
  - Error handling routine
    - Fault tolerant; error recovery
    - Safe storage of data

Alternatives
- Affine Transformations
  - Use checksums instead of code duplication
- Multithreading
  - One data segment
- Multitasking
  - Cleaner? memory content
  - Communication overhead
  - Coverage, latency

Research Direction
- Implement Stutter Step Mode with SUIF
  - Different instruction types
- How to Check
  - R3000 simulator or Verilog model
  - Fault modeling