New architecture for Valgrind

Ivo Raisr
Tomáš Jedlička
THE FOLLOWING IS INTENDED TO OUTLINE OUR GENERAL PRODUCT DIRECTION. IT IS INTENDED FOR INFORMATION PURPOSES ONLY, AND MAY NOT BE INCORPORATED INTO ANY CONTRACT. IT IS NOT A COMMITMENT TO DELIVER ANY MATERIAL, CODE, OR FUNCTIONALITY, AND SHOULD NOT BE RELIED UPON IN MAKING PURCHASING DECISIONS. THE DEVELOPMENT, RELEASE, AND TIMING OF ANY FEATURES OR FUNCTIONALITY DESCRIBED FOR ORACLE'S PRODUCTS REMAINS AT THE SOLE DISCRETION OF ORACLE.
Porting Valgrind to sparcv9

sparcv9/Linux and sparcv9/Solaris
Overview

• The story
• Hackers
• Why SPARC?
• sparcv9 ISA highlights
• Design decisions and problems to solve
• Current status
The Story

- Gained traction after upstreaming Solaris port
- Initially sparcv9/Solaris
- Recently sparcv9/Linux
- Maintained as separate forks
Hackers

• Collaborative effort:
• Ivo Raisr
• Tomáš Jedlička
Why SPARC?

• SPARC
• History begins in 1980’s
• Joint effort of Sun (now Oracle) and Fujitsu
• Designed for enterprise workloads
• Several terabytes of memory
• Several thousands of CPUs
sparcv9 ISA Highlights 1.

• RISC architecture
• load/store memory access, properly aligned
• Explicit stack support (nothing like push/pop)
• Instruction format is OP %r1, %r2/imm, %rd
• Hypervisor in the firmware
sparcv9 ISA Highlights II.

- Registers
- General purpose registers and aliasing into %i, %o, %l
- Widowing state register + spill/fill traps
- Floating point registers and its variants %f, %d, %q (4, 8, 16 bytes)
\[ \text{LOCAL}[7:0] = R[23:16] \]
\[ \text{IN}[7:0] = R[31:24] \]
\[ \text{OUT}[7:0] = R[15:8] \]
\[ \text{GLOBAL}[7:1] = R[7:1] \]
\[ R0 = 0 \]
sparcv9 ISA Highlights III.

• **Delayed Control Transfer Instructions (dCTI)**

• Control transfer instructions (jmp, branch...) have a delay slot

• Sequence of instructions:
  - `bcc %xcc/%icc, <address>`
  - `add %o0, 1, %o0`

  - behaves actually as:
    - Evaluate the branch condition
    - Execute `add %o0, 1, %o0`
    - Set nPC to `<address>`
    - Transfer control to `<address>`, setting PC to `<address>` and nPC to `<address+4>`

• Possibility to annul the delayed instruction if the branch is not taken.
sparcv9 ISA Highlights IV.

• **Address Space Identifier** (ASI)
  • Alters MMU behaviour during load/store operations
  • One instruction can do many different things, based on ASI value
  • Example: LDDF loads 8 bytes by default, however with a different ASI:
    – *LDSHORTF* loads 1 byte with ASI D0
    – *LDSHORTF* loads 2 bytes with ASI D2
    – *LDBLOCKF* loads 64 bytes with ASI F0
    – *LDDFA* loads 8 bytes with other ASIs
  • ASI can be immediate or register (%asi)
  • Sharing the same opcode: ldda [address] %asi, register
  • Actual instruction is known only at runtime, not at VEX translation time
  • Used in: hardware specialized optimizations, OpenSSL, stack unwinding
sparcv9 ISA Highlights V.

- **Application Data Integrity (ADI)**
- Dynamic Tainting technique implemented in sparc chip MMU and memory allocators.
- Taint mark (version) stored both in memory/cache and in the pointer passed from the allocator.
- Taint mark (version) match is checked by MMU with every load or store.
- Trap generated and signal sent to the process when versions do not match.
sparcv9 ISA Highlights VI.

• Syscalls and Fast traps
• Syscall handling via trap table
• Fast trap based syscall
• OS specific feature
<table>
<thead>
<tr>
<th>EVC_FAILADDR</th>
<th>EVC_COUNTER</th>
<th>%g0</th>
<th>%g1</th>
<th>%g2</th>
<th>%g3</th>
</tr>
</thead>
<tbody>
<tr>
<td>%g4</td>
<td>%g5</td>
<td>%g6</td>
<td>%g7</td>
<td>%o0</td>
<td>%o1</td>
</tr>
<tr>
<td>%o2</td>
<td>%o3</td>
<td>%o4</td>
<td>%o5</td>
<td>%o6</td>
<td>%o7</td>
</tr>
<tr>
<td>%l0</td>
<td>%l1</td>
<td>%l2</td>
<td>%l3</td>
<td>%l4</td>
<td>%l5</td>
</tr>
<tr>
<td>%l6</td>
<td>%l7</td>
<td>%i0</td>
<td>%i1</td>
<td>%i2</td>
<td>%i3</td>
</tr>
<tr>
<td>%i4</td>
<td>%i5</td>
<td>%i6</td>
<td>%i7</td>
<td>%f0</td>
<td>%f2</td>
</tr>
<tr>
<td>%f4</td>
<td>%f6</td>
<td>%f8</td>
<td>%f10</td>
<td>%f12</td>
<td>%f14</td>
</tr>
<tr>
<td>%f16</td>
<td>%f20</td>
<td>%f22</td>
<td>%f24</td>
<td>%f26</td>
<td></td>
</tr>
<tr>
<td>%f28</td>
<td>%f30</td>
<td>%d32</td>
<td>%d34</td>
<td>%d36</td>
<td>%d38</td>
</tr>
<tr>
<td>%d40</td>
<td>%d42</td>
<td>%d44</td>
<td>%d46</td>
<td>%d48</td>
<td>%d50</td>
</tr>
<tr>
<td>%d52</td>
<td>%d54</td>
<td>%d56</td>
<td>%d58</td>
<td>%d60</td>
<td>%d62</td>
</tr>
<tr>
<td>%pc</td>
<td>%npc</td>
<td>%y</td>
<td>%asi</td>
<td>%fprs</td>
<td>%gsr</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMSTART</th>
<th>CMLEN</th>
<th>CC_OP</th>
<th>CC_DEP1</th>
<th>CC_DEP2</th>
<th>CC_NDEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSR_RD</td>
<td>FSR_FCC</td>
<td>FSR_CEXC_OP</td>
<td>FSR_CEXC_DEP1_HI</td>
<td>FSR_CEXC_DEP1_LO</td>
<td>FSR_CEXC_DEP2_HI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FSR_CEXC_DEP2_LO</td>
<td>FSR_CEXC_NDEP</td>
<td>EMNOTE</td>
<td>scratchpad</td>
<td></td>
</tr>
</tbody>
</table>

Oracle®

Copyright © 2017 Oracle and/or its affiliates. All rights reserved.
Design Decisions and Problems I.

- Register windows support
- Instructions in branch delay slot
- Lazy evaluation of CCR and FSR.cexc
- Instructions with %asi register in opcode
- Basic emulation of 128 bit integer operations
- Program counters PC and nPC
- Hundreds of misaligned warnings from compiler
- Represent a syscall returning 5 return values
- Can Valgrind leverage ADI?
Design Decisions and Problems II.

• Decorating `Iex_Load` and `Iex_Store` with `ASI`
• Teach Memcheck about `ASI_PRIMARY_NOFAULT`

```
struct {
    IREndness end;  /* Endian-ness of the load */
    IRType ty;      /* Type of the loaded value */
    IRExpr* addr;   /* Address being loaded from */
    IRExpr* asi;    /* SPARCv9 address space identifier */
} Load;
```
Current status

• Synced regularly with SVN upstream
• Support for sparcv9/Solaris stable
• memcheck and none tests:
  • == 731 tests, 35 stderr failures, 8 stdout failures, 0 stderrB failures, 0 stdoutB failures, 1 post failure ==
• Support for sparcv9/Linux at the beginning
• Both ports live at: https://bitbucket.org/iraisr/valgrind-solaris
Q&A Session