A Quick Tour of the x86isa Books

Shilpi Goel
shigoel@cs.utexas.edu

ACL2 Rump Session Talk
Released the x86isa books on 21st May, 2015 (books/projects/x86isa)

License: BSD 3-Clause

Today: ~120 files, ~100K lines (including comments, whitespace, & documentation)
Short-Term Goal

E.g.: Formal Analysis of an Optimized Data-Copy Program
Short-Term Goal

E.g.: Formal Analysis of an Optimized Data-Copy Program

Specification:
Copy data \( x \) from linear memory location \( l_0 \) to disjoint linear memory location \( l_1 \).
**Short-Term Goal**

**E.g.: Formal Analysis of an Optimized Data-Copy Program**

**Specification:**
Copy data $x$ from linear memory location $l0$ to disjoint linear memory location $l1$. 

![Diagram of linear memory location $l0$ and data $x$]
Short-Term Goal

E.g.: Formal Analysis of an Optimized Data-Copy Program

**Specification:**
Copy data $x$ from linear memory location $l_0$ to disjoint linear memory location $l_1$. 
**Short-Term Goal**

*E.g.: Formal Analysis of an Optimized Data-Copy Program*

**Specification:**
Copy data \( x \) from linear memory location \( l_0 \) to disjoint linear memory location \( l_1 \).

**Verification Objective:**
After a successful copy, \( l_0 \) and \( l_1 \) contain \( x \).
Short-Term Goal

E.g.: Formal Analysis of an Optimized Data-Copy Program

**Specification:**
Copy data $x$ from linear memory location $l0$ to disjoint linear memory location $l1$.

**Verification Objective:**
After a successful copy, $l0$ and $l1$ contain $x$.

**Implementation:**
Include the *copy-on-write* technique: $l0$ and $l1$ can be mapped to the same physical memory location $p$.

- System calls
- Page mapping
- Privileges
- Context Switches
Short-Term Goal

E.g.: Formal Analysis of an Optimized Data-Copy Program

**Specification:**
Copy data $x$ from linear memory location $l_0$ to disjoint linear memory location $l_1$.

**Verification Objective:**
After a successful copy, $l_0$ and $l_1$ contain $x$.

**Implementation:**
Include the *copy-on-write* technique: $l_0$ and $l_1$ can be mapped to the same physical memory location $p$.
- System calls
- Page mapping
- Privileges
- Context Switches
Long-Term Goals

• Get more miles: Boot/run a serious OS (like FreeBSD) on the x86 is a model
  ➡ Support more x86-64 features

• Verify more serious programs
  ➡ E.g., FreeBSD/Linux code for context switching
  ➡ Use tools like codewalker to make life easier
What do the x86isa books contain?

*Modeling (x86isa/machine)*

- A formal, executable x86 ISA model (64-bit mode)

- x86 state
- Specification of x86 instructions (311 opcodes)
- Instruction fetch, decode, and execute function (step function)
- Run function

- Single core
What do the x86isa books contain?

➡ A formal, executable x86 ISA model (64-bit mode)

Modeling (x86isa/machine)

- x86 state
- Specification of x86
- Instruction fetch, decode, execute
- Run function
- Single core

2.1.1 Global and Local Descriptor Tables

When operating in protected mode, all memory accesses pass through either the global descriptor table (GDT) or an optional local descriptor table (LDT) as shown in Figure 2-1. These tables contain entries called segment descriptors. Segment descriptors provide the base address of segments as well as access rights, type, and usage information.

Each segment descriptor has an associated segment selector. A segment selector provides the software that uses it with an index into the GDT or LDT (the offset of its associated segment descriptor), a global/local flag (determines whether the selector points to the GDT or the LDT), and access rights information.

Figure 2-2. System-Level Registers and Data Structures in IA-32e Mode
### Modeling: Verification Effort vs. Utility

<table>
<thead>
<tr>
<th>Programmer-Level Mode</th>
<th>System-Level Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verification of application programs</td>
<td>Verification of system programs</td>
</tr>
<tr>
<td>Linear memory address space (2^{64} bytes)</td>
<td>Physical memory address space (2^{52} bytes)</td>
</tr>
<tr>
<td>Assumptions about correctness of OS operations</td>
<td>No assumptions about OS operations</td>
</tr>
<tr>
<td>~3.3 million instructions/second</td>
<td>~912,000 instructions/second (with 1G pages)</td>
</tr>
</tbody>
</table>

Simulation speed measured on an Intel Xeon E31280 CPU @ 3.50GHz, 8 cores, 32GB RAM
Modeling: Verification Effort vs. Utility

FreeBSD read system call semantics

System-level Mode

User Space (Ring 3)
- MOV %rax, 3
- SYSCALL
- MOV %rbx, %rax
- ...

Kernel Space (Ring 0)
- ...
- ...
- SYSRET
- ...

Programmer-level Mode

save user state

restore user state
What do the x86isa books contain?

*Modeling (x86isa/machine)*

- A formal, executable x86 ISA model (64-bit mode)
What do the x86isa books contain?

**Modeling (x86isa/machine)**
- A formal, executable x86 ISA model (64-bit mode)

**Simulation (x86isa/tools/execution)**
- Executable file readers and loaders (ELF/Mach-O)
- A GDB-like mode for dynamic instrumentation of machine code
- Examples of program execution and debugging
What do the x86isa books contain?

**Modeling (x86isa/machine)**
- A formal, executable x86 ISA model (64-bit mode)

**Simulation (x86isa/tools/execution)**
- Executable file readers and loaders (ELF/Mach-O)
- A GDB-like mode for dynamic instrumentation of machine code
- Examples of program execution and debugging

**Reasoning (x86isa/proofs)**
- Helper libraries to reason about x86 machine code
- Proofs of various properties of some machine-code programs
What do the x86isa books contain?

**Modeling (x86isa/machine)**
- A formal, executable x86 ISA model (64-bit mode)

**Simulation (x86isa/tools/execution)**
- Executable file readers and loaders (ELF/Mach-O)
- A GDB-like mode for dynamic instrumentation of machine code
- Examples of program execution and debugging

**Reasoning (x86isa/proofs)**
- Helper libraries to reason about x86 machine code
- Proofs of various properties of some machine-code programs

**Documentation**
A Personal Note

• I made a decision to make my work a part of the ACL2 Community books

• Even though it’s not really ready for primetime...

• Why? Apart from the obvious technical benefits (keep up with changes in ACL2, books I depend on), this has been incredibly motivating.
A Personal Note

- I made a decision to make my work a part of the ACL2 Community books
- Even though it’s not really ready for primetime...
- Why? Apart from the obvious technical benefits (keep up with changes in ACL2, books I depend on), this has been incredibly motivating.

Thank You!
Model Validation

How can we know that our model faithfully represents the x86 ISA?
Validate the model to increase trust in the applicability of formal analysis.
Task A: Validate the logical mode against the execution mode

Task B: Validate the execution mode against the processor + system call service provided by the OS
Programmer-level Mode: Execution Mode

A Common Lisp Distribution

ACL2

x86 ISA Model

SYSCALL

Return

Operating System
Programmer-level Mode: Execution and Reasoning

Execution Mode

\[ x_0 \xrightarrow{} x_1 \]

ENV

Logical Mode

\[ x_0 \xrightarrow{env} x_1^' \]

env

ENV'