Updates on the Linux Capable ACL2 x86 Model

Yahya Sohail and Warren A. Hunt, Jr.
yahya@yahyasohail.com, hunt@cs.utexas.edu
https://yahyasohail.com, https://cs.utexas.edu/~hunt

Original Authors: Shilpi Goel, W. Hunt, Jr., Matt Kaufmann
UT Contributors: Cuong Chau, J Strother Moore
Additional UT Contributors: Soumava Ghoush, Keshav Kini, Robert Krug, Ben Selfridge, Rob Sumners, Nathan Wetzler
Centaur Technology Contributors: Anna Slobodova, Jared Davis, Sol Swords
Kestrel Contributors: Alessandro Coglio, Eric Smith
Sponsors: Centaur Technology, DARPA, Intel, NSF

May 12, 2025

Table of Contents

The x86 Model

History

Completeness and Fidelity

What's New

What's Next

The x86 Model

- The x86 model is an x86 simulator written in ACL2
- Since it's written in ACL2 it is both an executable simulator and a formal model we can prove theorems about
- It can be used to prove theorems about x86 programs
  - Assigns semantics to machine code
  - Correctness proof of a `wc` program
  - Supervisor software, like Zero-Copy
- To our knowledge, most complete formal model of x86 ISA

The x86 Model

- The x86 model is an x86 simulator written in ACL2
- Since it's written in ACL2 it is both an executable simulator and a formal model we can prove theorems about
- It can be used to prove theorems about x86 programs
  - Assigns semantics to machine code
  - Correctness proof of a `wc` program
  - Supervisor software, like Zero-Copy
- To our knowledge, most complete formal model of x86 ISA
- Linux capable; Let's start the demo

Abridged History

- In ∼2004, Hunt modeled the y86 ISA used in Bryant and O'Hallaron's architecture textbook
- Around 2009, Hunt created a simple x86-ISA model
- In 2012, Hunt and Kaufmann documented a more complete ACL2 x86-ISA model (UTCS Technical Report)
- In ∼2015, Goel's PhD work included adding x86-ISA instructions, supervisor mode, and memory management
- In ∼2017, Cuong Chau added floating-point support (SSE 1 and SSE 2 instructions)
- Later, Alessandro Coglio [Kestrel] and Goel added support for 32-bit instructions
- In 2023, Sohail added a timer, interrupts, console I/O, etc. so Linux could be booted, and run user programs. Also, added TLB.

Completeness and Fidelity

- Supports
    - most of the integer ISA
    - portions of the vector ISA extensions
    - long (64-bit) mode with paging
    - 32-bit mode (but not 32-bit paging)
- Model of *a TLB*
- Models of custom timer (using instructions executed as a proxy for time) and TTY peripherals

Completeness and Fidelity

- Supports
  - most of the integer ISA
  - portions of the vector ISA extensions
  - long (64-bit) mode with paging
  - 32-bit mode (but not 32-bit paging)
- Model of *a TLB*
- Models of custom timer (using instructions executed as a proxy for time) and TTY peripherals
- This is enough to boot (minimally modified) Linux and run GCC

What's New

- The Linux capable model, presented at last ACL2 workshop, merged into the community books
  - Guard verified
  - Many bugfixes
  - Peripheral models
  - Cosimulation based validation tool
  - Linux patch
- Attachable stobj memory
- TLB cache model
  - Speed up address translation
  - Lemmas and proofs in community books updated to account for the TLB

What's New (continued)

- Better testing
  - Sol Swords wrote `asmtest` testing framework
  - I wrote `testgen` for `asmtest`
    - Automatically generate tests using Intel's XED library's instruction database
    - Supports instructions with immediate, GPR, and XMM operands
- Bugfixes, more instructions, updated Linux patch
- Documentation for TLB and Linux boot

What's Next

- Model is far from complete
  - Multiprocessing with memory ordering semantics
  - Better MMIO semantics
  - More instructions
  - Improve validation

What's Next

- Model is far from complete
  - Multiprocessing with memory ordering semantics
  - Better MMIO semantics
  - More instructions
  - Improve validation
- More applications
  - The programs verified in the community books are still largely simple, mostly proof of concept
  - Would be cool to see correctness proofs of “useful” software in the community books
  - Supervisor software proofs are still very limited; maybe a good next step is correctness proof of program with a basic unikernel

What's Next

- Model is far from complete
    - Multiprocessing with memory ordering semantics
    - Better MMIO semantics
    - More instructions
    - Improve validation
- More applications
    - The programs verified in the community books are still largely simple, mostly proof of concept
    - Would be cool to see correctness proofs of "useful" software in the community books
    - Supervisor software proofs are still very limited; maybe a good next step is correctness proof of program with a basic unikernel
- Is x86 still worthwhile target?

Let's Check on Linux