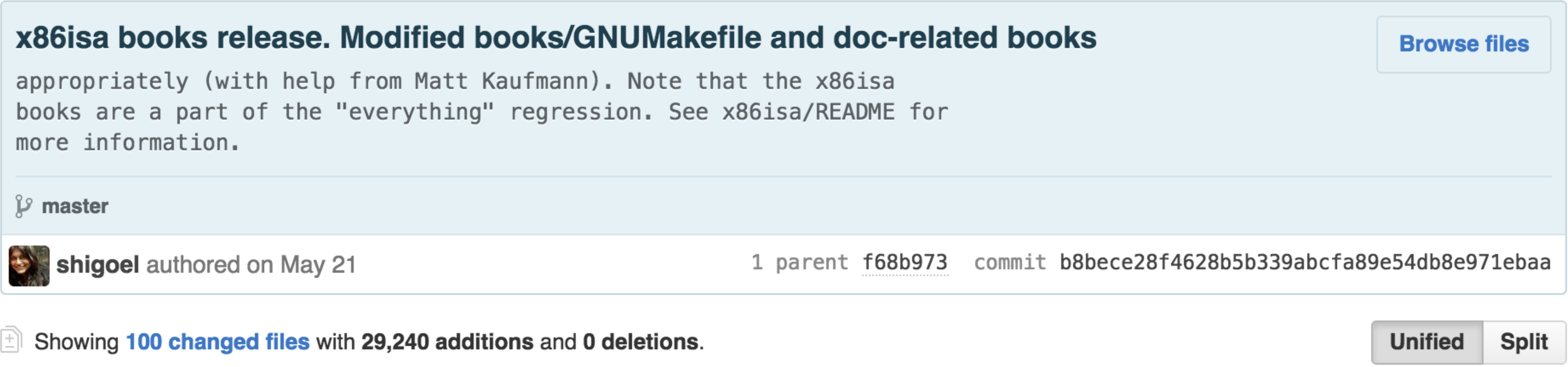
# A Quick Tour of the `x86isa` Books

Shilpi Goel
shigoel@cs.utexas.edu

ACL2 Rump Session Talk

*2nd October, 2015*



**x86isa books release. Modified books/GNUMakefile and doc-related books**

```
appropriately (with help from Matt Kaufmann). Note that the x86isa
books are a part of the "everything" regression. See x86isa/README for
more information.
```

**master**

**shigoel** authored on May 21 1 parent f68b973 commit b8bece28f4628b5b339abcfa89e54db8e971ebaa

Showing **100 changed files** with **29,240 additions** and **0 deletions**.

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

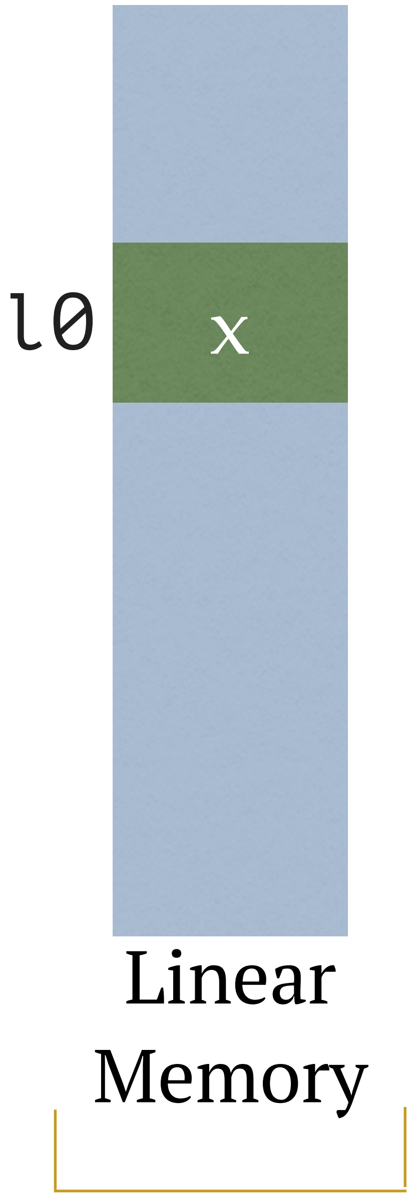
Specification:
Copy data `x` from linear memory location `l0` to
disjoint linear memory location `l1`.

# 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`.

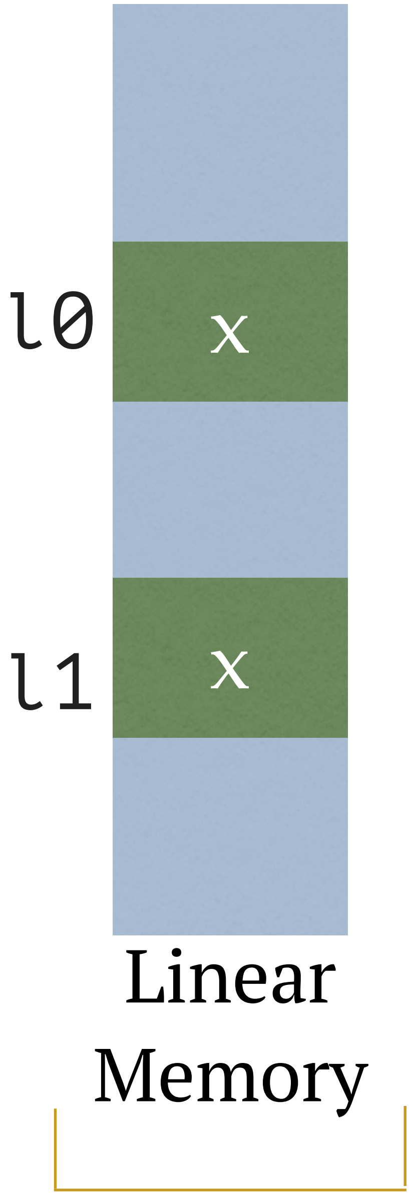


Programmer
-level mode

# 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`.



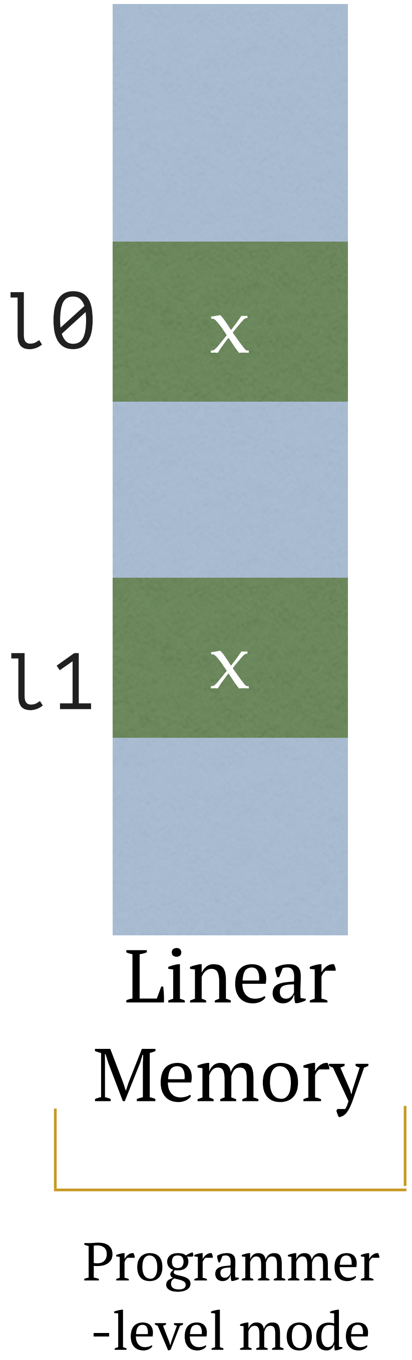
Programmer
-level mode

# 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`.



# Short-Term Goal

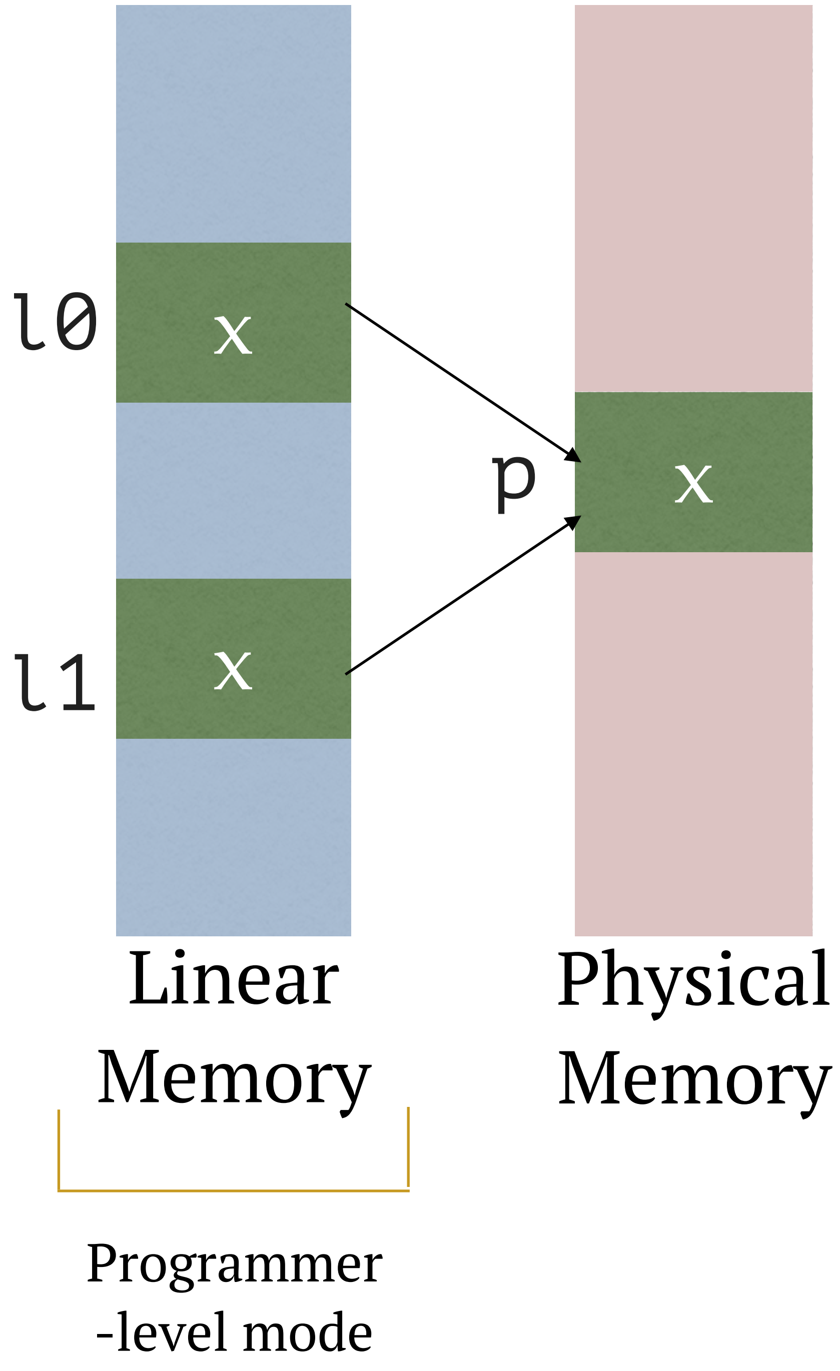
## E.g.: Formal Analysis of an Optimized Data-Copy Program

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

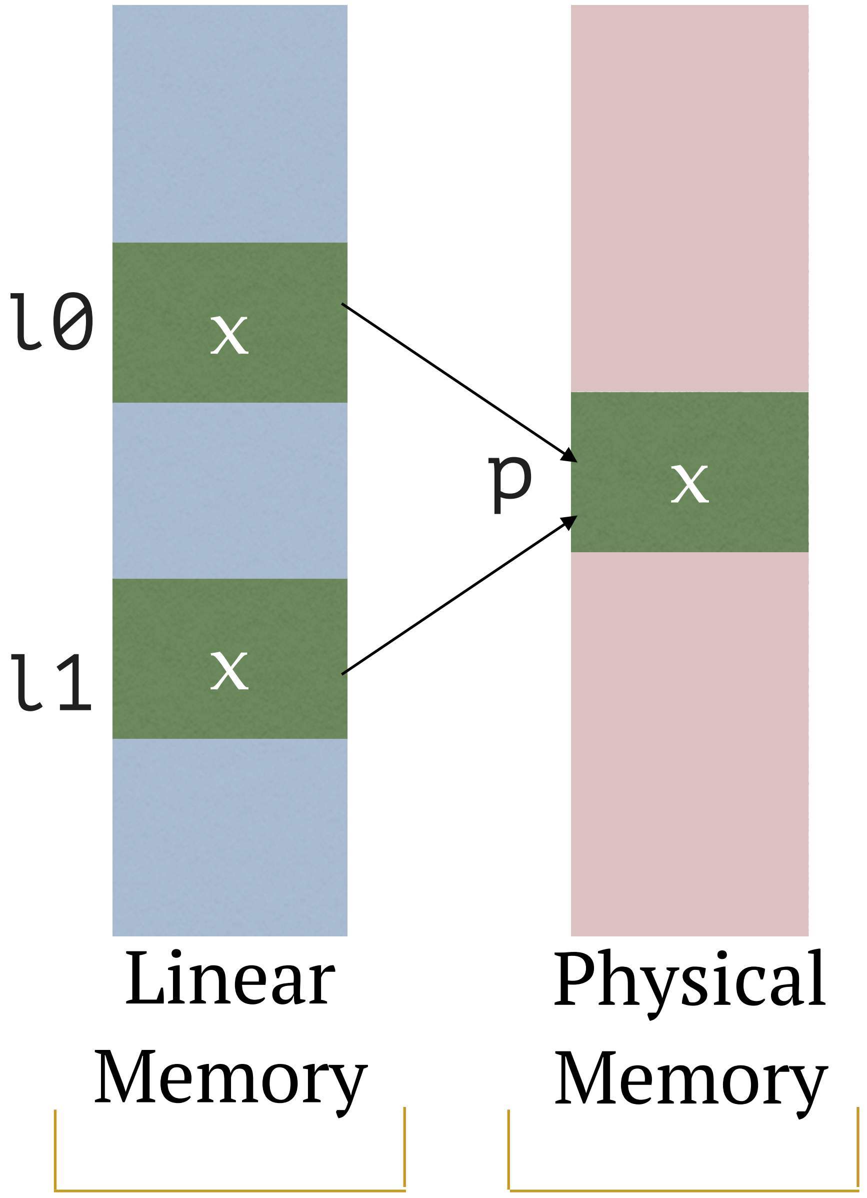
## E.g.: Formal Analysis of an Optimized Data-Copy Program

Specification:
Copy data x from linear memory location 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



Programmer -level mode

System- level mode

# Long-Term Goals

- Get more miles: Boot/run a serious OS (like FreeBSD) on the `x86isa` 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

*Modeling* (x86isa/ma

➡ **A formal, execut**

- x86 state
- Specification of x8
- Instruction fetch,
- Run function

- Single core

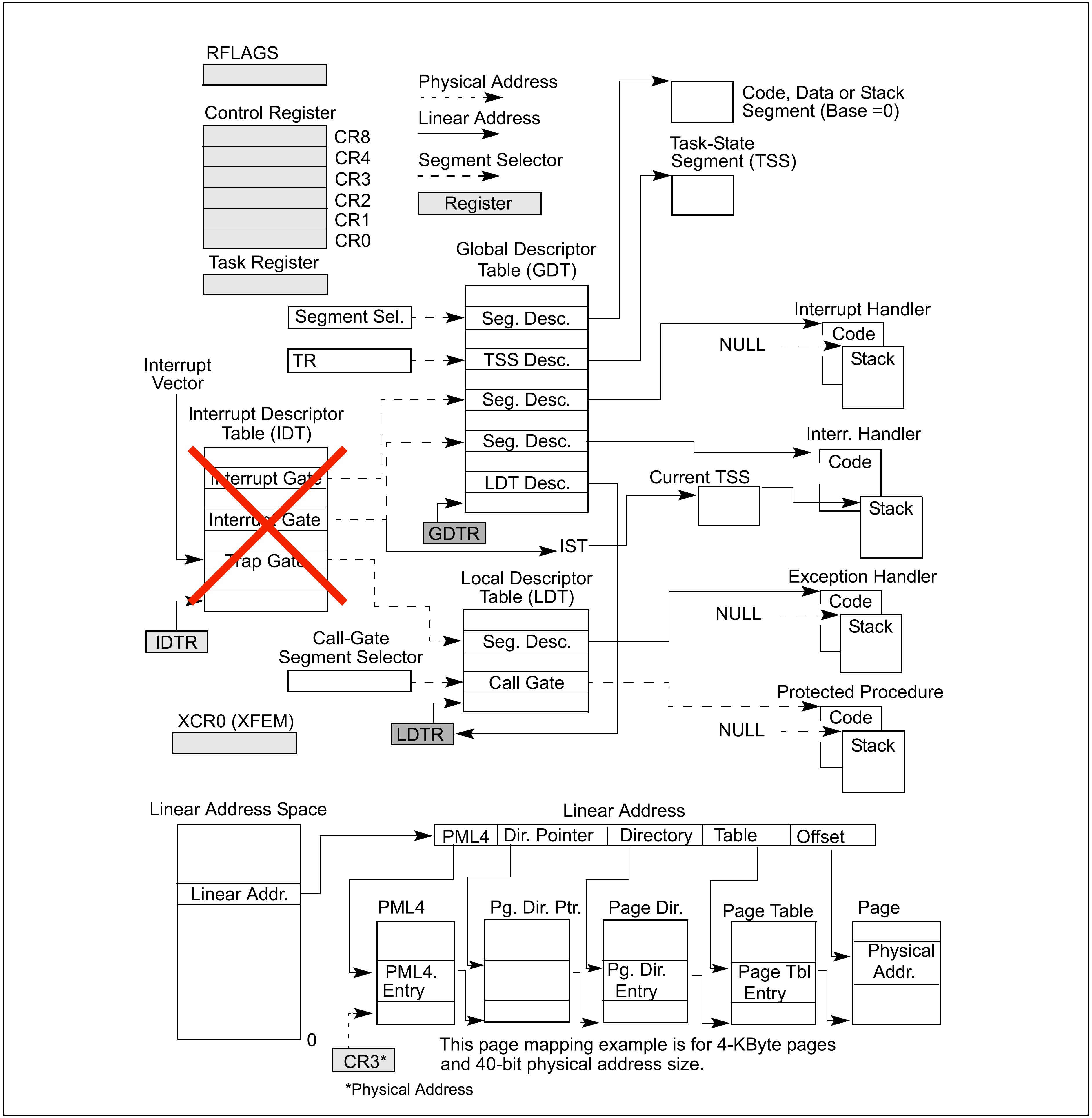


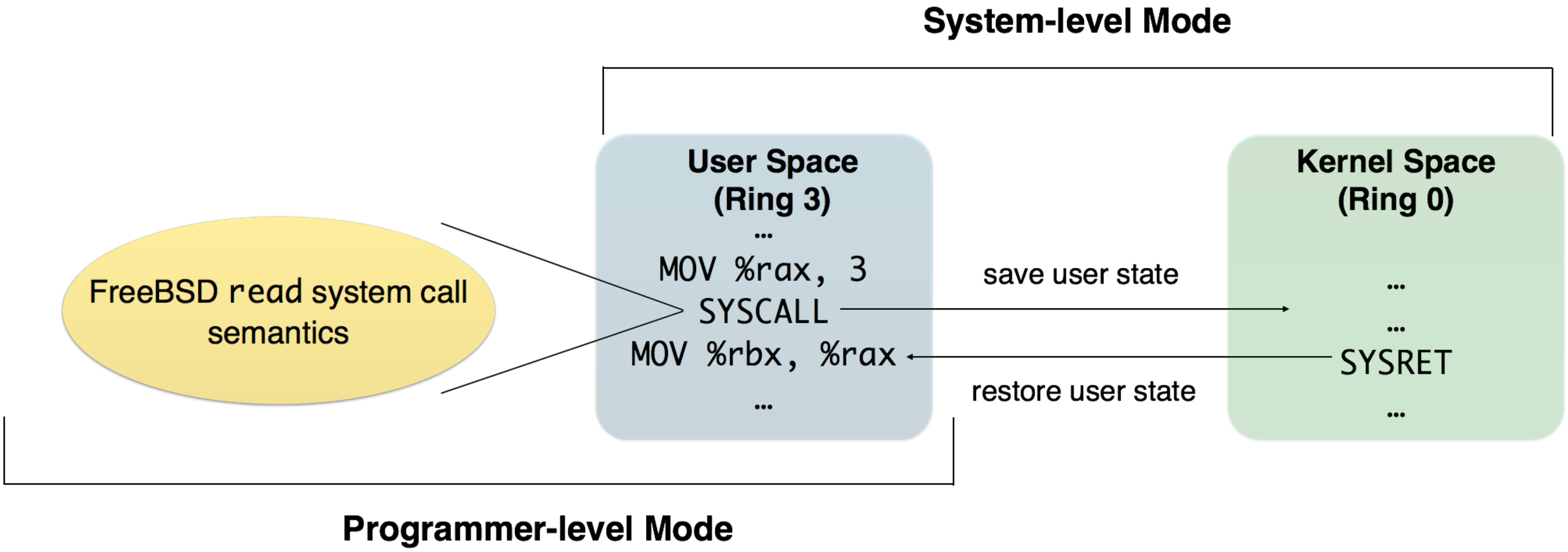
Figure 2-2. System-Level Registers and Data Structures in IA-32e Mode

# Modeling: Verification Effort vs. Utility

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

Simulation speed measured on an Intel Xeon E31280 CPU @ 3.50GHz, 8 cores, 32GB RAM

# Modeling: Verification Effort vs. Utility



# 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

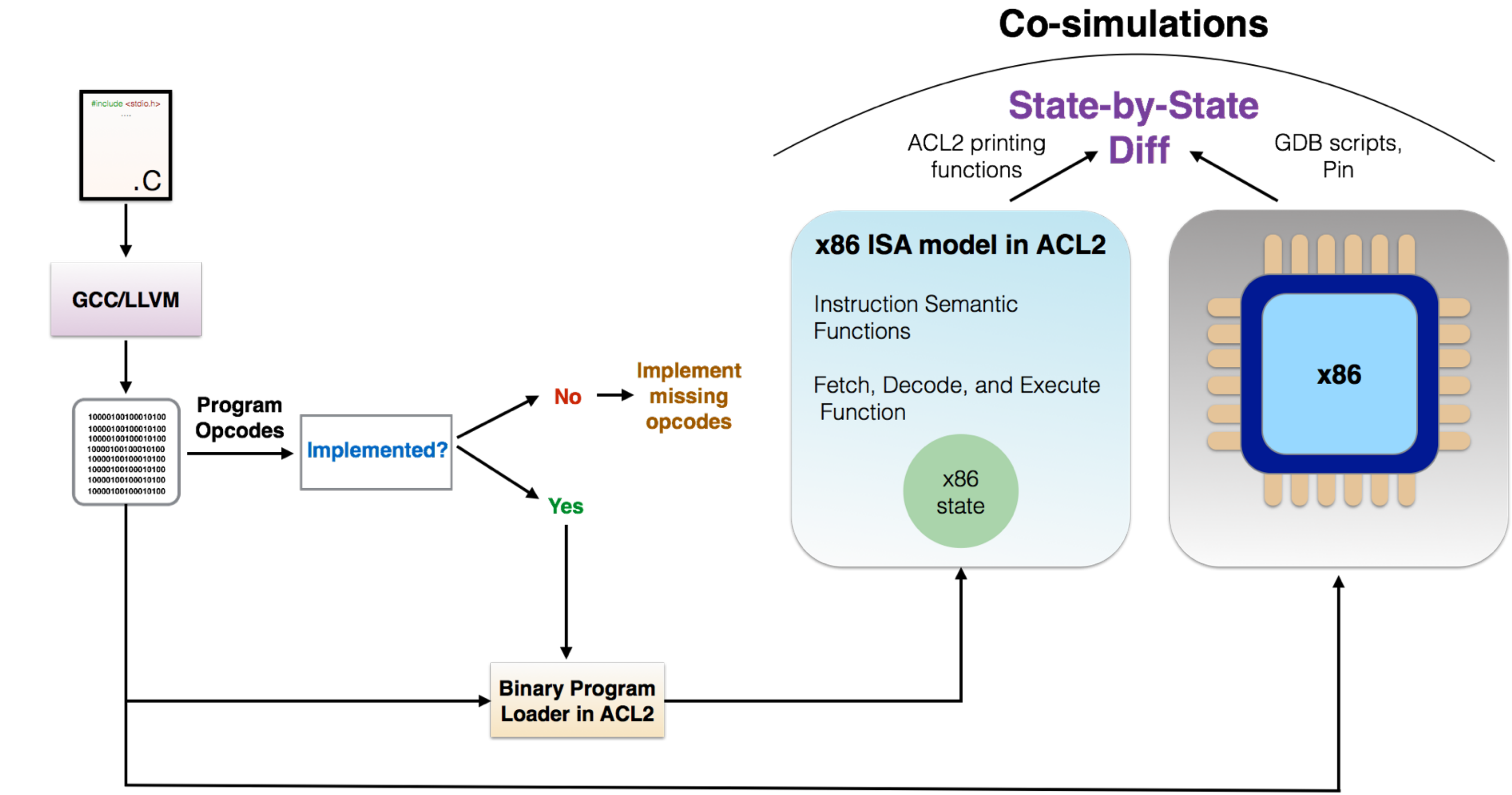
- I made a decision to make my work a part of the ACL2 Community books
- Even though it’s not really ready for primetime…
- Why? Apart from the obvious technical benefits (keep up with changes in ACL2, books I depend on), this has been incredibly motivating.

## Thank You!

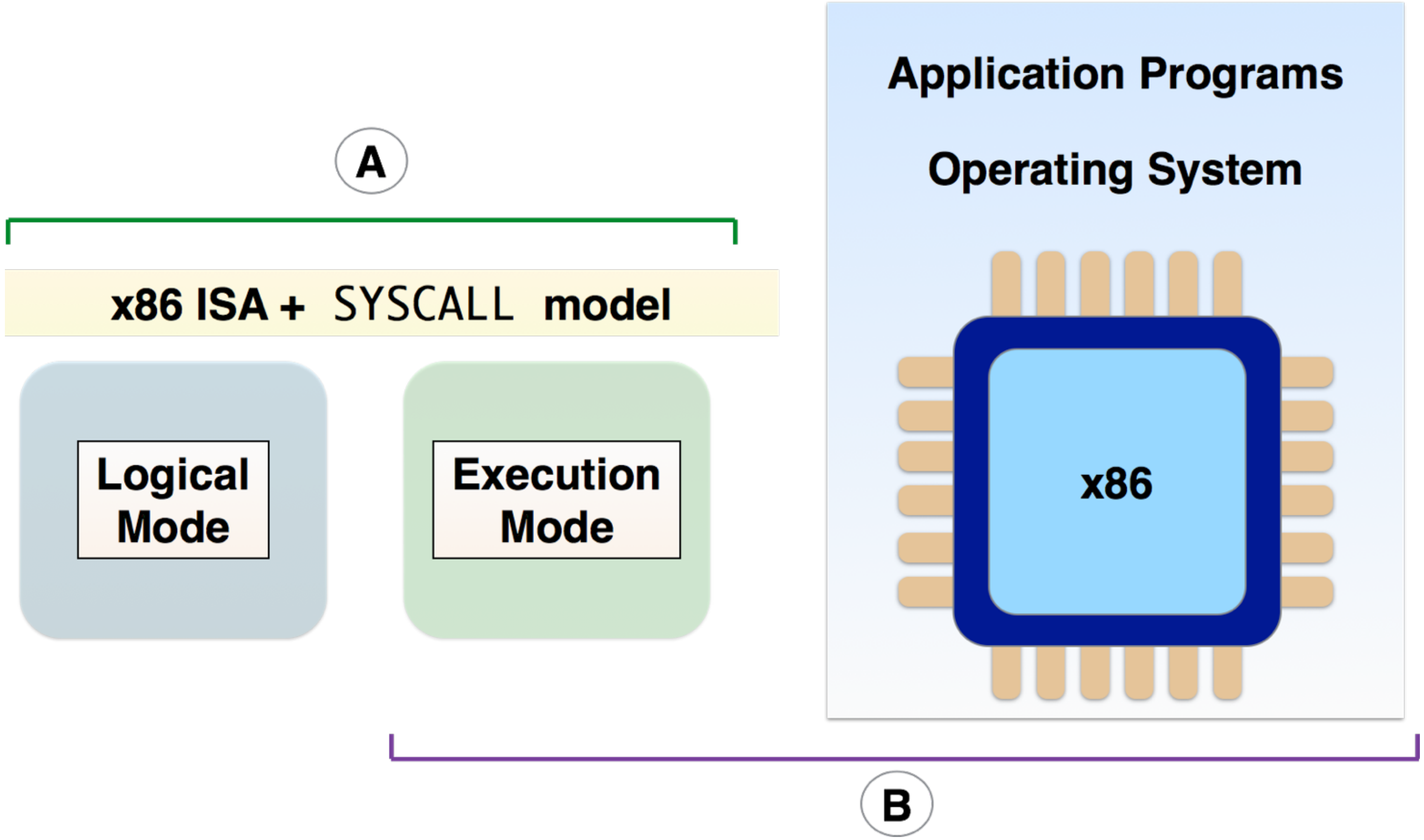
# Model Validation

*How can we know that our model faithfully represents the x86 ISA?*

Validate the model to increase trust in the applicability of formal analysis.



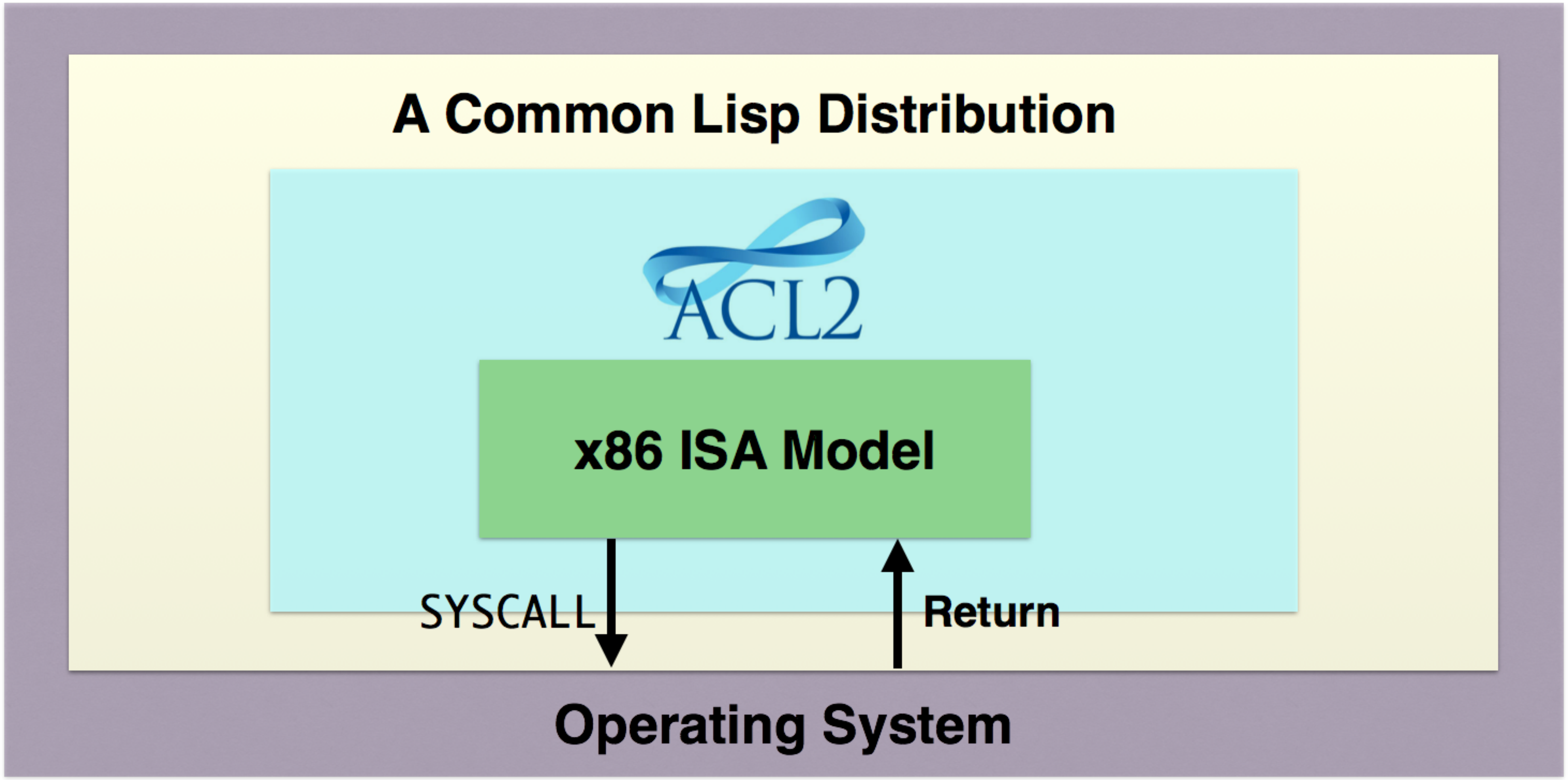
# Programmer-level Mode: Model Validation



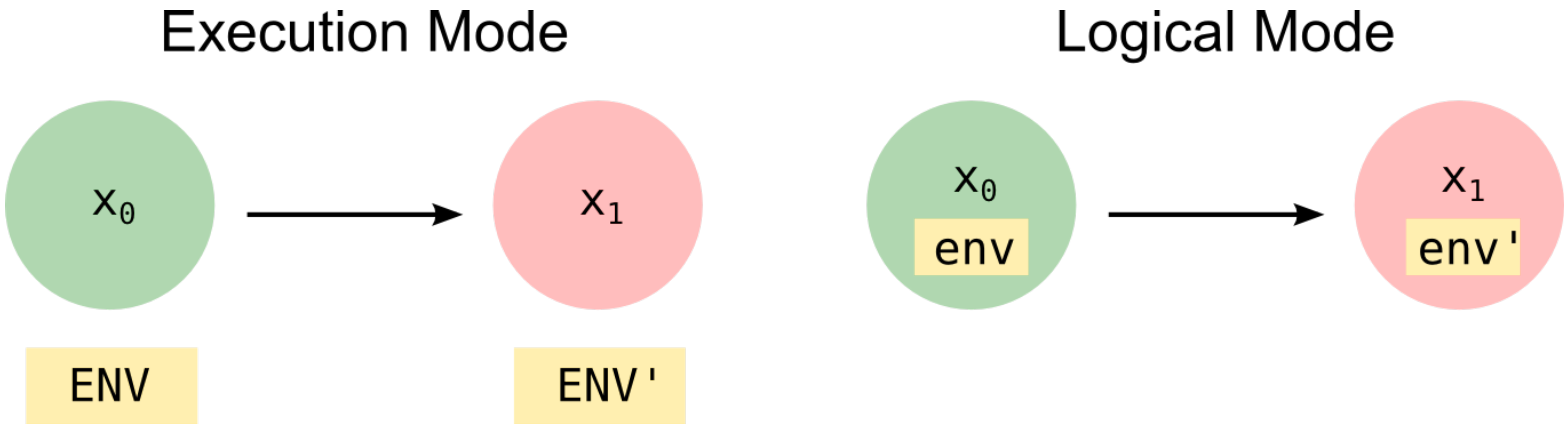
**Task A:** Validate the logical mode against the execution mode

**Task B:** Validate the execution mode against the processor + system call service provided by the OS

# Programmer-level Mode: Execution Mode



# Programmer-level Mode: Execution and Reasoning



Execution Mode

$x_0$

ENV

$x_1$

ENV'

Logical Mode

$x_0$

env

$x_1$

env'