Analysis and Validation of Semantic Mistranslation Errors in Emulators

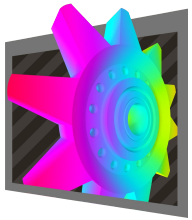
Christian Krinitsin, Theofilos Augoustis,
Sebastian Reimers, Pramod Bhatotia

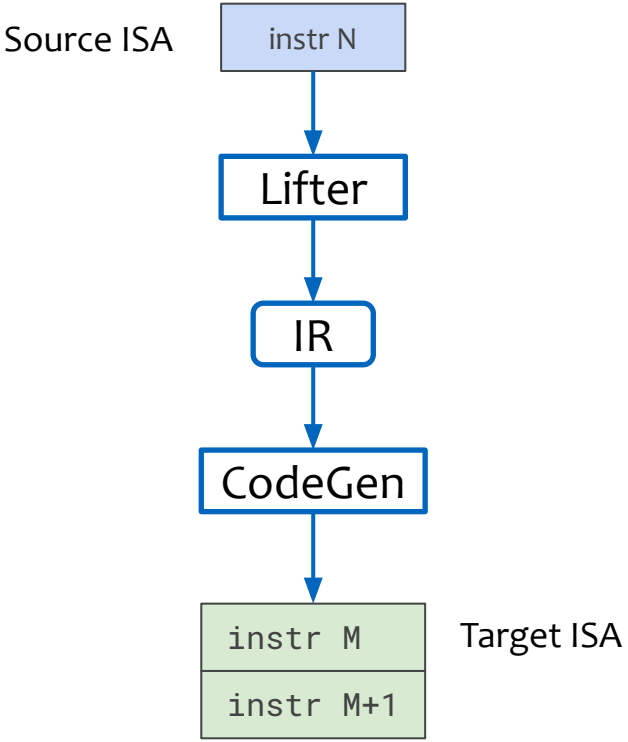
30.09.2025

Program Emulators

Emulators have a critical role:

- Enable cross-platform execution
- Only support for legacy software



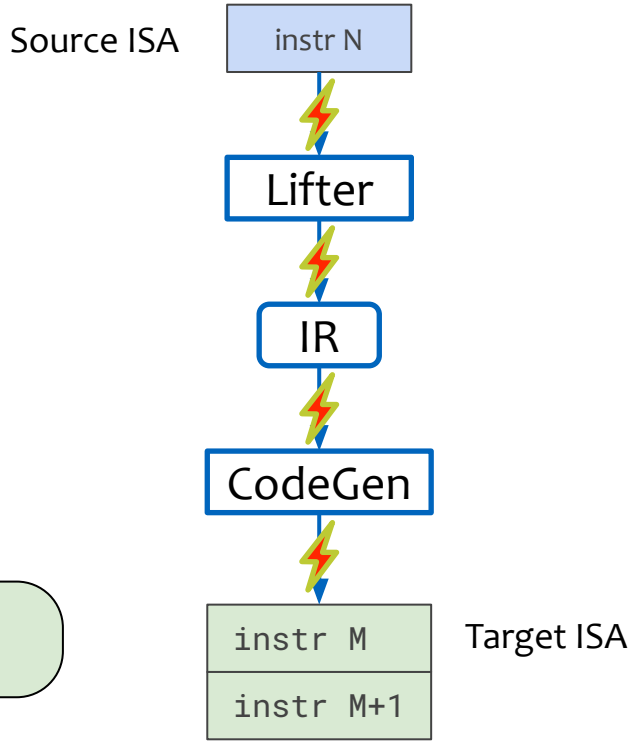


Program Emulators

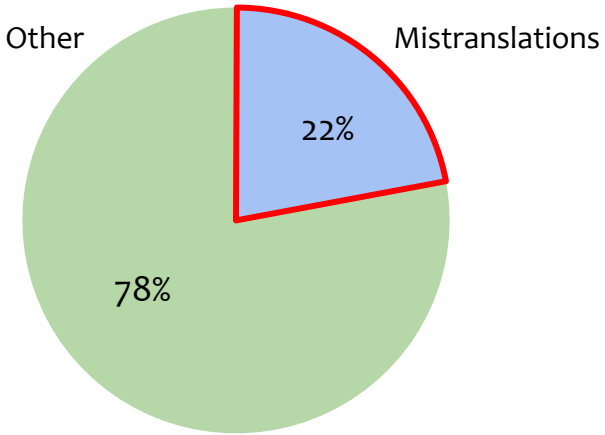
Bugs in emulators:

- Hard to find
- Subtle errors vs. program crashes
- Better tools are needed to validate emulators

What types of bugs exist in user-mode emulators and how can we detect them?



QEMU bug analysis

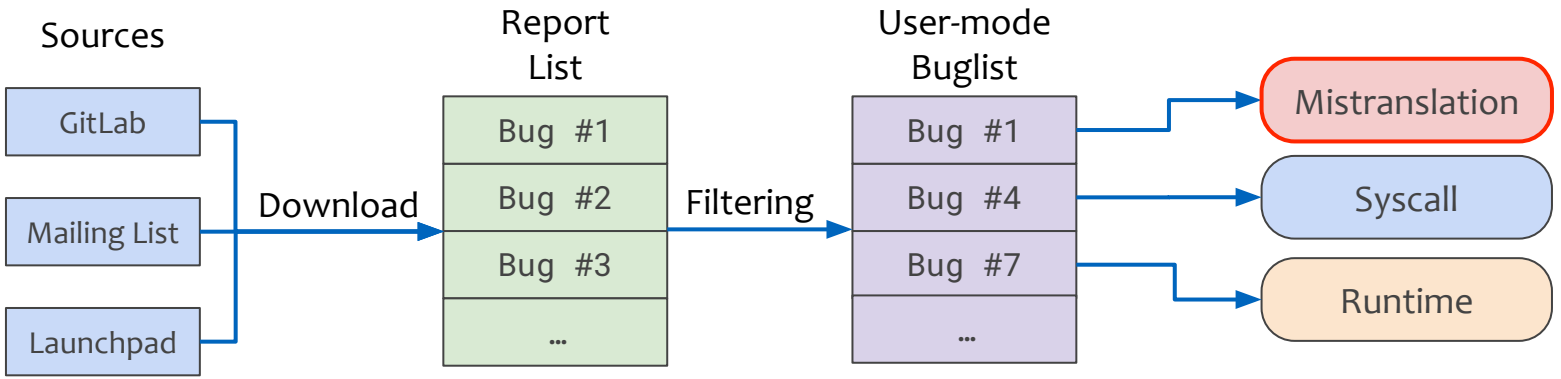


Manual classification of QEMU user-mode bugs

We lock the following results:

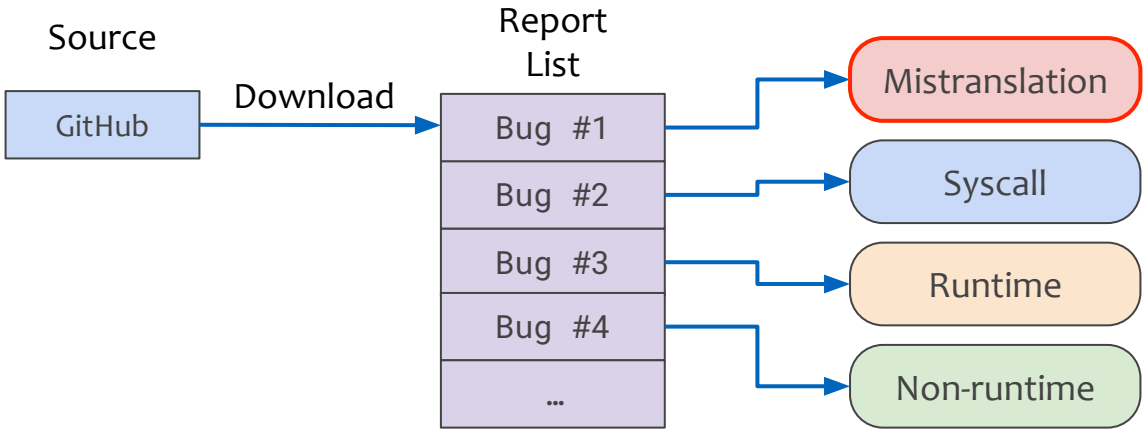
- 5812 bugs in total
- 551 user-mode bugs
- 119 mistranslation bugs

Overview: QEMU bug analysis



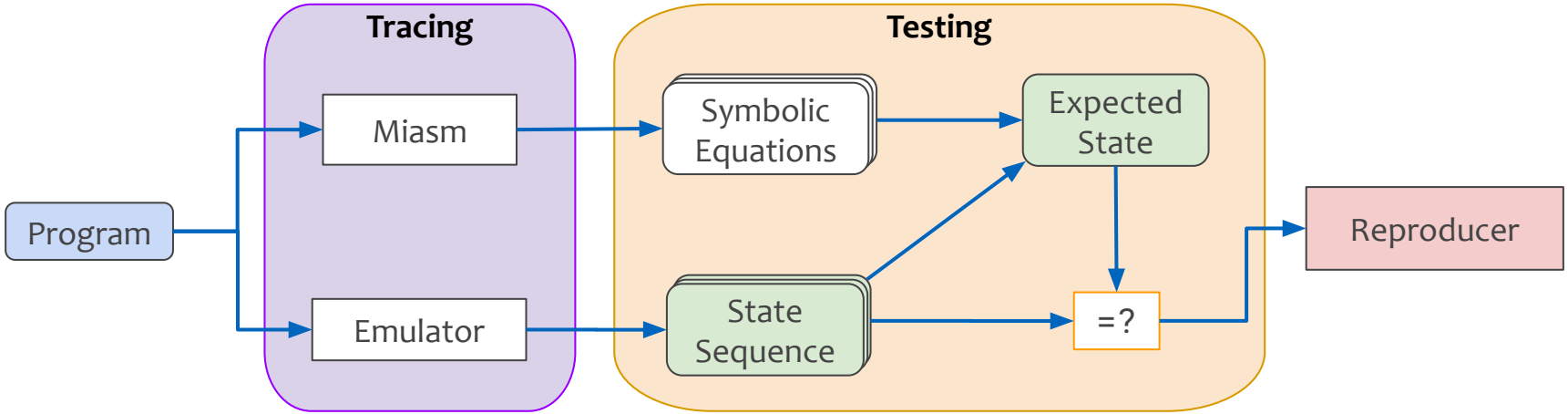
Filter all reports for user-mode bugs and classify each bug using **LLMs**

Box64 and FEX bug analysis



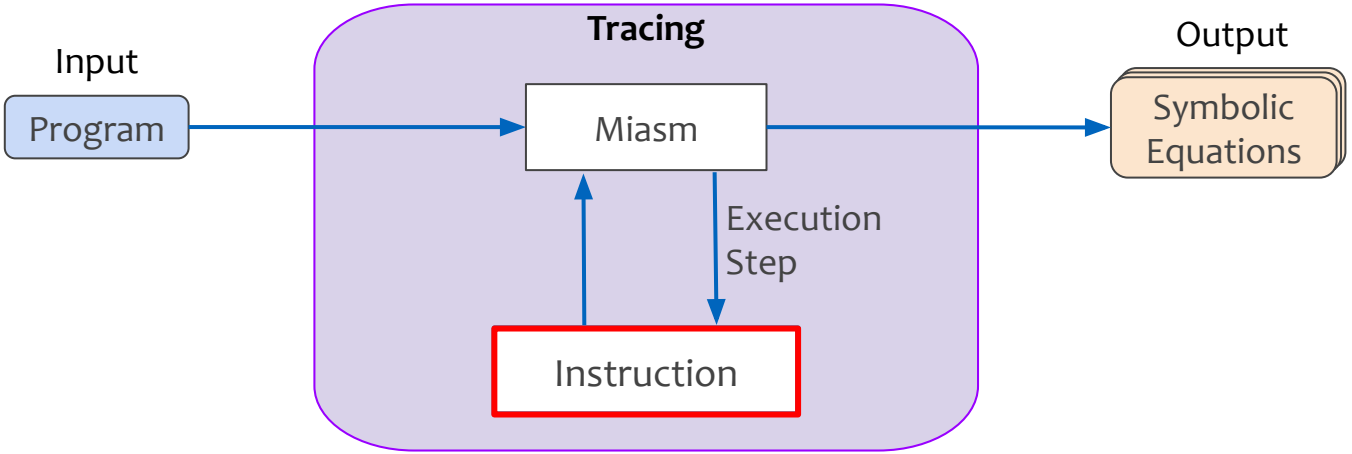
User-mode **only** emulators: discard non-runtime issues in one step

Focaccia



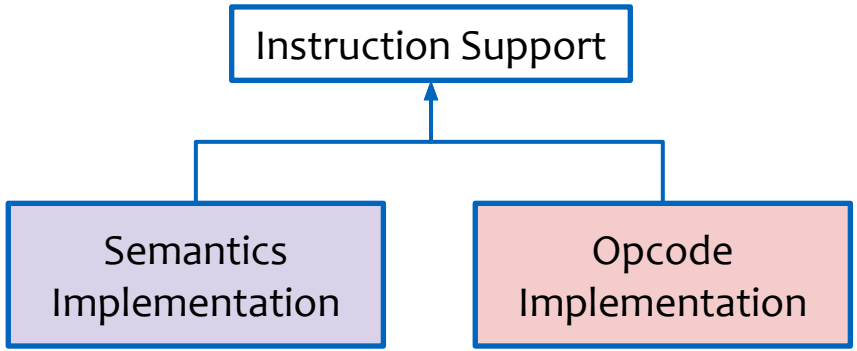
Focaccia can detect bugs and create reproducers

Symbolic Execution Backend - Miasm



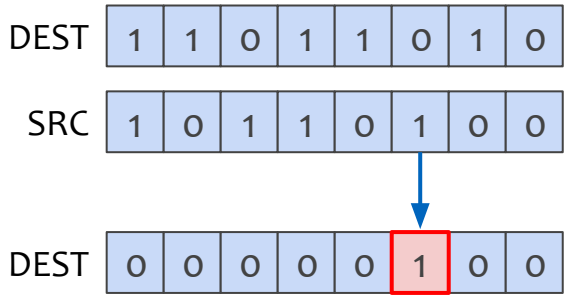
Focaccia's effectiveness depends on Miasm's completeness

Overview: Miasm Extension

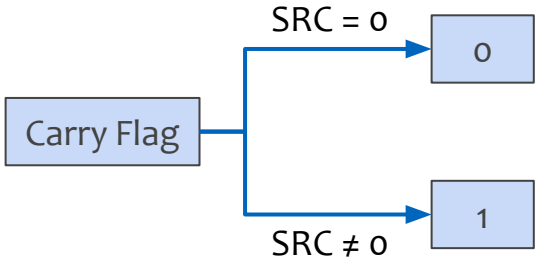


Implementation of instruction consists out of **two** different steps

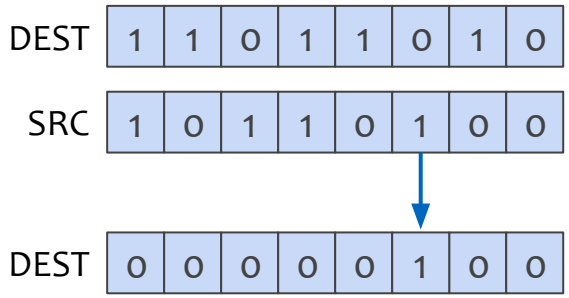
Miasm extension - QEMU BLSI bug

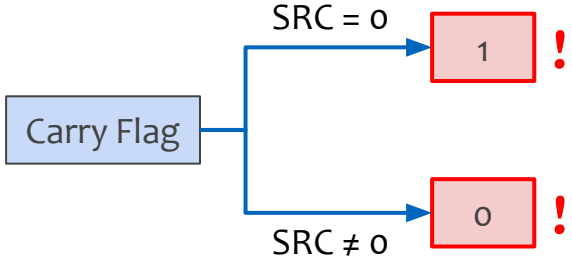


DEST = -SRC & SRC



Miasm extension - QEMU BLSI bug





Implementation of the instruction semantics can be **simple**

Miasm extension - BLSI Opcode

Opcode/ Instruction
VEX.LZ.0F38.W0 F3 /3 BLSI r32, r/m32
VEX.LZ.0F38.W1 F3 /3 BLSI r64, r/m64

Instruction Operand Encoding	
VEX.vvvv (w)	ModRM:r/m (r)

Purpose of opcode fields:

- VEX prefix with set constants
- Opcode
- Opcode extension in *reg* field
- Operands

Miasm extension - VEX prefix

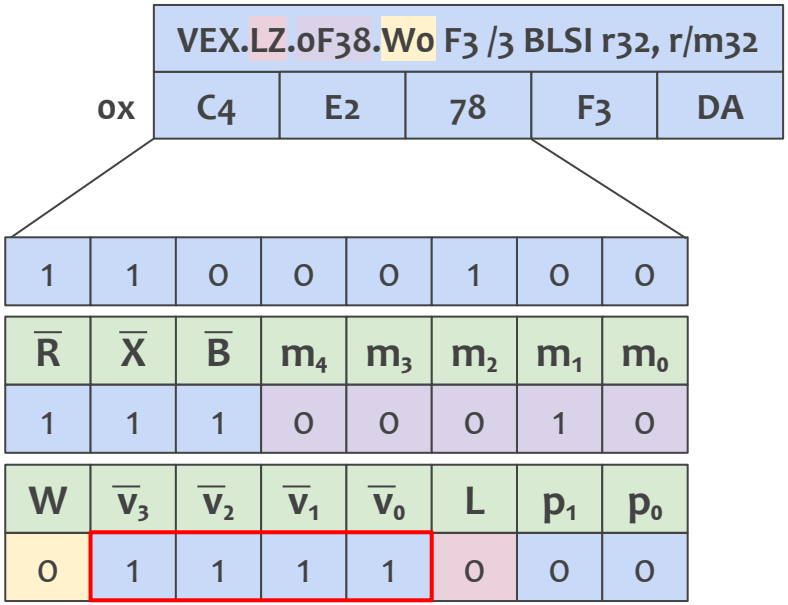
VEX prefix							
1	1	0	0	0	1	0	0
$\overline{R}$	$\overline{X}$	$\overline{B}$	m_4	m_3	m_2	m_1	m_0
W	$\overline{v_3}$	$\overline{v_2}$	$\overline{v_1}$	$\overline{v_0}$	L	p_1	p_0

Purpose of VEX fields:

- **R; X; B:** Fourth bit for register index fields
- **m:** Opcode extension
- **W:** Specifies 64-bit operands
- **v:** Additional source register index
- **L:** Specifies vector length of 128/256 bit
- **p:** Additional prefix bytes

Most of the logic is **already implemented** for the predecessor, the REX prefix

BLSI encoding - pre disassembly

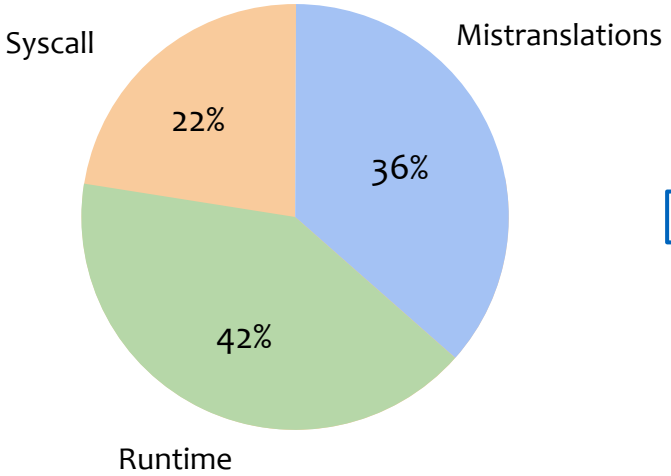


v = 0 ⇒ **eax**

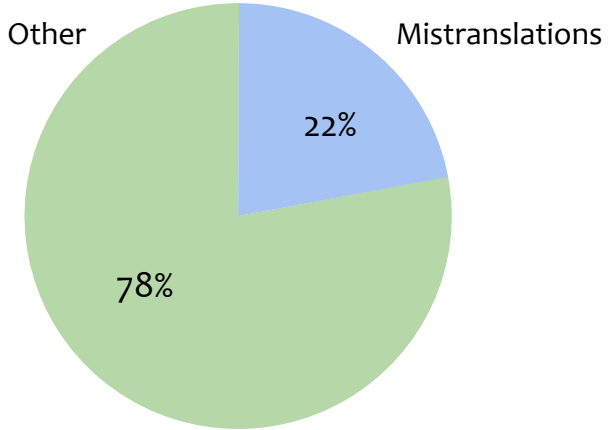
Evaluation Plan

- Bug study:
  - Setup: QEMU/Box64/FEX user-mode bug reports
  - Criteria: Significance (% mistranslation bug reports, quality)
  - LLM: Gemma3 (27b) using *Ollama*
  - Baseline: Manual classification on QEMU reports
- Implementation:
  - Setup: Miasm
  - Criteria: Effectiveness (# bugs reproduced, # newly supported instructions)
  - Benchmark: x86-64 Bit manipulation instructions (BMI)
  - Target Architecture: x86-64

Evaluation - QEMU

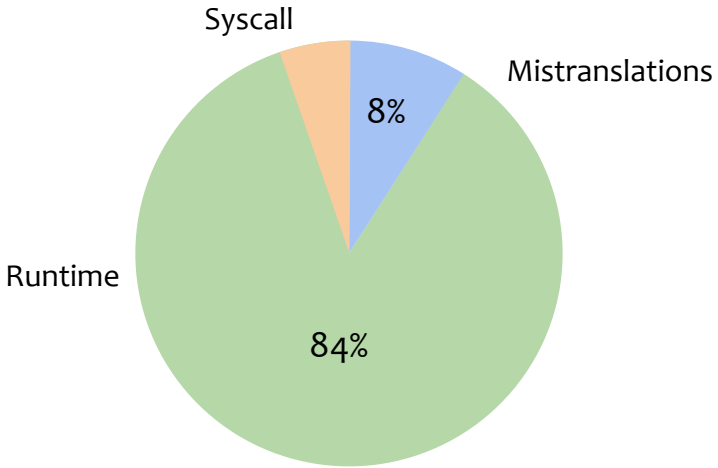


Classification of QEMU user-mode bugs using the Gemma3 (27b) LLM



End result of classification after manually filtering out false positives

Evaluation - Box64

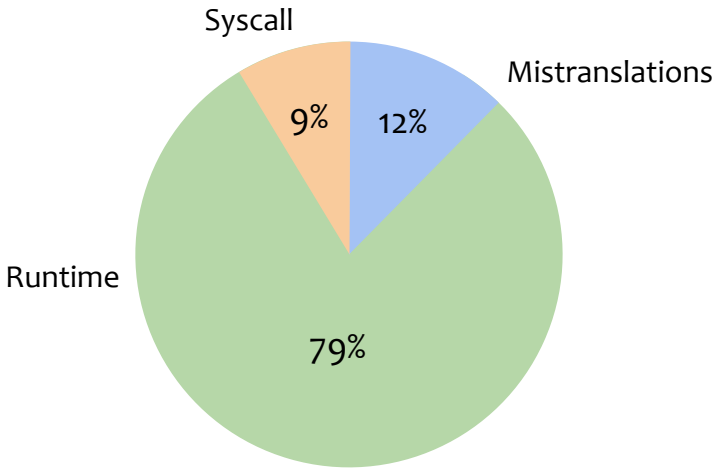


Classification of Box64 user-mode bugs using the Gemma3 (27b) LLM

Quality of bug reports is low:

- No user debugging
- Root causes are not discussed
- Potentially hide mistranslation bugs

Evaluation - FEX



Classification of FEX user-mode bugs using the Gemma3 (27b) LLM

Quality of bug reports is even lower:

- Half of reports are issued by the maintainers
- ‘Mistranslations’ are rather performance issues
- Results not meaningful for bug study

Evaluation - Reproducing BLSI bug

1. [ERROR] Content of register CF is false. Expected value: 0x0, actual value in the translation: 0x1.

Expected transformation: Symbolic state transformation 0x401040 -> 0x401045:
 [Symbols]
  ZF = (RAX & (-RAX))?(0x0,0x1)
  SF = (RAX & (-RAX))[63:64]
  OF = 0x0
  CF = RAX?(0x1,0x0)
  RBX = RAX & (-RAX)
  RIP = 0x401045
 [Instructions]
  BLSI RBX, RAX
Actual difference: Snapshot (x86_64): {'RSP': '0x0', 'RIP': '0x5'}

Focaccia's output for validating QEMU 7.2 on the BLSI instruction

Evaluation - BMI instructions

Instruction	Extension	Bug reproduced on	Supported
BLSI	BMI1	x86-64	✓
BEXTR	BMI1	x86-64	✓
ANDN	BMI1	-	✓
BLSMSK	BMI1	x86-64	✓
BLSR	BMI1	x86-64	✓
TZCNT	BMI1	-	✓
BZHI	BMI2	x86-64	✓

Conclusion

Takeaways:

- Mistranslation bugs are a **significant** cause of errors in user-mode emulators
- Data-driven bug analysis **aids** validation
- Validation in Focaccia needs a **correct** and **complete** symbolic execution backend
- Many mistranslations occur for **uncommon** instructions

Next steps:

- Expanding the symbolic execution backend of Focaccia to support more instructions
- Reproducing more bugs from the bug study