Silhouette: Efficient Protected Shadow Stacks for Embedded Systems

Jie Zhou
University of Rochester

Yufei Du
University of Rochester

Zhuojia Shen
University of Rochester

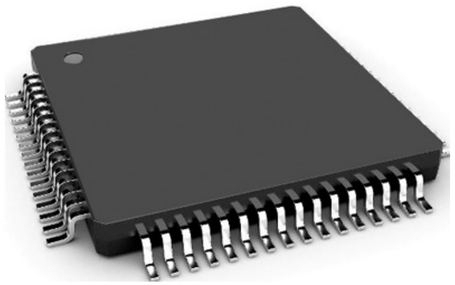
Lele Ma
College of William & Mary

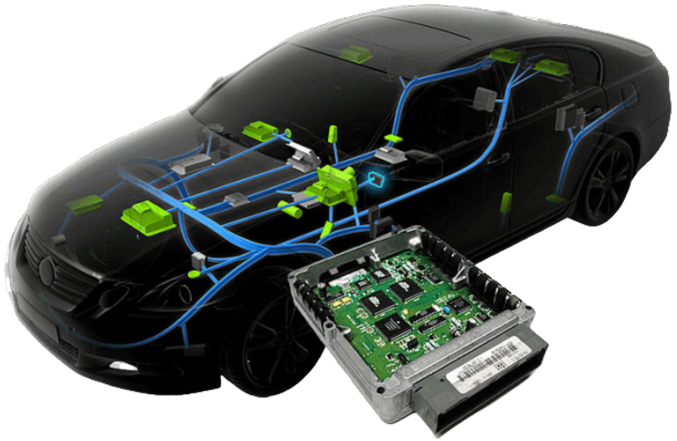
John Criswell
University of Rochester

Robert J. Walls
Worcester Polytechnic Institute

Presented at *USENIX Security 2020*

Microcontroller-based Systems are Almost Everywhere



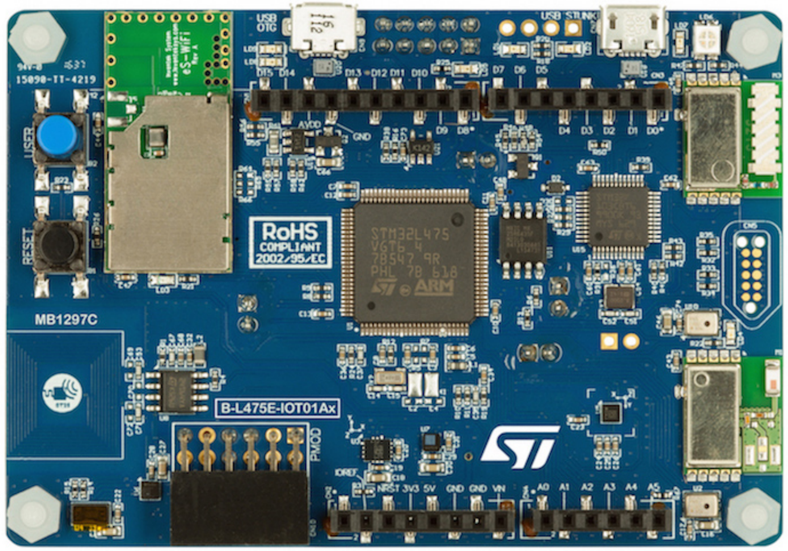


ECU

Bluetooth module

Wi-Fi module

Microcontroller-based Embedded Devices

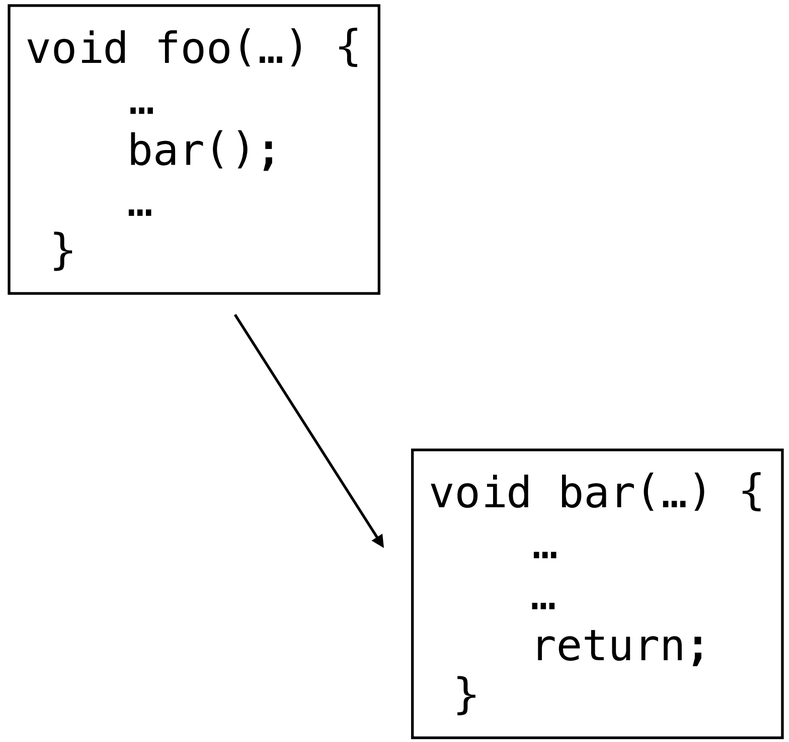


- Limited CPU speed
- Limited memory
- Real-time constraints
- Frequent direct operations on hardware

THE C PROGRAMMING LANGUAGE

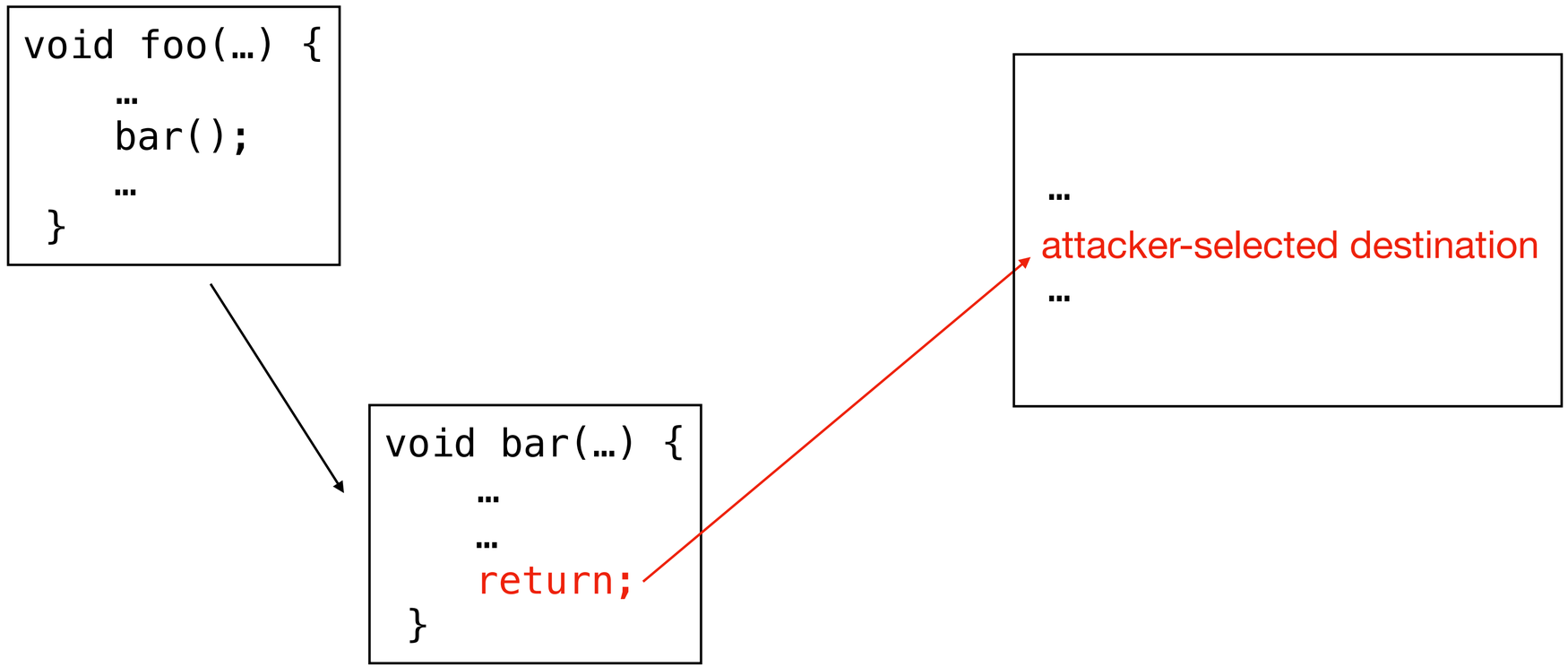
C is Not Memory Safe

Control-flow Hijacking: corrupting control-data to divert control flow to attacker-selected destinations

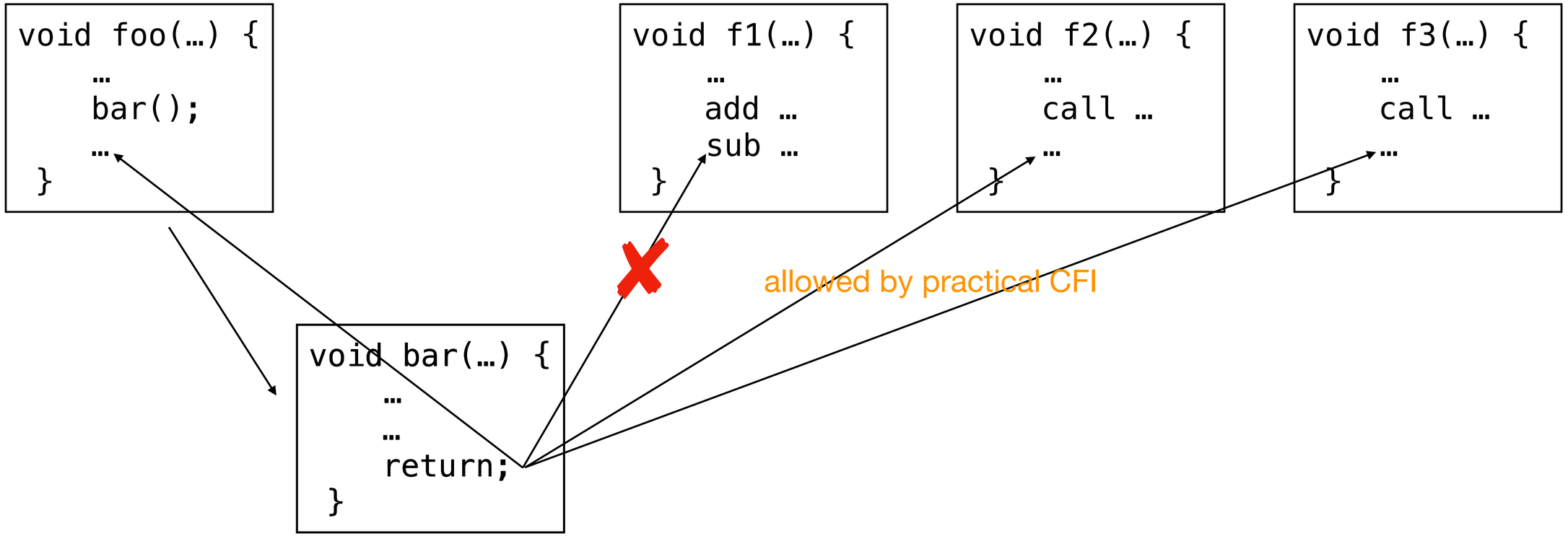


C is Not Memory Safe

Control-flow Hijacking: corrupting control-data to divert control flow to attacker-selected destinations



Control-Flow Integrity (CFI)



Common weakness of practical CFI*: allowing a return instruction to return back to ***multiple*** places

*Exploited by Out of Control *@Oakland'14,* ROP is Still Dangerous *@USENIX Security'14,* Control-flow Bending *@USENIX Security'15,* etc.

Silhouette

Silhouette: a compiler-based defense that

- guarantees the integrity of return addresses
- coarse-grained forward-edge CFI
- low performance overhead (1.3% and 3.4% overhead on two benchmark suites)

- Developed for **ARMv7-M** due to its popularity
- Also working on other ARM embedded processors

Outline

- **Silhouette Design**
- **Evaluation**
- **Summary**

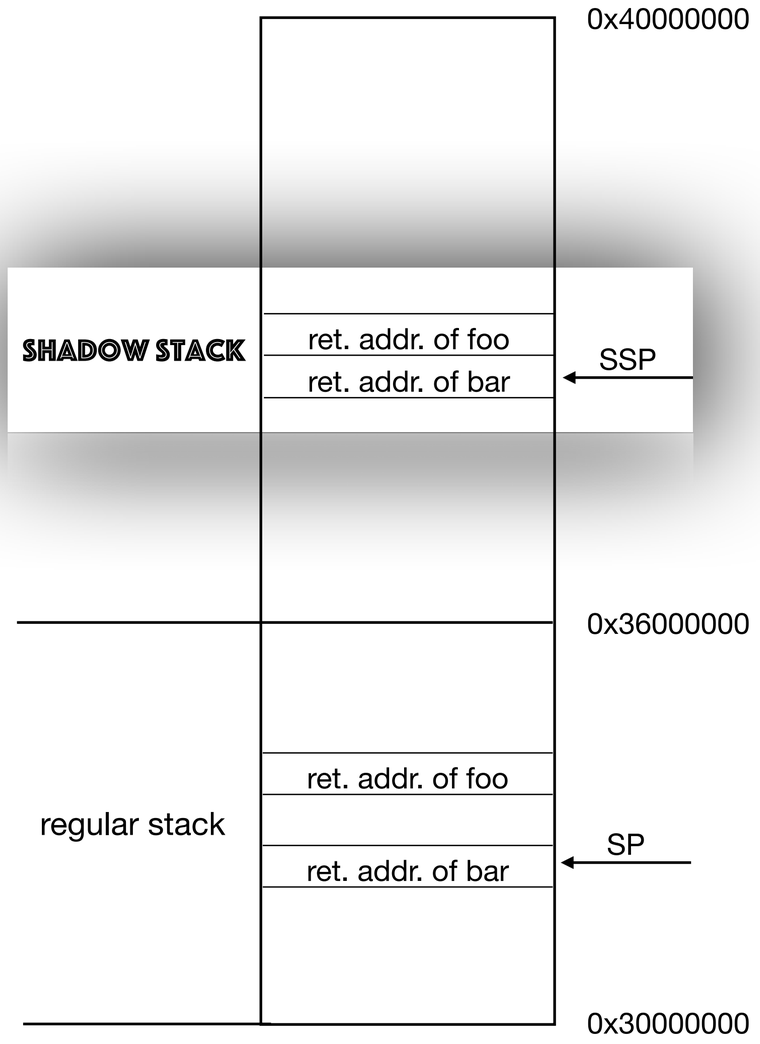
Outline

- **Silhouette Design**
- Evaluation
- Summary

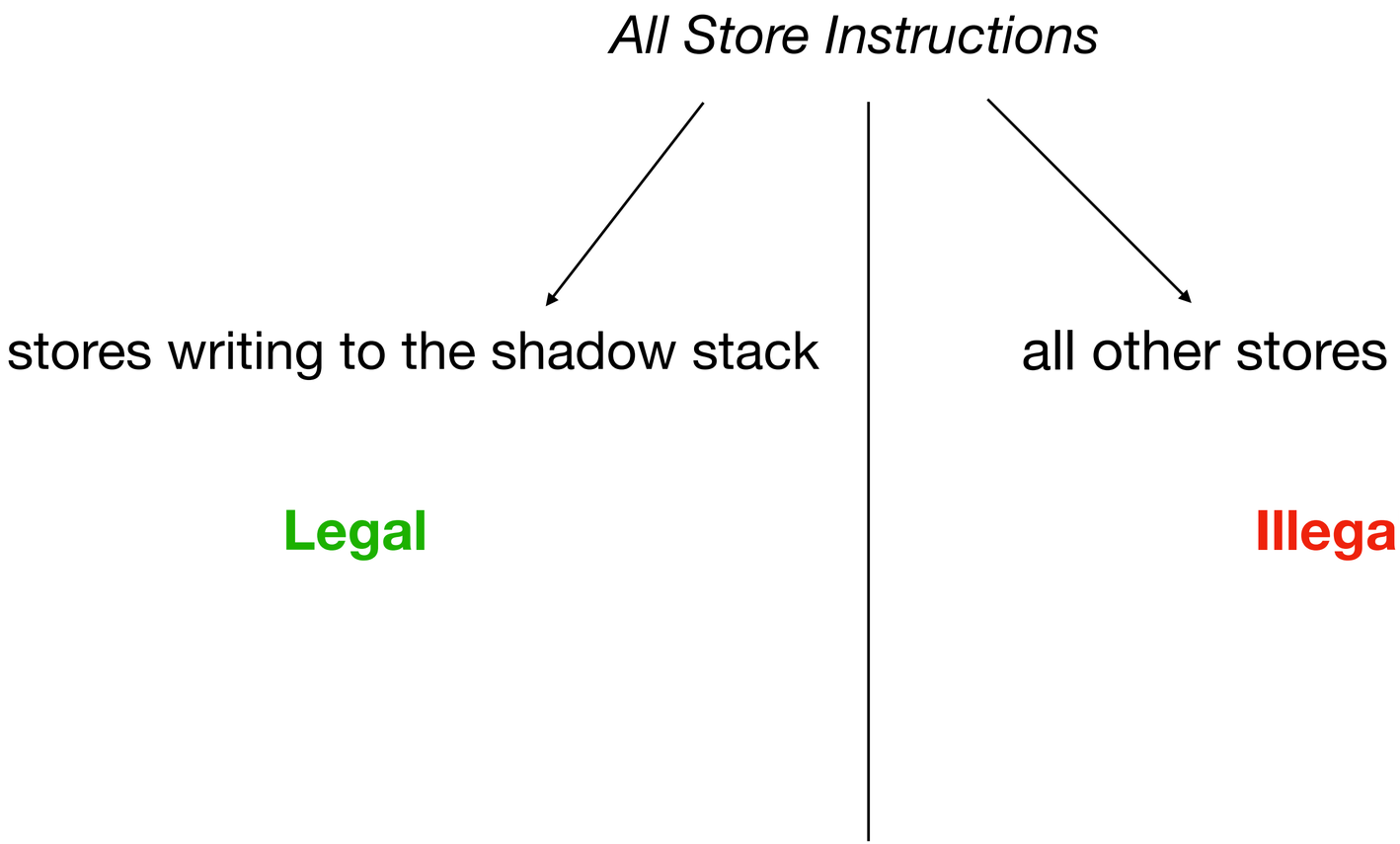
Shadow Stack

Protecting *return addresses*

Shadow stack itself also needs protection!



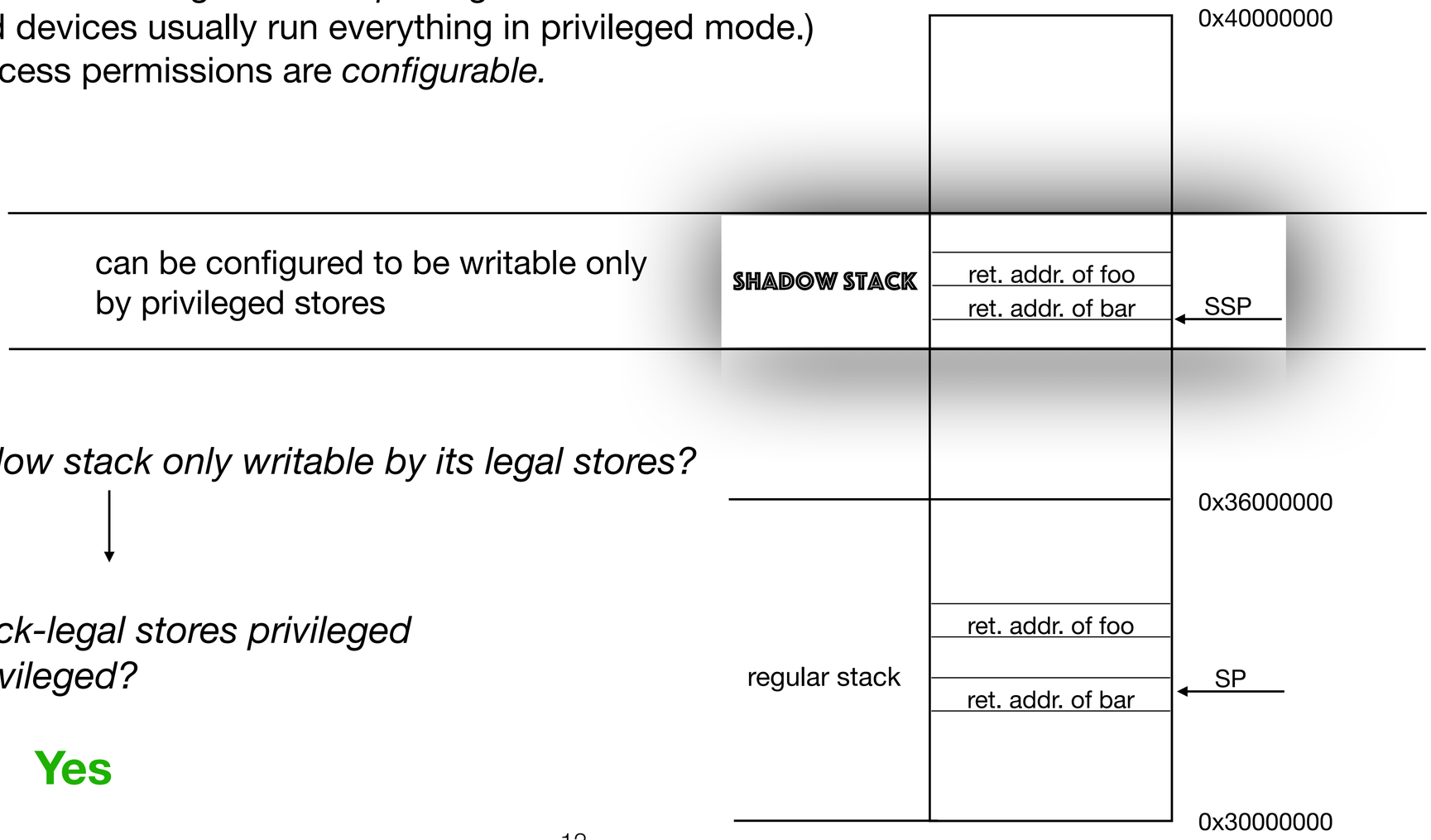
From a Shadow Stack's Point of View



Can we make the shadow stack writable only by its legal stores?

Background on ARMv7-M

- Execution Mode: *Privileged* and *Unprivileged.*
  (Embedded devices usually run everything in privileged mode.)
- Memory access permissions are *configurable.*



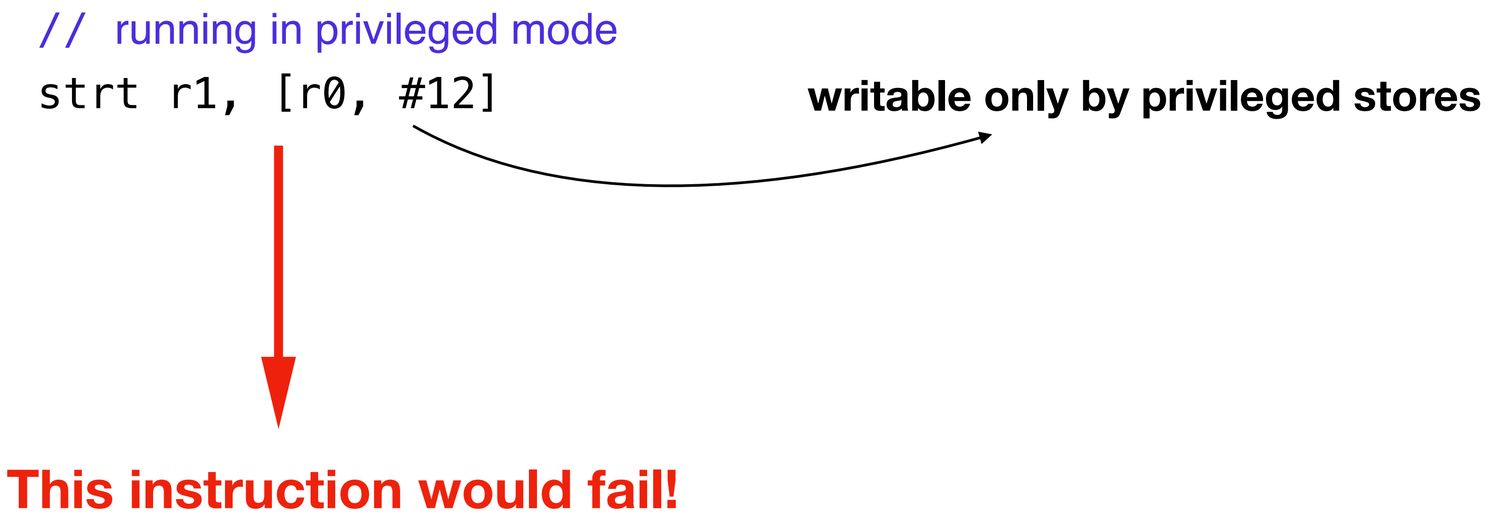
Can we make the shadow stack only writable by its legal stores?

Is it possible to make
- *only the shadow-stack-legal stores privileged*
- *all other stores unprivileged?*

Yes

Unprivileged Store

Act as if running in **unprivileged** mode when running in **privileged** mode.



Use Unprivileged Store to Protect Shadow Stack

- Configure the memory region for shadow stack to be writable only by privileged stores.

- Transform **all** stores to be unprivileged stores except
  - shadow-stack-legal stores
  - those that require to run as privileged such as some I/O-related operations.

Effect: even if memory is corrupted and control flow is diverted, illegal store instructions do not have write access to corrupt the shadow stack.

Store Hardening

Store Instructions of ARMv7-M

	Addressing Mode	Number of Types
Normal Store Instructions	source register, base register, offset register, immediate, left shift, write back, store multiple, floating-point stores	over 40
Unprivileged Store Instructions	source register, base register, immediate	3

Comparison of Normal and Unprivileged Store Instructions

Store Hardening Examples

```
// example 1

str r0, [r1, #4]
```

no performance overhead
no code size overhead

```
strt r0, [r1, #4]
```

```
// example 2

str r0, [sp, #-12]
```


performance and code size overhead

```
sub sp, #12
strt r0, [sp, #0]
add sp, #12
```

Forward-edge Control-flow Issues

- Transform **all** stores to be unprivileged stores except
  - shadow-stack-legal stores
  - **those that require to run as privileged**

Forward-edge Control Flow	How Silhouette Handles Them
Indirect Function Calls	Restricted by Label-based Forward-edge CFI
Large `switch` Statements	Compiled to Bounds-checked TBB or TBH instructions
Computed `goto` Statements	Transformed to `switch` statements

Silhouette Architecture

Simplified Architecture of Silhouette

Security guarantee:

- Return instruction always returns to its legal destination
- Forward-edge control flows are restricted to selected destinations

Outline

- Silhouette Design
- **Evaluation**
- Summary

Experiment Setup

Evaluated both performance and code size overhead

Development board: **STM32F469**

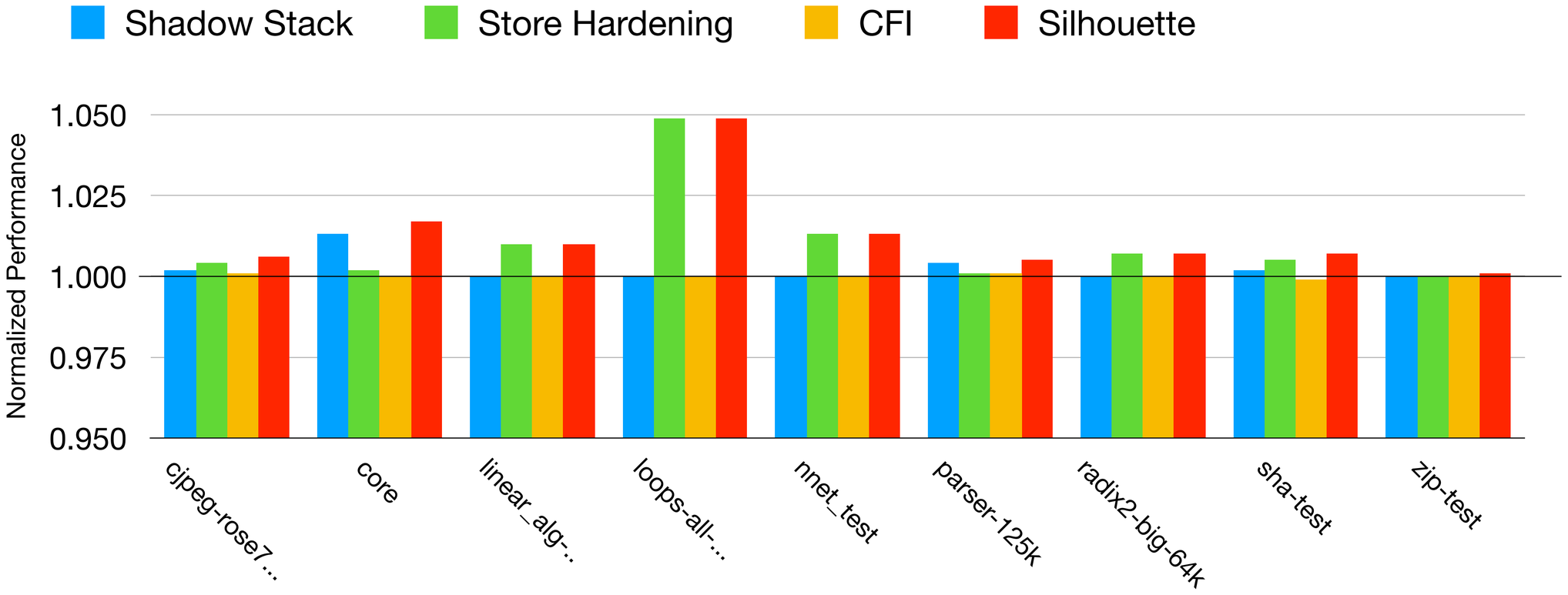
- Cortex-M4 processor, run at 180 MHz
- 384 KB SRAM
- 16 MB SDRAM
- 2 MB Flash Memory

Benchmarks: all 9 programs in **CoreMark-Pro**
29 programs in **BEEBS**

Base compiler: Clang/LLVM 9.0

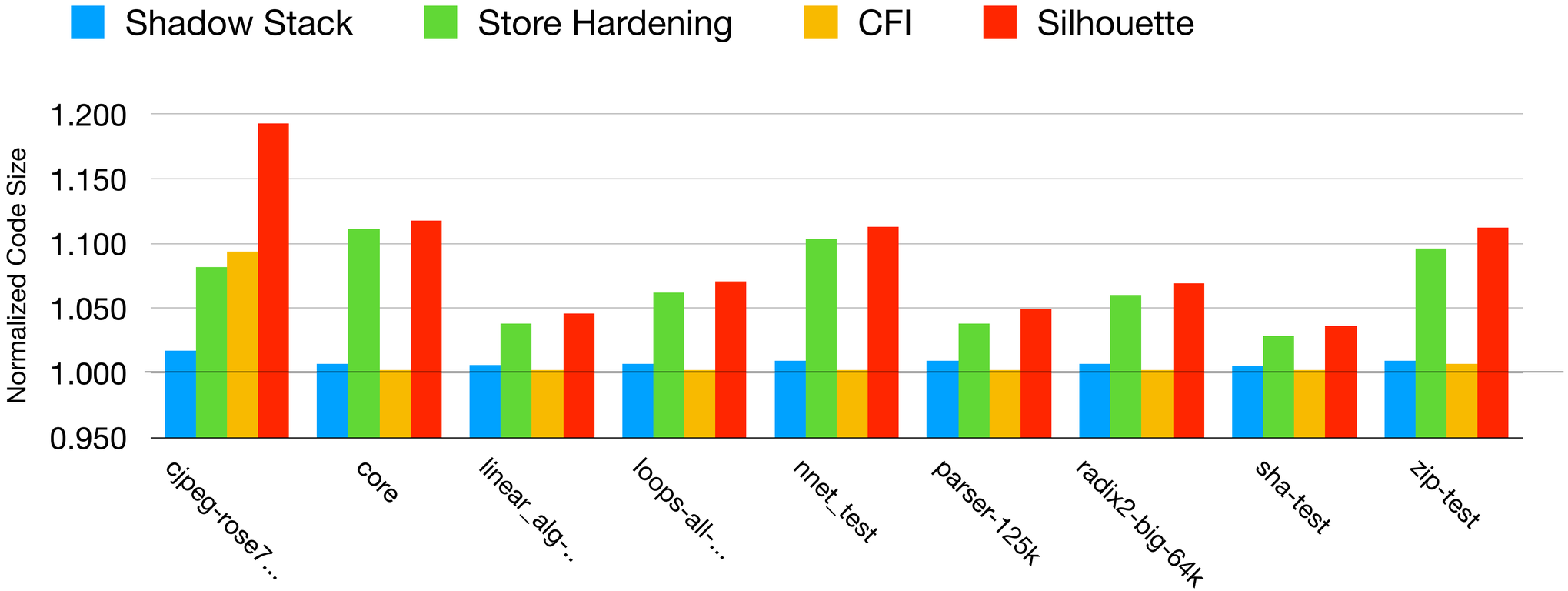
Optimization level: -O3

Performance on CoreMark-Pro Benchmarks



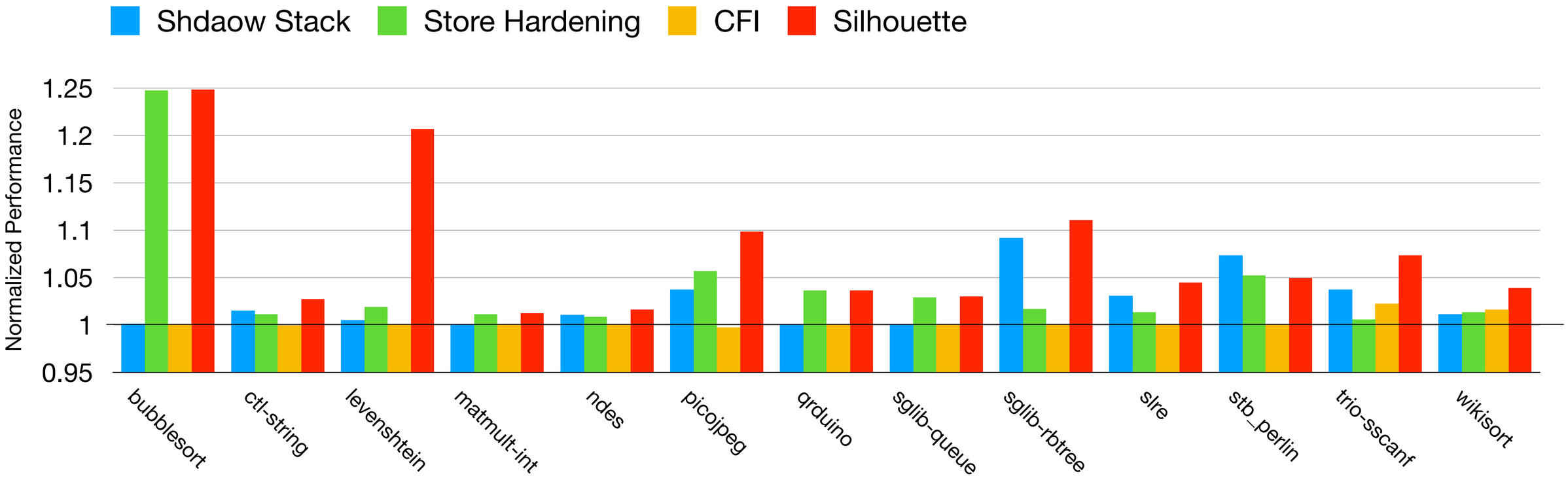
	Shadow Stack	Store Hardening	CFI	Silhouette
Min	0	0	-0.1%	**0.1%**
Max	1.3%	4.9%	0.1%	**4.9%**
Geo. Mean	0.2%	1%	0	**1.3%**

Code Size on CoreMark-Pro Benchmarks



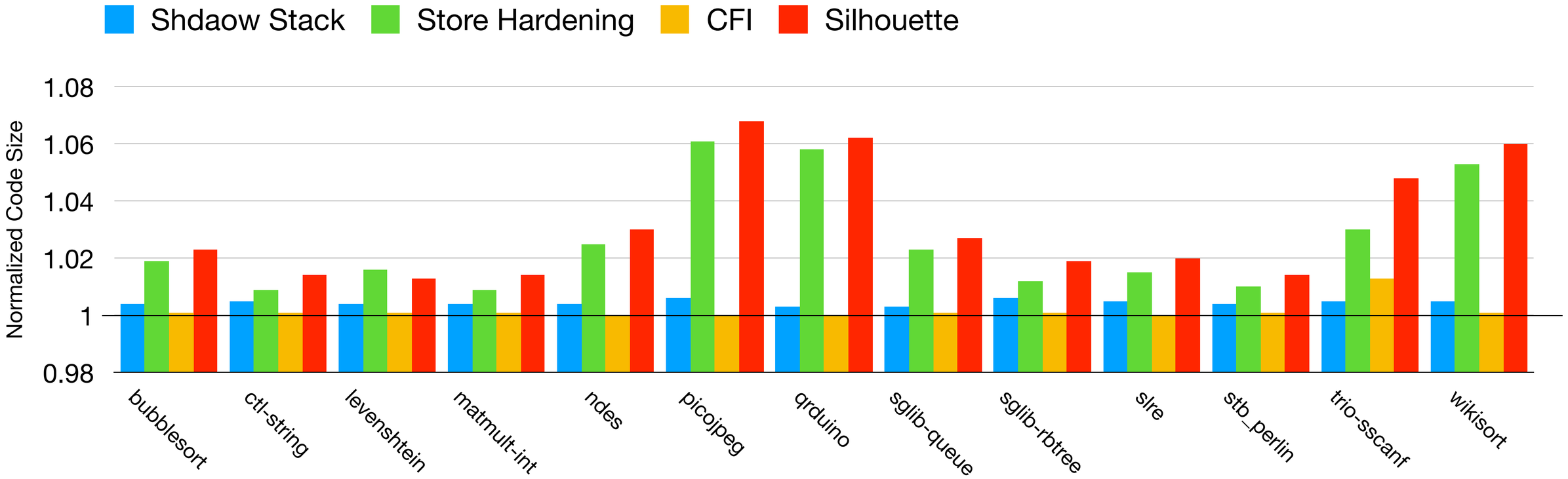
	Shadow Stack	Store Hardening	CFI	Silhouette
Min	0.5%	2.8%	0.2%	**3.6%**
Max	1.7%	11.1%	9.4%	**19.3%**
Geo. Mean	0.8%	6.8%	1.2%	**8.9%**

Performance Overhead on BEEBS Benchmarks



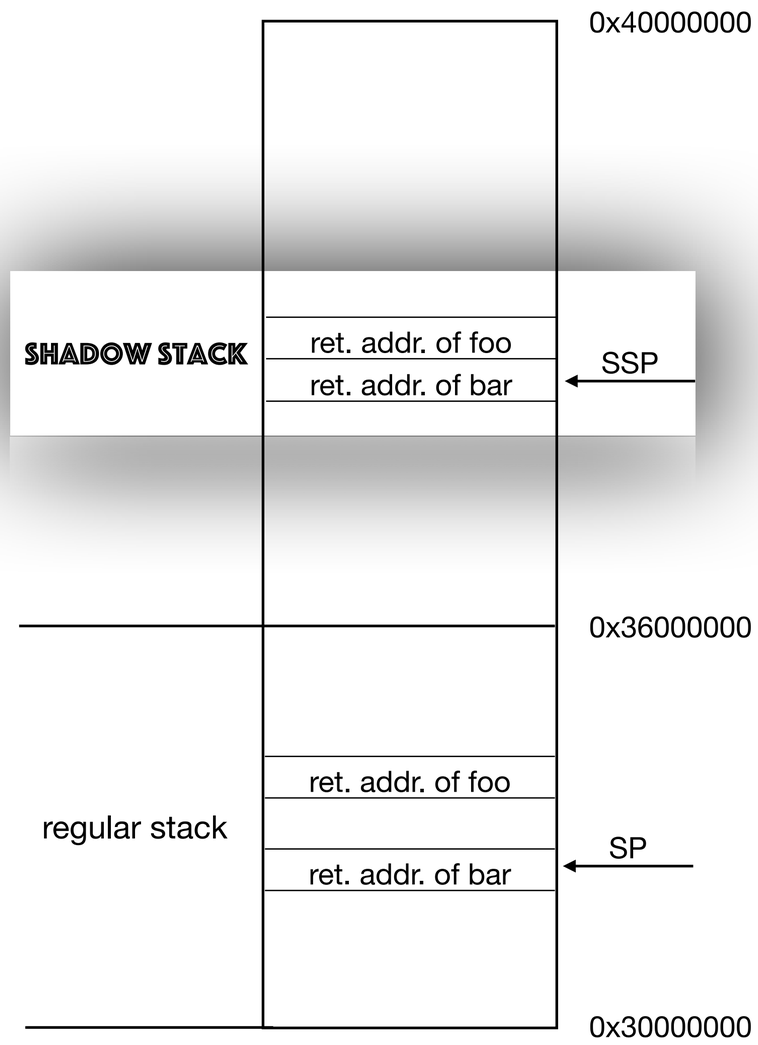
	Shadow Stack	Store Hardening	CFI	Silhouette
Min	0	-0.3%	-0.3%	**-0.3%**
Max	9.2%	24.7%	2.2%	**24.8%**
Geo. Mean	1.1%	1.8%	0.1%	**3.4%**

Code Size on BEEBS Benchmarks



	Shadow Stack	Store Hardening	CFI	Silhouette
Min	0.3%	0.5%	0	**0.9%**
Max	0.6%	6.1%	1.3%	**6.8%**
Geo. Mean	0.4%	1.8%	0.1%	**2.3%**

Silhouette-Invert



Writable only by unprivileged stores
*but **not** by privileged stores?*

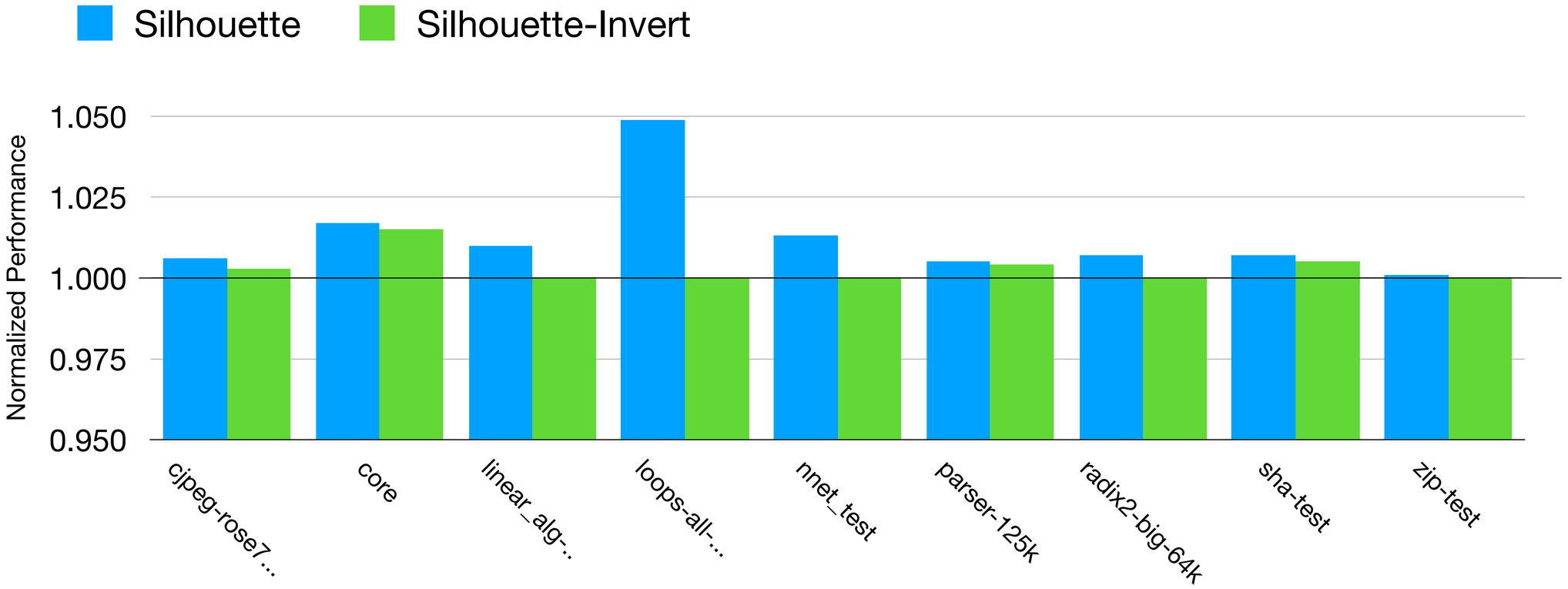
Silhouette-Invert

- Configure shadow stack to be unprivileged-write-only
- Transform shadow-stack-legal stores to be unprivileged
- Leave all other stores unchanged

Not supported on ARMv7-M

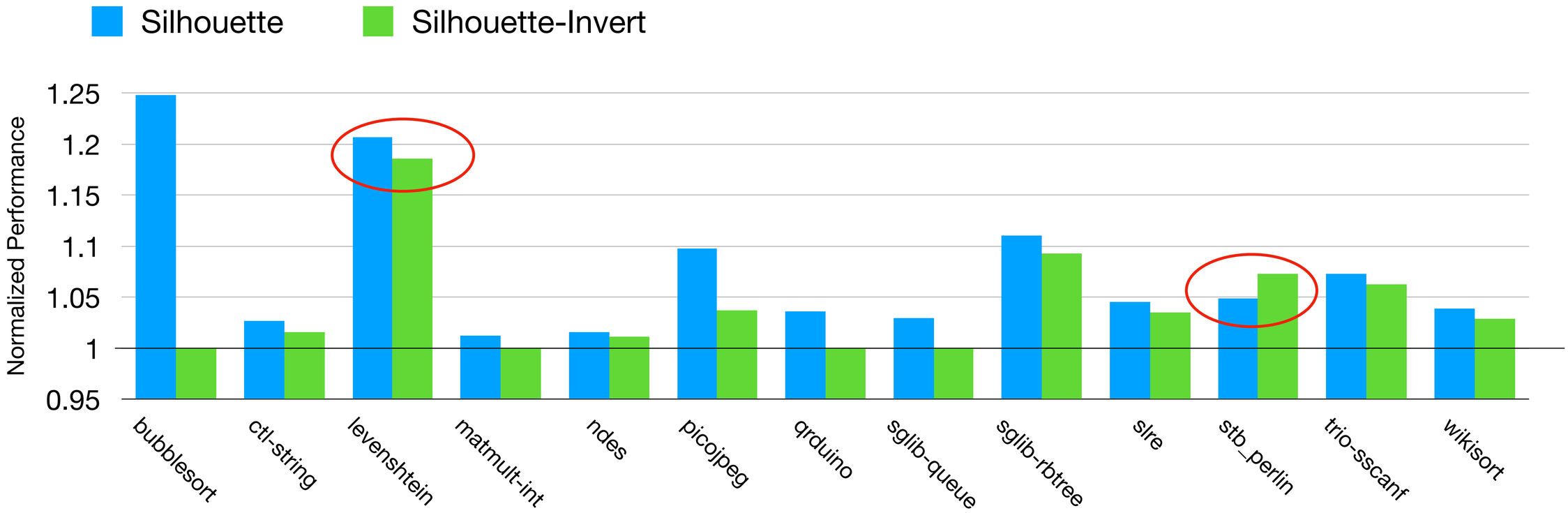
Proposed two solutions with minor hardware changes.
See the paper for details.

Silhouette v.s. Silhouette-Invert on CoreMark-Pro



	Silhouette	Silhouette-Invert
Min	0.1%	0
Max	4.9%	1.5%
Geo. Mean	1.3%	0.3%

Silhouette v.s. Silhouette-Invert on BEEBS



	Silhouette	Silhouette-Invert
Min	-0.3%	0
Max	24.8%	18.6%
Geo. Mean	3.4%	1.9%

Summary

- Silhouette: an efficient defense to protect return addresses for ARM embedded systems
- Low performance and code size overhead
- Silhouette-Invert:
  - Further decreases performance and code size penalty
  - Minor hardware change
- Open-Source: https://github.com/URSec/Silhouette

Contacts:
Jie Zhou (jzhou41@cs.rochester.edu)
Zhuojia Shen (zshen10@cs.rochester.edu)
John Criswell (criswell@cs.rochester.edu)

Question?