RISC-V® Summit

Compiler Code Size Density

Nidal Faour
Toolchain Engineer
Western Digital

Agenda

- Code size on RISCV
  - why there is a problem?
- Research
  - Addressing the problem
- Solutions
  - GCC patch example
- Going forward
  - Research
  - Techniques
  - Test-cases: Embench
- Summary

Code size on RISC-V

Code size on RISCV

- RAM is the most expensive resource in the world of embedded devices.
- RISC-V ISA does not provide a multi operations in a single instruction form.
- RISC-V GCC & LLVM are 10-20% behind other ISA

Research

Research

- Trying out RISC-V on a real project
- Inspect the assembly for both targets, RISC-V and WD legacy vendor
- Finding weak points: flows where the compiler should have made a better decisions.

✗ • ISA limitations will not be addressed

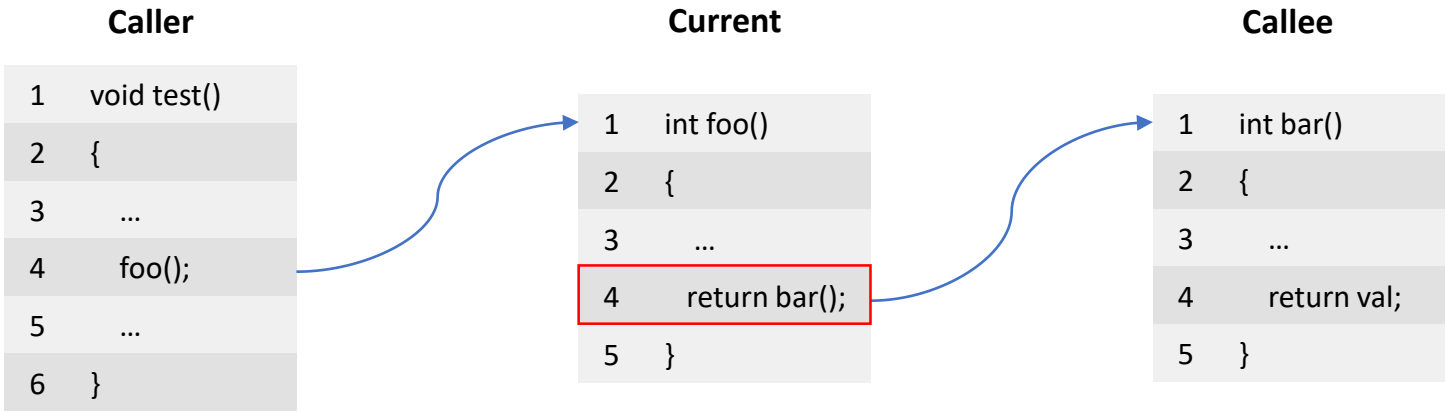
✓ • Compiler immaturity can be improved

Solutions

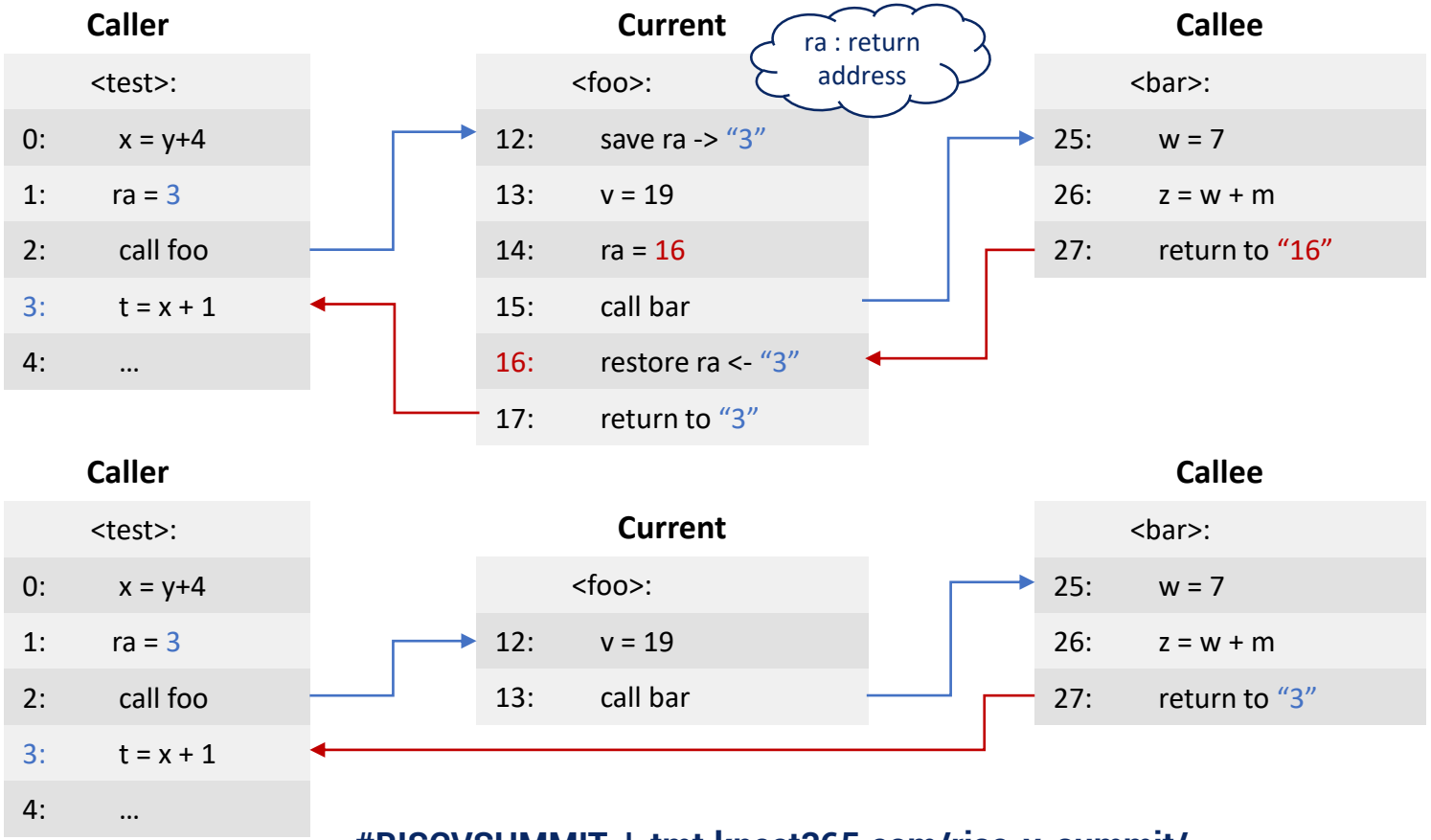
Solution – Issue & fix A

- save/restore patch:
  Unneeded prologue/epilogue functions. The GCC compiler had injected these calls where it should not.

  Example: On a tail call the return to the **caller** can be done from the **callee**.



Solution – Issue & fix A



Solution – Issue & fix B

Common st/ld base address:

- In a stream of memory access, the GCC compiler was not compress aware.
- The access was made using the address with a large offset which left us with non compressed 32 bits instructions
- Using a register to rebase the address then step in small offsets, gave the opportunity to use compressed 16 bits instruction
- Using the compressed instruction gave the same stream of instruction a reduction of 10% in code size.

Solution – Issue & fix B

Load instruction	= 4 bytes	Load instruction	= 4 bytes
		Rebase instruction	= 4 bytes
Store/load	= 4 bytes	C. Store/load	= 2 bytes
Store/load	= 4 bytes	C. Store/load	= 2 bytes
	= 12 bytes		**= 12 bytes**

Load instruction	= 4 bytes	Load instruction	= 4 bytes
		Rebase instruction	= 4 bytes
Store/load	= 4 bytes	C. Store/load	= 2 bytes
Store/load	= 4 bytes	C. Store/load	= 2 bytes
Store/load	= 4 bytes	C. Store/load	= 2 bytes
	= 16 bytes		**= 14 bytes**

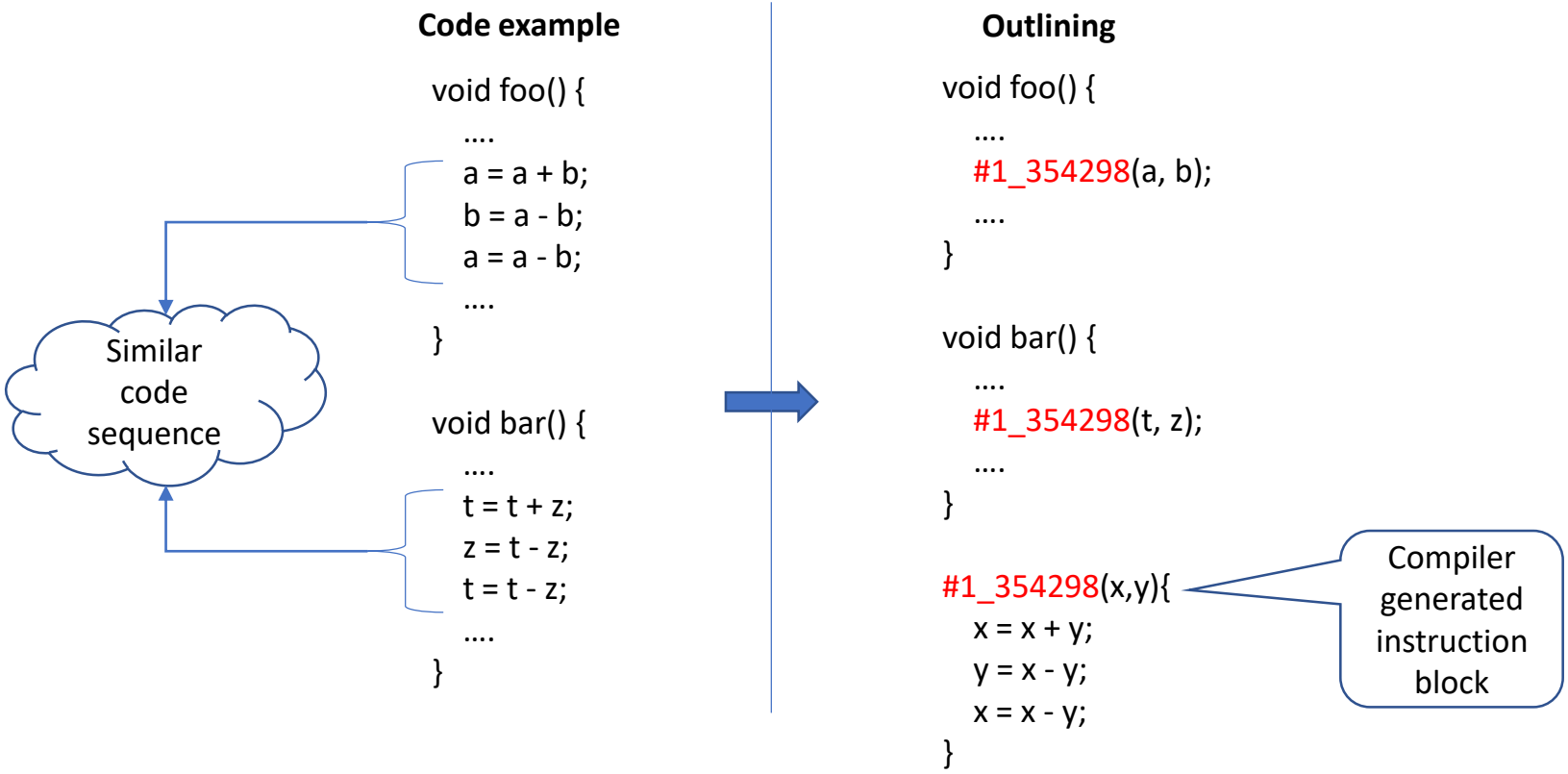
$$f(x) = C + \frac{x}{2} \cong \frac{x}{2} \quad \rightarrow 50\% \text{ max}$$

Going forward

Going forward

- Same research is on going over LLVM
- Outlining will be provided for the LLVM compiler
- More density test cases will be added to the Embench

Going forward - outlining



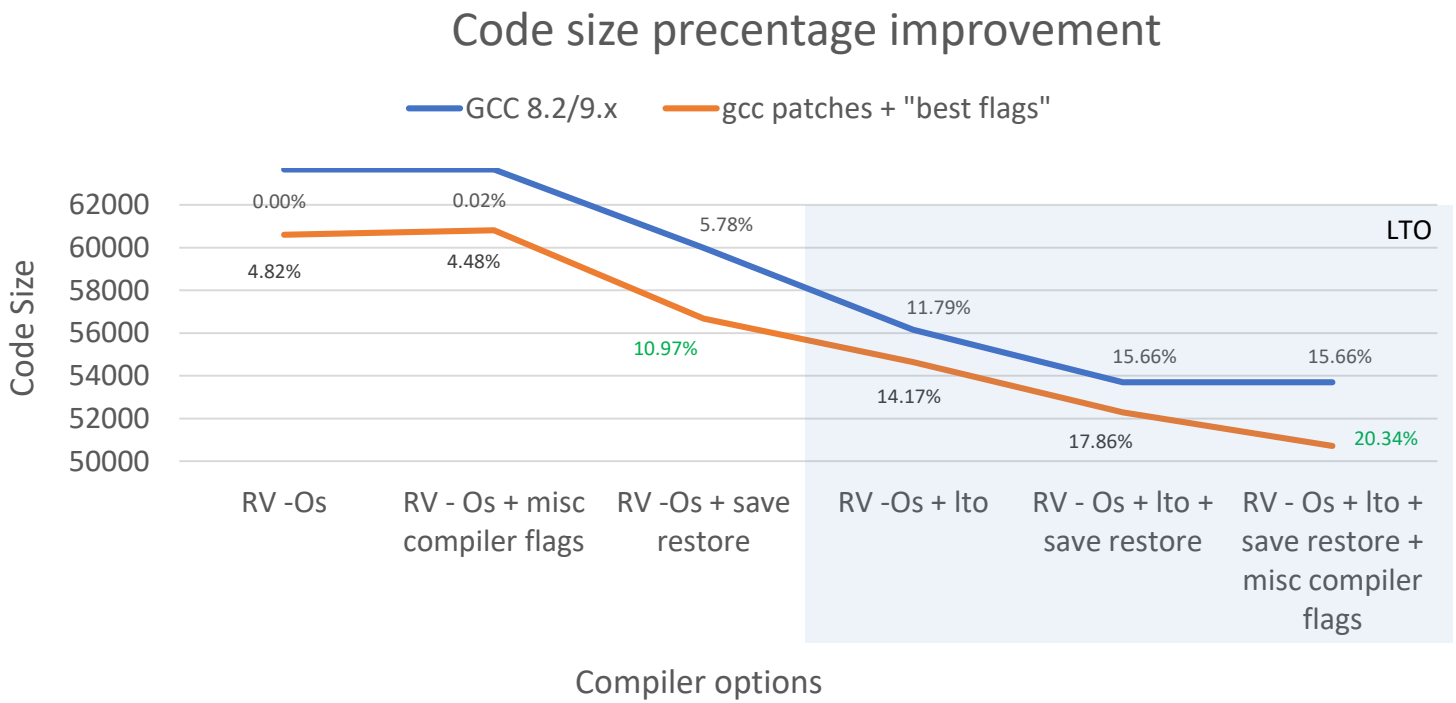
Going forward - Embench

- Code size measurement
- Lack of code size open benchmarks
- Embench – benchmark suite for embedded systems
- [illegible] cases for c[illegible]

https://embench.org/
https://github.com/westerndigitalcorporation/riscv32-Code-density-test-bench

Summary

- Current effort results – 5% improvement to code size



Thank you