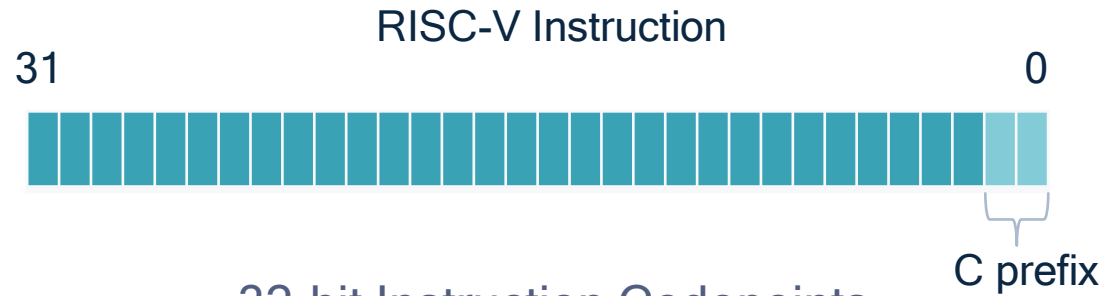# Qualcomm

A Case to Remove C from App Profiles
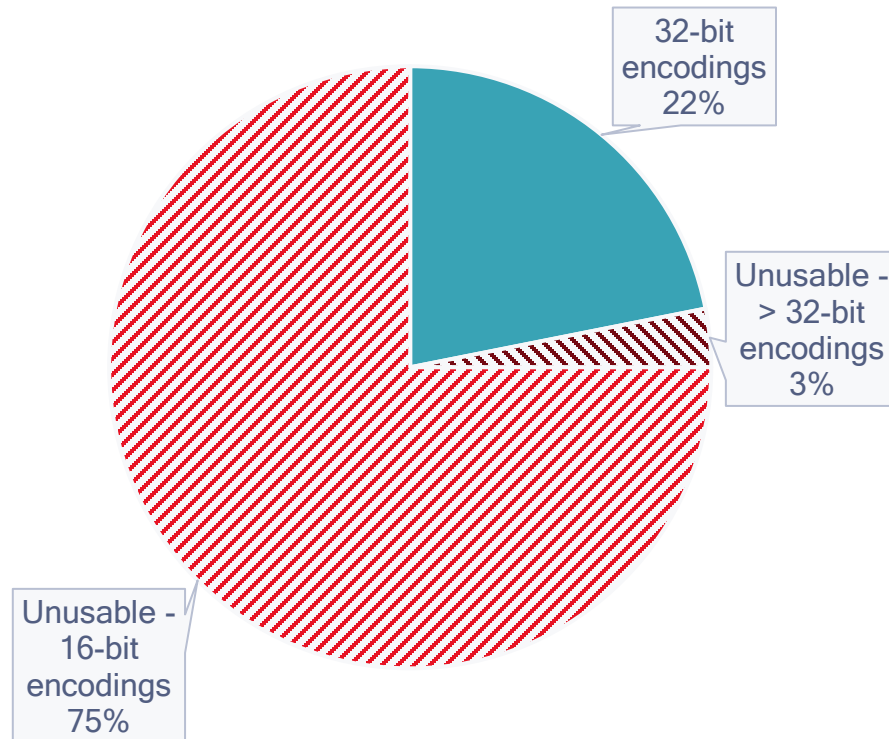
Derek Hower, James Ball, Conrado Blasco, Manu Gulati

Qualcomm Technologies, Inc.

C Extension – Compressed (16-bit) instruction encodings

RISC-V Instruction

31   0

C prefix

32-bit Instruction Codepoints

- 32-bit encodings 22%
- Unusable - > 32-bit encodings 3%
- Unusable - 16-bit encodings 75%

Facts

- C defines compressed re-encodings of common 32-bit instructions
  - **No new instructions** – always 1:1 mapping between 16/32-bit
- 2 LSBs of instruction identify length
  - 3 of 4 values indicate 16-bit
  - 1 of 4 values indicate >= 32-bit
- C binaries are packed: 32-bit encodings can be unaligned

Problems with C in high performance designs

Pushes RISC-V to large opcodes

- RISC-V is nearly out of 32-bit opcodes
- **Forced transition** to > 32-bit opcodes will *degrade code size*
  - Eventually, *RV64 with 32-bit encodings will have better code size than RV64C + large encodings* as new instructions are added and used

Code Size - SPEC 2006 GEOMEAN

Legend: aarch64, x86_64, rv64g
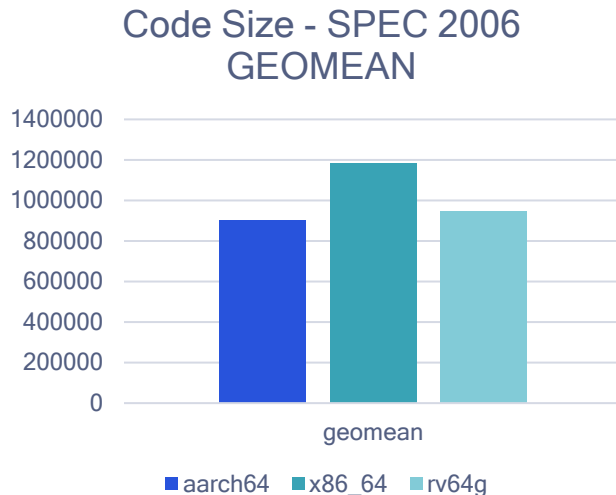
Substantial design complexity

- Unaligned fetch is **challenging to design, verify**
  - Cache line, page crossing instructions
  - Increased wire delay/muxing
- Leads to designs that are:
  - **more expensive** (NRE)
  - **slower** (extra pipe stages)
  - **buggy** (see Intel Jump Code Conditional)

> "Under complex micro-architectural conditions involving branch instructions bytes that span multiple 64 byte boundaries (**cross cache line**), unpredictable system behavior may occur."
> – Intel Xeon Erratum SKX102
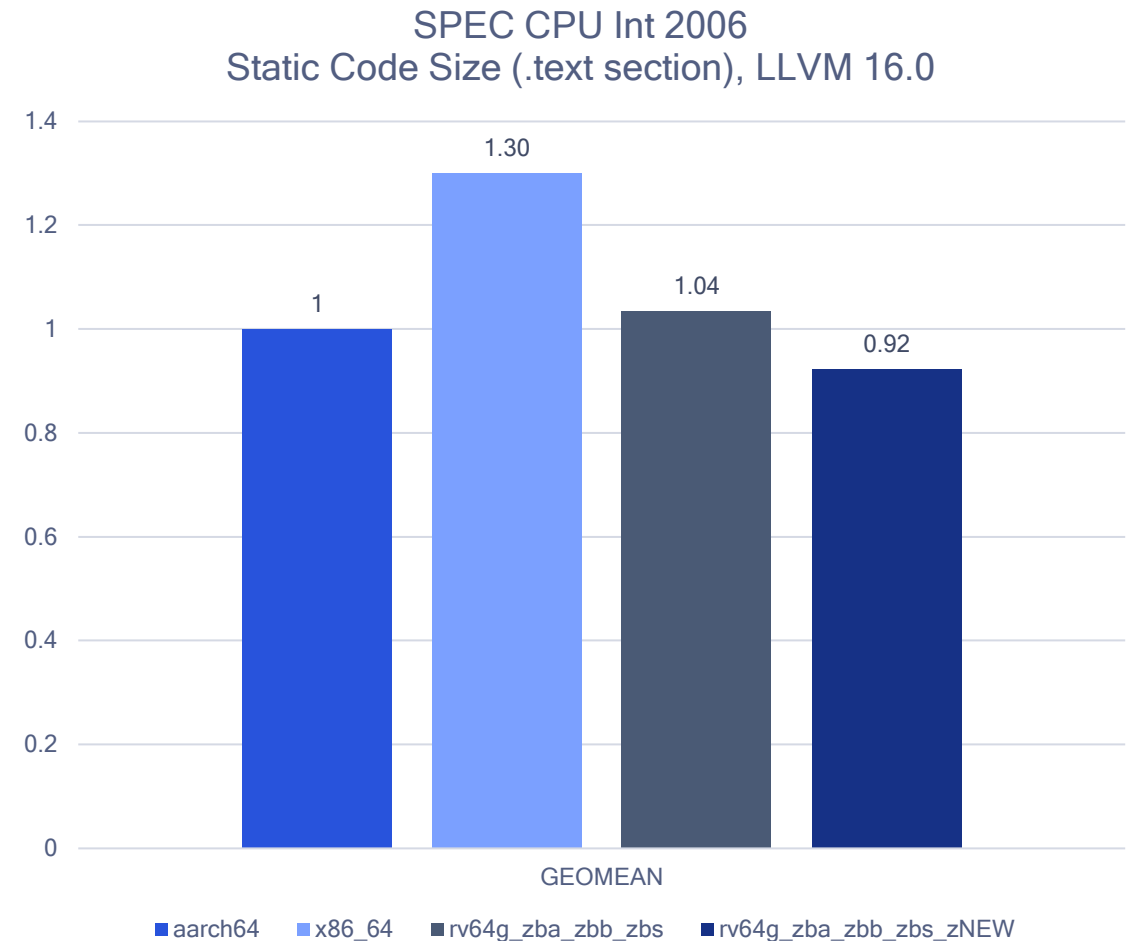>
> Fix results in 0-8% slowdown.

Performance benefit is modest

- Best case: 2-3% speedup
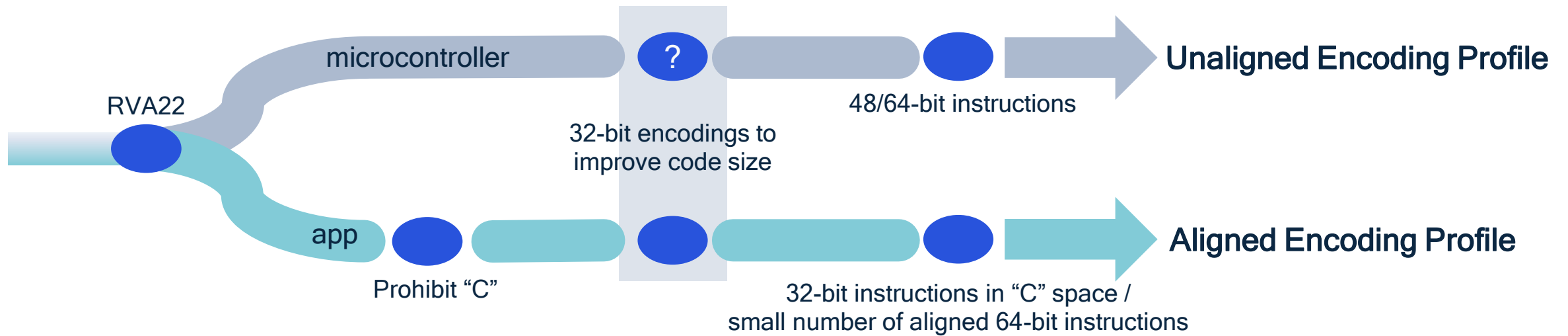- Often: slowdown (net negative when program fits in icache)

Speedup
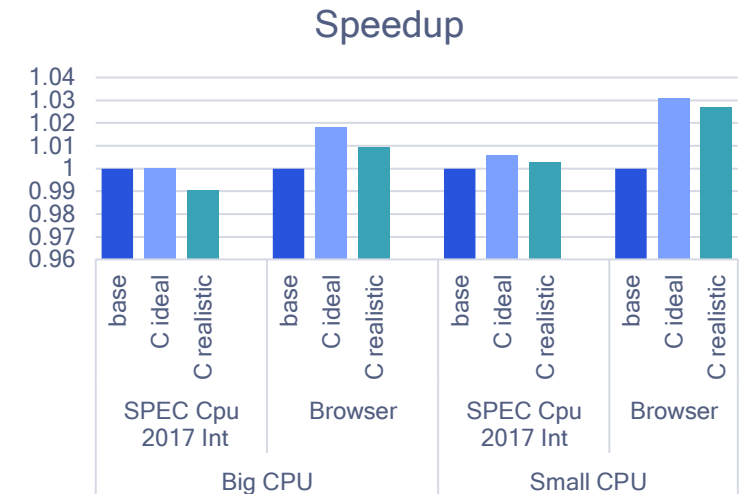
C is not needed

- RV64G is already competitive on code size

- RV64G + 32-bit instructions for code size is *best in class*
  - More ld/st addressing modes
  - Ld/st pair
  - Conditional immediate branches
  - Move pair

- RV64GC has little performance benefit in beefy designs

SPEC CPU Int 2006
Static Code Size (.text section), LLVM 16.0

GEOMEAN

■ aarch64 ■ x86_64 ■ rv64g_zba_zbb_zbs ■ rv64g_zba_zbb_zbs_zNEW

Diverging profiles

- Remove C from application profiles

- Once removed, the C opcode space can be reclaimed to keep code size down long term

- **It's not too late**: commercial distros have not picked a base

  - But, time is running short

Thank you