Instruction Fusion Limit Study for RISC-V

Elizabeth Ho[1]*, Jonathan Woodruff[1]

[1]Department of Computer Science and Technology, University of Cambridge

Abstract

This paper explores the limits of instruction fusion for RISC-V. We characterise instruction fusion rules and algorithms in a general framework, demonstrating a reduction in effective instruction count by over 50%.

Instruction Fusion

Instruction fusion, otherwise known as macro-op fusion, is common practice in modern x86 and Arm processors. They look for opportunities to merge multiple assembly instructions into a single fused instruction within the instruction pipeline.

The RISC-V ISA subscribes heavily to the RISC philosophy of keeping the ISA simple instead of introducing complex instructions that are important for one application but might be rarely used in others. This allows microarchitects to design for RISC-V more easily, as they have fewer instructions to implement.

However, this flexibility is gained at the expense of performance. Microarchitects that design more complex chips might benefit from more complex instructions, as they can implement the logic to execute those without having to fetch and decode multiple instructions to do the same amount of work.

Instruction fusion is a good way to allow gains in performance while maintaining the simplicity and flexibility of the ISA. Microarchitects can choose to fuse certain combinations of instructions into one for their specific application, giving the performance benefits of treating the work as a single instruction, while not bloating the ISA with unnecessary instructions for other applications [1].

Therefore, we have reason to think that RISC-V is in a unique position to exploit instruction fusion. Its core ISA is much simpler than alternatives like ARM and x86, but its wide range of applications call for a method to merge complex functionality into a single instruction.

To prove this, we perform a limit study of instruction fusion opportunities using instruction trace histograms of the SPECInt2006 benchmarks.

Related Work

There has been significant previous work to explore the benefits that instruction fusion brings to RISC-V.

Research in the area was inspired by a 2016 paper by Celio et al. [1], who demonstrated a 5.4% effective instruction count decrease by finding common fusion pairs in the SPECInt benchmark. Multiple papers have been published successively about the subject, which include compiler optimisations [2], fusing non-contiguous memory instructions [3] and fusion in multi-core processors [4].

A common theme across all previous work is a focus on pairwise instruction fusion. We believe that considering longer sequences of instructions may uncover further fusion opportunities. The approach of finding common pairs of instructions is also suboptimal, as these can change depending on the application as well as the workload. To the best of our knowledge, no limit study on instruction fusion has been performed before.

Fusion Rules

Our fusion framework takes rules to be functions that take in a block of instructions and return whether this block is fusable. A wide variety of fusion rules can be expressed in this form, many more than the common fusable pairs presented in previous literature.

For our experiments, we chose to generalise rules defined in previous literature [1]. This gives us a practical idea of what might be feasible with instruction fusion, while also preserving the spirit of a limit study in considering broad classes of instructions and arbitrarily long fusable blocks. We have chosen the following base fusion rules:

1. an arbitrary sequence of arithmetic instructions
2. an arbitrary sequence of arithmetic instructions ending in a memory instruction
3. an arbitrary sequence of arithmetic instructions ending in a branch instruction
4. an arbitrary sequence of arithmetic instructions ending in a memory or branch instruction

Fusion Algorithm

For each instruction, there could be multiple possible fusion rules that would be able to fuse it with the next. To minimise the effective instruction count, we

*Corresponding author: syh38@cam.ac.uk

want to find the rule that fuses the highest number of instructions into a single fusable block.

To do this, we employ a greedy algorithm that looks through the instruction histogram in address order, and, for each instruction,

1. iterates through all the possible rules,
2. stops considering rules that are unable to fuse this instruction with the block of instructions that have been fused so far, and,
3. if there are no rules left in consideration, considers this instruction as the start of a new fusable block.

Fusion Results

We define the fusion rate to be the percentage of instructions that are fused away after performing fusion, and is calculated using the formula:

$$\text{Fusion Rate} = \frac{\text{Original Count} - \text{Effective Count}}{\text{Original Count}}$$

where a fused instruction block counts as a single microarchitectural instruction.

Computing fusion opportunities based on the four instruction rules detailed above demonstrates that the potential upside of instruction fusion is very high. As seen in Figure 1 below, a fusion rate of over 50% can be achieved in the limit, meaning that the resulting effective instruction count is less than half of the original number of instructions. Fusing pairs alone gives close to 30% fusion rate, suggesting that instruction fusion might be able to provide a big benefit without adding substantial amounts of complexity.

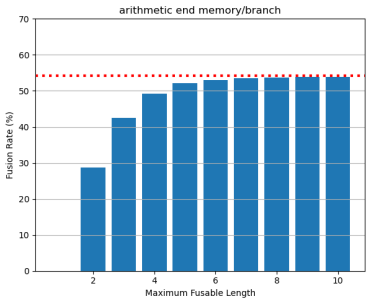

Figure 1: *Fusion rate versus maximum fusable length. Aggregated results across the SPECInt2006 benchmark programs. The dotted red line shows fusion rate at the limit.*

Comparison with Superscalar Processors

Superscalar processors also try to speed up the instruction pipeline by executing multiple instructions at the same time if they are independent from each other. This functionality can be mimicked using instruction fusion by specifying fusion rules that only fuse independent instructions, i.e. they don't share any operands. The results are shown in Figure 2 below.

Perhaps unsurprisingly, the additional restriction that fused instructions must be independent significantly reduces the fusion rate, with a value of 30% in the limit.

Another result is that fusing pairs of independent instructions retains most of the benefit of fusing arbitrarily long sequences of them. One explanation for this is that longer sequences of instructions are less likely to be independent. This suggests that a dual-issue core would be able to capture most of the benefit to running independent instructions in parallel, not considering instruction reordering.

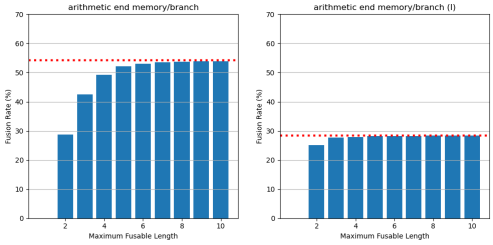

Figure 2: *The same fusion rule, but with an additional restriction that instructions must be independent on the right.*

Conclusion

Instruction fusion is a promising technique that allows microarchitects to capture the improved performance of more complex instructions without bloating the ISA.

We hope that there will be further work to support research and implementation of instruction fusion in RISC-V microarchitectures so that we are able to capture the potential benefits that instruction fusion will bring.

References

[1] Christopher Celio et al. *The Renewed Case for the Reduced Instruction Set Computer: Avoiding ISA Bloat with Macro-Op Fusion for RISC-V*. 2016.

[2] Jian-Yu Shen and Shih-Wei Liao. *Evaluating and Enhancing Performance through Macro-Op Fusion Optimization with RISC-V*. 2024.

[3] Sawan Singh et al. *Exploring Instruction Fusion Opportunities in General Purpose Processors*. 2022.

[4] Yaojie Lu and Sotirios G. Ziavras. *Instruction Fusion for Multiscalar and Many-Core Processors*. 2017.