Why Branch Misprediction Rate is Lower in WebAssembly Compared to Native Environments

Your Name

March 4, 2025

1 Introduction

Modern CPUs employ branch prediction to mitigate the performance cost of control flow changes. While WebAssembly binaries often contain more branch opcodes than their native counterparts, empirical evidence shows that the branch misprediction rate in WebAssembly is lower. This report section explains why this counterintuitive phenomenon occurs.

2 Structured Control Flow in WebAssembly

WebAssembly enforces a structured control flow using high-level constructs instead of allowing arbitrary jumps. The main constructs are:

- **`block`:** A `block` defines a region of code with a single entry and exit point. An unconditional branch (`br`) targeting the block's label immediately exits the block, similar to a `break` statement.
- **`loop`:** A `loop` represents an iterative control structure. Within a loop, an unconditional branch (`br`) targeting the loop's label jumps back to the beginning of the loop, effectively continuing iteration.
- **`if/else`:** The `if` construct conditionally executes one of two blocks. It evaluates a condition (from the top of the stack), then executes either the `then` or `else` block, each of which is a structured code block with a common exit.

Because these constructs are built using a combination of branch opcodes (e.g., `br`, `br_if`, and `br_table`), a single high-level structure (such as an `if/else` or a loop) may translate into several branch instructions. Although this increases the total branch count, the pattern is very regular.

3 Impact on Branch Prediction

The additional branch opcodes introduced by WebAssembly's structured control flow are highly predictable:

- **Predictable Safety Checks:** Many branches are used for bounds and stack checks that compare against constant values. These conditions almost always evaluate the same way (for example, "not taken"), so the branch predictor quickly learns the pattern.
- **Regular Patterns:** The structured nature forces branches into a fixed, repetitive pattern. Even though there are more branch instructions, the CPU's branch predictor is able to achieve high accuracy because the outcome of each branch is almost deterministic.
- **Simpler Conditions:** In contrast, native code might have fewer branch instructions overall, but they may depend on more complex, data-dependent conditions that are harder to predict.

Thus, despite the increased number of branch opcodes in the WebAssembly disassembly, their predictable behavior leads to a lower overall branch misprediction rate compared to native code.

4 Experiment Result

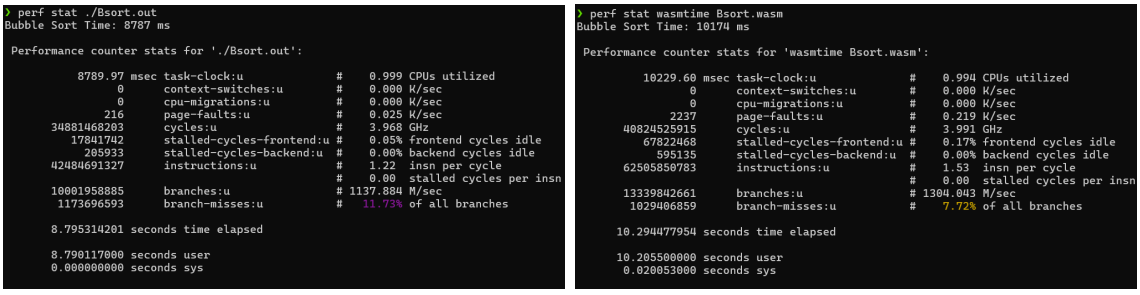

Figure 1: Comparison of performance statistics from native and WebAssembly environments.

5 Conclusion

In summary, WebAssembly's use of structured control flow results in a larger number of branch opcodes. However, because these branches are inserted in highly regular, predictable patterns (primarily for safety checks), modern CPU branch predictors can more accurately

predict their outcome, resulting in a lower branch misprediction rate than in native code where branches, though fewer, may be more unpredictable.