Emma Berry
Dr. Yang
ELEN 511
9 December 2023

Report on HW Speculation Implementation Using Tomasulo's Algorithm

Introduction:

This project entailed extending a simulation of the Tomasulo algorithm to include hardware speculation, specifically addressing how the algorithm handles branch mispredictions. The Tomasulo algorithm is critical for achieving instruction-level parallelism and deals with hazards in a way that minimizes pipeline stalling. Our focus was on integrating branch prediction with speculative execution into the existing Tomasulo framework.

ISA and Processor Architecture:

The ISA includes:

- Arithmetic instructions: add, sub, mul, div.
- Memory access instructions: load (ld) and store (st).
- Extended instructions: addi (add immediate), bne (branch if not equal).

The processor architecture consists of:

- Functional Units: ADD, MUL, and MEM.
- Reservation Stations: ADD_RS, MUL_RS, LD_BUF, ST_BUF.
- Sixteen general-purpose registers and a word-addressed memory.

Implementation Overview:

The implementation follows the Tomasulo algorithm with an additional emphasis on hardware speculation. To this end, a ROB (reorder buffer) was implemented. The primary steps include Issue, Execute, Write Result, and Commit. The core algorithm was simulated in a C program, iterating over instruction cycles to mimic the processor's behavior.

Key Features:

- **Speculative Execution**: Instructions are executed speculatively, assuming the branch will be taken.

- **Branch Misprediction Handling**: On detecting a branch misprediction, subsequent instructions are flushed from the pipeline and the state is reset to the correct path.
- **Commit Stage**: Instructions are committed from the ROB, ensuring correct order and handling mispredictions.

Experiment: Handling a Branch Misprediction

Test Instruction Sequence:

1. ld r0, 0(r1)
2. mul r4, r0, r2
3. st r4, 0(r1)
4. addi r1, r1, 1
5. bne r1, r2, -4

Scenario:

The first prediction of the BNE instruction is wrong, requiring the algorithm to correct its path after the misprediction.

Results:

1. **Initial Execution**: Instructions were issued and executed as per the Tomasulo algorithm with speculation.
2. **Branch Misprediction**: When the BNE instruction was executed, it was found that the branch should not have been taken.
3. **Corrective Action**: The algorithm correctly identified the misprediction, flushing subsequent instructions and resetting the instruction number to the correct path.
4. **Continued Execution**: The algorithm then continued execution from the correct instruction, adhering to the Tomasulo algorithm's principles.

Conclusion:

The implementation of the Tomasulo algorithm with ROB successfully demonstrates handling of branch mispredictions. The project highlights the algorithm's efficiency in maintaining high levels of instruction-level parallelism even in the presence of control hazards like branch mispredictions. This implementation underscores the importance of speculative execution in modern processor architectures for optimizing performance.

See attached text file for the full output.