

Hardware-based Speculation

- With multiple issue, control dependencies becomes a bigger burden;
- branch prediction is good, but not good enough for issuing multiple instns per clock.
- last example – stall per loops
- more branches, more penalties

Ideas!

- to overcome control dependencies speculate on the outcome of branches, and execute the program as if the guesses are corrected. (ie. fetch, issue, execute, intns as if the guess is right)
- What if the guess is incorrected?
- Soln: Using hardware: speculation

Approach

- Combine dynamic branch prediction to choose which instn to execute
- speculation to allow execute of instn before control dependencies are resolved
- dynamic scheduling to deal with the scheduling of different combination of basic blocks

Used in

- PowerPC 603/604/64/64 MIP R10 000/R12 000 Intel Pentium II/III/4
Alpha 21264 AMD K5/K6/Athlon
- implement speculation based on Tomasulo.

Extending Tomasulo

- extending Tomasulo to support speculation -must separate the bypassing of results among instn (speculative instn) from the actual completing of an instn
- can allow an instns to execute and to bypass its results without allowing the instn to perform any updates that cannot be undone, until we know that the instn is no longer speculative
- using bypassed value is like performing a speculative register read, since we do not know whether the instn providing the source register value is providing the correct result until the instn is no longer speculative.
- when an instn is no longer speculative, we allow it to update the register file or memory; this step is called instn commit

- out-of-order execution but in-order commit

Reorder Buffer

- Reorder buffer it holds the result of an instruction between the time the operation associated with the instn completes and the time the instruction commits is also used to pass results among instn that may be speculative is another source of operands for instn
- With Tomasulo, when a instn write a results, subsequently issued instn can find result in register file.
- Here: the reg file is not updated until instn commits
- each entry in ROB has four fields: instn types, destination field, value field, ready field

Store

- Store still takes 2 steps. 2nd step performed by instn commit
- every instn has a place in the ROB until it commits

Four Steps

1. Issue: get instn from instn queue; issue instns if there is an empty reservation station and an empty slot in ROB send operands to reservation station if avail: either from registers or the ROB.
2. execute: if one of more operands is not yet available, monitor CDB while waiting for the value to be computed. begin execute when both operands available
3. write result: when result is available, write to CDB and from the CDB to ROB as well as any reservation station waiting for result
4. Commit: when instn reaches head of ROB, its value is present in the buffer, update the reg with the result and remove the instn from the ROB. Store is similar, except writing to memory instead of register Branch with incorrect prediction reaches the head of ROB,

flushed ROB, and execution is restarted with the correct branch.