Chapter 3 Review

CPI = IDEAL CPI + STALL CYCLES/INSTR

- For more performance, we need to reduce the IDEAL CPI and pipeline stalls

Chapter 3 Review

- ***Dynamic scheduling*** (hardware, this chapter) and ***static scheduling*** (software, next chapter) are techniques for reducing stalls
- We will examine ***superscalar*** processors also that reduce the IDEAL CPI

Chapter 3 Review

- How do we extract more ***Instruction Level Parallelism*** (**ILP**) from the small number of instructions in a basic block?
- First, we looked at what limits parallelism.

Chapter 3 Review

- **Data dependencies** (True Dependencies)
  - Instructions that exchange data set up a data flow which is the fundamental limit to the amount of parallelism
- **Name dependencies** (antidependence and output dependence) (instructions that are not linked by data flow, but write to or read from some common register or memory location) do *not* fundamentaly limit parallelism.
  - Can be resolved by register renaming

Chapter 3 Review

- ***Control dependencies*** (which are imposed by branches) can be violated in certain cases where the outcome of the program is not affected.
- The two properties of a program that we need to preserve are the
  1. Data flow
  2. Exception behaviour
  - We are free to modify the program to get higher performance as long as we preserve above two properties.

Chapter 3 Review

- Dynamic scheduling increases ILP by letting instructions behind stalls execute.
  - Out-of-order exectution
  - Out-of-order completion creates WAW and WAR hazards
- Name dependencies for registers (and therefore potential WAW and WAR hazards) can be resolved through *register renaming*.
  - A relatively small set of architectural registers are mapped to a larger set of registers.

Chapter 3 Review

- Tomasulo's algorithm: an organization that incorporates dynamic scheduling and register renaming
- Registers are renamed to "reservation stations" associated with each function unit.
- The ID stage of the classic pipeline is separated into two stages:
  - Issue – check for structural hazards (empty reservation station)
  - Execute – execute the instruction in a FU when its operands are available
- If one or both operands are not available, then place a pointer in the reservation station indicating which FU will produce them (this is how register renaming is implemented)

Chapter 3 Review

- Two examples of Tomasulo's Algorithm

Chapter 3 Review

- Dynamic memory disambiguation
  - Avoids name hazards involving loads and stores that share the same effective address
- Dynamic branch prediction
  - Predict the behaviour of a branch based on its past behaviour

Chapter 3 Review

- 1-bit branch prediction
  - Store a 1-bit prediction (taken/not taken) in a branch history table indexed by the lower bits of the branch address
  - Will mispredict some important cases, especially loops
- 2-bit branch prediction
  - Only change prediction if there are two mispredictions in a row
  - 2-bit predictors do as well in practice as *n*-bit predictors

Chapter 3 Review

- (*m*,*n*) Correlating Branch Predictors
  - 2-level predictors: use the history of the last m branches to choose from 2^m *n*-bit predictors

Chapter 3 Review

- Tournament branch predictors
  - Multilevel branch predictors that combine local and global predictors with an algorithm to choose between them
- High-performance instruction delivery
  - Branch target buffer – store the predicted target address
  - Branch folding buffer – store the predicted target instruction
  - Return address predictors

Chapter 3 Review

- Multiple Issue Machines
  - Issue more than one instruction per clock cycle
  - Potentially lower CPI below 1
  - H/W issue: superscalar machine (this lecture)
  - S/W issue: VLIW (next chapter)
- Simple Superscalar MIPS
  - Statically scheduled
  - 2-issue: can issue up to 2 intructions/clock
  - 1 int and 1 fp instruction

Chapter 3 Review

- Hardware Speculation
  - Execute instruction stream based on branch predictions
  - Need to be able to back out of instructions when the speculation was incorrect
  - Will implement by modifying Tomasulo's algorithm

Chapter 3 Review

- Key idea: instructions must complete in order
  - in-order issue
  - out-of-order execution
  - in-order completion
- Extra "commit" stage added to pipeline
  - Instructions must commit in-order

Chapter 3 Review

- Reorder Buffer (ROB)
  - All writes are to the ROB until instruction commits and is allowed to modify the real registers and memory
  - Takes over the role of register renaming from the reservation stations
  - Also ensures precise exceptions