Computer Systems Organization

CSE 521/560M
Lecture 10
Prof. Patrick Crowley

Plan for Today

- Thoughts on HW 1
- Questions
- Today’s discussion
Multiple Issue Alternatives

- Superscalar (hardware detects conflicts)
  - Statically scheduled (in order execution)
  - Dynamically scheduled (out of order execution)
- VLIW (compiler schedules)
  - Needs advanced compiler techniques (e.g., trace scheduling)

Impact of Superscalar on IF

- IF: Need to fetch more than 1 instruction at a time
  - Simpler if instructions are of fixed length
  - Simpler if restricted to single L-cache lines
- Sample of recent microprocessors
  - Dual issue: Alpha 21064, Sparc 2, Pentium, Cyrix
  - Triple issue: Pentium Pro (P6)
  - Quad issue: Alpha 21164, PowerPC 620, Sun UltraSparc, HP PA-8000, MIPS R10000
The Decode Stage

- (Simplification: dual issue and static scheduling)
- ID: Look for conflicts between the (say) 2 instructions
  - Of course need for more than one functional unit
  - If one integer unit and one f-p unit, only check for structural hazard, i.e., check opcodes
  - Slight difficulty for f-p load/store that use the integer unit (multiple access to f-p register file – dual port it; RAW hazard resolved as in single pipelines)
  - The load can now delay up to 3 instructions

Alpha 21064

- IF- Prefetcher fetches 2 instructions (8 bytes) at a time
- Prefetcher contains branch prediction logic
  - (4 entry JSR stack; 1 bit/instruction in the I-cache + static prediction BTFNT)
- Swap stage: initial decode yields 0, 1 or 2 instruction issue
- Conditions for 2 instruction issue
  - The first instruction must be able to issue (in order execution)
  - Load/store can issue with an operation except stores cannot issue with an operation of different format
  - An integer op. can issue with a f-p op.
  - A branch can issue with a load/store/operation
Alpha 21164

• Main differences w.r.t. 21064 (besides caches)
  – Up to 4 instructions issued/cycle
  – Two integer units
  – Two f-p units (one add, one multiply)
  – Slightly different pipeline organization

Alpha 21164

• Instruction issue (4 static stages: S0-S3)
  – IF during S0. Gets 4 instructions at a time
  – Up to 4 instructions decoded and buffered during S1
    (some predecoding done if-cache miss when brought in).
    Also branch prediction in this stage.
  – Put up to 4 instructions in slots during S2 (check for
    structural hazards; a few priority rules). A new block of 4
    instructions will be brought in only when the current 4
    are all gone.
  – Instruction issue in S3 (check data dependencies and
    activates forwarding; “scoreboarding”)
Pentium

- Dual integer pipeline (one of them used for f-p)
- Decode 2 consecutive instructions I1 and I2. Issue both iff
  - I1 and I2 are “simple” instructions (no microcode)
  - I1 is not a jump instruction
  - No WAR and WAW hazard between I1 and I2 (I1 precedes I2)

Multiple Issue for Dynamic Scheduling

- Issue in order, execute out of order
- Requires possibility of issuing concurrently dependent instructions (otherwise little benefit over static scheduling)
  - Update “tables” during issue stage
  - Restrict concurrent issue to units using different register files (or res. Stations). Only conflicts left are for load/store/move (from int to f-p regs)
PowerPC 620

• Issue stage. Up to 4 instructions issued/cycle except if a structural hazard exists
  – No reservation station available
  – No rename register available (every “result” is renamed)
  – Reorder buffer is full (needed to commit instructions in order)
  – Two operations for the same unit. Only one write port/set of reservation stations
  – Miscellaneous use of special registers
• Above, first 3 are due to program final two are due to implementation

Pentium Pro (P6 architecture)

• Fetch-decode transforms instructions into micro-operations (mops) and stores them in a global reservation station. Does register renaming (RAT=register alias table)
• The dispatch (i.e., issue)-execution unit issues mops to functional units that execute them and temporarily stores the results (the reservation table is 5-ported, hence 5 mops can be issued concurrently)
• The retire unit commits the instructions in order
General Speculative Execution

- Allow execution of an instruction before knowing that it is to be executed
  - Branch prediction (instructions after the branch executed but not completed before branch is resolved)
  - Moving code across basic block boundaries (instructions are “executed” before the branch on which they really depend is issued)
  - Conditional instructions
  - Out of order execution

Instruction Commit Step

- Need of a mechanism to
  - Know when an instruction has completed non-speculatively
  - Know whether the result of an instruction is correct
  - “Complete” instructions in order. This commits the instruction (easy in single pipeline since commit is last stage of the pipe)
- Note that in Tomasulo’s solution
  - An instruction completes as soon as it has finished executing (i.e., result put in register unless there is a WAW hazard)
  - Leads necessarily to imprecise exceptions (unless handled by software tests)
Reorder Buffer

- Extend Tomasulo’s scheme with a reorder buffer
- Reorder buffer entry contains (this is not the only possible solution)
  - Type of instruction (branch, store or ALU or load)
  - Destination (none, memory address, register)
  - Value
- Replaces store buffers
- Reservation station tags and “true” register tags are now IDs of entries in the reorder buffer
- Reorder buffer implemented as a circular queue

4 Instruction Stages

- Tomasulo, 3 stages: issue, execute, write
- Now 4: issue, execute, write, commit
1. Issue (often called dispatch)
   - Check for structural hazards (reservation station busy, reorder buffer full). If one exists, stall the instruction and those following.
   - If issue is possible, send values to reservation station if they are available in either the registers or the reorder buffer. Otherwise send tag.
   - Allocate an entry in the reorder buffer and send its number to the reservation station.
4 Stages cont’d

2. Execute (like Tomasulo)
3. Write
   • Broadcast on common data bus the value and the tag (reorder buffer number). Reservation stations, if any match the tag, and reorder buffer (always) will grab the value.
4. Commit
   • When instr. At head or ROB has its result in the buffer it stores it in the real register (for ALU) or memory (for store) and is deleted. If the instruction is a branch with correct prediction, do nothing and delete. If the instruction is a branch with incorrect prediction, flush the buffer and restart with correct target address (do it ASAP)

MIPS, Tomasulo + ROB
Assignment

• Readings
  – For Wednesday
    • Ash: Ch. 8
  – For Monday
    • Commentary: The Microarchitecture of Superscalar Processors, Smith and Sobi