Lecture 8: More ILP stuff

Professor Alvin R. Lebeck
Computer Science 220
Fall 2001

Admin

• Thursday office hours to Wednesday 10am
• Homework #2 Due Today
  – Submit simulator code, fareed will post instructions
• Homework #3 assigned
• Please email me your projects, need my approval...
• Project Proposal (October 2)
  – Short document
  – Short presentation
• Papers to read on web page
Summary

• **Branch Prediction**
  – Branch History Table: 2 bits for loop accuracy
  – Correlation: Recently executed branches correlated with next branch
  – Branch Target Buffer: include branch address & prediction

• **Superscalar and VLIW**
  – CPI < 1
  – Dynamic issue vs. Static issue
  – More instructions issue at same time, larger the penalty of hazards

• **SW Pipelining**
  – Symbolic Loop Unrolling to get most from pipeline with little code expansion, little overhead

• **What about non-loop codes?**
  – How do you get > 1 instruction

---

Trace Scheduling

• **Parallelism across IF branches vs. LOOP branches**

• **Two steps:**
  – *Trace Selection*
    » Find likely sequence of basic blocks (*trace*) of (statically predicted) long sequence of straight-line code
  – *Trace Compaction*
    » Squeeze trace into few VLIW instructions
    » Need bookkeeping code in case prediction is wrong
Trace Scheduling

Reorder these instructions to improve ILP

Fix-up instructions
In case we were wrong

HW support for More ILP

- Avoid branch prediction by turning branches into conditionally executed instructions:
  
  if (x) then A = B op C else NOP

  - If false, then neither store result nor cause exception
  - Expanded ISA of Alpha, MIPS, PowerPC, SPARC have conditional move; PA-RISC can annul any following instr, IA-64 predicated execution.

- Drawbacks to conditional instructions
  - Still takes a clock even if “annulled”
  - Stall if condition evaluated late
  - Complex conditions reduce effectiveness; condition becomes known late in pipeline
**HW support for More ILP**

- **Speculation**: allow an instruction to issue that is dependent on branch predicted to be taken without any consequences (including exceptions) if branch is not actually taken (“HW undo” *squash*)
- Often try to combine with dynamic scheduling
- Tomasulo: separate *speculative* bypassing of results from real bypassing of results
  - When instruction no longer speculative, write results (*instruction commit*)
  - execute out-of-order but *commit* in order

**Speculation**

- 4-way issue
- All of B2 issued speculatively
- Must be squashed
- Could have execute B1 in speculative mode
HW support for More ILP

- Need HW buffer for results of uncommitted instructions:
  - reorder buffer
    - Reorder buffer can be operand source
    - Once operand commits, result is found in register
    - 3 fields: instr. type, destination, value
    - Use reorder buffer number instead of reservation station
    - Instructions commit in order
    - As a result, it's easy to undo speculated instructions on mispredicted branches or on exceptions

![Figure 4.34, page 311](image)

Four Steps of Speculative Tomasulo Algorithm

1. **Issue**—get instruction from FP Op Queue
   - If reservation station and reorder buffer slot free, issue instr & send operands & reorder buffer no. for destination.

2. **Execution**—operate on operands (EX)
   - When both operands ready then execute; if not ready, watch CDB for result; when both in reservation station, execute

3. **Write result**—finish execution (WB)
   - Write on Common Data Bus to all awaiting FUs & reorder buffer; mark reservation station available.

4. **Commit**—update register with reorder result
   - When instr. at head of reorder buffer & result present, update register with result (or store to memory) and remove instr from reorder buffer.
Limits to ILP

- Conflicting studies of amount of parallelism available in late 1980s and early 1990s. Different assumptions about:
  - Benchmarks (vectorized Fortran FP vs. integer C programs)
  - Hardware sophistication
  - Compiler sophistication

Initial HW Model here; MIPS compilers

1. **Register renaming**—infinite virtual registers and all WAW & WAR hazards are avoided
2. **Branch prediction**—perfect; no mispredictions
3. **Jump prediction**—all jumps perfectly predicted => machine with perfect speculation & an unbounded buffer of instructions available
4. **Memory-address disambiguation**—addresses are known & a store can be moved before a load provided addresses not equal

1 cycle latency for all instructions
Upper Limit to ILP
(Figure 4.38, page 319)

More Realistic HW: Branch Impact
(Figure 4.40, Page 323)
Selective Branch Predictor

- 8096 x 2 bits
- 2048 x 4 x 2 bits
- Branch Addr
- Global History
- 8K x 2 bit Selector
- 11 Taken
- 10 Taken
- 01 Not Taken
- 00 Not Taken
- Taken/Not Taken
- Choose Non-correlator
- Choose Correlator

More Realistic HW: Register Impact

Figure 4.44, Page 328

Change 2000 instr window, 64 instr issue, 8K 2 level Prediction

Infinite 256 128 64 32 None
More Realistic HW: Alias Impact

Figure 4.46, Page 330

Change 2000 instr window, 64 instr issue, 8K 2 level Prediction, 256 renaming registers

Perfect Global/stack perf; Inspec. None
heap conflicts Assem.

Realistic HW for '9X: Window Impact

(Figure 4.48, Page 332)

Perfect disambiguation (HW), 1K Selective Prediction, 16 entry return, 64 registers, issue as many as window

Infinite 256 128 64 32 16 8 4
Braniac vs. Speed Demon (Spec Ratio)

- 8-scalar IBM Power-2 @ 71.5 MHz (5 stage pipe) vs. 2-scalar Alpha 21064 @ 200 MHz (7 stages)

Discussion

- Technical discussion
- Style of presentation
- Relate topics to what we already know

- Next Time: Energy/Power

- Next Tuesday: Project proposal presentations