Computer Science 703

Advance Computer Architecture

2006 Semester 1

Preparation for Examination

James Goodman

Department of Computer Science

2006 YEAR PRESENTATION The University of Auckland | New Zealand

Examination Information

- Time: Monday, June 12: 9.15am
- Open-book, notes
- Calculators permitted

2006 YEAR PRESENTATION The University of Auckland | New Zealand

Required Readings 1

You are responsible for material in these papers

- P. Sweazey and A.J. Smith, "A class of compatible cache consistency protocols and their support by the IEEE Futurebus," *Proc. Thirteenth International Symposium on Computer Architecture (ISCA-13)*, Tokyo, Japan, pp. 414-423, June 1986.
- M. Herlihy and J.E.B. Moss, "Transactional Memory: Architectural Support for Lock-Free Data Structures," *Proc. International Symposium on Computer Architecture (ISCA-93)*, ACM Press, 1993, pp. 289-300.
- R. Rajwar & J.R. Goodman, "Speculative Lock Elision: enabling highly concurrent multithreaded execution," *34th Annual International Symposium on Microarchitecture (MICRO-34)*, December 2001, pp. 294-305.

2006 YEAR PRESENTATION The University of Auckland | New Zealand

Required Readings 2

You are responsible for material in these papers

- James E. Smith, "Characterizing computer performance with a single number," *Communications of the ACM*, Vol. 31, #10 , (October 1988), pp 1202-1206.
- W.A. Wulf, "Compilers and computer architecture," *IEEE Computer*, **14**(8), pp. 41-47, 1981.
- Smotherman, "Understanding EPIC Architectures and Implementations," *ACM Southeast Conference*, 2002
- R.M. Russell, "The CRAY-I computer system," *CACM*, **21**(1), pp. 63-72, 1978.

All papers available at URL:
http://www.cs.auckland.ac.nz/compsci703s1c/lectures/

Other Readings

You will likely find these readings be helpful in understanding the lecture material

- Hennessy & Patterson, “Distributed shared-memory architectures,” Chapters 2, 3, &6 from *Computer Architecture: A Quantitative Approach* (3rd Ed.), 2003.
- W.-H. Wang, J.-L Baer, & H.M. Levy, “Organization and performance of a two-level virtual-real cache hierarchy,” *ISCA-16*, pp. 140-148, June 1989.
- Mark Hill, “Processors should support simple memory-consistency models,” *IEEE Computer*, **31**(8), pp. 28-34, August 1998.
- R. M. Tomasulo, "An efficient algorithm for exploiting multiple arithmetic units," *IBM Journal of Research and Development*, Vol. 11, January 1967, pp. 25-33.

2006 YEAR PRESENTATION The University of Auckland | New Zealand

Notes

Notes for lectures appear at class URL

http://www.cs.auckland.ac.nz/compsci703s1c/lectures/

2006 YEAR PRESENTATION The University of Auckland | New Zealand

First-week topics

Moore’s Law

Multiprocessing, Multithreading, & Multicores

Multiprocessing Issues

- Lost updates
- Memory ordering
  - Sequential Consistency, Processor Consistency, Release Consistency
- Cache coherence
  - Snooping and directories

2006 YEAR PRESENTATION The University of Auckland | New Zealand

Threads and Thread programming

From assignment, you should have experience with threads

- Creation
- Use of Mutex
- Condition variables
- Thread-safe, MT-safe functions

Interconnection networks & topologies

- Crossbar vs. buses
- Direct vs. indirect networks
- Trees, fat trees, mesh, torus, ring, hypercube
- Perfect shuffle, Omega
- Topological measures (diameter, degree, bisection bandwidth)
- Avoiding deadlock in routing, virtual circuits
- Store-and-Forward routing
- Wormhole & Cut-through routing

2006 YEAR PRESENTATION The University of Auckland | New Zealand

Simulation

- General Simulation techniques
  - time-based, event-based and process-based simulation
  - validation of results
- Architecture-specific simulation
  - instruction emulation
  - trace collection, reduction, processing
- Simulation tools

2006 YEAR PRESENTATION The University of Auckland | New Zealand

Scalable Memory Systems

Interaction of

- Virtually-addressed cache
- Multi-level cache
- Cache coherence
- Non-blocking cache

2006 YEAR PRESENTATION The University of Auckland | New Zealand

Scalable Memory Systems (2)

- Directory-based protocols vs. snooping
  - Snooping has serious limitations of scale
  - Directory-based is always slower, but scalable
  - Basic protocol is simpler (3 states), but requires more serial events
- Maintaining a sharing list in the directory
- Distributed writes (why they are slow)
- Dealing with races

Cache Coherence-MOESI Model

- Attributes
  - ownership
  - exclusiveness
  - validity
- Five states
  - Allowed state changes
  - Permitted combination of states
- Example of lock contention

Better programming Models

- Critical sections
- Atomic RMW operations
  - T&S, T&T&S, Atomic Swap
  - Compare & Swap
  - LL/SC
- Notion of transactional memory
  - Atomic insertion of a transaction (linearizability)
  - Hardware support (SLE)
  - Implementing Transactional Memory
    - Hardware
    - Software

Since the Test…

- Evaluating Performance
- ISAs (Wulf)
  - Importance of Regularity, Orthogonality, & Composability
  - Primitives, not solutions
  - Run-time vs. Compile-time trade-offs

Instruction-Level Parallelism

- How to capture ILP
  - Discover inter-instruction dependences
  - Assign insructions to functional units
  - Determine when instructions execute
- Branching
  - Cost of branching
  - Prediction: costs and problems
  - Speculation

ILP: OoO Execution

- Hazards: RAW, WAR, WAW
- Busy Bits for synchronization
- Tomasulo Algorithm
- Register Renaming
- The “Imprecise interrupt”

2006 YEAR PRESENTATION The University of Auckland | New Zealand

EPIC Architectures and Itanium

- How much to do at compile time?
- Four architectural models
  - VLIW
  - Dynamic VLIW
  - Epic
  - Superscalar
- Bundles
  - Indicating dependences
  - compressing instructions
  - handling multiple branches
- Predicated execution
- Compiler hints for memory hierarchy
- Control speculation
  - Dealing with exceptions
- Data speculation: the ALAT

2006 YEAR PRESENTATION The University of Auckland | New Zealand

Cray Architectures

- Multiple register sets
- Vector instructions
  - vector registers
  - loads/stores with a stride
  - chaining
  - very high memory bandwidth
- Memory hierarchy (no cache)