

Supporting Material for Course Proposal

“Comparison of Contemporary Processor Architectures from the Software Point of View”

Gernot Heiser

March 19, 2002

1 Rationale

This proposal resulted from discussions between Manuel Chakravarty, Jingling Xue and myself. There it was found that research students working at the hardware-software interface do, in general, not have sufficient background in computer architecture, in particular advanced features of modern processors. There is obviously too much material to expect a fair coverage in an introductory-level architecture course such as COMP3211/COMP9211.

COMP4211 (according to the handbook entry and the material on the web page) is concerned with design of processors, and thus fulfils an important role in providing insight into processor internals and preparing students for research in computer architecture.

However, research students working in compilers, language design, operating systems and embedded systems need to learn more about the features that specifically impact low-level system software. While not necessarily interested in processor design issues, they need to understand the full range of such features, and their combinations, in contemporary processors. And they need to understand the implications these features have on software design. In particular, they should be able to answer questions such as:

- How can the software designer abstract over the various instantiations of a particular architectural feature across the range of available architectures, and what software designs will support portability?
- What implications do architectural decisions have on program performance and what techniques may be used to obtain maximum performance from a particular processor family?
- How likely is a particular approach to be tied to a particular processor family, or to a particular processor model within a family?
- What architectural features are available but have not been properly exploited to date? How could they be used?
- How are issues like data layout, locks, low-level parallelism, context switching rates, probability of taking `if` vs. `else` branches, working sets, etc., affect performance or power consumption on a specific architecture?
- What are the likely developments in the next few years? For example, will caches grow at all levels or will there be more levels? Will associativity increase or decrease?

The proposed course is meant to address these issues. It is an advanced-level architecture course which is, however, not primarily aimed at students doing (or intending to do) research in computer architecture. Instead it targets research students in related areas who need to accept the architecture as given, but need to understand what they can do with it.

In summary, the introduction of this course is seen as an important part of the strategy to build up a world-class research group in the general “systems” area at UNSW.

2 Formalities

2.1 Consultation

The proposal was prepared by Gernot Heiser, in consultation with Jingling Xue, Manuel Chakravarty and Gabi Keller. It is strongly supported by all of them. Gabi has volunteered to teach a small part of it (on multiprocessor interconnects).

A second round of consultation occurred with Hossam Elgindy, Sri Parameshwaran and Oliver Diesel. This has led to some amendments of the proposal.

Research students working in operating systems were also consulted for their opinion on what material they would have considered beneficial to their research.

2.2 Target audience

Postgraduate research students in operating systems, embedded systems and compilers.

2.3 Delivery mode

The appropriate delivery mode for the course will require some experimentation. However, for the first offering, the following model will be tried.

There will be a combination of lectures and student presentations. The lecturer will present and discuss architectural features. Students will then present actual architectures as case studies, discussing which features the particular architecture provides, in which combination, and why. They will then, with participation from the whole class, discuss the programming model of that architecture. They will afterwards (with the benefit of the discussion) submit a report on the particular architecture.

2.4 Assessment

Grading is pass/fail only. Assessment is based on the seminar, the written report, and participation in class.

3 Contents

3.1 Handbook entry

Examination of contemporary computer architectures, comparing and contrasting their software-visible features (caches, memory management unit, pipelining and instruction-level parallelism, instruction set architecture, register files). Examination of the effect of these features on the design and implementation of operating systems, compilers, run-time systems, etc. Discussion of software techniques for dealing with these architectural features.

The course is aimed at providing research students in the fields of systems and compilers with the relevant advanced architecture background and an idea of where architectures are likely to head in the next 5-10 years.

3.2 Specific topics covered

The below topics are covered, not from the processor architect's point of view (which would focus on why things are done a particular way and what the underlying assumptions and tradeoffs are) but from the point of view of the software architect, who needs to do the best with the given hardware:

Instruction set architecture: RISC vs CISC, VLIW, ILP, addressing modes, synchronisation primitives, pipelining, issue slots, load and branch penalties, delay slots, out-of-order execution, EPIC, SIMD instructions, predication, status flags

Cache memory architecture: virtual/physical, split/unified, associativity, latency, coherency, write buffers, store order...; software cache management (colouring)

MMU design: page sizes, TLB size and associativity, split/unified TLBs, TLB tagging, segmentation, support for sharing, hardware vs software reload, multi-level TLBs, pinning of entries; software algorithms and data structures (e.g., TLB caches)

Multiprocessors and interconnection architecture: SMP, NUMA, single-chip multiprocessors, symmetric multithreading; inter-processor communication mechanisms and latencies

3.3 Architectures covered

Most likely covered:

- x86 (IA-32),
- x86-64 (AMD),
- IA-64,
- MIPS,
- Alpha,
- SPARC,
- PowerPC,
- ARM.

Possibly also covered: Super Hitachi, embedded M68k???, Transmeta