A Microprocessor Survey Course: Exploring Advanced Computer Architecture in Practice

Kevin Skadron

Department of Computer Science
School of Engineering and Applied Science
University of Virginia
Charlottesville, VA 22904

skadron@cs.virginia.edu

Abstract

A seminar that surveys state-of-the-art microprocessors offers an excellent forum for students to see how computer architecture techniques are employed in practice and for them to gain a detailed knowledge of the state of the art in microprocessor design. Princeton and the University of Virginia have developed such a seminar, organized around student presentations and a substantial research project. The course can accommodate a range of students, from advanced undergraduates to senior graduate students. The course can also be easily adapted to a survey of embedded processors. This paper describes the version taught at the University of Virginia and lessons learned from the experience.

1. Introduction

An important component of studying computer architecture is seeing how concepts are applied in practice. Although textbooks often present excellent case studies, they typically break computer architecture down into discrete topics, the case studies often illustrate only the concepts from a given chapter, and textbooks are limited in the extent to which they can explore a number of microprocessors in detail. Textbooks also often have difficulty exposing interrelationships among different aspects of the processor, and they typically omit the manufacturing and economic factors that influence processor designs. In practice, decisions about the design of one processor component frequently influence the design of the rest of the processor, and manufacturing and economic factors often weigh heavily in design decisions.

In the Proceedings of the Workshop on Computer Architecture Education, June 10, 2000, Vancouver, B.C.—held in conjunction with the 27th International Symposium on Computer Architecture.

Princeton and the University of Virginia (U.Va.) have developed a seminar that compares and contrasts a number of state-of-the art microprocessors. The goal is not only to consolidate students' knowledge, but to expand their knowledge beyond the level of an advanced computer architecture course. This course was first taught several years ago by Skadron's advisor, Douglas W. Clark, at Princeton University. This paper describes the newest version of the course taught at U.Va. and lessons learned from the experience.

This seminar also affords students the opportunity to practice public speaking, the opportunity to gain experience in finding and interpreting technical documentation, and the opportunity to conduct preliminary research. The course can accommodate a variety of levels of knowledge; for example, we have allowed undergraduate students to participate in the seminar without diluting its challenge to advanced students. While undergraduates participate less than the more knowledgeable graduate students and pursue less aggressive research projects, they gain immensely by obtaining early exposure to advanced concepts and building interest in the area of computer architecture. Because the course serves as rich source of research ideas, we do recommend including a substantial research project.

2. Course Overview

The course opens with a short period of lectures by the instructor. The bulk of the course comprises two sequences of presentations, organized around the matrix shown in Figure 1. The first sequence, comprising most of the first half of the course, consists of detailed student presentations of four or five state-of-the-art microprocessors and corresponds to the rows in the matrix. The sec-

	ISA	I-stream	D-stream	...
Alpha 21264	A	B	C	D
Pentium III	E	F	...	
Athlon	...			
UltraSPARC-III	...			

Figure 1. Student-team matrix. The students (A, B, C, etc.) choose a processor and a "cross-cutting" design theme for which they are responsible in their team presentations.

ond sequence, corresponding to the columns, comprises the latter half of the course and consists of further student presentations that explore *cross-cutting* design themes. Students sign up for a processor and a theme (in other words, for a position in the matrix) at the beginning of the course. Teams—both processor and theme—should be balanced according to students' interests, experience, and maturity.

This "matrix format" is the core of the course's organization. It permits each student to leverage the research from the first presentation to prepare for the second presentation. Yet the two presentations' very different focuses push the student to become an expert not only on one microprocessor, but also on one cross-cutting theme and its impact on different processors.

The course's three phases are next described in more detail.

2.1. Introductory Material

To help bring all students in the class to a common level, the instructor spends 2–3 weeks lecturing, reviewing advanced topics in computer architecture and explaining economic and manufacturing considerations that may influence microprocessor design.

Although we have not done so, a test might be offered at the end of this introductory phase. A test will help the students cement the material from these lectures, and will provide an additional component for the overall class grade, which is otherwise based on presentations, participation, and the research project.

Because some of the material in these lectures does not appear in current textbooks, students are likely to ask for handouts that cover the new concepts. This is especially true if a test is required.

2.2. Microprocessor Presentations

The class is divided into teams, one per processor. These teams consist of the rows in Figure 1 above. Each team spends a week (about 2.5 class hours) presenting in detail the instruction set and design of their assigned processors. Students should include any available performance data, like SPECcpu [23, 24] scores, branch-prediction accuracies, and cache hit rates.

In the class, we surveyed four processors—in its most recent version, we covered the Compaq Alpha 21264 [8, 13], the Intel Pentium III [5, 7], the AMD Athlon [3, 4], and the Sun UltraSPARC-III [9, 21]. A set of embedded processors could be substituted for a course emphasizing issues in embedded-system design.

Each team member is responsible for one major area of the microprocessor's organization—the cross-cutting themes. Their preparation in this area then serves as a foundation for their work on their next presentation that focuses on their cross-cutting theme of choice. The themes we covered are:

Instruction set architecture. Material to cover includes novel instructions, new instructions, unusual or absent addressing modes, branch delay slots, and so forth.

Instruction stream. Material to cover includes instruction-cache and branch-predictor organization and effectiveness, and instruction-prefetch facilities.

Data stream. Material to cover includes data-cache organization and effectiveness, write-buffer and TLB organization, prefetching facilities, load-store ordering constraints, and so forth.

Register renaming and instruction issue. Material to cover includes renaming structures,

reservation-station and instruction-window organization, functional-unit topology, instruction wake-up, selection, and issue policies, the instruction-commit procedure, and any noteworthy exception handling (*e.g.*, imprecise exceptions are the default in the Alpha 21164 [6]).

Memory management. Material to cover includes support for memory consistency, cache coherency, inter-process communication, and context switching, as well as page-table organization and paging policies.

For smaller classes, "memory management" can be mostly omitted, and "register renaming and instruction issue" can be folded into the other topics. Although students are responsible for only one aspect of the processor, we have found that preparing for the group presentation leads them to become well-versed in most other aspects as well.

Presentations are interactive, with the class encouraged to ask questions, and comment or speculate on design choices. Participation by the audience is an important component of the grade. Fortunately, students rapidly become comfortable and pursue a lively back-and-forth dialogue. The instructor serves chiefly as a moderator.

In preparing for their presentation, students gather relevant source material from the published literature, the web, and inquiries made to manufacturers. Students might find, for example, a hardware manual, a whitepaper describing the rationale behind some choices, articles from the trade press discussing the merits of different processors, and papers from the research literature describing advanced microarchitectural techniques employed by this processor.

The teams assemble the most useful material into a book, which can be spiral- or velo-bound, and distribute it to class members before the presentation. The book should include a table of contents, an index of other literature that might be useful, and copies of the overheads used in the team presentations. These books are outstanding references: we find ourselves consulting them regularly for details on these processors, and copies have been in hot demand by other members of the faculty and graduate-student body.

2.3. Cross-Cutting Theme Presentations

After the microprocessors have been presented, students reassemble into new teams based on their chosen theme—these teams represent the columns from the matrix in Figure 1. In this phase, the students produce additional booklets, tracking down any additional research literature that describes the techniques used by the different processors. They should also seek further literature that describes relevant, recently-proposed techniques not yet seen in any microprocessor. In their presentations, they explore the technical, manufacturing, and economic factors that led designers to make the often widely disparate choices embodied in the different processors. The design factors that generated the most discussion in our most recent version of this course were:

- The choice of in-order execution by Sun for the UltraSPARC-III, when the other processors we studied, and indeed most high-performance microprocessors, have chosen out-of-order implementations.

- The different processors' widely varying instruction-issue capabilities. The Pentium III, for example, has just five instruction-issue ports, and three are dedicated to processing loads and stores. In contrast, the Athlon has nine ports. Nevertheless, the two processors achieve remarkably similar performance.

  The Alpha 2126, with six issue ports, lies between these extremes, but has the interesting feature of two register-file *clusters*. Its four integer pipelines are statically divided between the two clusters, with two pipelines and one copy of the register file forming one cluster. Results write to their home cluster normally, but require an extra cycle to broadcast their results to the other cluster. This means that instructions needing a result produced by the other cluster may suffer a penalty.

- The benefit of architected branch-delay slots. Many students were puzzled by the existence of delay slots, since they create difficulties for multi-issue architectures.

- The UltraSPARC-III's novel treatment of writes. Unlike the other processors in our survey, the UltraSPARC-III uses a write-through first-level

cache with an unusual write buffer organization called a *write cache* [11]. It can be read directly [20]—rather than requiring some kind of flush—when needed data still resides in the buffer. The write cache is unusually large and associative compared to a conventional write buffer, and follows an LRU rather than a FIFO writeback policy.

Although we have not yet done so, in future versions of the class we will require students to quantitatively evaluate some of the choices made by the different processors, using a simulation package like SimpleScalar [1, 2, 22]. This constitutes an additional portion of the assigned grade.

Evaluations might include a performance comparison of different branch-predictor, cache-hierarchy, or functional-unit-topology organizations, a comparison of in-order vs. out-of-order issue, or a comparison of different instruction-window sizes or organizations. More ambitious projects might explore the tradeoffs among better caching schemes, better compiler optimizations, and larger instruction windows; or the impact of the instruction set on the compiler's ability to generate efficient code schedules.

2.4. Research Project

As the class proceeds, students choose a topic for a final research project. They may work independently or in teams. The topic should be tractable in the space of a single semester, should be adapted to the student's level (undergraduates vs. senior graduate students), and should be a topic that can provide the basis for future, publication-quality research. In the most recent iteration of this course, Skadron's class of fifteen students produced a conference publication [16], a workshop publication [10], two technical reports [18, 25], and two eventual senior theses that are currently being combined in preparation for a conference submission [12, 14]. Topics included new branch prediction techniques, further evaluation of existing branch-prediction techniques, efficacy of confidence prediction, new disk-prefetching algorithms, costs of result bypassing, and effectiveness of benchmark sampling for simulation. The most novel projects, both proposed in part by the students themselves, were:

- A paper exploring *differential multithreading* [10], a technique for switching among multiple instruction streams in response to pipeline stall conditions. By relying on the pipeline organization of an in-order, single-issue processor as might be found in various embedded-processing environments, differential multithreading provides an inexpensive way to improve processor throughput at low cost.
- A paper describing a new software architecture for bytecode interpreters [18]. This paper explores how to take the techniques used by CPUs to exploit ILP processors and translate them to the bytecode execution engine embodied in the interpreter software. By using the fine-grained, low-cost multi-threading supported by simultaneous multithreading [15] or chip multiprocessing [17], the bytecode interpreter can exploit *bytecode-level parallelism* or BLP.

Students found their project useful for different reasons. Undergraduates found it useful as a first experience with the sort of research they might conduct in graduate school. Beginning graduate students found it useful as an opportunity to explore whether research in computer architecture interested them, and more advanced graduate students found it a way to explore a new research idea.

If the instructor does not wish to guide research projects, they can be replaced with simpler exercises—perhaps like those mentioned in Section 2.3—or with a final paper in which students describe how they would design a processor and justify their choices with findings from the published literature and with the outcome of the class discussions.

2.5. Grading

In assigning the course grade, we chose to emphasize the final paper and the presentations, as these comprised the bulk of the students' work. A significant portion of the grade was also assigned to class participation, as a way of encouraging active classroom discussion and debate. We suggest a grading scheme like the following:

- final paper (35%)
- presentations (25%)
- participation (15%)
- initial test (10%)
- simulation-based evaluation of cross-cutting theme (15%)

3. Benefits

The seminar has been very well received. Students enjoyed it, faculty felt it make a significant contribution to the graduate curriculum, and it will be added as a regular course offering at Virginia. In particular, students felt that they benefited from:

- Seeing the diversity of design decisions embodied in different processors (see Section 2.3 for examples).
- Gaining a detailed knowledge of most of the major state-of-the art microprocessors. This becomes especially valuable in the long run, in choosing how to conduct various future research projects, and when writing papers, in relating research choices to the current state of the art.
- Learning advanced concepts not covered in the core graduate computer architecture course, and seeing them put into practice (*e.g.*, various branch-prediction and caching techniques).
- Seeing the impact of different instruction-set architectures on design decisions. The need for IA-32 processors to decode the CISC instructions into RISC instructions and the consequent penalties (extra pipeline stages, longer branch misprediction penalties) is the most obvious example. The decision by Alpha architects to add sub-word load and store instructions after the first generation of the 21164 is another example.

Undergraduates also benefited from the exposure to advanced computer architecture concepts and from the opportunity to conduct research for their project. In our experience, the undergraduates who have taken this seminar do excellent projects. This not only helps them establish credentials for acceptance into top graduate schools, but also helps prepare them for graduate-level research.

In addition, the books assembled by the student teams serve as extremely useful references for years after the course has been taught, the students get the opportunity to pursue research in computer architecture, and some students produce publication-quality results.

4. Lessons Learned

Start research project early. For many students, this project will be their first substantial research project. Many if not most projects also require using a simulation package like SimpleScalar [1, 2, 22]. In order to have time to produce useful work, it is vital to select a topic within the first month and begin work right away. Unfortunately, this means students will not be able to hear all the processor presentations before choosing a topic, but starting the project early seems more important.

Give students a menu of research projects. Less advanced students require substantial guidance in their choice of project. Giving them *carte blanche* leads to many rounds of discussion before a topic can be found that is suitable as well as feasible in one semester. Despite the success we experienced in letting students find their own projects, we instead recommend that the instructor come up with a menu of research projects that touch on the different cross-cutting themes. Students who wish to pursue topics not on the menu can negotiate their topic as necessary.

Require milestones. Students taking other classes have a natural tendency to postpone long-range projects in favor of more immediate homework. Indeed, some students delay starting their project too long, and turn in projects that are weak and/or late. Problems have also been encountered when students misunderstand their project.

To combat these problems, students should write a project proposal and also turn in at least two interim reports. These documents can be quite short. The goal is to create milestones that require students to make steady progress. The proposal need merely describe the problem to be studied, the basic approach, and some milestones for completed work. The first interim report should indicate that the student has begun implementation (and therefore understands and has gained some experience with the research infrastructure to be used), and the second report should indicate that the student has completed most of the implementation, and should also include the first portions of the final report (*e.g.*, introduction and motivation, approach, etc.).

Most of our students used SimpleScalar [1, 2, 22], and we found it useful to provide concrete familiarization exercises designed to acquaint them with modifying the code to obtain various measurements. These exercises took the students on a tour of the code, required

them to use a variety of the configuration and statistics-gathering tools provided by SimpleScalar, and required them to understand the code well enough to learn where to instrument it to obtain various kinds of statistics.

Give immediate feedback on presentations. Promptly returning a grade and a critique of each student's presentation permits the students to do a better job with their second presentation. More importantly, it helps them develop better presentation skills. Fortunately, most of the presentations were outstanding; common problems were chiefly restricted to formatting of the presentation overheads.

The entire team can be assigned a single grade, or each student can receive an individual grade.

Require rigor in cross-cutting-theme phase. The discussion of cross-cutting themes is prone to a great deal of handwaving unless rigor is required. Requiring the students to perform and present the results of some quantitative evaluations helps substantially (see Section 2.3). Also important is requiring students to find relevant research literature that describes new techniques employed by the processors and literature describing new algorithms that extend these techniques. Literature discussing manufacturing and economic factors helps, too.

Everybody must participate. We have permitted students to audit the course, but required everyone attending the seminar to participate in the student presentation teams. This ensures widespread participation. At the beginning of the sequence of presentations, it helps for the instructor to ask questions and make observations. This seems effective at provoking broad class participation.

Some students are shy. While class participation was excellent overall, some students are shy or feel intimidated by students who are more knowledgeable or more outspoken. Even though these students often give excellent presentations and do excellent work, if they participate rarely, this harms their participation grade and can have a non-trivial effect on their overall class grade. We have not yet come up with a good solution for this problem, except to speak with students who tend to be quiet and be supportive of their efforts. It also helps for the instructor to treat naive questions with patience, in order to establish an environment of trust.

Provide tools and base configurations. In order to maximize the students' productivity, the instructor needs to provide infrastructure. This includes:

- Installing the simulation package used by most students—we used SimpleScalar [1, 2, 22].
- Installing benchmarks—we used the SPEC95 benchmarks [24].
- Providing instructions on proper simulation (fast-forward intervals, appropriate inputs, appropriate measurement methodology) [19].
- Finding adequate computer time for all the students. (We also required all simulations to be "niced".)
- Providing graphing tools. Many students are familiar with Excel, but are not familiar with how to assemble suitable graphs, so templates are useful. Students using LaTeX may also be unaware of how to include Excel figures in a LaTeX document, so the instructor needs to provide instructions.

5. Conclusions

This paper has described a microprocessor survey course that can accommodate a range of students, including advanced undergraduates and senior graduate students. The seminar is centered around student presentations and a research project. During the first phase of presentations, the organization of four or five high-performance microprocessors are described in detail. In the second phase, major cross-cutting design issues are explored. The class examines the rationale behind diverse design choices, the tradeoffs that drive these choices, and new research that will affect these cross-cutting issues. We have also provided a number of detailed suggestions on how to best implement such a seminar.

Acknowledgements

I would like to thank Douglas Clark for his helpful comments and for his assistance when I prepared this course.

References

[1] T. M. Austin and D. C. Burger. Simplescalar 3.0 prerelease. Sept. 1998.

[2] D. C. Burger and T. M. Austin. The simplescalar tool set, version 2.0. *Computer Architecture News*, 25(3):13–25, June 1997.

[3] K. Diefendorff. K7 challenges Intel. *Microprocessor Report*, pages 1, 6–11, Oct. 26 1998.

[4] K. Diefendorff. Athlon outruns Pentium III. *Microprocessor Report*, Aug. 23 1999.

[5] K. Diefendorff. Pentium III = Pentium II + SSE. *Microprocessor Report*, Mar. 8 1999.

[6] Digital Semiconductor. *Alpha 21164 Microprocessor: Hardware Reference Manual*, Apr. 1995.

[7] L. Gwennap. Intel's P6 uses decoupled superscalar design. *Microprocessor Report*, pages 9–15, Feb. 16, 1995.

[8] L. Gwennap. Digital 21264 sets new standard. *Microprocessor Report*, pages 11–16, Oct. 28, 1996.

[9] T. Horel and G. Lauterbach. UltraSPARC-III: Designing third-generation 64-bit performance. *IEEE Micro*, pages 73–84, May-June 1999.

[10] J. W. Haskins, Jr. and K. Skadron. Differential multithreading: Recapturing pipeline stall cycles and enhancing throughput in small-scale embedded microprocessors. In *Proc. of the Workshop on Complexity-Effective Design*, June 2000.

[11] N. P. Jouppi. Cache Write Policies and Performance. In *Proc. of the 20th International Symposium on Computer Architecture*, pages 191–201, May 1993.

[12] P. Juang. Classification-based hybrid branch prediction. Senior thesis, University of Virginia School of Engineering and Applied Science, Apr. 2000.

[13] R. E. Kessler, E. J. McLellan, and D. A. Webb. The Alpha 21264 microprocessor architecture. In *Proc. of the 1998 International Conference on Computer Design*, pages 90–95, Oct. 1998.

[14] A. V. Lanning. Pipelined branch prediction: Characterizing wrong-history misprediction. Senior thesis, University of Virginia School of Engineering and Applied Science, Apr. 2000.

[15] J. L. Lo, S. J. Eggers, J. S. Emer, H. M. Levy, R. L. Stamm, and D. M. Tullsen. Converting thread-level parallelism to instruction-level parallelism via simultaneous multithreading. *ACM Transactions on Computer Systems*, 15(3):322–354, Aug. 1997.

[16] C. Metz. Safety checking of kernel extensions. In *Proc. of the 2000 USENIX Technical Conference*, June 2000.

[17] K. Olukotun, L. Hammond, and M. Willey. Improving the performance of speculatively parallel applications on the Hydra CMP. In *Proc. of the 13th International Conference on Supercomputing*, June 1999.

[18] K. Scott and K. Skadron. BLP: Applying ILP techniques to bytecode execution. Tech. Report CS-2000-05, University of Virginia Department of Computer Science, Feb. 2000.

[19] K. Skadron, P. S. Ahuja, M. Martonosi, and D. W. Clark. Branch prediction, instruction-window size, and cache size: Performance tradeoffs and simulation techniques. *IEEE Transactions on Computers*, 48(11):1260–81, Nov. 1999.

[20] K. Skadron and D. W. Clark. Design issues and tradeoffs for write buffers. In *Proc. of the Third International Symposium on High-Performance Computer Architecture*, pages 144–155, Feb. 1997.

[21] P. Song. UltraSparc-3 aims at MP servers. *Microprocessor Report*, pages 29–34, Oct. 27 1997.

[22] SimpleScalar Tools Home Page. WWW site: http://www.cs.wisc.edu/~austin/simplescalar.html, Jan. 1998.

[23] Standard Performance Evaluation Corporation. SPEC CPU2000 Benchmarks. WWW site: http://www.specbench.org/osg/cpu2000, Dec. 1999.

[24] Standard Performance Evaluation Corporation. SPEC CPU95 Benchmarks. WWW site: http://www.specbench.org/osg/cpu95, Dec. 1999.

[25] B. S. White and K. Skadron. Path-based target prediction for file system prefetching. Tech. Report CS-2000-06, University of Virginia Department of Computer Science, Feb. 2000.