A Microprocessor Survey Course for Learning Advanced Computer Architecture

Kevin Skadron
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
skadron@cs.virginia.edu

Abstract

A course 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. The University of Virginia has developed such a course, 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 or DSPs. This paper describes the course and lessons learned.

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.

The University of Virginia has developed a course that explores advanced computer architecture concepts by comparing and contrasting a number of state-of-the art microprocessors. The course 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 research. The course can accommodate a variety of levels of knowledge; for example, the Virginia version accommodates a mixture of undergraduate and graduate students without overtaxing less advanced students or providing a diluted experience for those who are more advanced. 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. Graduate students take a more active role in leading the discussion, mentor the undergraduates, and pursue more aggressive research projects. This course works well for both computer-science and computer-engineering majors, but an advanced, undergraduate-level architecture class should be a prerequisite (*e.g.*, ECE 435 at Virginia), so that students have studied pipelining in detail and have at least been exposed to concepts like caches, branch predictors, and instruction scheduling. Finally, it is important to note that the course serves as rich source of research ideas, and it is therefore useful to include a substantial research project.

This course grew out of a seminar first taught by Douglas W. Clark at Princeton University. This paper, which updates the experiences reported in [10], describes how the course has been structured at Virginia, presents lessons learned after teaching this course twice, and evaluates how well the course meets its goals.

2 Course Overview

The course opens with a sequence of lectures by the instructor to provide background on modern processor architectures. This in turn sets the stage for the bulk of the course, which 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 second 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 (*i.e.*, a position in the matrix) at the beginning of the course. Teams—both processor and theme—should be balanced according to students' interests and experience. This format 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.

To appear in the SIGCSE 2002 Symposium, Feb. 27–Mar. 3, 2002, Northern Kentucky, USA.
Permission to make digital or hard copies of part of all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

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

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.

The last segment of the class is a sequence of lectures by the instructor on papers from the research literature that cover the most interesting cutting-edge topics that arose during the students presentations and research.

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

2.1 Introductory Material

To help bring all students in the class to a common level and prepare them for studying modern architectures, the instructor spends the first part of the course lecturing. The purpose of these lectures is to introduce or review the organization of modern microprocessors; describe cutting-edge features seen in today's processors, such as out-of-order execution, advanced branch prediction, and trace cache; and explain economic and manufacturing considerations that may influence microprocessor design. This is especially true if a test is required. We did not use a textbook, but a text of the instructor's choosing can certainly be used effectively. Because some material in these lectures does not appear in current texts, students are likely to ask for handouts that cover the new concepts.

A test should be offered at the end of this introductory phase. A test helps the students cement the material from these lectures, and provides an additional component for the overall class grade, which is otherwise based on presentations, participation, and the research project. This course has been taught twice at Virginia, once without and once with the test. Giving a test seemed to make a small difference in student attentiveness but made a substantial difference in the instructor's ability to identify which concepts were giving the students, and especially the less advanced students, difficulty. These topics can then be reviewed before student presentations begin.

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 (http://www.specbench.org) 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 [7], the Intel Pentium III [3], the AMD Athlon [2], the Pentium 4 [5], and the Intel Itanium [4]. A set of embedded processors or digital signal processors (DSPs) could be substituted for a course emphasizing issues in embedded-systems 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.

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, 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, the experience at Virginia is that 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.

The students also assemble a web page with links to all relevant online documents as well as their own presentation overheads and any summaries they have written. To ensure that the students take the website seriously, experience has shown that it must be a small part of the grade.

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 have been:

- The choice of explicitly parallel, in-order issue by Intel for the Itanium, 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 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 merits of small vs. large first-level caches. Some processors like the Alpha 21264 use large, 64 KB caches, while others like the Pentium 4 use small, 8 KB caches.

To help ground the cross-cutting discussions in facts rather than hand-waving, experience has shown that students must bolster their analysis with data obtained either from publications and the web, or through a quantitative evaluation of their own, perhaps using a simulation package like SimpleScalar [1]. The quantitative evaluation can be made a separate requirement and constitute 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 Advanced Material

After the student presentations, the remaining weeks can be used to read and discuss papers from the research literature that either explain issues that were not resolved during the presentations, or that extend current techniques with new ideas. About one paper per class seems to be the best rate at which to cover research papers.

These discussions were led by the instructor, with a loose lecture format that encouraged frequent student comments and discussion. To help students assimilate the papers, they were required to submit a one-page synopsis of each paper in which they identify the key idea, the most important contribution and most important flaw, and in which they pose any questions they had about the paper. These questions can be used to ensure that the students understood the paper and can serve as additional talking points during class.

2.5 Research Project

As the course 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. Students find their projects useful for different reasons. While some students may not in fact complete enough work for publication, the orientation toward publication helps generate a challenging project on topics of current interest and serves as a valuable experience in several ways. Undergraduates find it useful as a first experience with the sort of research they might conduct in graduate school. Beginning graduate students find it useful as an opportunity to explore whether research in computer architecture interested them, and more advanced graduate students find it a way to explore a new research idea.

In the first year this course was taught, the students did work that led to two conference papers, two workshop papers, and one technical report. The most recent year led to once conference publication, two abstracts at a work-in-progress session, and served as the genesis of two projects that are still in progress. The most novel projects, both proposed mainly by the students themselves, were:

- A paper exploring *differential multithreading* [6], 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 many embedded-processing environments, differential multithreading provides an inexpensive way to improve processor throughput at low cost. The student author says that this idea grew directly out of the in-class discussions.

- A paper describing a new software architecture for bytecode interpreters [9]. 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 multithreading techniques, the bytecode interpreter can exploit *bytecode-level parallelism* or BLP. (This technique has since been superseded by hardware support for Java translation.)

- A paper studying power issues related to branch prediction [8].

If research projects are not desired, 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.

2.6 Grading

In assigning the overall grade, the course emphasizes the final paper and presentations, as these comprised the bulk of the students' work. A significant portion of the grade was also assigned to class participation, to encouraging active classroom discussion and debate. The most recent grading scheme used was the following: project proposal (5%), midterm progress report (10%), final paper (30%), presentations (25%), participation (10%), initial test (10%), quantitative evaluation of cross-cutting theme (5%), website (5%).

3 Benefits

The course has been very well received. Students enjoyed it, and faculty felt it makes a significant contribution to both the graduate and undergraduate curricula. 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 computer architecture courses, 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. One example is the need for IA-32 processors to decode the CISC instructions into RISC instructions; the consequent penalties (extra pipeline stages, longer branch misprediction penalties); and how this makes a trace cache beneficial. 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, opportunities that many undergraduate courses do not have time to provide. In our experience, the undergraduates who have taken this seminar do excellent, quite sophisticated projects. This not only helps them establish credentials for acceptance into top graduate schools or for interviewing with top companies, 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

A number of lessons were learned during the first year in which this course was taught. As mentioned earlier, we found that a test after the first round of lectures is helpful for several reasons; that the website must be a small part of the grade in order to ensure compliance; and that quantitative analysis must be part of the discussions. In addition, incorporating the following suggestions made the second year run more smoothly.

Start the research project early. For some students, this project will be their first substantial research project. Many if not most projects also require using a simulation package like SimpleScalar [1]. 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.

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, and many of the resulting projects end up being similar to what would have appeared on the menu to begin with. 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, without a forcing function, 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 and the misunderstanding is not exposed until too late.

To combat these problems, students should write a project proposal and also turn in at least one interim report. Weekly reports are even better. These documents can be quite short. The goal is to create milestones that require students to make steady progress.

Most of our students used SimpleScalar [1], and we found it useful to provide concrete familiarization exercises designed to acquaint them with modifying the code to obtain various measurements. These exercises take the students on a tour of the code, require them to use a variety of the configuration and statistics-gathering tools provided by SimpleScalar, and require 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. The entire team can be assigned a single grade, or each student can receive an individual grade.

Everybody must participate. Students can also audit the course, but it is best to require 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, make observations, and aggressively pick up on student comments and turn them into discussions. 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. One-on-one conversations outside of class seem helpful in reassuring students that it is safe to speak up and in making them feel supported. Naturally, it also helps for the instructor to treat naive questions with patience, in order to establish an environment of trust.

Keep the class small. Group sizes of more than four students does not give each student enough presentation time and makes it too easy for a subset of the group to do most of the work. This places on upper limit on the class size that depends on the number of processors covered.

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].
- Installing benchmarks and explaining their use.
- Providing instructions on proper simulation (fast-forward intervals, appropriate inputs, appropriate measurement methodology) [11].
- 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.
- Providing template web pages. This helps the pages that the students produce look reasonably uniform. Some help with web-authoring tools may also be necessary.

5 Evaluation

The goals of the course were to provide students with knowledge about advanced computer-architecture techniques, a working knowledge of the architecture of current microprocessors, and an understanding of the tradeoffs that lead to different design decisions in different processors.

The best indicator of whether the course met its goals was the sophistication of the synopses the students wrote and their questions and discussion during the last phase of the course ("Advanced Material," Sec. 2.4). Most students were able to write about and discuss nuanced issues that related material from the research papers and the design of the microprocessors we studied. The difference in the students' sophistication in the latter stages of the course was a marked improvement compared to the level of discussion in the first round of presentations. Direct feedback also confirmed that the students felt they were meeting all three goals.

6 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. A number of detailed suggestions on how to best implement such a seminar have also been described. An interesting possibility is that the same general format used here might also be applied to pedagogical advantage in other courses and topic areas—operating systems, for example. We also have not tried the course as an undergraduate-only course. This might be equally successful, as long as the students are sufficiently motivated and participate actively in discussion; but the research project might be better replaced with more concrete measurement exercises or with a term paper.

The most recent course materials can be found online at http://www.cs.virginia.edu/˜skadron/cs854_uproc_survey/

Acknowledgments

I would like to thank Doug Clark for his helpful comments and for his assistance when I prepared this course. I would also like to thank the reviewers for their helpful comments. This work was supported in part by the National Science Foundation under grant no. CCR-0082671.

References

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

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

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

[4] K. Diefendorff. HP, Intel complete IA-64 rollout. *Microprocessor Report*, Apr. 2000.

[5] P. N. Glaskowsky. Pentium 4 (partially) previewed. *Microprocessor Report*, Aug. 2000.

[6] J. W. Haskins, Jr. and K. Skadron. Inexpensive throughput enhancement in small-scale embedded microprocessors with block multithreading: Extensions, characterization, and tradeoffs. In *Proceedings of the 20th IEEE International Performance, Computing, and Communications Conference*, pages 319–28, Apr. 2001.

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

[8] D. Parikh, K. Skadron, Y. Zhang, M. Barcella, and M. Stan. Power issues related to branch prediction. In *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture*, Feb. 2002. To appear.

[9] K. Scott and K. Skadron. BLP: Applying ILP techniques to bytecode execution. In *Proceedings of the 2nd Annual Workshop on Hardware Support for Objects and Microarchitectures for Java*, Sep. 2000.

[10] K. Skadron. A microprocessor survey course: Exploring advanced computer architecture in practice. In *Proceedings of the Workshop on Computer Architecture Education*, June 2000.

[11] 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.