Computer Architecture Education at the University of Illinois

Josep Torrellas
Computer Science Department
University of Illinois at Urbana-Champaign, USA
torrella@cs.uiuc.edu

June 1998

Abstract

This short paper briefly describes the computer architecture courses at the University of Illinois Urbana-Champaign. It then examines the opinions of computer architecture undergraduate and graduate students who have taken these courses. Based on these opinions, suggestions are made on how to improve the curriculum.

1 Introduction

The University of Illinois at Urbana-Champaign has for many years been a major center for computer architecture education and research. It has educated people who have later become industry leaders in computer architecture or joined the faculty of other major computer architecture universities. In addition, it has excelled in computer architecture research. To list a few areas, it has contributed with major advances in parallel computer architecture, internet technology, compilation techniques, performance evaluation, networking and operating systems.

Currently, the University has about 8 faculty members whose main interest is computer architecture. These faculty are associated with the Departments of Computer Science and of Electrical and Computer Engineering. Active areas of research include microarchitecture, processor, memory and I/O architectures, uniprocessor and multiprocessor systems design, networking, compilation techniques, and performance modeling and evaluation among others. In addition, there are well over 10 other faculty with interests that overlap with computer architecture.

The University has a large body of undergraduate and graduate students in computer architecture. For example, in Computer Science alone, every semester roughly 150 advanced undergraduate or graduate students take the intermediate course in computer architecture. Furthermore, every other semester, roughly 50 graduate students take the advanced course in computer architecture. Similar numbers are recorded in Electrical and Computer Engineering.

This short paper briefly describes the computer architecture courses at the University of Illinois (Section 2). Then, it examines what the undergraduate and graduate students who have taken the courses think about them (Sections 3 and 4). We finish with a discussion in Section 5.

2 The Computer Architecture Curriculum

The computer architecture courses are cross-listed in the Computer Science and the Electrical and Computer Engineering departments. Most of them are taught by both departments, with some variations across departments. A semester system is followed.

2.1 Undergraduate Courses

The undergraduate courses tend to emphasize basic concepts and hands-on experience. The curriculum starts with a course on logic design. This course is usually taken by sophomores. Its purpose is to train the students in combinational and sequential networks, data path and control unit design and related issues. An important part of the course is a laboratory where students use tools to generate and simulate circuits.

After logic design is mastered, junior students take the course in basic computer architecture. This course introduces the basic architecture concepts, including assembly programming, pipelining and memory hierarchies. The students are required to perform logic design and simulation of several parts of a computer.

In parallel or after this course, students take a hardware laboratory course. This course actually comes in several flavors, and is taken by both undergraduate and graduate students. The work involves the use of field-programmable gate arrays (FPGA) or other commodity components to design systems like memory or cache controllers, processors, memory systems or network components. The purpose of the course is to gain hands-on experience in building a system.

The last undergraduate course is computer organization. This course is taken by both seniors and first-year graduate students. The goal of the course is to give a solid base in computer architecture and give insight into where the research areas are. The main topics covered are pipelining, instruction level parallelism, memory hierarchies, I/O, and multiprocessors. The material is covered in a fairly high-level manner, with an emphasis on quantitative analysis. The course also includes exercises to design and simulate several parts of a computer.

Finally, there is a VLSI design course taken by both graduate and undergraduate students. The course covers tools for design capture, simulation, verification, logic minimization, and timing analysis. Students design and simulate various circuits. One of them is a large system, for example a cache with its control circuitry.

2.2 Graduate Courses

The graduate courses are high-level in nature and emphasize concepts. In general, they can be taken in any order. One course teaches parallel programming. It is taken by both graduate and advanced undergraduate students. The course examines how to program in the shared-memory, message passing, and data parallel paradigms. Students are required to write programs for several classes of parallel machines. They get accounts in several machines from supercomputing centers.

A second graduate course teaches advanced uniprocessor architecture. This course focuses on advanced uniprocessor microarchitecture and compilation issues. It covers issues like advanced instruction issue, branch prediction, register renaming, advanced pipeline design, functional unit organization, predication and advanced code optimization techniques.

A third graduate course teaches parallel architectures. It covers the major architectural issues in current parallel machines. Its focus is on issues, not on descriptions of specific machines. It covers issues like cache coherence protocols, latency-handling techniques, multiple processors on a chip, multithreading, processors in memory, fast networking, operating system support and new workloads. The students are required to examine a research topic in detail.

Another graduate course teaches tools and techniques for performance modeling, monitoring, evaluation, tuning, and debugging of parallel machines. It examines both analytical and experimental methods. It describes existing performance tools.

In addition to these courses, there is a wide variety of courses on special topics in computer architecture. They are not offered on a regular basis, and the actual topic depends on the interest of the faculty member offering the course. Similarly, there is a wide variety of computer architecture research seminars, either offered by the department or offered by individual research groups within the department.

Finally, there are several other courses that complement the computer architecture education of graduate students. Some of the most important ones are those on intermediate- and advanced-level compilers, intermediate- and advanced-level operating systems and programming languages.

3 The Undergraduate Experience

We conducted an informal poll among undergraduate students who had taken the courses described. While their opinions varied, several broad issues emerged.

The students valued the courses as formative, up to date, and relevant to their later careers. They especially valued the hands-on experience that they gathered with the designs in the hardware laboratories. Similarly, they valued the machine problems that allowed them to evaluate the performance of different parts of a computer through simulations. Finally, they valued the broad-based view given to them by the most advanced undergraduate course on computer architecture. In that course, many issues are covered from a high-level point of view, and their relationships analyzed.

Asked about how to improve their education, they suggested more hands-on problem sets that give them insight into the practical implications of the concepts studied. One example that they gave was problem sets on the impact of code blocking on cache performance. Making small changes to the code and seeing wide performance variations on a real machine was an eye opener for many students.

Another improvement would be providing them with some research experience. The students who have had a research experience are almost always enthusiastic about it.

Finally, students suggested increasing the fraction of the instruction devoted to parallel systems. Many students, after they graduate, are developing some type of parallel software or testing hardware that supports some type of concurrency.

4 The Graduate Experience

We also conducted the poll among graduate students who had taken the graduate courses described. In general, the courses are seen as very good vehicles for a student to catch up with the state-of-the-art research and advanced development in the field. In addition, for the students with industrial experience, the courses provide rational explanations to observed behaviors and common-wisdom assumptions.

One of the most positive experiences is the use of simulation and tracing tools in the courses. In addition to giving insight into how a CPU, cache or disk performs, the simulations give a flavor of what research is like. At the beginning of a graduate student's career, such experiences may trigger career decisions that have a long-term impact.

A second resource very valued by the students is the many research seminars that take place in the department. While some of the seminars may seem too deep for junior graduate students, it gives them some background and forces them to think in a broader mind set. Large research-group meetings are particularly valued.

There are, however, some weak points in the curriculum that need to be addressed. One of them is that there is not enough emphasis on low-level, VLSI-like, implementation issues. One example is that caches are studied in great detail from a high-level viewpoint, including the effect of different organizations and parameters. However, how all these issues are actually implemented in VLSI and how area and timing are affected are insufficiently covered.

Another weak point is that there is some duplication of material across courses. Prominent examples of topics suffering duplication are virtual memory and networking. These topics are usually seen by students in other courses. Given the amount of material that needs to covered in the computer architecture courses, any material that is already covered in other courses needs to be eliminated from the former.

Finally, a criticism of the computer architecture discipline is that it is not rigorous enough. Many of the methods and techniques used are based on rules of thumb and tradition rather than provable optimality. In spite of the current emphasis on quantitative analysis, computer architecture is seen as significantly "softer" than other engineering disciplines and sciences.

A related complaint is that the contents of the most advanced computer architecture courses vary significantly depending on the instructor of the course. This may be due to the youth of the computer architecture discipline, which prevents studying the issues with enough perspective. As a result, not everybody agrees on what the most important issues are. Alternatively, it may be due to the relative lack of rigor that was mentioned above.

5 Discussion

The analysis of the students' opinions suggests some ideas to improve the computer architecture curriculum.

The undergraduate curriculum seems to need relatively few changes. We need to keep the laboratories, for they provide valuable hands-on experience to the students. We need to increase the emphasis on machine problems and homeworks that give insight into the practical implications of the concepts discussed in class. It can argued that this can be easily done given the many simulation tools that we have available and the fact that computing hardware is widespread. Finally, we should strive to give more research experiences to undergraduate students.

The graduate curriculum can certainly be improved. In our effort to expose the students to the wide range of new architecture ideas, we cannot neglect to cover the arguably less-glamorous, low-level implementation issues. A possible way to address this issue is to offer courses that cover the boundary between computer architecture and VLSI. This approach seems particularly suitable at this time when large increases in the number of on-chip transistors are driving architectural changes.

It is important that junior graduate students gain insight into what computer architecture research is, early in their careers. One possible way to accomplish this is to offer a course where students simulate parts of a computer. The course could be organized into several two-week long periods. Each of these periods would be focused on one component of the computer, for instance CPU, cache or disk. At the beginning of each period, the instructor would discuss the fundamental issues associated with the corresponding component. Then, the students would be provided with a simulator of that component. They would have to make some modifications to it and simulate certain aspects. This format would allow the students to gain valuable insights without having to code too much. This type of course can have high impact on junior graduate students considering a career in computer architecture.

Finally, there are the issues of relative lack of rigor in computer architecture and wide variations in material covered by different instructors. These issues are more fundamental and harder to address. Maybe the discipline is too young and changing too rapidly to make significant progress on these fronts for now.

Acknowledgments

The author would like to thank all the students who participated in the informal poll, especially Sujoy Basu, Zhenzho Ge, Yi Kang and Vihn Lam.