Computer Architecture Education at the University of Illinois: Current Status and Some Thoughts

Josep Torrellas
Center for Supercomputing Research and Development
and Computer Science Department
University of Illinois at Urbana-Champaign, IL 61801, USA
torrella@cs.uiuc.edu
February 1996

Abstract

The University of Illinois has traditionally been a major center for computer architecture education and research in the nation. This short paper briefly describes the computer architecture curriculum at the University of Illinois and discusses a few ideas for improving the computer architecture education of our students. In particular, to improve the education of undergraduates, we suggest a higher emphasis on parallel architectures and programming, and the provision of practical experiences. For graduate students, we argue about the need to develop close links with industry and educate graduate students from other scientific and engineering disciplines.

1 The Computer Architecture Curriculum

Computer architecture courses at the University of Illinois are cross-listed in the Computer Science and the Electrical and Computer Engineering departments. They are often taught by both departments. We follow a semester system. The major undergraduate courses in computer architecture are:

- **Logic Design**. This course is taken at the sophomore or junior level. The major topics are combinational and sequential networks, ALU design, and control unit design. The course includes laboratory time that involves schematic capture and simulation of circuits. One of the books used is Mano's [3].
- **Basic Computer Architecture**. This course, often taken at the junior level, focuses on basic architecture concepts like assembly programming, pipelining, and memory hierarchies. Books that have been used include Patterson and Hennessy's [5] and Feldman and Retter's [1].
- **Hardware Laboratory**. This course, which comes in several flavors, is taken by both undergraduate and graduate students. The work typically involves the use of field-programmable gate arrays (FPGA) to design systems like memory or cache controllers, processors, memory systems, or network components.
- **Computer Organization**. This course is taken by both seniors and first-year graduate students. It is intended to give a solid base in architecture. The main topics are pipelining, instruction level parallelism, memory hierarchies, I/O, and multiprocessors. The book used is Hennessy and Patterson's [2].
- **VLSI Design**. This course is taken by both graduate and undergraduate students. It covers tools for design capture, simulation, verification, logic minimization, timing analysis, etc. Students design and simulate a large system. The book used is Mead and Conway's [4].

The major graduate courses in computer architecture are:

- **Parallel Programming**. This course is taken by graduate and advanced undergraduate students. It teaches how to program the different types of parallel machines. Students are required to write significant programs for several classes of parallel machines.
- **Advanced Uniprocessor Architecture**. This course focuses on advanced uniprocessors, including microarchitecture, instruction-level parallelism, and other compiler issues.
- **Parallel Architectures**. This course covers the major architectural issues in current parallel ma-

chines. It focuses on distributed shared-memory machines, multicomputers and, to a small extent, data-parallel machines. The focus is on issues, not on descriptions of specific machines.

- **Performance Evaluation, Tuning, and Debugging of Parallel Systems**. This course focuses on the performance of parallel systems. It examines tools and techniques for performance modeling, monitoring, evaluation, tuning, and debugging of parallel machines.
- **Special Topics**. There is a wide variety of courses on special topics in computer architecture. They are not offered on a regular basis.

Finally, there are several other courses that complement the computer architecture education of graduate students:

- **Compilers: Intermediate and Advanced Levels**. These two courses include front- and back-end issues, advanced code optimizations, and transformations for the automatic parallelization of codes.
- **Operating Systems: Intermediate and Advanced Levels**. These two courses cover traditional operating systems, as well as distributed operating systems and networks.
- **Programming Languages**.

2 Improving Undergraduate Education

Undergraduate education in computer architecture can be improved by emphasizing three issues: the study of parallel computer architectures and programming, the adoption of practical approaches, and the provision of research and industrial experiences.

The first issue, namely the emphasis on parallel architectures and programming, is a consequence of the rapid advances in computer systems. Indeed, people in industry who work with computers will increasingly need to master the concepts of parallel systems. Unfortunately, this task is difficult without some background from school. This is because this field changes rapidly and previously-accepted ideas often become obsolete. An obvious example of this is the quick-paced change of the hot keywords in the popular computer press. Furthermore, the relevant information is hard to find beyond specialized articles because there are very few books that contain up to date information. Consequently, an engineer can easily fail to develop the critical mass of knowledge necessary to keep up with the field and, as a result, feel discouraged.

Past experience indicates that, after undergraduate students are taught the basics of parallel systems, they show a high interest in pursuing the subject further. Teaching them some intermediate concepts in parallel systems prepares them to work with more advanced material on their own if they need to later.

The second aspect, adopting practical approaches, is key to motivating students. This means relating the lecture material to the instructor's own experience building and debugging systems. It also means that, as an architectural concept is introduced, it pays-off to examine its implications on the operating system or compiler. An obvious example is the TLB, which has clear operating system and software implications. Finally, a good way of increasing the practical component of our computer architecture courses is to use research tools developed in academia. In particular, it would be nice to include the software of a multiprocessor tracing and simulation tool in the most popular computer architecture textbooks.

Finally, research and industrial experiences for undergraduates nicely complement classroom education. Past experience giving research experiences to undergraduates, however, produced mixed results. While some instances worked and others did not, they all required a lot of time from the professor. Industrial experiences should always be encouraged.

3 Improving Graduate Education

Graduate education in computer architecture can be improved by developing closer links with industry and by paying more attention to graduate students from other scientific and engineering disciplines.

Most people agree that developing close links with industry is crucial. Graduate students should be encouraged to spend one or more summers in industry. Otherwise, they miss one crucial aspect of their education: the experience of a real computer systems design environment. In addition, of course, their interaction with industry can be the source of new research ideas.

Finally, it is necessary to make computer architecture education more accessible to graduate students from other scientific and engineering disciplines. The reason for this is that large parallel machines increasingly require programmers to understand many details of the architecture to run programs with acceptable performance. This trend, already obvious, is likely to become more marked in the future. Often, however, the programmers of these machines are physicists or chemists who have little background in computer architecture. Consequently, we need a graduate course that covers the most important topics of undergraduate computer architecture courses. This course should be designed for scientists or engineers who have no background in computer architecture, are very motivated, have extensive experience programming at least one parallel machine, and want to get up-to-speed in computer architecture very fast.

4 Conclusion

The next few years will be a time of change for computer architecture education. There are at least two important challenges to meet. The first one is the need to put more emphasis on parallel architectures and programming issues in undergraduate education. The second one is the need to educate graduate students from other scientific and engineering disciplines. These challenges need to be addressed soon.

References

[1] J. M. Feldman and C. T. Retter. *Computer Architecture: A Designer's Text Based on a Generic RISC.* McGraw-Hill, New York, NY, 1994.

[2] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[3] M. M. Mano. *Computer Engineering Hardware Design.* Prentice Hall, Englewood Cliffs, New Jersey.

[4] C. Mead and L. Conway. *Introduction to VLSI Systems.* Addison-Wesley, New York, NY, 1980.

[5] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design: The Hardware / Software Interface.* Morgan Kaufmann Publishers, San Francisco, CA, 1994.