The Computer Architecture Sequence at Michigan State University

Richard J. ENBODY

Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824-1226
enbody@cse.msu.edu

Abstract

This paper outlines the computer organization and architecture courses in the Department of Computer Science and Engineering at Michigan State University.

1 Introduction

Computer Architecture has been a key component of the Michigan State University's Computer Science degree since it was established in 1967. The four founding members of the department were Electrical Engineers, some of whom came to build MSU's first computer (MISTIC) in 1956. Two are still on the faculty and a third is a Professor Emeritus. The influence of those fledgling computer builders remains.

Currently the cornerstone of our undergraduate architecture sequence is a simulator developed by Dr. Reid, one of those who built the MISTIC computer almost forty years ago. The emphasis of our computer architecture course has been the construction of a simulated processor capable of running a simple operating system and executing programs. That is, the students build something, and it has to work.

Students in the computer organization course build small circuits using the simulator. Students in the graduate courses generally do not build things, but work with real hardware as described in our other paper.

2 Undergraduate Architecture Sequence

Michigan State University uses the semester system with fifteen weeks of classes per semester.

2.1 CPS 320 Computer Organization and Assembly Language Programming

This course is a required four-credit course with three hours of lecture and two hours of laboratory per week. Prerequisites are an Introduction to Computer Programming (CS1) course using C++ and a course in Discrete Mathematics. The current text is *Architecture, Organization and Programming* (Maccabe; Irwin, 1993). The official description is:

> Machine representation of data and instructions. Machine organization, primary storage, registers, arithmetic local unit, control unit, operations. Assembly language programming, interface to high level languages. Assemblers and loaders.

This course develops an understanding of:

- the organization of digital computing systems,
- the representation of and operations on basic data types,
- the process of translating and executing a computer program.

The primary vehicles to achieve these objectives are the study of general concepts in this area, and the study of a specific computing system which illustrates these concepts. Students use a simulator to construct combinational and sequential circuits. They also write assembly language and C programs in a UNIX environment on a SPARC system.

2.2 CPS 420 Computer Architecture

This course is a required four-credit course with three hours of lecture and two hours of laboratory per week. The prerequisites are Data Structures and Algorithms (CS2), Computer Organization, and Theory of Computing. The textbook is *Computer Organization and Design* by Patterson & Hennessy (Morgan Kaufmann). The official description is:

> Digital logic and sequential machine design. Computer organization, control unit and arithmetic logic unit implementation. Input-output, memory organization, parallel operations. Digital system simulation.

In this course the student use simulation to design and implement a central processing unit for an up-to-date architecture. Simulation components ranging from gates to complex devices such as multiplexers, arithmetic-logic units, and read-only memories are explored. The design project includes the following:

- development of the instruction set
- implementing the datapath and datapath control
- memory interface
- input-output interface
- development of modules representing subsystems
- testing of individual modules

The students learn to understand issues related to timing in digital systems such as pipelining, data hazards, and branch hazards; learn algorithms for doing arithmetic in digital processors; and learn to design and analyze the effects of various choices of number of levels of cache, main memory, and other secondary memory devices. The students are able to analyze basic performance characteristics and compare different processors based on clock speed, number of cycles per instruction, and instruction mix. Finally, students come to understand design issues related to input and output, including the use of memory-mapped input-output, buses, direct memory access, and programmed input-output.

A valuable characteristic of the current simulator is that, in addition to the normal values of ONE and ZERO, it also has a TRANSITION value between ONE and ZERO. This third state makes the simulator more realistic than a simpler two-state simulator, and greatly enhances the educational value of the course. Also, the simulator is hierarchical so students can work at the gate level, register-transfer level, or at a higher level.

2.3 CPS 474 High-Performance Computing

This course is an optional three-credit course offered once a year. It has two hours of lecture and two hours of laboratory per week. The prerequisites are Computer Architecture and Linear Algebra. The text is *High Performance Computing* by Severance and Dowd (O'Reilly and Associates). The official description is:

> Programming of high-performance supercomputers. Hardware, algorithms, numerical accuracy, compilers, vector, multiple-instruction multiple-data-stream, and single-instruction single-data-stream machines.

This course focuses on how to program high performance computing systems. The course views computer architecture from a performance perspective. Students learn how to develop their own benchmarks to measure the performance of the various components of a computer. Students study how the central processor, memory hierarchy, floating point numbers, compilers and parallel processing architecture impact their programs. The course examines single processor and multiple processor systems from an architecture and programming perspective.

2.4 Experience With Our Undergraduate Courses

The Department has been quite satisfied with the architecture component of the curriculum. In particular, the large project to develop a simulated CPU in CPS 420 plays an important role in the curriculum well beyond its immediate value as a tool for understanding computer architecture. The project is developed in a series of modules over the course of the semester and, when taken as a whole, it is a very large project. For most students it is the first time they have worked on a computer project which spans fifteen weeks of work. As a first experience it provides students with a very structured approach to a large project and prepares them for the Software Engineering course and our Capstone Design Course. Software Engineering can be viewed as the engineering of large programs so the experience of working on a fifteen-week project in CPS 420 provides a foundation for appreciating the concepts of large system design. Since the Capstone Design Course also involves a semester-long project the experience in CPS 420 is invaluable.

However, the architecture project provides another benefit students must use modules they construct. The stages of the project involve construction of the components of the CPU to be simulated. The components must interact with each other and some use components designed by the student earlier in the course. Eventually the components are combined into a working computer. The complexity of the components make them almost impossible to exhaustively test so students frequently find themselves having to fix modules which they thought worked correctly. The process of "eating one's mistakes" is a valuable learning

experience. The concepts of correct design and thorough testing are reinforced. An interesting characteristic of the course is that the a clever student can be beaten by a meticulous student.

The response to our CPS 474 course on High Performance Computing is a study in contrasts. Good, inquisitive students who have taken the course speak enthusiastically about the insight they gained on computing from taking the course. However, enrollment for the course continues to be quite low in spite of having high-quality instructors for the course. Other elective courses such as software engineering, graphics, and databases continue to out draw it.

3 Graduate Architecture Courses

3.1 CPS 820 Advanced Computer Architecture

This course is a three-credit course with three hours of lecture per week, and Computer Architecture as a prerequisite. More than half the lectures are devoted to supplementary topics, but otherwise it covers the first six chapters of *Computer Architecture: A Quantitative Approach, Second Edition* by Hennessy & Patterson. Supplementary topics this semester are benchmarking, SPEC95, performance monitoring (PERFMON), DRAM technology, disk technology, EPIC & IA-64, Alpha 21264, P6 "backside" bus, PCI "frontside" bus, and asynchronous processor design.

The official course description is

> Instruction set architecture. Pipelining, vector processors, cache memory, high bandwidth memory design, virtual memory, input and output. Benchmarking techniques. New developments related to single CPU systems. Objectives In this course students will study advanced concepts in computer architecture. The emphasis of this course is single-CPU systems.

This semester's emphasis was on the analysis of computer architectures. We worked with software tools which allow access to the set of special on-chip counting registers on UltraSparc and PentiumPro microprocessors. The availability of this set of tools allowed us to investigate properties previously inaccessible outside the major chip producers. (See other paper submitted to the ISCA Workshop on Education.) We also purchased a copy of the SPEC95 benchmark suite for our use.

3.2 CPS 822 Parallel Processing Computer Systems

This is a three-credit course with three hours of lecture per week, and Advanced Computer Architecture as a prerequisite. The official course description is:

> Massively parallel SIMD processors, multiprocessor architectures, interconnection networks, synchronization and communication. Memory and address space management, process management and scheduling. Parallel compilers, languages, performance evaluation.

The objective of this course is to consider a number of topics pertaining to the architecture of parallel computing systems. From the prerequisite CPS 820 course, students know that computer architecture is concerned with organization, software, and algorithms. This course covers just two sections of the Henessy and Patterson text in order to be certain that students have a common ground and understand the basic terminology. Students read a number of papers that cover a spectrum of topics. Several videos of presentations by industry leaders on certain aspects of parallel computing are also be studied. In addition, students are responsible for submitting short critical reviews of the assigned papers and being part of a project group.

3.3 CPS 920 Topics in Computer Architecture

A three-credit topics course is offered in the Spring semester. The choice of topic and emphasis of the course is up to the instructor. The last offering covered the latest developments in microprocessor architecture, i.e. the significant developments since the latest texts and beyond what was covered in the Advanced Computer Architecture course.

3.4 Experience With Our Graduate Courses

Our CPS 820 Advanced Computer Architecture course is one of the courses in greatest demand in the graduate program. Most graduate students take the course, and students in Electrical Engineering who have an interest in Computer Engineering also take the course. It varies in its emphasis depending on who teaches it, but has tended to have some experimental component to it. We believe that students should be experimenting with architectures. The course is well received, with negative comments coming from students who wish to build something rather than study or experiment with concepts.

The Parallel Architecture course has often had programming as a component. The belief is that some amount of programming is needed to experience the unique characteristics of parallel architectures. Students enjoy playing with different machines and languages. Providing local hardware is difficult, but remote access has generally be relatively easy. Programming is not emphasized in this course since there is a sister course: CPS 838 Parallel Algorithms. As symmetric parallel architectures have become more prevalent and the variety of commercially available parallel architectures has decreased interest has waned in the course. It is still popular, but is no longer the first course to fill.

4 Related Courses

In addition to these specific architecture courses there are a number of systems courses which attract similar students. Examples of such courses are operating systems and networks course at both the undergraduate and graduate level, as well as a topics course. Courses such as hardware arithmetic, VLSI design, and implementation of neural networks are available in our Department of Electrical and Computer Engineering (formerly the Department of Electrical Engineering).

5 Conclusion

Computer Architecture has been an important component of the computer science program since its inception thirty years ago. We feel that it is a core component to our identity as a program. However, the field of computer science is constantly changing and there are those who feel that in-depth study of computer architecture is no longer central to every computer science student's education.