An Active Learning Environment for Intermediate Computer Architecture Courses

Jayantha Herath, Sarnath Ramnath, Ajantha Herath*, Susantha Herath

St. Cloud State University, St. Cloud, MN 56301

*Marycrest International University, Davenport, IA 52807

herath@stcloudstate.edu http://web.stcloudstate.edu/jherath/CompArch-2

Abstract

Most computer science, information systems and engineering programs have two or more computer architecture courses but lack suitable active learning and design experience in the classroom. Computer architecture at the intermediate level should focus on the implementation of basic programming constructs in different instruction set architectures. To accommodate such features we developed an undergraduate computer architecture course with hands-on classroom activities, laboratories and web based assignments. To assess the course we distributed the course modules among 200 computer architecture instructors. This paper describes our experience in developing active learning course modules.

1. Introduction

During last fifteen years, we have been experimenting with methods to improve the quality and efficiency of teaching computer architecture courses for undergraduate computer science and engineering students. Our goal has been and continues to be to help them become good computer scientists in a relatively short period of time with both theoretical understanding and practical skills so that they can enter and make an effective contribution to the profession. Traditionally, computer architecture subject matter has been presented to a less than enthusiastic student body in a relatively passive classroom environment. In general, this chalk-talk instructional process consists of multiple copying stages: the instructor first copies notes from a textbook to his note book, then the instructor copies those notes onto the blackboard, thereafter the students copy notes into their note books. Moreover, each instructor allocates considerable chunk of his/her time to prepare or update the same course material in each offering. In addition, there is both local and national need for high-quality trained labor with the ability to stay current with the technological advances in the computer architecture field .

Growth of any undergraduate computer science or engineering program will largely depend on the strength of the computer architecture curriculum. To address the deficiencies in traditional curriculum [4-10] and to satisfy the current needs, we redesigned our computer architecture course sequence with fundamentals to incorporate rapidly changing computer related technologies so that our graduates will be current with the technologies before they graduate. It is hypothesized that the learning rate can be increased if both the instructor and the student are active at the same time. Thus the performance of the students can be improved dramatically by converting the traditional passive classroom into an active hands-on learning environment. Designing a course with learning-by-doing modules and making it available for all the

instructors on-line [1] reduces the course preparation time for instructors, reduces multiple copying steps in the learning process, strengthen the abilities and increase the enthusiasm of both traditional undergraduate students as well as the adult learners.

Goals and Objectives

The main objective of this project was to develop computer architecture course modules for intermediate level undergraduate students and the faculty. These active learning modules are central to achieve the following goals:

- To provide the students an efficient, rigorous and engaging learning environment with necessary tools and training to become proficient in the computer architecture subject matter in a relatively short period of time.
- To provide architectural details necessary to implement basic programming constructs learned in CS-1 and CS-2 with hands-on skills, integration, team-work and hence to enhance the quality of the graduates.
- To use performance focused learning at all levels of curriculum to illustrate the principles of computer architecture.
- To provide the faculty and students modifiable on-line courseware with state-of-the-art hardware and software practice.

Following sections outline the details of course plan, goals achieved, difficulties encountered, assessment plan future work and summary.

2. Detailed Course Plan

The course, outlined below, will address the ways of reducing the deficiencies in the existing curriculum [4-10]. When developing and delivering the computer architecture subject matter for computer science majors, we believe that the prime factor to be focused on in any step is processor performance in implementing programming language constructs. Our curriculum consists of three semester courses to help master the computer architecture subject matter in a technology integrated classroom laboratory. First course of this sequence will cover fundamentals of architectural design [11-13]. The laboratories for this course consist of hardware and software simulations of combinational and sequential digital circuits. This foundation will help to develop the skills from gate level to register transfer level component integration in design. The intermediate level course that we designed introduce both complex instruction set and reduced instruction set processor architectures, instruction set manipulations with I/O, memory, registers, control and procedures [1-2][14-16] to the students. The laboratories for this course consist of hardware and software simulations of programming constructs in CISC and RISC architectures. After completion of intermediate course the students will be able to learn architectural details of any other processor. The third course is focusing on the advanced concepts in architecture involving parallel/distributed computations and special purpose architectures to provide both depth and breadth to the subject matter. Parallel processing and special purpose processing concepts in the undergraduate curriculum has been the focus of several curriculum improvement efforts for some time [3][17-18]. The students should be able to understand the importance of parallelism in enhancing performance and its benefits as an application programmer, a systems programmer, an

algorithm designer, and a computer architect. A course sequence with the features outlined above could help our students develop design skills in several different architectures before their graduation. The undergraduate curriculum, graduate programs and industry will definitely appreciate the graduates with such design skills.

Topics for Computer Architecture II

At the intermediate level we introduced processor design and focused on the implementation techniques of basic programming constructs such as I/O, arithmetic expressions, memory operations, register operations, if-else-for-while control and functions in several different instruction set processor architectures. Two complex instruction sets and one reduced instruction set processor architectures were introduced in our course. Students learned that proper instruction set design, memory management and I/O management techniques will also lead to the performance enhancement. Increasing performance of the processor by reducing the program execution time is considered at each design and implementation. Focusing on the importance of performance when designing the processor helped to maintain the momentum and enthusiasm in the classroom. Often the students were excited to observe the register level manipulations in the processors. They also enjoyed discovering the processor and controller designs in an active classroom. Comparing different architectures including pipeline techniques and abstracting the essentials of the processor architectures at this level generated the required enthusiasm to the learning and teaching. The required textbook for this course is Paterson and Hennessey [2]. We are looking for ways to integrate rapid prototyping of the systems to the course using web based tools.

Hardware/Software Laboratories

To provide architectural concepts with hands-on skills, integration, team-work and hence to enhance the quality of the graduates, we added pre-lab, in-lab and post-lab assignments to complement the classroom activities. Table 2 summarizes the educational experience gained from these laboratories.

Table 2. Educational experience

Experience	Level	Application
Prelab	**Analysis, synthesis**	**Design circuits and programs to perform a specific simple task**
Closed Labs	**Application, analysis, synthesis, evaluation**	**Design, implement and test circuits and programs to perform a specific task within a given period of time**
Open Labs	**Application, Analysis, synthesis, evaluation**	**Design circuits and programs to perform a difficult task**
In-class activities	**Application, Analysis, synthesis**	**Cost reduction, performance improvement, integration**
Tests	**Analysis, synthesis**	**Architecture design related questions**

To reflect student-centered design and analysis processes, classroom activities were modified to accommodate skills in performance improvement and cost reduction when designing processors. In general, pre laboratory assignments helped the students explore and create on their own. They synthesized the classroom instructions with other resources to produce hardware and software and then to test and to debug. In the classroom, each student provided with a computer and tool kit to extend the concepts they learned in the pre lab assignment. Less challenging design problems

that can be solved within a given period of time were assigned as in-class closed-laboratory assignments. A post-lab assignment helped the students to analyze the use of in-class activities. More challenging and time consuming problems were assigned as post laboratories. Students were active in both laboratory and in the classroom while thinking and experimenting on a machine with the architectural concepts introduced in the classroom. After completing each project, students submitted a report discussing their experience. First, each student worked alone as the sole developer of the hardware and software. Towards the end of the semester two to four students were allowed to work in a team to design, construct and complete the projects. The group was responsible for developing definitions and specification of a larger hardware/software product to solve a problem as their final project. The course helped students become proficient in the subject matter in a relatively short period of time.

3. Goals Achieved

We created the active learning course material that will enhance students' high level skills: teamwork, analysis, synthesis, performance comparison and active participation in the classroom. To reflect the inclusion of several different instruction set architectures we created hands-on hardware and software laboratory assignments. Our computer science students received the instructions based on the course material developed in Spring 2002 and Fall 2001.

Our active learning course modules enabled students to learn architectural concepts more effectively and efficiently thus providing students an opportunity to function well in an increasingly competitive technical society. The classroom activities provided the students with opportunities for analysis, synthesis, and verification of correctness in building larger systems all during traditional class time. Such modifications increased the enthusiasm in the classroom, addressed the needs of both traditional undergraduates and adult students, the needs of the industry and provided necessary tools and training for the student to become proficient in the computer architecture subject matter in a relatively short period of time. To our knowledge, no other computer architecture course used our approach. Therefore, our course modules and experimental results will be very useful for the other computer science and engineering programs nationally. Table 3 depicts the Indicators/Measurements of goal attainment of all three courses.

Table 3. Indicators/Measurements of goal attainment

Entry level	Intermediate level	Goal Attainment
Gate level design and analysis	**Design, analysis and performance improvement of architectural components, processors, controllers**	**Parallel processing, system design, analysis and performance improvements**
Exams	**Exams**	**Exams**

Architecture Symposium

A computer architecture symposium [21-22][24] was organized at the end of the Spring'02 semester to stimulate our undergraduate and graduate students, computer science and engineering faculty in tri-state and the local industry. We invited five excellent speakers from MIT, University of Minnesota, IBM T.J. Watson center and Oracle to deliver lectures based on their work. The symposium was well attended by the students, faculty and industry. Spring'02 semester started with introduction seven trillion FLOPS machine, then the students learned about 35 trillion FLOPS machine. At the end of the semester in the symposium students learned about the 185 trillion FLOPS machine

under development at IBM. This conference also helped our efforts to develop a core curriculum for Computer Science that presents an integrated view of hardware and software to the undergraduate students.

Difficulties

Incorporating several architectures into one course seemed overloading the students and faculty at the beginning. However, making our course modules available for the students at the beginning of the semester via web helped to eliminate this difficulty. Selecting a series of projects that increases enthusiasm in a diverse body of students was also a difficulty we encountered. Observing, helping and verifying the correctness of weekly work focusing on the analysis and synthesis of components was a time-consuming task. Trained student assistants helped in scheduling the laboratories and reduced the burden. However, attracting suitable student assistants and paying them sufficiently to keep them was also another difficulty we faced. Identifying suitable modern educational circuit boards for our experiments was another difficulty we faced.

St. Cloud State University, with six colleges, is the second largest university in Minnesota. The university enrollment is approximately 15,000 students drawn from MN, rest of the USA and foreign countries. The computer science department, among the 10 departments of College of Science and Engineering, is one of the two CSAB accredited departments in MN. The department consists of 180 undergraduate major students, 30 graduate students and 10 full time faculty members. We have two departmental laboratories with 50 PCs for introductory programming classes, architecture and operating systems. Most of the graduates enter industry or graduate school after graduation. Computer science department offered the computer architecture I course twice during the academic year 2002/03 for about ninety students. Computer architecture II course is offered twice a year during the academic year 2002/03 for about sixty students. Computer architecture III course is offered once during the academic year 2002/03 for about forty students. We graduated 25 students this year.

4. Course Assessment

The course material developed was evaluated by soliciting the criticism from the faculty and students. Student learning was evaluated using many different ways. The background knowledge and preconception checks were performed in the form of a simple questionnaire/worksheet that the students will fill in prior to working on the lab assignments. The students were asked to explain the concepts they have learned so that the instructor can measure student learning. Faculty and teaching assistants regularly observed the team work. Recording experiences from laboratory assignments was an essential part of the student work. Student groups submitted weekly project reports. Group-work evaluations were also used to assess the course. In the larger lab projects, students worked together in groups. Each member turned in an evaluation of his/her own learning experiences gained by being part of a team. To reinforce the learning, a test was scheduled after the completion of each module. Excellent students performed well in all levels and had complete understanding of the subject matter. Very good students were strong in many areas but weak in some. Average students showed weaknesses in some levels. Poor students could not perform well in many areas. Classroom opinion polls and course-related self confidence surveys were also performed to receive the feedback. In the future, comments from the industrial advisory committee and accreditation board member's

site visit and reviews from other instructors will be used to evaluate the project performance. Within our large university system we will have opportunities to test our designs which could possibly extend to other faculty and students. We are currently in contact with many computer architecture instructors to find ways to improve the courses we teach.

Dissemination of Course Modules Among Instructors

To disseminate the findings of this project, laboratory manuals, course notes and other related information, the web is heavily used. Before the start of Spring'02 semester, we contacted approximately 600 computer science departments using our distribution list and informed the availability of our course modules for their classroom use and review with no charge. More than 200 computer architecture instructors requested the course modules. We distributed our lecture notes among them via e-mail. A better version of our course material is now available to others for classroom use [19-20]. It is important to note that we have successfully completed the introduction to computer architecture project earlier and distributed the course material to more than 200 instructors. We will continue assessing the course material through faculty and student feedback for next few semesters. We will continue to share the experience gained from this experiment with the rest of the computer architecture community. Progress of this project will be reported to the MnSCU Center for Teaching and Learning.

5. Summary and Future Work

Traditionally, computer architecture courses are presented, with complexity and confusion, to a less than enthusiastic student body and often delivered in a relatively passive classroom environment. In general, learning takes place if both the instructor and the student are active at the same time. To promote this in the classroom and to overcome the above mentioned deficiency, we developed an intermediate computer architecture course with hands-on classroom activities, laboratories and web based tools and distributed among many computer architecture instructors. Other deficiencies encountered in the traditional learning environment such as instructor's preparation time and multiple copying stages involved in the learning process were also addressed. Availability of properly designed and developed on-line course materials, with a series of hands-on laboratories as well as classroom activities will definitely reduce both instructors' preparation time and multiple copying stages, and increase student learning rate. Such on-line courses could help both traditional students and adult learners to explore the computer architecture area while developing their design and analysis skills. Modifiability and flexibility of course material at the instructor's end will contribute very much to the faculty development. Often, the students are confused because of not having a well-defined focus in the classroom activities. This computer architecture course is designed to complement the activities performed in CS-1, CS-2 and computer architecture-1 courses. The subject matter provides the gateway for advanced studies in computer architecture and other areas. The course helped to understand the implementation details of basic programming constructs in CISC and RISC architectures. Performance issue is considered in all alternative designs. This courseware helped students to be active in the classroom and increased the enthusiasm in learning computer architectures. Hardware description programming experience allows description of the structure, specification using a familiar programming language and simulation before being manufactured. As a result, students as designers can quickly compare alternatives

for high performance and test for correctness. We are planning to use a industry-standard hardware description programming language [23] in both first and second level courses. Developing a clustered computing environment will be useful for the laboratories in the third course of the sequence. Educational circuit boards with several processors that communicate with each other through dedicated channels will be a good alternative for the advanced course. Virtual environments with variety of visualization systems are matured enough to aid students' understanding of miniaturized complex processor architecture. Through such platforms students will learn to appreciate the instruction set architecture. In the future revisions we will explore the feasibility of incorporating such virtual environments in the computer architecture classroom [7][8] and then improving upon them in successive iterations.

Acknowledgments

This project has been supported by the MnSCU Center for Teaching and Learning through the Bush/MnSCU Learning by Doing Program.

6. References

1. A Web-Based Computer Architecture Course Database, Edward F. Gehringer http://www.csc.ncsu.edu/eos/users/e/efg/archdb/FIE/2000CACDPaper.pdf
2. Computer Organization and Design: Hardware/Software Interface, Second Edition, John L. Hennessy and David A. Patterson, 1997 http://www.mkp.com
3. Computer Architecture: A Quantitative approach, Third Edition, John L. Hennessy and David A. Patterson, 2002 http://www.mkp.com
4. The Undergraduate Curriculum in Computer Architecture, Alan Clements, http://www.computer.org/micro/mi2000/m3toc.pdf
5. Teaching Design in a Computer Architecture Course, Daniel C. Hyde, http://www.computer.org/micro/mi2000/m3toc.pdf
6. Rapid Prototyping Using Field-Programmable Logic Devices, James O. Hamblen, http://www.computer.org/micro/mi2000/m3toc.pdf
7. PUNCH: Web Portal for Running Tools Nirav H. Kapadia, Renato J. Figueiredo, and José A.B. Fortes, http://www.computer.org/micro/mi2000/m3toc.pdf
8. Building Real Computer Systems Augustus K. Uht, Jien-Chung Lo, Ying Sun, James C. Daly, and James Kowalski, http://www.computer.org/micro/mi2000/m3toc.pdf
9. HASE DLX Simulation Model Roland N. Ibbett, http://www.computer.org/micro/mi2000/m3toc.pdf
10. An Integrated Environment for Teaching Computer Architecture Jovan Djordjevic, Aleksandar Milenkovic, and Nenad Grbanovic, http://www.computer.org/micro/mi2000/m3toc.pdf
11. Digital Design, 3/e **2002** Morris Mano, http://prenhall.com
12. Digital Design: Principles and Practices, Updated Edition, 3/e **2001** John Wakerly, http://prenhall.com
13. Digital Design Essentials and Xilinx 2.1 Package, 1/e **2002** Richard Sandige, http://prenhall.com
14. Computer Systems Organization and Architecture John Carpinelli (2001), http://awl.com
15. COMPUTER ORGANIZATION, Fifth Edition V. Carl Hamacher, Zvonko Vranesic, Safwat Zakay, http://mhhe.com
16. Computer Systems Design and Architecture, 1/e **1997** Vincent Heuring , Harry Jordan, http://prenhall.com
17. Parallel Computer Architecture: A Hardware/Software Approach David Culler and J.P. Singh with Anoop Gupta, August 1998 http://www.mkp.com
18. Readings in Computer Architecture, Mark D. Hill, Norman P. Jouppi, and Gurindar S. Sohi, September 1999, http://www.mkp.com
19. Computer Architecture I Preliminary version http://web.stcloudstate.edu/jherath/CompArch-1
20. Computer Architecture II Preliminary version http://web.stcloudstate.edu/jherath/CompArch-2
21. Hardware/Software Interfacing for High Performance Symposium -02 http://web.stcloudstate.edu/jherath/Conference.htm
22. The RAW Microprocessor, M.B. Taylor etal, MICRO 2002March http://dlib2.computer.org/mi/books/mi2002/pdf/m2025.pdf
23. VHDL Primer, A, 3/e **1999** Jayaram Bhasker, http://prenhall.com
24. VLSI Digital Signal Processing Systems: Design and Implementation, K. K. Parhi, 1999, http://wiley.com