Using Streaming Media and Java Applets to Introduce Computer Organization

Gene A. Tagliarini and Edward W. Page

Department of Computer Science, Clemson University

Abstract

Technology-based curriculum delivery is rapidly becoming a focus of activity across engineering programs. This paper describes the use of two World Wide Web technologies to develop and present materials in support of an introductory course in computer organization. The technologies are streaming media presentations and Java applets. The streaming media presentations supplement classroom presentations by providing brief lectures that review content. The Java applets provide practice in converting among integer representations, animations of data flow within a computer, and a detailed simulation of the behavior of a simple central processing unit. These resources are accessible for student use via the web.

Introduction

Correlated with increased awareness of the Internet has been a rising interest in using the Internet as a vehicle for teaching. With the advent of Internet courses, students are no longer required to be physically present on a campus to obtain the benefit of instruction. In addition, their participation in course activities no longer requires that a student's schedule be synchronized with an instructor's schedule. The asynchronous learning networks web site [7] is a significant resource whose content addresses issues ranging from pedagogic philosophy [2, 3, 5] to implementation technology [1, 6, 8] relevant for on-line course developers.

The course for which the web materials described here are being developed and tested is a sophomore level introduction to computer organization taught as part of a curriculum in Computer Science. As such, the course presents the principle components of a computer data path including memory, input/output devices, and the central processing unit (CPU) with special emphasis being placed on the CPU.

A major focus of the class lies in developing a register transfer-level simulation of a CPU as described by Mano [4]. The primary goal of developing the simulation is that the student will gain a practical understanding of the flow of information within a CPU. In addition, the student develops a grasp of the operation of the control unit and its influence upon the register transfers that implement the semantics of instructions. Thus, students are confronted with a key issue in computer design: the influence that the semantic complexity of instructions exerts upon the difficulty of CPU design.

Students are provided with an abstract description of the functionality of the CPU. The description is formulated in terms of register transfers, which are the movements of information along the CPU data path. CPUs use register transfers to accomplish many activities such as retrieving instructions so that they can be interpreted and executed, presenting addresses to memory so that values may be stored or retrieved, and sending values to the arithmetic and logic unit so that new values can be computed.

As the students simulate the processor, they are also required to deal with a variety of data representations that are commonly used in modern CPUs. In the process, they are required to develop programs to convert between hexadecimal, binary, and decimal numeric representations.

For many of the students taking the course, the simulation is the first large program that they have written from scratch. While no specific programming language is required, the majority of the students employ Java. Once various data structures have been initialized, the logical content of the program consists of a simple loop that repeats the processes of fetching, decoding, and executing instructions as well as checking for interrupt service requests. Arrays are the most complex data structures that are required. The program must simulate the semantic activity specified for each instruction by fetching individual instructions, parsing each instruction, and then executing each instruction by transferring information between the affected registers or memory cells and setting any appropriate CPU flags.

Typically, the course has been taught using a traditional lecture format with the students availing themselves of whatever additional resources they might acquire. With the emergence of vehicles for web-based delivery, the authors sought to supplement classroom presentations with streaming mini-lectures and Java applets that students access asynchronously via the web [9]. The mini-lectures rehearse material presented in class. The Java applets provide opportunities for drill and practice as well as a benchmark for the performance of the students' simulations.

Streaming Media Mini-Lectures

In order to use the web efficiently as a delivery vehicle for lecture content, the supplements are being developed using streaming media technology from RealNetworks [10]. Streaming media files enable the user to deliver audio and video content over the web while being parsimonious with respect to the client's computer memory resources. Conventional media file transfers require that the entire file be copied from the server to the client before the file is played. With streaming media files, only a portion of the file needs to be transferred prior to initiating the playback. As the media file is played, additional content is transferred and buffered so that it can be presented when needed. Thus, only so much of the client computer's memory resources are consumed as are absolutely required. In fact, the entire media file is likely never to exist intact on the client machine.

Developing streaming mini-lectures involves a two-step process that uses Microsoft PowerPoint and a "plug-in," known as RealPresenter, available from RealNetworks. The first step is to develop and narrate a PowerPoint presentation as one would normally. The RealPresenter plug-in is then used to convert the narrated presentation to the streaming media format. In order for students to use the streaming media presentations, they must have a web browser that is equipped with a streaming media player, such as RealPlayer. RealPlayer is also available from RealNetworks and can be downloaded over the web.

Processor Components

The mini-lecture on processor components presents an overview of the four principle elements commonly used in modern CPU designs: the arithmetic and logic unit (ALU), special purpose registers, general purpose registers, and the control unit. The course includes a segment on the design and implementation of ALU functions. The basic concepts of sequential circuits are presented using flip/flops, and then flip/flops are used to create both general and special purpose registers. Students are not expected to have been previously exposed to gate-level logic circuit design.

From the student's viewpoint, the most complex aspect of CPU design is the development of the control unit. Since a CPU simulation must mimic the behavior of the registers and the data path as they would respond to the control signals issued by a control unit, the simulation models the control unit. Consequently, one slide presentation gives a detailed description of every state change that occurs within the CPU for every clock cycle. The presentation is cast in the context of programmable logic arrays (PLAs), which are commonly used to develop control circuitry. In addition, the PLA diagrams enable the description to include concrete detail for every step in the CPU's instruction cycle. For example, Figure 1 illustrates a PLA diagram that represents every action that occurs on the CPU data

path for a memory access instruction. The discussion that accompanies these slides normally occupies several class meetings.

CPU Organization

The mini-lecture on CPU organization presents three architectural approaches to processor design: the stack machine, the accumulator machine, and the general register machine. In addition, to providing the student with an overview of these architectures, the mini-lecture rehearses the historical development of computer organization and relates it to the development of memory technology. The presentation also provides a context in which to describe the "von Neumann bottleneck," which occurs when both the program and its data are stored in memory, and some of the organizational strategies that have been developed to reduce the negative impact of the bottleneck.

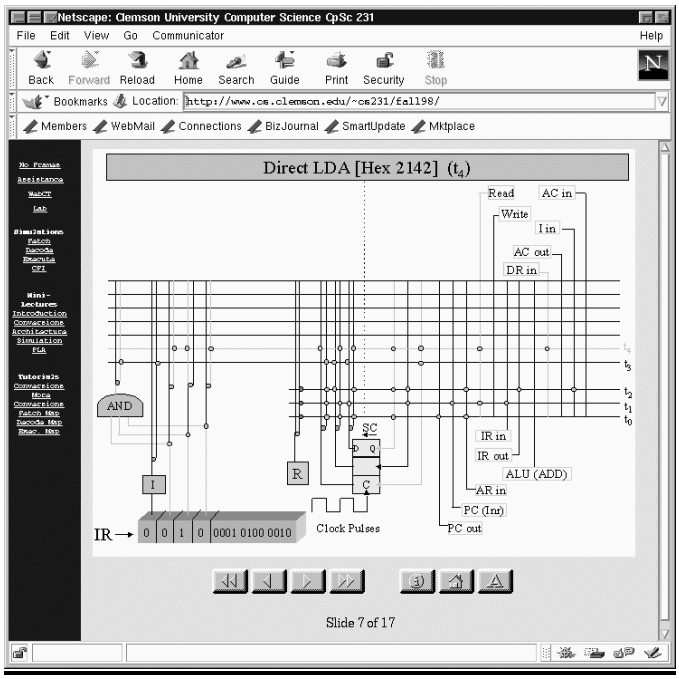

Figure 1. A PLA diagram for a data access instruction

Integer Data Representations

Figure 2 illustrates one of the slides from the mini-lecture that presents the process of converting from one integer representation to another. The presentation begins with radix conversions between decimal and binary. After presenting techniques for radix conversion, the mini-lecture applies conversion algorithms to illustrate how fixed representation widths are employed to store integers using the unsigned, signed-magnitude, and two's-complement formats. The current mini-lecture does not include conversion of floating point values but a mini-lecture is planned to address this need.

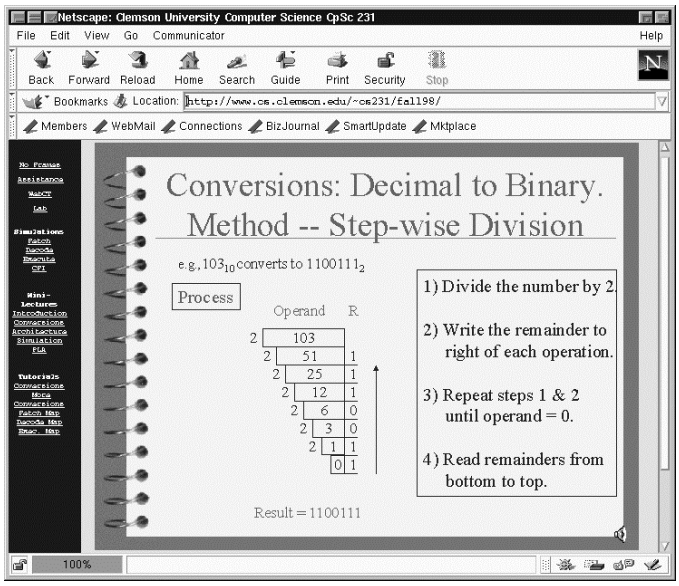

Figure 2. A slide from the integer format conversions mini-lecture

Interactive Applications

Conversions

One important advantage of using the web to deliver supplemental course material lies in the potential to engage the student interactively. For example, from the student's perspective, the mini-lectures on numeric data conversion are essentially passive. However, applets have been developed to enable students to appraise their ability to carry out radix conversions.

Figure 3 illustrates the user interface of a Java applet that allows students to practice converting integer representations. The student may input a value in a field corresponding to a particular integer format (unsigned, signed-magnitude, or two's-complement). By clicking on the corresponding button, the applet calculates and displays all of the appropriate values for the other integer formats. Thus, students can test their skills and, since use of the conversion applet is not monitored, they can receive non-threatening feedback immediately.

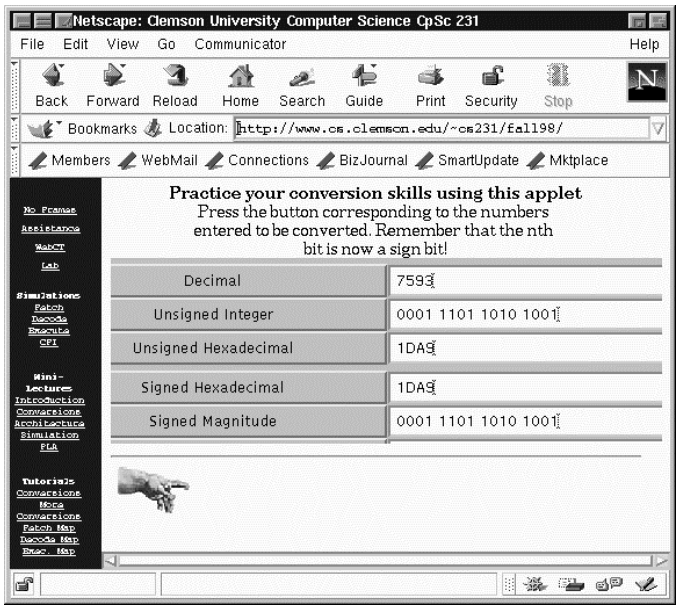

Figure 3. The user interface of the conversion practice applet

Animations

A second advantage of using the web as the delivery vehicle for course material is the ability to animate activity. Animation has been used to demonstrate concepts that depend upon the flow of information within a CPU. Even though the information flow can be described statically using register transfer notation, students grasp sequences of data movement better when they visualize it flowing on the data path. For example, Figures 4 and 5 illustrate two steps in the flow of information during the FETCH phase of an instruction cycle. Figure 4 shows the flow of information from the program counter (PC), which contains the address of the instruction to be fetched, as the PC's contents are copied into the address register (AR) so that it can be presented to the memory.

Figure 5 illustrates the flow of information back from the memory to the instruction register (IR). In parallel to the data transfer from memory to the instruction register, the content of the program counter is updated so that it represents the address of the next instruction. This animated presentation of the flow of information along the data path enables the student to step through the images that portray the register content transfers sequentially. In addition, the affected registers and data paths are highlighted using contrasting colors in the web display. In Figures 4 and 5, the highlights are shown using shading for the registers that are used and arrows to indicate information flow along the data path.

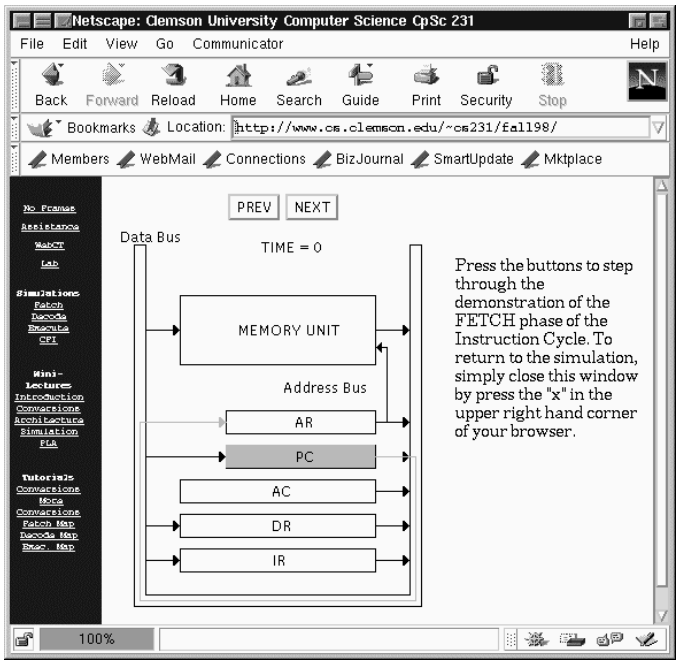

Figure 4. Step one of the FETCH phase of the instruction cycle

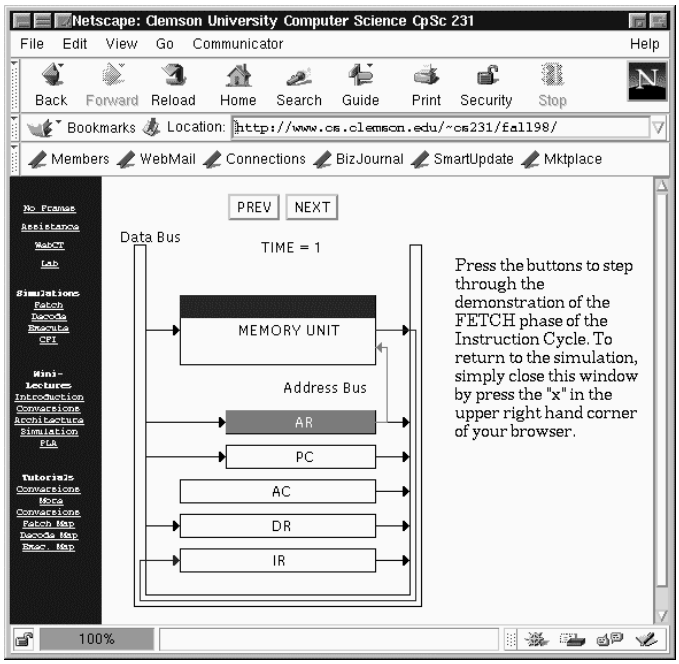

Figure 5. Step two of the FETCH phase of the instruction cycle

CPU Simulations

A centerpiece of the work that the students complete during the course is programming project in which they develop a simulation of a CPU. The project is divided into four intermediate stages, FETCH, DECODE, EXECUTE, and CHECK_FOR_INTERRUPTS, corresponding to the four phases of a model instruction cycle. Each stage of the

assignment builds upon the previous stages so that at the completion of each stage, the simulations exhibit more of the behavior of the CPU.

To support the students' development efforts, a collection of four Java applets was created to simulate the correct register transfers for each stage of the assignment. Students use the applets as benchmarks to determine whether their simulations are correct. In addition to modeling the register transfers, the applets modify and display the CPU status flags so that the entire state of the machine is mimicked.

Figure 6 illustrates the machine state display for one instruction used in the EXECUTE phase assignment. Notice that the instruction has been parsed and that the status flags are displayed. Also notice that the content of memory can be displayed so that the effects of memory modifying instructions can observed.

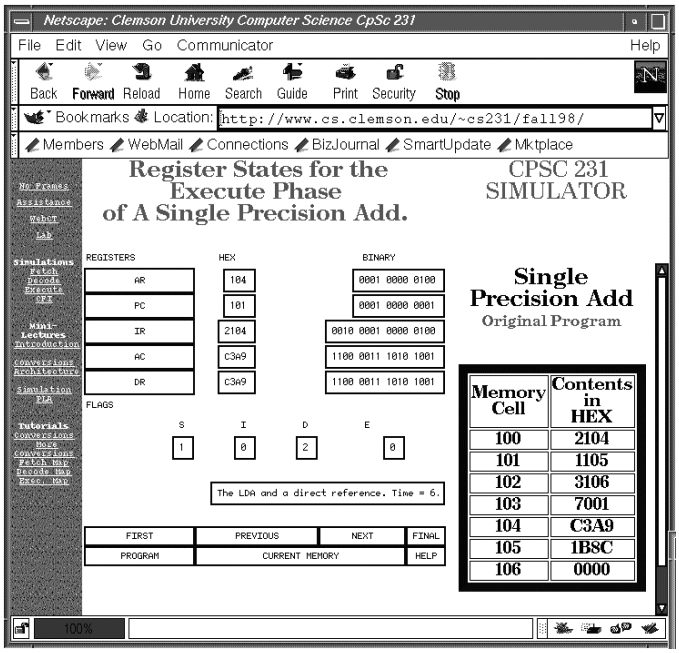

Figure 6. The state of the computer after executing a data access instruction

The EXECUTE and CHECK_FOR_INTERRUPTS applets also keep track of the cumulative number of CPU clock cycles since the start the program. As a by-product of incorporating time measurements in the applets, students have the capacity to model scripted asynchronous requests for service from peripherals.

A fifth simulation applet was developed to mimic the register states after each individual clock cycle. Having a simulation that proceeds cycle by cycle enables students to monitor every state change and to understand how each stage of interrupt processing affects the state of the CPU. This has proved particularly useful for helping students grasp the actions that occur when an interrupt is handled.

The cycle-by-cycle simulation has also been useful when students are attempting to locate errors in their simulations because the effects of numerous register transfers may be summarized in the state of the machine at the end of a phase of the instruction cycle. The cycle-by-cycle applet enables the student to see the effects of the register transfers at the finest temporal granularity.

Students may use a suite of programs that are stored with the applets as benchmarks. Alternatively, they may test their simulations by using a web interface applet that enables them to input arbitrary machine language programs.

Observations, Conclusions, and Future Work

During the first semester utilizing the mini-lectures and interactive applications, student response has been very favorable. The popularity of the support materials has been particularly high among international students who may have difficulty with spoken English.

In addition, students have repeatedly reported using the number conversion mini-lecture and the conversion applets to practice prior to testing. While radix conversion mini-lecture currently addresses only integer conversions, a mini-lecture is planned to present and illustrate decimal conversion using the IEEE 754 floating-point format.

Using PLAs to describe the details of control unit implementation and behavior introduces a significant level of complexity. However, the PLA diagrams provide a graphical method of presenting the control unit, which is otherwise characterized by an abstract mathematical notation. By using complementary presentations, the needs of students having either global or analytical learning styles are addressed.

Finally, the course supplement development plan includes implementing applets to simulate both an assembler and a disassembler for the machine language of the CPU. These tools will facilitate student activities, such as assembly language programming or program interpretation, that employ higher levels of abstraction than the machine language provides directly.

Acknowledgements

The authors gratefully acknowledge the development efforts of Jimmy Felder, Eric Senn, and Ariel Wolfer. In addition, partial funding for this research was provided by the National Science Foundation through the Southeastern University and College Engineering Education Coalition under cooperative agreement EEC-9727411.

References

1. Caviedes, Jorge, "A technological perspective of anytime, anywhere education," ALN Magazine, Vol. 2, No. 1, http://www.aln.org/alnweb/magazine/vol2_issue1/jorge.htm, March 1998.

2. Irani, Tracy, "Communication potential, information richness, and attitude: a study of computer mediated communication in the ALN classroom," ALN Magazine, Vol. 2, No. 1, http://www.aln.org/alnweb/magazine/vol2_issue1/aln_exper.htm, March 1998.

3. Jaturaputpaibul, Pitak, "Presentation of on-line courses," ALN Magazine, Vol. 1, No. 2, http://www.aln.org/alnweb/magazine/issue2/pitak/pitak.htm, August 1997.

4. Mano, M. Morris (1993) Computer System Architecture, 3rd edition, Prentice Hall, Englewood Cliffs, NJ.

5. Reinhart, Cornel J., "Reflections on learning and teaching at a distance: the America in the sixties course," ALN Magazine, Vol. 2, No. 1, http://www.aln.org/alnweb/magazine/vol2_issue1/aln_exper.htm, March 1998.

6. Thaiupathump, Choonhapong, Martine, Dawant, and Bourne, John, "Building an automated FAQ system (AUTOFAQ): building frequently asked questions (FAQs) pages for ALNs," ALN Magazine, Vol. 2, No. 1, http://www.aln.org/alnweb/magazine/vol2_issue1/choon.htm, March 1998.

7. http://www.aln.org

8. http://www.aln.org/alnweb/magazine/issue2/pitak/real.htm

9. http://www.cs.clemson.edu/~cs231/fall98

10. http://www.real.com

Gene A. Tagliarini

Dr. Tagliarini received BA and MA degrees in Mathematics from the University of South Florida in 1970 and 1971, respectively. In 1989, he received the PhD in Computer Science from Clemson. He has taught courses at both the graduate and undergraduate level for over 25 years. Dr. Tagliarini has served as the Principal Investigator for both Department of Defense and industry. He has been active in research in the area of biologically inspired computing for the past 10 years. His work led to a systematically designed neural network to solve one of the largest published real applications. He has developed neural networks for classifying sonar returns, matching fingerprints, compressing image files, classifying the spectra of minerals, optimizing solutions to resource allocation problems, and finding solutions to constraint satisfaction problems. He has published over 30 research papers in the areas of biologically inspired computing, computational intelligence, evolutionary computation, and multimedia.

Edward W. Page received the BS degree from Clemson University, the MS degree from the University of Alabama in Huntsville, and the PhD degree from Duke University, all in Electrical Engineering. Dr. Page is an active researcher in the field of computational intelligence and has served as Principal Investigator or Co-Principal Investigator on research projects totaling more than $4 million in telecommunications, artificial intelligence and neural networks. He has served as a consultant to a number of government and industry organizations, including the Congressional Office of Technology Assessment, and has published more than 60 papers.

Edward W. Page