Using a PDP-11/10 to Teach Content and History in Computer Organization Courses

Douglas Harms **Dave Berque**
DePauw University
Computer Science Department
Greencastle, IN 46135
{dharms, dberque}@depauw.edu

Abstract
This paper describes our use of a 1970's era PDP-11/10 to support an hour long module which we have incorporated into our computer organization course for the past few semesters. The module is designed to reinforce standard concepts such as number systems and two-pass assemblers while at the same time exposing the students to some historical issues. After providing some background information about the PDP-11/10 we explain the motivation for our approach. Then we describe a fifteen minute instructional video which we have produced on the topic of developing programs on the PDP/11-10. The video is available from the authors either on VHS tape or as a QuickTime file, thereby allowing instructors at other schools to try our approach by showing the video to their students.

1. Introduction

The ACM/IEEE-CS Joint Curriculum Task Force's landmark report, *Computing Curricula 1991* [1], contains recommendations for including historical content in several of the knowledge units which comprise its model curriculum. For example, the knowledge units related to operating systems suggest that students should be exposed to "...operating system development, including historical information about the development of architectural support for changes in software, and the economic and technical forces that drive operating system development" [1]. In spite of the legitimacy that this report lends to including historical issues in the curriculum, discussion of such issues has been relatively limited in the education literature. For example, the proceedings *ACM SIGCSE Technical Symposium on Computer Science Education* for 1999 and 2000 collectively lists only 2 of the 27 "Birds of a Feather Sessions" as dealing with an historical issue. More dramatically, of 54 workshops, 10 seminars, 30 panels, and 148 papers described in the proceedings, none deal primarily with historical topics [2, 3].

Although publications dealing with the relationship between history and computer science education have not been frequent, several authors have written on this topic. For example, Katz [7] has outlined concerns about the accuracy of historical topics covered in standard text books, while Gal-Ezer and Harel [6] have pointed out the importance of Computer Science educators being well-grounded in the discipline's history, and Lee [9] has illustrated ways in which old technologies such as Napier's Chessboard Calculator can be used as teaching tools.

In the remainder of this paper we follow up on Lee's theme by showing how we have used an old (circa 1972) PDP-11/10 to help teach several instantiations of the standard Computer Organization course. After a brief review of PDP-11/10 history, architecture, and supporting peripherals, we describe how we integrated a short PDP-11/10 demonstration into our Computer Organization course in order to reinforce standard concepts such as number systems, loaders, and two-pass assemblers while simultaneously providing historical perspective. Finally, we describe a fifteen minute video tape about the PDP-11/10 which we have produced with assistance from our media center. This video makes it possible for instructors at other schools to try our approach without needing access to a PDP-11/10.

2. PDP-11/10 System Overview

The PDP-11/10 was introduced in June 1972 and is one of the earliest members of the very successful PDP-11 family of minicomputers. The PDP-11/10 CPU implements the basic instruction set of the architecture, which consists of 59 instructions and twelve addressing modes (register direct, register deferred, auto-increment, auto-decrement, auto-increment deferred, auto-decrement deferred, index, index deferred, immediate, absolute, relative, and relative deferred). Operands are either byte (8-bit) or word (16-bit) values, and the addressing modes permit operands to be in one of eight 16-bit general purpose registers, in memory (possibly accessed via a register), within the instruction itself, or on a

SIGCSE 2001 2/01 Charlotte, NC USA

stack. Subroutine call/return instructions permit easy implementation of recursive subroutines.

The PDP-11/10 implements a 16-bit address bus which limits the address space to 64KBytes. Memory is implemented using magnetic core with a 950 nsec cycle time and is byte-addressable with consecutive even/odd addressed bytes addressable as one 16-bit word. The first 512 bytes of memory are reserved for trap and interrupt vectors and the upper 8KBytes of the 64KByte address space is used for memory-mapped I/O device registers. No Memory Management Unit is offered for the PDP-11/10.

All system components (i.e., CPU and I/O devices) connect to and communicate with each other via a single bus called the UNIBUS. A wide variety of UNIBUS peripheral devices are available, including terminals, line printers, card readers/punches, paper-tape readers/punches, magnetic tape units, disk drives, and clocks. (Descriptions of many peripherals available for UNIBUS systems can be found in [4].)

A front panel consisting of several lights and toggle switches is included with the system. (See Figure 1.) This allows an operator to load and examine registers and memory, reset the system, and control execution of a program (start, stop, and single-step).

chips), core memory (each 16KByte frame implemented on two 8.5" x 15.75" printed circuit boards and one 8.5" x 10.5" printed circuit board holding the actual core memory), and up to twelve peripheral cards. An entire computer system consists of one or more 22" x 30" x 72" mounting cabinets holding the system unit and peripherals, weighs several hundred pounds, and dissipates several thousand watts of power.

Despite its physical size the PDP-11/10 does not implement several features common on modern computers. For example, the CPU does not implement instructions for integer multiplication, integer division, or floating point operations. (An optional Extended Arithmetic Element (EAE) card implements integer multiplication and division; no floating point processor was offered for this CPU.) There is no processor state (e.g., supervisor vs. user) and thus no method to prevent a program from executing any legal instruction (e.g., HALT) or accessing any memory location (including "system" routines and I/O device registers). A detailed description of the PDP-11/10 can be found in [5].

3. Using a One Day PDP-11/10 Module to Reinforce Course Concepts: Live Demo Approach

For the past several years DePauw University has used various VAX systems to support its Computer Organization course. In addition to describing standard topics such as

Figure 1
PDP-11/10 Front Panel

DEC provided some software with the PDP-11/10. Specifically, the machine's paper tape software system included an editor, an assembler, several loaders, an online debugger, an I/O executive, and a math package. On disk systems DEC provided a simple disk operating system (DOS), a job-stream processing system (BATCH), and a time-sharing system (RSTS-11). BASIC and FORTRAN were also available.

The PDP-11/10 computer is physically large by today's standards, as shown in Figure 2. For example, the 19" x 10.5" x 23" system unit weighs 110 pounds and houses the power supply, CPU (implemented on two 8.5" x 15.75" printed circuit boards consisting of approximately 240 SSI

number systems, digital logic, machine language, and assembly language, the course outline has generally included a statement of several broader goals including the following: "...students will gain an appreciation for the historical development of technology, and the people who make technological advancements". Several instructors have approached this goal, in part, through a variety of supplemental activities. For example, the students often read *The Soul of a New Machine* [8] which describes the experiences of a team that is designing a new computer. Students have generally found the reading to be both enjoyable and relevant, and they have participated enthusiastically in the follow-up discussion which is skillfully led by a colleague in the English department. We have also

invited a faculty member from the physics department to visit the class to briefly explain and demonstrate vacuum tube technology.

Figure 2

Starting with the Fall 1998 Semester, the authors have added an additional hour-long "PDP-11/10 Module" to the course. Each student attended a one hour demonstration of the PDP-11/10 in action. The demonstration included a discussion of the historical context of the PDP-11/10, an overview of the PDP-11/10 hardware that included taking the computer apart to show the physical sizes of components such as the CPU and core memory, operation of the front panel to load and execute PDP-11 machine language programs, and the procedure for using the paper tape tools (loaders, ED11 editor, and PAL11 assembler) to develop and execute assembly language programs.

The PDP-11/10 and the associated development environment are obviously archaic and cumbersome compared to contemporary systems. However, we believe that our careful demonstration of selected features of the PDP-11/10 has been of great benefit to our Computer Organization students. The primary benefits to students are summarized below.

Concrete reminders that machine language is fundamentally different from high level languages.

Programming the PDP-11/10 is a very hands-on and physical experience for students: the students help toggle machine language programs into memory using only the switches and lights on the front panel, and data files exist as very physical collections of holes on paper tape. This user interface is very different than the windowing systems they are used to using in other courses and provides a very real sense that they're "not in Kansas anymore" when programming in machine and assembly languages.

Concrete experience with abstract concepts such as bit, byte, file.

The PDP-11/10 presents students with physical representations of binary data. Switches on the front panel are physical representations of binary 0 (switch down) and 1 (switch up); similarly, lights on the front panel represent 0 (light off) and 1 (light on). Holes on the paper tape represent binary ones, and the absence of holes represent zeroes. Each row of paper tape stores one byte (see Figure 3), and an entire paper tape stores a sequence of bytes (i.e., a file); paper tape "files" are normally labeled with a handwritten name (i.e., the filename), and a collection of tapes is a directory. One "deletes" a paper tape file by physically throwing it in the trash can (which, interestingly enough, is recoverable until the custodian empties the trash!).

Students often understand an abstract concept (e.g., binary data) better when they have experience with a concrete realization (e.g., holes in a paper tape). For example, students have been known to get emotionally excited when they realize the holes on a paper tape are the ASCII values of the text they typed in an editor and "saved."

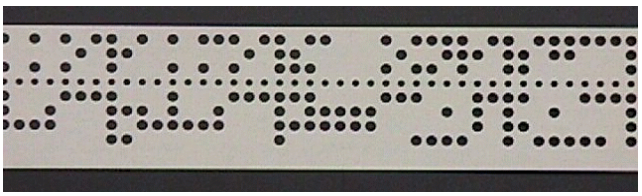

Figure 3

Concrete experience with binary, octal, and hexadecimal numerals.

Seeing the switches on the front panel of the PDP-11/10 being used to "toggle" a program into memory gives students direct experience with the relationship between octal and binary as well as the advantages of octal over binary. Because octal is used to express 16-bit values in the PDP-11 (for reasons beyond the scope of this paper), the switches and lights on the front panel are organized in groups of three with each group labeled alternately with red and blue coloring (see Figure 1.) Each colored group represents one octal digit, and as students learn about toggling they very quickly understand that octal "5" is represented by "up-down-up" in the switches and "on-off-on" in the lights. This physical representation helps students learn the octal-binary

conversion table. Also, students very quickly appreciate the fact that it is much easier to think about and toggle "106527" than it is to think about and toggle "1000110101010111".

Experience using loaders.
The PDP-11/10 does not have any ROM and students understand they have to explicitly load every program they need. Using the front panel to toggle a program into memory is the only method of loading a program provided by the PDP-11/10 hardware; after seeing several programs toggled in, students appreciate that this method of loading is only practical for very small programs.

Our PDP-11/10 does not have any disk drives, but instead relies on paper tape as the primary means of external storage. A paper tape is a collection of bytes, and students realize the possibility that a paper tape could represent the memory image of a machine language program. The problem, of course, is getting the data from paper tape into memory. Because the paper tape reader is an I/O device it is possible to write a program that reads the contents of a paper tape and stores it into memory; this program is called a loader. How is the paper tape loader (called the bootstrap loader) loaded into memory? By toggling! Thus, students see the bootstrap loader (which is a 14 word program) toggled into memory, used to load the absolute loader from paper tape (the absolute loader is larger and more robust than the bootstrap loader). Then the students see how the absolute loader is used to load all subsequent paper tapes during the program development life-cyle. These tapes include the editor, the assembler, the assembly language source program, and the machine language image.

Following this experience with paper tape loaders students easily extend the concept of a loader to other storage devices such as disks. They also appreciate that the initial toggling of the loader can be avoided if the bootstrap loader is burned in ROM and automatically executed when the computer is turned on.

Although we have not done any formal assessment of the effectiveness of this approach, anecdotal evidence indicates the students found the demonstration to be interesting and useful. They were especially intrigued by the paper tape system, the front panel, and the physical size of the computer, particularly compared to the limited computational power of the computer.

4. Using a PDP-11/10 Reinforce Course Concepts: The Video Taped Approach

Since the PDP-11/10 is almost thirty years old it is difficult to find replacement parts for it, and it is not clear how much longer we will be able to keep it in working order. As a way of preserving our ability to demonstrate the machine to our students, we recently made an instructional video with the help of two students who have taken the Computer Organization course and who also have had courses in television production. The video runs for approximately fifteen minutes and focuses on the topic of developing programs on the PDP/11-10 and other similar computers. To help the students relate the video to topics from the Computer Organization Course, the video is accompanied by a written study guide. The video is available from the authors either on VHS tape or as a QuickTime file.

We tried to make the video interesting to our audience in part by constructing a simple story line which unfolds during the video. We begin with a discussion between a modern-day computer science professor and his student during which the student complains about how time consuming it is to write and debug C++ programs. The professor tells the student that things used to be much more tedious in "the old days" and he offers to use a use a Holodeck Program to take the student on a trip to see an older computer. The pair enters the Holodeck and with the help of some special effects they find themselves in a 1970 era machine room complete with old line-printer art. In the machine room they encounter a programmer who is humorously dressed in 1970's clothing and who is busily working to write a program on his brand-new PDP-11/10. During the ensuing dialog the professor-student pair asks the 1970's programmer a number of questions about how this machine works. As the questions are answered, the visitors learn how to toggle in the bootstrap loader using the switches on the front panel. Now that the machine is ready to load and execute other programs, the programmer demonstrates how to edit, assemble, save (on paper tape), and execute simple programs. Along the way points are made about a number of issues including the hardware configuration, the use of the octal number system, how bits, bytes, and files are represented on paper tape, the two-pass assembly process, etc. Whenever possible these points are related to concepts that are typically covered in the computer organization course. For example, the notions of bits, bytes, files, filenames, and directories on a paper-tape system are contrasted to their modern day counter parts.

After producing the first version of the video tape we field-tested it with a group of twelve undergraduate computer science students. We asked the students to respond to written questions about the content of the video, and also invited them to participate in a focus group which gave them the opportunity to make suggestions about how the video could be improved. Their answers to our questions confirmed that they had learned a significant amount from the video. For example, the students were able to correctly respond to questions about the process of running programs on the PDP-11/10, and about the role of paper tape in storing files. On the other hand, the students had some constructive criticisms of the video. Most notably, many of the students pointed out that one section of the video moved too quickly, and they urged us to add a summary of the major points covered in

that section. Other students suggested that graphical overlays could be used to more clearly point out the different parts of the PDP-11/10. Subsequently we have revised the video in response to these suggestions. For example, a graphical overlay is used to show how the switches on the front panel are grouped together, and how groups of three switches can be used to represent a single octal value.

During the Fall semester of 2000 we showed the PDP-11/10 video to the second author's Computer Organization class. The class had an enrollment of twenty-four students, and twenty-three of the students were present on the day the video was shown. Each of these students completed a short survey which gave the authors the chance to gather some very preliminary data about the effectiveness of the video (at least from the student viewpoint). The survey asked the students to react to several statements using the following rating scale:

1 = strongly agree
2 = agree somewhat
3 = neutral
4 = disagree somewhat
5 = strongly disagree.

We have not done any statistical analysis of the results, but the scores were generally positive. For example, when asked to react to the statement "*The video was interesting*" the mean response was 1.3. When asked to react to the statement "*Watching the video was useful because it helped me to understand the history of computers better*" the mean response was 1.7. Similarly, when asked to react to the statement "*Watching the video was useful because it helped me to understand one or more concepts we have discussed in class this semester*" the mean response was 1.6. Finally (and thankfully) when asked to react to the statement "*Watching the video was a waste of time*" the average response was 4.8.

5. Future Work

We are interested in learning whether faculty at other institutions will find the video to be useful in their classes. Toward this end, we invite interested faculty to contact the authors via email to make arrangements to obtain copies of the video and study guide. We welcome comments about the effectiveness of the video as well as suggestions for improvement.

We are also considering making follow up videos that stress other aspects of the PDP-11/10, for example its hardware architecture including core memory and bus. If the original video proves to be successful we may undertake this project in the coming years.

6. Acknowledgments

The authors wish to acknowledge DePauw University undergraduate students Matt Farrell and Brian Goad for producing and editing the video described in this paper as part of a course that they took under the direction of Brad Boeke, director of Television Operations at DePauw University.

References

[1] ACM/IEEE-CS Joint Curriculum Task Force. *Computing Curricula 1991 Report of the ACM/IEEE-CS Joint Curriculum Task Force.* ACM Press and IEEE Computer Society Press, 1991.

[2] *Proceedings of the 1999 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM Press, New Orleans, March 1999.

[3] Proceedings of the 2000 ACM SIGCSE Technical Symposium on Computer Science Education. ACM Press, Austin, Texas, March 2000.

[4] Digital Equipment Corporation. *PDP-11 Peripherals Handbook*. 1973.

[5] Digital Equipment Corporation. *PDP-11/05/10/35/40 Processor Handbook*. 1973.

[6] Gal-Ezer, J., and Harel, D. What (Else) Should CS Educators Know? *Communications of the ACM*, September 1998, Vol. 41, No. 9, pages 77-84.

[7] Katz, K. The Present State of Historical Context in Computer Science Texts: A Concern, *SIGCSE Bulletin*, December 1995, Vol. 27, No. 4, pages 43-50.

[8] Kidder, T. *The Soul of a New Machine*, Avon Books, New York, 1981.

[9] Lee, J. Those Who Forget the Lessons of History are Doomed to Repeat It, Or Why I Study the History of Computing, *IEEE Annals of the History of Computing*, Vol. 18, No. 2, IEEE Press.