DOCUMENT RESUME

ED 129 568 SE 020 360

AUTHOR Roehrkasse, Robert C.
TITLE The Impact of Minicomputers and Microcomputers on the Software-Oriented Curriculum.
PUB DATE Jun 75
NOTE 12p.; Paper presented at the annual conference of the American Society for Engineering Education (Colorado State University, Ft. Collins, June 16-19, 1975)

EDRS PRICE MF-$0.83 HC-$1.67 Plus Postage.
DESCRIPTORS *Computer Oriented Programs; *Computers; Curriculum; *Curriculum Design; Engineering; *Engineering Education; *Higher Education
IDENTIFIERS *Minicomputers

ABSTRACT

Discussed are the requirements of a software-oriented engineering curriculum that also includes use of computer hardware. Three areas are identified as necessary in such a curriculum: functional area users, systems programming, and mini-micro technology. Each of these areas is discussed in terms of instructional methods and suggested topics. Course descriptions are given for such a curriculum at the United States Air Force Academy. (MH)

Event Number 3215 (Microcomputers/Microprocessing in Engineering Education).

American Society for Engineering Education Annual Conference
June 16-18, 1975
Colorado State University
Fort Collins, Colorado 80521

The Impact of Minicomputers and Microcomputers on the Software-Oriented Curriculum

ROBERT C. ROEHRKASSE
ASSOCIATE PROFESSOR OF COMPUTER SCIENCE
UNITED STATES AIR FORCE ACADEMY, COLORADO 80840

The views expressed herein are those of the author and do not necessarily reflect the views of the United States Air Force or Department of Defense

IMPACT OF MINICOMPUTERS AND MICROCOMPUTERS ON THE SOFTWARE-ORIENTED CURRICULUM

Our initial experience with the mini/microcomputer scene began with the 1974 Fall Semester. The first course was structured around a microcomputer and a minicomputer. The microcomputer system was an integer machine which Captain Larson of our Electrical Engineering Department had built around an INTEL 8008 chip (8 bit word) with a Teletype 33 as a peripheral device. We had a cross-assembler on our Burroughs 6700. Our programs were loaded from paper tape, and our input was from the teletype keyboard. The minicomputer was an airborne computer of Westinghouse vintage (18 bit word). It was a fractional machine. We used this machine by running in a simulator mode. Again, our Burroughs 6700 was the host machine. As a prerequisite, we required only the course in computer programming using ALGOL which all Air Force Academy cadets are required to take.

What did we learn from our initial experience? We found that our students had difficulties. First, using the microcomputer required that input from the teletype be converted to an internal representation suitable

for calculations and another character conversion to provide a suitable representation to be printed on the teletype. Although our cadets are exposed to internal machine representation and number systems, there is no need to really use those notions in any higher-level language program exercise. Additionally, we had a mandatory exercise that required that they handle double precision operations and maintain one-place decimal accuracy on the integer machine. The fractical machine usage exposed the students to scaling and to the need to consider scale factors throughout a sequence of computations. Again, our students had some difficulty. As this was an airborne computer, the input and output data was routed to sensors. Hence, the need for an understanding of analog-digital conversion and vice versa. Here, even our instructors needed assistance from the Academy's Electrical Engineering Department.

It therefore becomes apparent that a software orientation does not provide the necessary skills to function in the mini-micro world. The content of a computer science curriculum is usually determined by which ever academic discipline first introduced the computer at the institution. If electrical engineering, the hardware aspects will be included. If more means-end oriented as from industrial engineering, business

administration, mathematics, or astronautics, the software aspect will tend to dominate. Ours has had the latter evolution.

What are we doing? We recognize that there is greater need to understand the hardware aspects and the internal manipulations a computer goes through. To provide this knowledge, we will introduce a computer architecture course this fall semester which we hope will provide the necessary understanding. The course will examine the basic functional computer structures including memory devices, word structure and addressing modes, arithmetic units, and input-output devices. But, further consideration of the mini-micro revolution indicates that even greater changes ought to be made in the software-oriented curriculum. The curriculum should serve three needs: functional area users, systems programming, and mini-micro technology.

Let's consider the first need, functional area users. I think here we are seeing the same packing-down phenomenon with computer usage that we have with mathematics. That is, topics like calculus are taught in high school, perhaps even earlier. More and more people from the functional areas or academic disciplines are using the computer. They are programming the computer themselves. Where

these users in the past posed a problem and then requested that a program be written, they now do, and want to do, the programming themselves. Yet, to be effective they need more than just a working knowledge of some computer language. These people must now concern themselves with data structures and representation, auxiliary storage media, input/output considerations, viz., program efficiency. These considerations were once the province of those who called themselves programmers. We have a reasonable body of knowledge accumulated about such techniques. The challenge is how to convey this material to these new computer users. The traditional curriculum allocates from one semester to three semesters to cover this material -- much too long from a user's vantage point. The challenge remains: How can we compress the teaching--learning process? What will remain of the traditional programmer's province? The trend will probably be toward the open shop and remote access methods.

We now turn our attention to systems programming, the activity which is concerned with enhancing a computer system by extending the hardware features through compilers, assemblers, operating systems, and IO subroutines, etc. Heretofore, the hardware was designed and constructed.

Then software was developed to compensate for any design faults and to facilitate access by the end user. The microprocessor may provide an economic technology for designing and constructing future computers in terms of a hardware-software trade off. Also,increasing software costs and decreasing hardware cost should provide an economic incentive. No longer would we seek systems software to compensate for design faults, but rather seek to optimize some measure of "computer utility." However, we will continue to need to educate a cadre to do whatever it is that systems programming becomes as new computer architectures are developed.

Finally, let's consider several developments spawned by the mini-micro technology. These developments involve applications. Simply said, computer power is rapidly becoming economically available for everyone -- your own personal computer. Well, perhaps not really, but computing power is certainly becoming cheaper and cheaper. The challenge is to find the most economical and effective method to access the computing power so you have just enough for your application. Many applications have become economical and feasible by being structured around a microprocessor. The resulting microcomputer is just a design component in these applications. As such, I see

these areas as the province of the electrical engineer. On the other hand, there are those applications where the computer has a more visible role. Three access methods that these applications employ will be considered: stand-alone, network,and data communications.

First,you could get your computing power as a stand-alone system. System components are then your concern. We now have independent peripheral devices, add-on memory, etc. You can tailor your own system with a different vendor supplying each component. Obviously, there is a definite need to known more than the software. There are many hardware considerations.

Another method is through a network. There are two areas of investigation. The first is an attempt to see how much computer power can be gotten through some notion of parallel processing by interconnecting minicomputers in varying numbers and ways. Those large machines (STAR, ILLIAC IV) which exploit parallel processing are quite expensive and have only been considered effective on specific problems which tend to be scientific in nature. Those architectural designs structured from minis are described as distributed, ring, daisy, hierarchial, etc. Localized processing is the other area. Various sites are linked via the network. Each site then has a degree

of processing capability suited to its tasks. Again, the hardware considerations surface.

Thirdly, data communications provide another method of access. Let's take data communications to be that subsystem of a computer system whose purpose is the transmission of digital code between a sending and receiving digital device in some way other than by direct electrical connection. You could be concerned with data communications as part of a network or as the connection between a remote device and a computer site. Here's a world of buzz words like modem, teleprocessing telecommunications, multiplexor, concentrator, etc. The point to ponder is the importance of the hardware. It would be very difficult, if at all possible, to compensate for a poorly designed or integrated system through software. The vendor's task is to persuade you to buy his product. Yours is to insure that those components selected are functional when integrated.

Hopefully we have observed that hardware is a critical element in the applications which exploit mini-micro technology. The software-oriented curriculum does not address these hardware considerations. Granting that the more traditional large-scale systems will remain, the mini-micro technology is the scene of the revolution. The ideal individual to flourish in this environment is

an electrical engineer turned systems programmer or a systems programmer turned electrical engineer. The former would appear as the easier transition.

As for our curriculum, we will accomodate these mini-micro needs by introducing courses in data communications, mini-micro usage, and graphics. However, the ultimate curriculum objective is maintaining a flexible structure that can address changes in computer technology.

Comp Sci 485, Computer Architecture 1(1) Fall 75-76

Logical design of computers and machine organization. This course examines the functional basis of various computer structures including memory devices, word structure and addressing, arithmetic units, and input-output equipment. Recent advances in computer organization. Includes several computer projects to illustrate basic concepts. Final report. Prereq: Beginning ALGOL Programming. Sem hrs: 2 1/2 fall.

Comp Sci 495, Telecommunications for the Computer Scientist 1(1) Fall 75-76

Topics include timesharing networks, data communications facilities, terminal devices, and analytical tools needed to design and evaluate computer-communications systems. Modems, multiplexors, concentrators, front-end processors and EIA interface hardware functions will be studied in detail. Host computer software support such as message control programs, frond-end software, performance and diagnostic routines, and line protocals used to control the external environment will be investigated. Emphasis will be on problem recognition and trouble-shooting skills needed for correcting errors that often occur among the many complex components. Each student will be required to perform a comparative analysis of terminal equipment for a hypothetical teleprocessing network. As a term project, each student will write a technical paper about a major teleprocessing network operated or used by an Air Force organization (e.g., ALS, WWM/CCS, AUTODIN, ARPA). Concepts and techniques presented are designed to broaden the scope of the computer science field for the future Air Force officer and to provide the background necessary to effectively function with a variety of specialists involved in one of the many USAF computer-communications networks. Prereq: Intermediate ALGOL Programming and department permission.

COMP SCI 495, Small Computers and Computer Graphics Spring 75

Basic concepts of mini and micro computers including internal arithmetic, internal representation of data and instructions, architecture concepts, assembly language

programming, I/D programming, and interrupt programming. The course also examines basic concepts of interactive computer graphics including display processing hardware, graphics peripherals, high level graphics programs and display file compilers, and display processor programming. Includes several computer projects to illustrate basic concepts. Final report. Prereq: Intermediate ALGOL Programming.