WIS-CS-74-209
The University of Wisconsin
Computer Sciences Department
1210 West Dayton Street
Madison, Wisconsin 53706

Received February 4, 1974

ON COMPUTING FACILITIES FOR COMPUTER SCIENCE

by

E. J. Desautels

Computer Sciences Technical Report #209

February 1974

ON COMPUTING FACILITIES FOR COMPUTER SCIENCE

by

E. J. Desautels

ABSTRACT

Computer science is viewed as field of study in which the experimental and pragmatic aspects, while initially slighted in some curricula, are becoming recognized as playing an increasingly important role. The need for appropriate computing facilities is examined, and resources not likely to be routinely available from a production computing facility are identified.

Equipment selection guidelines are discussed, and experience based on the operation of the University of Wisconsin Computer Sciences Department Computer Systems Laboratory and others is described. The author suggests that these and related experiences at other institutions should lead to the realization that it is not only desirable but necessary that a computer science program have its own laboratory, and whereas equipment costs made this impractical a few years ago, it is now economically feasible.

COMPUTER SCIENCE AND COMPUTER SCIENCE EDUCATION

In the 1968 recommendations for academic programs in computer science (2), the need for appropriate supporting laboratory facilities is stated, and the possibility that the acquisition of completely separate equipment dedicated to education and research in computer science is raised (p. 167 in (2)). However the recommendations did not go into any details on what these supporting facilities should be or how they should be used. Since then, little has been published discussing specifically what facilities are needed for an academic program in computer science, and what unique problems are involved in establishing and operating such facilities. In this report we shall focus on the special needs not likely to be satisfied by a shared central campus computing facility. This leads us to the need for the establishment of a separate laboratory for computer science instruction.

A good source of information on these needs is the "Minicomputers in the digital laboratory program" report of the National Academy of Engineering's Commission on Education (report VII of the COSINE series 28). It addresses the special computing needs in electrical engineering education, and describes facilities to meet these needs. Our report attempts to identify similar needs specific to computer science education, and describes special facilities to match these needs.

The laboratory experience reported upon is based directly on the use of Datacraft Corporation and Digital Equipment Corporation machines, with major peripherals purchased from over a dozen vendors. However many of the observations are based on experiences with several dozen of the one hundred odd computer installations on the Madison campus (22) representing many main-frame computer manufacturers.

Computer Science as a Discipline

Computer science is in its infancy, yet there are some established traditions. The Curriculum 68 recommendations emphasize the theoretical foundations, partly due to the strongly mathematically oriented background of the authors, but primarily due to the need to emphasize the fundamental, technology-independent concepts, as opposed to techniques and training of ephemeral value. Such an emphasis on fundamentals has led to some academic programs in which the mathematics requirements for an undergraduate major in computer science exceed the requirements for a mathematics major (38).

The realization that computer science has an important pragmatic or experimental component is evident through the number of papers on various aspects of systems performance (e.g. 24,29) and in the recent establishment of the special interest group on performance measurement and evaluation (SIGMETRICS, (1)). However too few student, in particular undergraduates, are systematically exposed to the pragmatic, experimental aspects of computer science, beyond those of writing and debugging programs for a black box system that never fails.

There are many textbooks of the "how to program computer XYZ" variety, but these are not computer science texts. Many of these even stop short of discussing important fundamentals such as interrupts, input-output and channel operations in detail. Recently some excellent undergraduate texts have appeared, in which the student is expected to master a real computer (e.g. Stone (37)).

In Salton's editorial entitled "What is computer science?" [32], four problem areas are suggested by H. Zemanek as central to computer science:

(1) a theory of programming, with emphasis...on a practical theory of algorithms concerned with the construction of economical and efficient programs;

(2) a theory of process and processor organization, which takes into account the finite dimensions of existing memories,..., and the desire for a reduction in computation and program production time;

(3) a theory of description for processes and computational structures in terms acceptable to the processor; and

(4) a theory of computer applications which would include all features common to most numeric and nonnumeric applications.

If one accepts this thesis, then it follows that computer science education must be concerned with measurement and evaluation of systems and applications on real processors, upon which theories may be based.

In Hosch's discussion of computer science education [21], he argues that the study of the specification, execution, and analysis of algorithms comprises what may be considered "pure" computer science, while techniques for applying these concepts constitute "applied" computer science. He suggests that at the undergraduate level, computer science education should concentrate on training fully competent programmers. However he claims it is not essential to become an efficient systems programmer or even to become intimate with the operation of a particular operating system; he believes an appreciation of hardware and software design features used in a variety of types and scales of computing systems is sufficient.

These views may be a reaction to some curricula which over emphasize the use of a particular operating system and one high-level language, in which students may learn all of the idiosyncracies of a system, without understanding the concepts underlying the system's implementation and operation.

Recent Curriculum Recommendations

The report entitled "a computer science course program for small colleges" [4] proposes a curriculum which requires one full time instructor, and expects that the computing services will be shared by the whole campus. It is unfortunate that provision for even a minimal configuration system was not recommended. After all, a small college engineering or science department probably has an oscilloscope costing as much.

The report entitled "curriculum recommendation for undergraduate programs in information systems" [7] discusses computer education for management, and describes the computer resources it needs as not materially different from that required for other computer related courses.

The recommended curricula that have been published say very little about what computing facilities are necessary, and how they should be used.

Hamblen's 1971 report on using computers in higher education [19] does not address itself to the specific issue of the role of computing in computer science education, and therefore does not indicate what is needed nor how it should be used.

It is to be hoped that a critical attitude towards the ability of computer hardware, mathematical libraries and miscellaneous software to properly approximate real arithmetic computations, let alone integer arithmetic, should be instilled in any computer related curriculum. The "horror" stories recounted by Kahan's superlative survey [23] should be required reading in any curriculum.

Computer Engineering and Computer Science

Departments of electrical engineering have played a major role in the history of computing (e.g. ENIAC, ILLIAC, etc.). The role of computing and computers in the electrical engineering curriculum has been thoroughly studied, as is evidenced by the sequence of the seven COSINE reports (28). The present state of affairs is reviewed in [34] .

It has been traditional in electrical engineering to require laboratory courses in which students build, test, measure and report on their experience, much as students in physics and chemistry are expected to do. The theory they have studied is put into practice, and shortcomings of the theory (due to simplifying assumptions) and the practice (due to experimental error, interference phenomena, etc) are experienced. In recent years, electrical engineering students have been building small computers in their laboratory courses. Textbooks such as Chu's [7] as well as commercially available kits and components [15], accompanied by guides and textbooks such as Bell, Grason and Newell's [6] have simplified the task of organizing a computer-oriented laboratory course in electrical engineering. These laboratory courses may now also use low-cost minicomputers to provide a much richer software experience than previously available.

COSINE report VII [28] states (on p. 4) that "the computer related educational objectives that must be accomplished by an electrical engineering or computer engineering program" are:

(1) To provide training and programming experience using high level languages.

(2) To provide training and programming experience using machine code and assembly level code.

(3) To teach the student the fundamentals of machine organization.

(4) To provide the student with an experimental facility to test the ideas and concepts presented in courses on Operating Systems.

(5) To provide the students with an experimental facility to test the ideas and concepts learned in courses dealing with interfacing and data processing.

(6) To provide the student with an experimental facility to test out ideas and concepts learned in courses on digital process control, digital testing and equipment monitoring.

(7) To provide students with an opportunity to understand by hands on experimentation, the relationships and interactions between hardware and software, and the problems presented by real-time programming problems.

(8) To provide students with an opportunity to utilize the computer as a system element.

These objectives are certainly applicable to students majoring in computer science, with just a few changes. The computer science major may not have a course in designing hardware interfaces, but he should be familiar with designing software interfaces (e.g. introducing a handler for a new device). The courses on digital process control, digital testing and equipment monitoring might be replaced by a course on real-time system design, testing and performance evaluation. Similarly using a computer as a system element may be done in terms of its use as an intelligent terminal or a node of a network of computers.

Computing Facilities for Computer Science

Instruction and research in computer science evidently needs a service facility which can routinely run programs in a variety of high level languages.

Any such facility either has assembly language and macro capabilities, or these capabilities can be readily implemented or simulated.

Such a service can be provided either by a shared central facility (i.e. "the computing center") or by a departmental facility. We do not wish to argue in favour of one or the other.

However, we wish to address ourselves to the special needs of a computer science program which are not likely to be met through either a shared facility or a production oriented departmental facility. In the sequel, we shall use "service system" to refer to the general purpose facility on which conventional programming assignments are carried out, while "laboratory system" will refer to the system on which the special needs described below will be met (in some instances, the service system and the laboratory system will be identical).

Needs

The Computer Science student must be just as experienced with the use and operation of computers as an Electrical Engineering student is with an oscilloscope. It is all to easy to relegate the responsibility of teaching the operation of a computer to a technical school; it is a disservice to a Computer Science student not to provide him with sufficient exposure to computing equipment so that he may personally set up and run his own computing experiments. The educational value of a "hands-on" facility has been argued elsewhere [12,35] and our experience coupled with feedback from our students and faculty (in many disciplines) confirms this view. It may be helpful to reflect upon Dijkstra's observation that "...our power to visualize processes evolving in time are relatively poorly developed" [16], and we should provide useful concrete experiences for the student's benefit.

Basic Concepts

Operation of a computing system is usually initiated by a person (an "operator") and is usually subject to intervention by that operator. At the lowest level of control, the computer console functions as the operator control panel, a very specialized input-output device. A few basic exercises suffice to gain familiarity with console operations, their use in initiating program loading (IPL, manually or via hardware bootstrap) and their use in resolving crashes (program debugging).

The care and feeding of peripherals can be introduced as subsequent laboratory exercises require. The fundamental concepts of serial asynchronous input-output, serial synchronous input-output, block transfers of data (direct memory access), sequential and random access device handling, interrupt handling, hardware error detection and recovery suggest laboratory exercises to illustrate and illuminate each of these.

These concepts can be tied together by the construction of (or analysis of an existing) operating system which provides for multiprogrammed control of several peripherals. Hardware/software measuring devices and techniques can be introduced in the context of studying performance of an operating system.

Facilities for telephone network access to the laboratory computer permit the study of modem control and communication protocols. Simple analogue input and output handling techniques should be examined, in the context of data acquisition and real-time control systems.

The concepts of process, task, queue, concurrent operations, resource allocation, protection, synchronization, etc. are so important to computer science that a student should have many opportunities to see instances of these in use, to "touch" them, change them and measure them.

Providing the Facilities

A computing installation large enough to support the routine computing needs of a Computer Science department should have most if not all of the needed physical facilities. In the case of a shared campus computing facility, a number of issues must be resolved. Block time must be made available to dedicate the shared facility to the exclusive use of the Computer Science laboratory; scheduling needs of either the computing center and the Computer Science lab may preclude this possibility. Arrangements must be made for paying for the block time. Rate structures at some institutions would preclude this use of the campus facility.

Maintaining the integrity and privacy of user files presents an additional complication. It would be necessary not only to dump all on-line files on a removable storage device, but also to erase the original on-line files, and then restore them afterwards. Magnetic tape files and disc pack storage conveniently accessable for normal production operations would necessitate the presence of supervisory personnel to guard against errors or foul play.

The problems of using a general-purpose departmental computing facility to provide a laboratory facility are similar, except that policy decisions are an internal matter.

A third solution is provided by acquiring separate facilities chosen specifically to meet the needs previously enumerated. Such a solution was economically impractical even five years ago, but is an eminently practical one now. The costs and advantages of a separate facility are discussed below, as well as its relationship to the general-purpose facility.

A fourth solution is provided through the use of "virtual machines". The simulation of a virtual machine for a graduate course at Cornell is described in [33]. An advanced system programming course for graduate students at Michigan, using a VM on MTS 67 is described in [3]. These two papers do not discuss cost considerations. Rosen suggests in [31] that general virtual machine systems tend to be inefficient. The use of virtual machines to provide large groups of undergraduates with the equivalent of hands-on experience with a mini-computer has not been investigated.

With the exception of the third solution, the above solutions confront the student with the problem described by Stark in [35]; namely that the student is overwhelmed by the complexity and the sheer bulk of the hardware and the software. Perhaps this is why these solutions can only be successful for advanced courses.

Can a Small Computer Illuminate Large System Problems?

In a stimulating article published in 1972 on the relations between computing centers and departments of computer science, [31] Saul Rosen (who is involved in both) stated it was becoming increasingly difficult for the central facility to allow the special privileges needed for really basic systems exercises and one answer was to get a small dedicated computer. He then stated that "unfortunately, small computers have small systems which, though interesting in their own right, do not provide insight into problems which are typical of large systems." No doubt this was representative of the state of affairs early in 1972. However a number of significant developments since then make it necessary to reconsider the situation.

The halving of end-user prices for mini-computer central processors, core and solid state memories, the development of low cost peripherals and mass storage, the introduction of low cost dynamic address translation hardware and other developments since 1972 make it possible to assemble a very sophisticated configuration for the cost of a small system, by 1972 standards. Of course one then needs a sophisticated operating system in order to provide insight into problems "which are typical of large systems". The Disc Monitor System [9] running on the Datacraft (commercially available since late 1972) provides many of the services of the larger systems (spooled I/O, multiple foreground jobs and batch queue, priority driven, dynamic memory allocation, checkpoint-restart, several language processors, support of reentrant processors, on-line files, synchronous and asynchronous communications, etc.). Such a system can illustrate many large-system problems, such as memory fragmentation, deadlocks, protection, resource management, scheduling, etc.

At this time, the UNIX system [30] is probably the most sophisticated system running on a configuration available for well under $100,000. This time-sharing system was developed at the Bell Telephone Laboratories for the PDP 11/45, and it embodies many new ideas in terms of command language design (e.g. concurrent processing of commands) and file organization, with the goal of enhancing user convenience and system power. It is the author's wish that the next generation of operating systems and the relevant courses may profit from the UNIX experience.

The Digital Equipment Corporation has several operating systems for its PDP-11 family. The RSX-11D probably represents its most sophisticated one to date, supporting multi-tasking, dynamic address translation, queued input-output, file management and other services usually associated with much larger machines.

SELECTING EQUIPMENT

Equipment Selection Guidelines

The guidelines provided in the COSINE report [28-VII] differ from the usual guidelines in selecting a computer; they are nonetheless quite reasonable. The object in a computer lab is not high speed or large capacity but availability of the widest variety of interesting computer hardware, options and devices, operating at a reasonable speed.

If access to a larger central system is possible, one can assume it can be used to provide exposure to some relatively more expensive options, such as single and double precision floating point hardware.

Program and Data Preparation

One of the most important considerations in providing facilities for one or more classes of students involves initial program and data preparation. If program preparation for non-laboratory programming work is done on keypunches, then the laboratory computer should take advantage of this and it is well worth getting a card reader (card readers in the 300-600 card/minute range are entirely adequate and reasonably priced).

If the service computing is done on-line, and keypunches are scarce, then the service computing facility's ability to produce machine-readable material should be exploited. This may involve using punched cards, punched paper tape, magnetic tape cassettes, 7/9 track mini-tapes, floppy disc cartridges, etc.

If the service computer supports remote batch stations, it may be practical to have the laboratory program preparation performed on the service facility and transmitted to the laboratory computer operating as a pseudo-remote batch station. However large system executives sometimes do not give the remote computer operator sufficient control, and in effect this mode of operation might require either non-trivial operating system changes (in addition to development of an RJE capability in the laboratory computer), or an attended operation (i.e., a laboratory operations staff).

Secondary Storage

It is essential that a laboratory have a mass storage device with a direct access capability. Should this capability be provided either by a non-removeable device (e.g., a fixed-head disc) or by a removeable device for which it is impractical to have many removeable media (e.g., some cartridge disc drives use cartridges in the $130+ range), it is very desirable to have a second mass-storage device with very low cost media, to inexpensively backup and restore the information on the first.

In a standard computing environment, mass-storage access is controlled, and files can be protected. In a computer laboratory environment, every user is priviledged, and no file is safe, unless it is locked up! In attempting to modify an operating system, a student may destroy it, not only in main memory, but also on the mass storage device. Thus a backup facility is very desirable.

The recent introduction of the floppy discs [39] promises to provide both the convenience of magnetic tape (low cost, portable, reusable) with the advantages of random access. Some systems are now being marketted with a floppy disc based operating system.

Microprogramming

Several small systems as well as medium and large scale systems have a microprogramming capability. This characteristic may be very important in optimizing a processor for a given application. From the conceptual and tutorial point of view, a ay system which lacks a microprogramming capability can simulate one. Since the simulation of a simple processor can usually be accomplished more easily and realistically than the simulation of a complete system, it is in part a matter of faculty interest as to the importance of a microprogramming facility in the laboratory. Certainly such a system should eventually be made part of a well rounded laboratory.

Options

Options and facilities found only on large scale systems a few years ago are becoming available on many small computers. Facilities to provide program protection (privileged instructions, memory protection) and dynamic address translation hardware for virtual memory systems are available on small systems (e.g., PDP 11/45, Datacraft 6024/4 VM).

Multi-Purpose Peripherals

Some alphanumeric CRT's have a limited graphics capability. This may be a practical interim solution should a full graphics display capability be precluded by cost considerations.

An electrostatic plotter-printer may function as a high-speed printer, as a plotter and as a graphics output device.

Vendor Selection

A 1973 Rand report [20] stated that over 50 suppliers of minicomputers had systems in the $4,000 to $20,000 range for a basic configuration (i.e., no disc, no high-speed I/O, etc). A trade magazine survey [36] listed some 135 minicomputer models in the $900 and up range (not necessarily equipped with a power supply). Details on some 200 models produced by 80 companies are given in [10].

If one wishes to consider independent sources for items such as printers, CRT's, storage devices, modems, etc., then the number of vendors quickly exceeds one thousand.

In such a rapidly developing field, it is wise to enquire of any item:

(a) does it exist?

(b) how many are in the field (how long)?

(c) can it be delivered by a given date?

(d) what evidence is there that it performs as promised?

(e) will it be delivered with complete documentation?

(f) are parts available from other sources?

(g) which options, if any, cannot be field installed

(h) what are its growth capabilities?

Some manufacturers are responding to the inroads made by independent peripheral suppliers by offering substantial discounts for full systems, in the 20-30% range.

The Rand report [20] provides a valuable framework within which competing mini-computer processors may be evaluated.

It briefly examines 15 systems of 9 manufacturers (Table 1). Though many of these systems have since been superceded, the report is nonetheless worthwhile.

Manufacturer	Model	Memory[1]	Price[2]
Digital Equipment Corp.	PDP 11/10	8	$6,996
	PDP 11/40	8	12,995
Data General Corp.	Nova 800	4	6,950
	Nova 1200	4	5,450
	Supernova SC	3+1S	14,250
General Automation	SPC-16/80	4	8,550
Hewlett-Packard	2100A	4	6,900
Honeywell	H316	4	8,400
IBM Corp.	System 7	4S	16,795
Interdata	Model 80	16S	14,900
Texas Instruments	TI-960A	4S	2,850
	TI-980A	4S	3,475
Varian	620/f-100	4	10,500
	620/L-100	4	6,400
	Varian 73	4S	14,500

Table 1. Processors examined in Rand report [20]

Note 1. Memory is specified in units of 1K (1024) 16 bit words of core, with S designating semiconductor memory.

Note 2. These prices are from the report and are thus no longer applicable.

Surplus or Used Equipment

Some laboratories have managed quite well with used or surplus equipment. If the equipment is military surplus, it may be a strictly military item, with no commercial counterpart. Then system expansion is possible only by designing and building the missing links. Furthermore military systems are often constructed with a fail-replace attitude which greatly complicates maintenance, short of having duplicate systems and a full stock of spare modules. The Minuteman guidance computer [5] from the Anti-Ballistic Missile system is a good illustration of the difficulties one can encounter.

From time to time, military and government surplus equipment which is slightly dated commercially available equipment, may be made available to institutions under attractive terms. More often than not, these systems might make sense housed in a production facility, but they are not usually competitive with today's mini-computers, when installation, maintenance and expansion costs are considered.

OPERATING A LABORATORY

Equipment Maintenance

A standard maintenance contract may appear expensive at first glance (ranging from several hundred to several thousand dollars per month). When one considers its value as an insurance policy (such contracts usually include all parts as well as labor) contrasted with the full costs of setting up and supervising an in-house maintenance staff, the terms of a contract may appear attractive.

Many small-computer installations have developed an in-house maintenance capability, and call upon outside help only when faced with problems requiring special training and equipment (e.g., fixing a disc drive). Such on-call help may not always respond immediately, the charges are on an hourly basis ($20-35 per hour), often with a minimum four hour charge, with the cost of parts not included.

In a laboratory with a mixed configuration (multiple vendors), it may not be possible to cover the laboratory with one maintenance contract. Having an in-house capability provides this degree of freedom. The cost of an in-house capability includes standard electronic test equipment (oscilloscope, VTVM, pulse generator, logic probe, extender cards, etc.), tools, supplies, and a stock of spare parts (integrated circuits, connectors, cables, fuses, indicator lamps, etc.).

If one accepts the possibility of having to shut down the laboratory due to critical equipment failure for even a several day period, then the spare parts inventory for an in-house staff may be kept quite low. For obvious reasons the electromechanical devices fail more frequently. Since these are usually some type of input-output device or a rotating storage device, a laboratory can continue functioning with some devices out of service if alternate devices are included in the configuration. An inoperable card reader may force users to the inconvenience of paper tape input for a short period; a bad disc might require users to fall back on a tape operating system.

Having an in-house capability makes it possible to select the most desirable item for a particular application, without being constrained to select the nearest approximation supported by the manufacturer. Then one proceeds to build an interface, document it, and prepare a set of diagnostics. The full cost of these activities tend to narrow the gap between the assumed lower cost of building it yourself and buying it ready to use. In some cases it will cost more to build it yourself, and it becomes difficult to justify this extra cost as "staff training". Yet in the long run, a modest in-house hardware design and construction capability, in addition to providing routine maintenance, allows one the freedom to think and plan beyond what is currently commercially available and encourages a dialogue between the computer engineer and the computer scientist.

Support Facilities

Whether the service computing is provided by a campus-wide center or a departmental system, full use of it can do much to enhance the ability of the laboratory to provide a wide range of experiences to a large number of students.

Assembling programs on a small computer can be either time-consuming or expensive. It will be time-consuming if slow peripherals are used on a minimal configuration, by students who have no other way to assemble their programs. A 200 line program (e.g., to illustrate timer interrupt handling, with a programmed time-of-day capability) may take half an hour to assemble once on a system with Teletype tape input-output (10 CPS; 4 minutes for each of 3 assembler passes, plus 10 minutes to load the assembler). On the other hand, the object code for this same program could be generated at the service center (with no great inconvenience to the student, especially if several runs are necessary to eliminate syntax and logic errors and be loaded into the laboratory system in one minute (200 lines giving 300 words, or 600 bytes, requiring 1 minute at 10 CPS).

Assembling programs in the laboratory can be made more convenient for the student, but at the expense of acquiring a high-speed input device (e.g., a card reader, a fast paper tape reader, a magnetic tape cassette, etc.) and a fast hard-copy output device (e.g., a line printer). Repeated assemblies in the laboratory are a waste of scarce resources (i.e., laboratory time).

Using an assembler for the laboratory system which operates on the service system can significantly increase the capacity of the laboratory. Such cross-assemblers are either available from the manufacturer or may be generated by the students themselves [11].

Transfer of assembler output from the service center to the laboratory may be accomplished via binary cards, paper tape, tape cassettes, 7/9 track tapes, etc. Some service centers provide routine inexpensive punched card output and impose extra charges or delays for other output media; the service center rate structure may influence the choice of transfer media.

Assembler output may also be obtained through time-sharing terminals equipped with a recording device, typically paper tape or magnetic tape cassettes. Some systems restrict time-sharing communications to a smaller subset of bit patterns than one may need for assembler output, and the media transfer time might be 30-40% longer than expected, with special encodings/decodings required.

It may be helpful in introducing the laboratory central processing unit to a class of students to have a simulator for it which operates on the service computer [25]. This is a useful teaching aid; the writing of such a simulator is also an instructive assignment. Having a printout of the changes in the machine state is very helpful to a student struggling with the subtleties of a machine reference manual.

The laboratory system itself may find it advantageous to operate as a remote batch station connected to the service center, or as an intelligent terminal connected to the service center. The security requirements and sophistication of the remote capabilities of the service center may make such arrangements practical only if the laboratory is completely attended.

Documentation

The laboratory user must have access to far more information than the typical computer user. In addition to manuals on the central processing unit, the assembler and the basic operating system, he will at one time or another need detailed information about each peripheral unit, about the specific unit assignments and options installed.

It is advisable to obtain permission to reproduce vendor supplied documentation, as the cost of a set of manuals may be excessive for the average student. It is not unusual to have to supplement vendor documentation to provide corrections and clarifications.

A more cumbersome documentation requirement is provided by the need to make software listings available for examination and study. A student can learn a great deal by careful scrutiny of listings of an operating system, of an assembler, of a compiler and so on. A vendor's policy on providing source code for his software should be well understood.

Management Problems

In a typical laboratory setting in the physical sciences and engineering, the laboratory equipment costs permit the laboratory to provide work stations for perhaps twenty or more students working independently, with laboratory exercises involving expensive machinery properly scheduled. This parallel processing of groups of students makes it possible to provide full time laboratory supervision; the supervisors assist the students, and protect the equipment.

In a computer science laboratory, the indispensable equipment item is an entire computing system. The experiments for which the laboratory was created necessitate the exclusive use of most of the system by a single student, with the unused items being of no value to others (unless they can be switched to a second computer system). Since such systems cost in excess of $20,000, this precludes outfitting many laboratory work stations. This leads to establishing an open laboratory policy, with 24 hour per day access, with no full-time supervision or operating staff.

Scheduling can be managed by having a sign-up procedure, with a time limit per day based on the time of year (i.e., liberal time limits in slack periods). The sign-up book should also be used to report any laboratory-related problems. The sign-up book is probably the only reliable source of laboratory use statistics (unless the laboratory has full time supervision).

The desire to make available a multiplicity of systems and software packages adds to the management problems (documentation, updates, maintenance, staff training, instructor education).

Experience with a Computer Laboratory

The University of Wisconsin Computer Sciences Department was able to acquire a Datacraft 6024/3 in the fall semester of 1971, around which a computer systems laboratory was established. A PDP 11/20 was acquired a few months later. The current configurations for each machine are given in the appendix. Prior to these acquisitions, the short term loan of a PDP-8 and a Hewlett-Packard 2100 system provided us with a transient facility. General-purpose computing services were then provided by the campus computing center UNIVAC 1108, now replaced by an 1110.

The choice of the Datacraft and the PDP 11 was governed by our conviction that each was interesting in its own right, each somehow representative of a large class of computer architectures, and fully capable of illustrating each of the important concepts the laboratory was to illuminate, in addition to being reasonably priced.

With each machine we began with the minimum amount of memory and a console teletype. We evolved into the present configurations by successive acquisition of each device, memory increment and option. The acquisition sequence was dictated by our needs, the available funding and the choice of devices then available. Further details on the equipment and its use is found in [12,13].

Should we not have concentrated all of the scarce funds on a single system? This certainly would have allowed one system to have a richer configuration. If we were convinced that one system was insufficient, should we not have acquired a copy of the first? This would certainly facilitate maintenance, ease the burden of documentation, and allow simple sharing of peripherals.

It is helpful to reiterate that the projects the students are to carry out in the computer lab require a hands-on access, one student at a time (possibly in teams of two for some larger projects), and that one class of 30 to 40 students can saturate a computer which is available 24 hours per day. Thus the need for more than one computer is felt very quickly at an institution which can have several classes using the computer lab.

The acquisition of a second computer also makes it possible to investigate the problems of computer to computer communications, which is a first step into an examination of computer networks.

Of course, investigations of this type effectively halve the capacity of the lab! They can however be accommodated through judicious scheduling.

We felt it was important to have two computers, and that they be as different from each other as possible (without necessarily being in the one-of-a-kind class). This makes it possible to engrave the student's mind with the notion that not all small computers are alike, and makes it possible to undertake comparative studies (e.g., which features of the addressing capability of each machine affect certain classes of applications).

<u>Costs</u>

A laboratory can be started with a capital investment as low as $5,000 for a CPU with a teletypewriter and 4KW memory. A configuration in the $25,000-50,000 range would provide medium speed input-output capabilities, a mass-storage device and a variety of options and peripherals.

Cost of a maintenance contract may be estimated using 1% to 2% of the capital equipment cost to obtain a monthly charge.

Staff costs, if fully accounted for, soon overshadow equipment costs. A laboratory supervisor with an assistant can provide a minimun of software and documentation support. This assumes an open, unattended laboratory setting (24 hour per day access) which may be difficult to implement at some institutions. We find that installing an ingenious missing-hardware detection circuits makes round the clock unsupervised access feasible (to date).

The cost of faculty supervision should not be overlooked. It probably would fall in the 1/4-1/2 full time equivalent range.

Costs reported by other institutions in establishing and operating computer science laboratories do not vary by an order of magnitude. The University of Massachusetts speaks of a $120,000 investment, with a $4,000-5,000 annual operating cost [17]. Their report stresses the necessity of developing a long range development plan. The University of Alberta managed to acquire some 8 computers for $80,000 by taking advantage of overstocking, educational and promotional discounts, etc. [26].

Growth and Development

The long range plans for a laboratory should take into account the need for a "critical mass" of equipment in the initial stages, followed by a pattern of incremental growth. Though a laboratory can begin with a minimal configuration (CPU and teletypewriter), it is very difficult to motivate other faculty to take advantage of such a limited resource, especially if they are accustomed to the amenities provided by the large central system. The critical mass could be defined as that configuration required to comfortably support an interesting operating system. It would usually include a CPU with at least 48K bytes of memory, a disc, a medium speed card reader and line printer, a teletypewriter and a CRT.

It is necessary that the curriculum be changed to reflect the laboratory aspects of computer science. This notion has also been advanced in another context. The paper entitled "separation of introductory programming and language instruction" [18] argues for a lecture course on concepts and an associated laboratory for coding experiments and experience. Similarly COSINE report VII assumes a formal laboratory experience will be coordinated with a sequence of computer related courses.

CONCLUSIONS

Whereas students in other disciplines have begun to rely more and more upon computer simulation to gain insight into the empirical foundations of their disciplines (e.g., chemistry [27]), it would be appropriate for students of computer science to experience more of an empirical flavour in their discipline, through the facilities of a computer sciences laboratory. Such a laboratory can be established and operated with a reasonable expenditure.

This report has stressed the use of PDP 11 and Datacraft equipment, because the author has access to this equipment. It is the author's considered opinion that although equipment selection should be a very deliberate endeavor, any number of small systems could be used as a basis for a laboratory. It is not so much what you have that counts as what you do with it.

REFERENCES

1. ACM SIG/SIC Reference Guide, Comm. ACM 6, 12 (December 1973), 783-784.

2. ACM Curriculum Committee on Computer Sciences, Curriculum 68, Comm. ACM 11, 3 (March 1968), 151-169.

3. Arden, B. W., Flanigan, L. K. and Galler, B. A., An advanced system programming course, Proc. IFIP (1971), 1510-1514.

4. Austing, R. H. and Engel, G. L., A computer science course program for small colleges, Comm. ACM 16, 3 (March 1973) 139-147.

5. Beck, C. H., Proceedings of the third meeting of the Minuteman computer users group, July 1971 (available from C. H. B., Tulane University).

6. Bell, C. G., Grason, J. and Newell, A., Designing computers and digital systems, Digital Press, Digital Equipment Corp., 447 pp., Maynard, Mass., 1972.

7. Couger, J. D. (ed.), Curriculum recommendations for undergraduate programs in information systems, Comm. ACM 16, 12 (December 1973), 727-749.

8. Chu, Y., Introduction to computer organization, Prentice-Hall, New York, 376 pp., 1970.

9. Datacraft Corporation, Disc monitor system general specifications. Preliminary specifications, August 1971, Latest edition, September 1973, Ford Lauderdale, Fla.

10. Datapro Research Corp., Datapro reports on minicomputers, Datapro Research Corp., Delran, N.J., 1973.

11. Desautels, E. J., A mini-computer support system, University of Wisconsin Computer Sciences Department Technical Report No. 116, February 1971.

12. Desautels, E. J., A small computer in a large university, Proceedings of ACM SIGPLAN Symposium on pedagogic languages with small computers, (January 1972), 28-32.

13. Desautels, E. J., Chow, V. and Schneider, M., Loosely coupled systems, University of Wisconsin Computer Sciences Department Technical Report No. 187, July 1973.

14. Desautels, E. J., A small computer's rule in computer sciences instruction and research, Proc. National Datacraft Users Organization, Madison, Wisconsin, September 1973.

15. Digital Equipment Corp., RTM Register transfer modules, Digital Equipment Corp., 50 pp., Maynard, Mass., 1973.

16. Dijkstra, E. W., GoTo statement considered harmful, Comm. ACM 11, 3 (March 1968), 147-148.

17. Eckhouse, R. H., Jr., The computer science laboratory, SIGCSE Bull. 4, 1 (March 1972), 42-45.

18. Fisher, P., Hankley, W. and Wallentine, V., Separation of introductory programming and language instruction, SIGCSE Bull., Third Symposium on Computer Science Education, (February 1973), 9-14.

19. Hamblen, J. W., Using computers in higher education, Comm. ACM 14, 11 (November 1971), 709-712.

20. Hollingworth, D., Minicomputers: a review of current technology, systems, and applications, Rand report R-1279, 104 pp., Rand Corp., Santa Monica, Cal., July 1973.

21. Hosch, F. A., Some comments on the role of computer science education, SIGCSE Bull. 5, 3 (September 1973), 13-17.

22. Instrumentation Systems Center, University of Wisconsin computer facilities, University of Wisconsin-ISC, Madison, Wisconsin, 1973.

23. Kahan, W., A survey of error analysis, Proc. IFIP, 1971, 1214-1239.

24. Lynch, W. C., Operating system performance, Comm. ACM 15, 7 (July 1972), 579-585.

25. Mac Callum, I. R., O'Mahoney, B. J. and Knowles, K., Simulation as an aid to machine code teaching, Computer Bull. 16, 10 (October 1972), 488-491.

26. Marshand, T. A. and Tartar, J., A course in minicomputer systems, 153-156, Third SIGCSE Symposium, SIGCSE Bull. 5, 1 (February 1973).

27. National Academy of Sciences, Computational support for theoretical chemistry, Report of a May 1970 Conference, National Academy of Sciences, Washington, D. C., 1971.

28. National Academy of Engineering, Commission on Education (COSINE) Task Force Publications.

    I. Some specifications for a computer-oriented first course in electrical engineering, September 1968.

    II. An undergraduate electrical engineering course on computer organization, October 1968.

    III. Some specifications for an undergraduate course in digital subsystems, November 1968.

    IV. An undergraduate computer engineering option for electrical engineering, January 1970.

    V. Impact of computers on electrical engineering education--a view from industry, September 1969.

    VI. Digital systems laboratory courses and laboratory development, March 1971.

    VII. Minicomputers in the digital laboratory program, April 1972.

    VIII. An undergraduate course on operating systems principles, June 1971.

29. Noe, J. D., University education in computer measurement and evaluation, SIGCSE Bull. 5, 2 (June 1973), 8-13.

30. Ritchie, D. M., and Thompson, K., The UNIX timesharing system, (to appear in Comm. ACM, 1974).

31. Rosen, S., Comments on relations between computing centers and departments of computer science, ACM SIGUCC Newsletter, (March 1972), 14-17.

32. Salton, G., What is computer science?, J. ACM 19 (January 1972), 1-2.

33. Shaw, A. C. and Weidermann, N., A multiprogramming system for education and research, Proc. IFIP (1971), 1505-1509.

34. Sloan, M. E., Coates, C. L. and Mc Cluskey, E. J., COSINE survey of electrical engineering departments, Computer (June 1973), 30-39.

35. Stark, R. H., Computer science needs its laboratory, SIGCSE Bull. 4, 1 (March 1972), 46-48.

36. Stein, P., Succeeding with minicomputers, Computer Decisions, (November 1973), 32-49.

37. Stone, H. S., Introduction to computer organization and data structures, McGraw-Hill, New York, 1972.

38. University of Wisconsin, Descriptive Information, Computer Sciences Department, Madison, Wisconsin, 1973.

39. Zschau, E. V. W., The IBM diskette and its implications for minicomputer systems, Computer, (June 1973), 21-26.

40. Timmreck, E. M., "Computer Selection Methodology, (Comp. Surveys 5, 4(December 1973) 199-222) is of limited value as a guide to selecting equipment for a computer laboratory. It reflects the thinking about large (batch?) systems that must have been typical of large organizations in the mid and late sixties. The references do not go beyond 1971.

APPENDIX:

University of Wisconsin Computer Systems Laboratory Equipment

Datacraft 6024/3 configuration

Processor: 32K x 24 bits, 1 microsecond core memory, memory protection, privileged instruction option, 8 bit channel, 24 bit channel, 24 DMA channel

Devices/interfaces: ASR 33 Teletype, Documation 600 CPM card reader, 120 Hz clock UW-PSL* synchronous interface, 6 Mbyte CDC cartridge disc, Dynastor dual drive floppy disc (1/4 Mbyte per drive), UW-PSL* 32 terminal multiplexor, own 16 bit interface. Universal Data System originate/auto answer modems.

*(University of Wisconsin - Physical Sciences Laboratory)

PDP-11/20 configuration

Processor: 20K x 16 bits, .95 microsecond core memory disc/dectape hardware bootstrap

Devices/interfaces: ASR 33 Teletype, 60 Hz clock, 64KW fixed-head disc, dual dectape drives, synchronous interfaces, 3 asynchronous serial interfaces (110 to 2400 baud), 4 asynchronous parallel interfaces (16 bits), modem interface, Remex paper tape punch (75 CPS), Digitek paper tape reader (300 CPS)

Shared devices

Centronics 160 CPS printer, Ann Arbor CRT display, ASR 38 Teletype, Video Systems CRT (upper/lower case), Colorado Video TV camera, 12 bit DAC/ADC, Digilog portable CRT, Omnitec portable acoustic coupler, Univac 1110 (via synchronous interfaces)