nate, but otherwise the authors are reasonably successful at achieving this goal. The writing style is informal, with enough interesting and humorous asides to make the book quite readable. Chemists who wish only a general overview of supercomputers may find this book to be useful.

LABORATORY LOTUS
A Complete Guide to Instrument Interfacing
by Louis M. Mezei
Prentice Hall, Englewood Cliffs, NJ
317 pages, hardbound $36.00, 1989

Reviewed by Harry E. Pence

Lotus 1-2-3ubiquitous business spreadsheet, is already widely used in chemistry laboratories to organize large data sets and to do simple, repetitive calculations. Mezei's‹f‹book describes how this software can not only do calculations and make graphs but also collect data from and control laboratory instrumentation.

Much of the book is devoted to explaining how to use either
Lotus 1-2-3 or Symphony to interface instruments having RS-232 communications. The presentation is extremely systematic and clear. The author provides brief summaries at the beginning and end of most chapters and also includes a running outline to make the discussion clearer. Sample programs are developed one section at a time to make these examples easier to understand. This modular approach also helps to simplify the explanation and minimize the chances for error.

The instruments chosen to serve as examples include the ACROSYSTEMS ACRO-900 interface, the Perkin-Elmer LS-2B
Filter Fluorimeter, the Mettler AE163 analytical balance with optional data interface, and the Bio-Tek EI-309 Microplate Reader, but for readers who don't have access to these specific instruments, the author explains how these lessons can be applied more generally to other instrumentation. The discussion is not limited to data collection and treatment, but also covers how Lotus can set up the instrument, take the sample, and accumulate the data.

The presentation covers both RS-232 as well as IEEE-488 communications. For older instruments that lack a modern communications port, instructions are provided on how to access the data through a chart recorder output, an analog voltage output, or a digital display. Although considerations of cost and convenience cause the author to express a preference for the basic spreadsheet programs, he also discusses how to use Lotus Measure®R for instrument interfacing.

The final chapter includes a general summary and some excellent general advice about developing a communications program for a laboratory instrument. This section offers some especially valuable tips on programming and troubleshooting. Appendices are provided that cover RS-232 cables and pinout specifications and also review the most useful Lotus functions and commands.

Among the profusion of books on Lotuss delightful to find one that is specifically aimed at chemists. It is even more pleasant to be able to report that the discussion is so systematic and lucid. This book should be both an excellent resource for those who are already dedicated Lotus users as well as an effective argument to convince new users that this software can play a valuable role in their laboratories.

PERSONAL COMPUTERS FOR SCIENTISTS
by Glenn I. Ouchi
American Chemical Society, Washington, DC, 1987
300 pages, paperbound, $22.95, hardbound $34.95

Reviewed by Harry E. Pence

Glenn Ouchi is a well-known advocate of laboratory uses of microcomputers, who has written articles for many widely-read computer journals, taught ACS short courses, and edited a newsletter on computing in the chemistry laboratory. He is well qualified to fulfill the promise that the book will "help you select and use a personal computer (PC) to solve problems in your work."

The initial section of the book covers the fundamental hardware and operating systems characteristic of microcomputers. The author does not presume very much prior computing knowledge on the part of the reader. The clear and concise explanations are accompanied by many excellent illustrations and line drawings. First-time computer users should find this book to be both attractive and readable.

The core of the book is the section on applications software. Ouchi emphasizes that when choosing a computer the essential consideration should be to identify the critical problems which are to be solved and then to select a machine that will run the software that is best able to accomplish those jobs. Advanced hardware capabilities, such as high performance or extended memory, are worthless unless the software needed for the tasks at hand will run on the computer. As many users have sadly learned, purchasing a computer based on the assurance that the
required software will be available "real soon now" will usually lead only to frustration and wasted time.

The author discusses the "big three" types of software: word processors, spreadsheets, and data bases, but also includes applications of major interest for scientists, such as graphics and statistics programs. Much of the software that he describes is widely©used for business applications, but he provides good examples of how those commercial packages are equally applicable to scientific projects. The realistic, sample experiments are excellent for showing

how each piece of software can be used, but if there were more emphasis on the types of jobs that each type of software could do, the book would become less dated as new releases of these packages have become available.

The final sections of the book deal with communications and interfacing. This material reviews some of the key topics fundamental to data communications, interfacing, and using electronic data bases. The treatment is rather brief, but not unreasonably so for this level. Each chapter includes a short list of references for further reading, and a glossary of commonly-used computer terms is provided.

This is a clear, well-written summary of the basic knowledge needed to use a microcomputer for scientific projects. Scientific computing is developing so rapidly that some of the material already seems somewhat dated, but there is also a great deal of sound advice that is independent of the most recent software releases. Well-written, introductory books of this type make an important contribution, especially for faculty or students who have had little previous experience with computers.

*Professor of Chemistry
SUNY-Oneonta
Oneonta, NY 13820

* * END * *

INFORMAL NOTES ON PROGRAMMING-LANGUAGES (AND SOME UTILITIES)

K. W. Loach, Chemistry Dept., SUNY Plattsburgh.

LOACHKW@SNYPLAVA.BITNET or (518)564-4116

This has been written on the basis of personal experience, talks with other users and general reading. The discussion is limited to high-level languages (i.e. languages intended to be intelligible to humans) and emphasizes systems suitable for chemists and available on personal computers. The sources and citations are not comprehensive. My opinions expressed here are avowedly influenced by personal taste, and by the following sources:

General Sources of Information:

PC Magazine, PC Tech Journal BytePC Week, Info World, Digital NewsDigital Review, Unix Review, DEC Professional,C Users Journal, Dr. Dobbs Journal, J. Chem. EducationTrends in Analyt. Chem.

Sources of Software:

Programmer'sShop,Campus Technology,PublicBrand Software,Programmer's Connection, Micro Warehouse ,Austin Codeworks, Programmer's Paradise ACS Software, Campus Technology, C User's Group, Computer Solutions Free Software Foundation
(addresses and phone-numbers of these sources can be obtained from many of the General Sources above, or from the author.)

WHAT IS A LANGUAGE?

What is a 'language', in the general sense? It is any system that allows the storage and execution of autonomous high-level computer instructions, and which allows the sequence of those instructions to be controlled by data comparisons. 'Storage' and 'autonomy' are essential to the definition, because they serve to distinguish languages from interactive command systems (which execute commands as they are entered by a human operator). In an interactive system, the operator makes decisions between commands, and the commands themselves lose autonomy and cease to be a 'program'.

The 'classical' languages are mostly familiar: Fortran, Cobol, Basic, Snobol, Pascal, Ada, C, Lisp, and APL; there are also the less familiar 'neoclassical' languages, e.g. Smalltalk, Forth, Prolog, and Awk. I call these all 'classical' (in spite of their very wide differences) because they are all instantly recognizable as languages. In contrast, a great deal of programming is now done with 'utilities' that are not thought of as 'languages'. I have heard my colleagues deny that they use programming languages, and yet the systems that they use (word processors, spreadsheets, equation solvers, shells, file managers, editors, etc.) all commonly have programming language capabilities.

There has been a strong convergence. While 'utilities' have been increasingly acquiring language capabilities, 'languages' have been adding utility-like features to become 'development systems' or 'programming environments'. Meanwhile, environments have become increasingly rich, with features such as object-oriented programming (OOP), hypertext, the graphical user interface (GUI), hot-linkage (dynamic data exchange or DDE), interoperability, open-systems and the definition of language standards.

I will discuss classical languages first, and then the environment issues, and then utilities with strong language features.

PART ONE: 'CLASSICAL' LANGUAGES

Fortran: For many chemists, Fortran is the programming language. It was one of the earliest of high-level languages, and is still widely used by scientists and engineers. In the seventies, when 'structured programming' became the fashion,