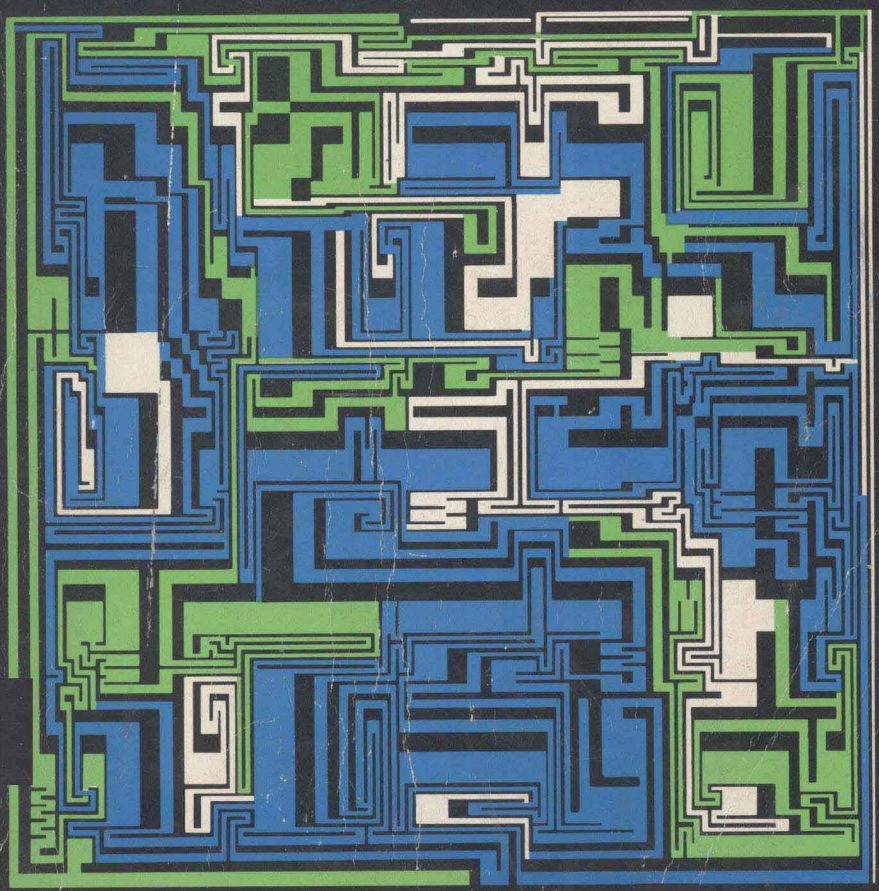
EWARD BOOK

$12.95

PASCAL PROGRAMMING FOR THE APPLE

T.G. LEWIS



A step-by-step guide with ready-to-run programs—
brings you up to date on the very latest technology!

Pascal Programming for the Apple

T. G. Lewis
Oregon State University

Reston Publishing Company, Inc.
A Prentice-Hall Company
Reston, Virginia

Library of Congress Cataloging in Publication Data

Lewis, Theodore Gyle
Pascal programming for the Apple.

Includes index.
1. PASCAL (Computer program language)
2. Microcomputers—Programming. I. Title.
QA76.73.P2L48 001.64'24 80-25382
ISBN 0-8359-5455-2
ISBN 0-8359-5454-4 (pbk.)

10 9 8 7 6 5 4

Printed in the United States of America

Preface

Pascal is the most influential contribution to programming made in the past 25 years. It is not only a programming language, but is also a statement about programming style. Thus, one major aspect of Pascal is its impact upon the way programs are conceived and then composed.

Second, the VLSI revolution in hardware is made dramatically available through Pascal software; and UCSD Pascal represents the first of a new generation of software systems for microcomputers. When combined with a microcomputer, Pascal is the greatest microcomputer software advance made since BASIC.

The Apple computer has capitalized on these most recent advances in technology in both hardware and software. Its graphics and sound reproduction hardware are intimately tied to the UCSD Pascal software. Their combined utilization is synergistic – greater than their sum. For example, the data-structures capabilities of Pascal naturally provide easy-to-use graphical symbols. The **type** COLOR = (BLACK, WHITE, BLUE, GREEN, VIOLET, ORANGE) fits into the graphics hardware support without language "extensions" usually found as nonstandard features in other languages.

It was because of this emerging synergism between hardware and software that this book was written. The following chapters are devoted to clarifying concepts of computing, programming style, devices, and information storage/retrieval. In particular, sections are devoted to teaching fundamentals of the Pascal System (Chapters 1-3), reviewing the Pascal language (Chapter 4), and applying this fundamental knowledge to many applications (Chapters 5-10).

Chapters 5-10 were designed to sharpen programming skills while simultaneously introducing new programming techniques. Financial applications are introduced in Chapter 5 and text processing using Pascal **strings** in Chapter 6. Chapter 7 explores a very important area of concern in small systems: how to implement large programs on small computers.

Chapter 8 covers the graphics modules provided in TURTLEGRAPHICS, and Chapter 9 is a short introduction to the musical tone generator, NOTE. Finally, in Chapter 10, we unify and bolster the **file** structure operations casually described throughout the book.

Using and learning about the Apple implementation of UCSD Pascal has been a tremendously stimulating experience. I have had great assistance from many people: Dr. John Couch and Susan Wells at Apple Computer, Evan Sakey at UCSD, and the helpful contributions of Dorothy Hyde, Sharon Bassett and the office staff at Oregon State University. Thanks go also to Peg Vorderstrasse and Ann Puig for their typing.

I hope you will enjoy reading the pages to follow. I know you will benefit from the experience if you take time to study each program in detail.

T. G. Lewis

Contents

The French Connection (The System)

"I chose the name because Blaise Pascal was the first (perhaps one of the first) person to build what we may reasonably call a digital calculator. He did so around 1642 to speed up the tedious calculations when helping his father who was a tax collector."

Professor Niklaus Wirth
July 16, 1979

1.1 PASCAL THE MAN (1623-1662)

He was a sickly child, raised by his father who intended to educate him in the classics and literature. But 12-year-old Blaise Pascal surprised his father by secretly teaching himself geometry. One day he showed his father proof of Euclid's thirty-second theorem — that is the sum of the angles of a triangle equals 180 degrees — and that began his scientific training. He wrote a book on geometry (conic sections) before he was 16 and invented the calculator at 19 years of age.

The seventeenth century was a time of great scientific advancement. It might be compared with the post-World-War-II period of expansion in scientific knowledge. Many great inventions and ideas were being thrust onto the English and French societies during Pascal's lifetime.

William Harvey, Galileo, Newton, Descartes, Boyle, Napier, Kepler, Huyghens, and others lived about the same time as Pascal. Thus it is not unusual that Pascal's genius led him to the formulation of Pascal's Law (hydrostatic pressure is proportional to a cross-sectional area of a fluid) and Pascal's Triangle (coefficients of the binomial expansion).

It may be surprising that Pascal invented an early version of mechanical calculators, since this required the design and construction of precision gears; but about 1641, at the age of 19, Pascal developed a working calculator similar

---

Personal correspondence with Niklaus Wirth concerning the name of his new programming language.

to the odometer system of gears used in speedometers today. It had been 1600 years since the previous advance in such machinery was made by Hero of Alexandria. Pascal constructed about 50 models of his calculator, of which 6 or 7 still exist.

Pascal led the way in "reasoning" machines while others of his time were inventing pendulum clocks (Huyghens, 1656), barometers (Torricelli, 1644), and fountain pens (1657). Even though he was awarded a patent for his calculator, the device was too costly and unreliable to become a commercial success; but his work set the stage for modern computer technology. Indeed, Pascal foresaw the modern-day implications of his invention when he remarked, "The arithmetical machine produces effects which approach nearer to thought than all the actions of animals."* This implication drove Pascal to question the power of the human mind. His contemporary, Descartes, asserted that reason alone separated mankind from unthinking machines and animals. Pascal, on the other hand, believed his calculator to be an example of a machine that could "reason" as well as function as an accountant.

Only free will distinguished humanity from the animal world. This notion caused him to ask questions about God and mankind's place in the world. In fact, Pascal was tormented by religious questions. On November 23, 1654, he experienced a two-hour vision during which his "heart felt God." He recorded the details of this experience and sewed a note to himself into the lining of his clothing. Unfortunately for science, this revelation convinced Pascal to abandon the world. He converted to Jensenism and produced his most notable work, *Pensés*, while contemplating his religious beliefs. In 1662 at the age of 39, Pascal died believing in the importance of both the heart and the mind. The mind is limited in what it can understand, Pascal wrote, while the remaining mysteries of the universe can only be understood through faith in God.

A thread of history connects the primitive calculator invented by Pascal to the sophisticated machines that run Pascal programs described in this book. Charles Babbage (1792-1871) was influenced by Blaise Pascal, Gottfried Leibniz, and Joseph Jacquard. Babbage's calculating engine was a mechanical computer capable of computing while under the control of a program. In 1937, while a graduate student at Harvard University, Howard Aiken became aware of Babbage's work. Aiken, with financial support from IBM, was responsible for constructing MARK I, the first general-purpose electronic computer. The microcomputers of today are descendants of the MARK I.

In some sense we have returned full circle to Blaise Pascal. Pascal, the man, believed in both reason and blind faith; the language Pascal is both a scientific tool based on simple reasoning about computing and a humanly understandable language based on the belief that simple, efficient languages are the most reliable

*J. Bonowski, and B. Mazlish, *The Western Intellectual Tradition*, Harper Torchbooks, N.Y., 1975, p. 240.

tools for writing computer programs. In the pages to follow we will be concerned about the programmer as well as the computer when implementation issues arise. For example, simple explanations for straightforward methods will be preferred over technically rigorous explanations. We will use examples that are easy to read rather than examples that demonstrate sophisticated algorithms. Finally, we will employ *short* programs rather than lengthy, complete programs so that the concepts will be illuminated.

1.2 PASCAL THE LANGUAGE (1971-)

Niklaus Wirth invented Pascal for two reasons:*

". . . to make available a language suitable to teach programming as a systematic discipline based on certain fundamental concepts clearly and naturally reflected by the language."

". . . to develop implementations of this language which are both reliable and efficient on presently available computers. . ."

It was the expressly stated goal of Wirth to develop a tool for ". . . the understanding of programs by human readers and the processing by computers."

This is also the goal of this book. We wanted to develop a document for human understanding of the Pascal language and for understanding the Pascal *environment*, which consists of both the Pascal system and the Apple computer.

It is important to understand not only the language Pascal but also the environment of the Pascal System, because the environment influences how the language is used. Figure 1.1 gives a user's view of the levels to be found within

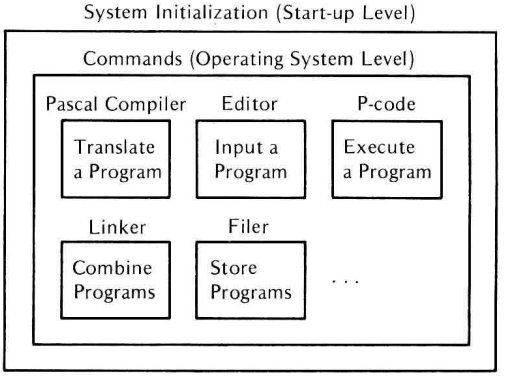


Figure 1.1. The Pascal Environment

*N. Wirth, "The Programming Language Pascal," *Acta Informatica*, 1, 35-63 (1971).

the Pascal environment. These levels constitute a kind of Pandora's Box with additional boxes inside.

Each box is at a level within the environment, and each level contains boxes for designing and implementing Pascal programs. We have shown only a few of the innermost boxes in Figure 1.1. For an in-depth description of these boxes, skip to Chapter 2.

The Pascal environment is automatically initialized when the Apple computer is started. The system is loaded from disk into the computer's main memory. When ready, the screen on your computer will announce itself:

```
Welcome APPLE1: TO
U.C.S.D. Pascal System II.5
Current Date Is 15-DEC-79
Command:E(dit,R(un,F(ile,C(omp,L(ink,X(exute,A(ssemble,D(ebug ?[I.5]
```

This prompt puts the Pascal System in the Command level where it waits for input of the first letter of the command desired. To open the editor box, type E, and to open the compiler box, type C, etc. Each command is accepted in its abbreviated form (usually first letter).

Figure 1.1 illustrates only the first 3 levels of the Pascal environment. In the next chapter we will examine each box in more detail, but for now we will explain the purpose of the boxes at level 1.

Assembler The Pascal environment includes a generalized machine language symbolic assembler. While we will not be concerned about its use here, it can be used to produce machine code programs. These programs may be tightly coded routines which are used by Pascal programs. In order to combine them with Pascal, we must use the Assembler and then the Linker.

Compiler All Pascal programs must be converted into an intermediate (condensed) form called *P-code*. The Compiler does this by reading a text file created by the user (see Editor), converting it into P-code equivalents and writing the P-code file out onto disk. The converted P-code version is processed further by first Linking its parts together and then interpreting the Linked parts using a P-code simulator (see execute).

Debugger This command is used to set breakpoints in Assembler programs. This is an aid to debugging programs; but since we are not concerned with the use of the Assembler, we will not use the Debugger.

Editor The Pascal environment includes several editor programs. We will describe only one editor – the screen editor. This program is used to prepare Pascal programs. It allows the user to type Pascal statements, edit errors, and save the edited programs on disks.

Filer The Filer is a program used to save programs on disk files, copy files, delete files, examine files, and set the date (calendar). Each file is named by the user using a dot notation.

VOL:NAME.TYPE

Thus, the disk volume VOL: is given first, followed by the file name NAME and the file extent .TYPE. The most common file extents are program text, .TEXT, program P-code, .CODE, and program data, .DATA.

Linker The Link command is used whenever separately compiled program units are to be combined into a single P-code module. If a program uses another program unit, then the Linker must combine the two into a single unit before either can be executed.

Run This command is actually three commands rolled into one. The most frequent commands – Compile, Link, and eXecute – are carried out in sequence whenever R is entered. The Compile step is skipped if the workfile (see Figure 1.2) is already compiled. The Link step is also skipped if not needed. The eXecute step is carried out by running the P-code file.

eXecute This command causes the P-code simulator to begin interpreting the P-code workfile, or some other .CODE file specified by the user.

? This command displays additional (less frequently used) commands:

User Restart
Initialize
Halt
? (return to command level)

The H command is particularly useful because it allows graceful termination of a session.

The explanations above discuss a workfile and a P-code simulator. What exactly are they?

The workfile is the single most important object in the Pascal environment (Figure 1.2). The commands available at the Command level access the workfiles as shown in Figure 1.2. To input a Pascal program the Editor builds a SYSTEM.WRK.TEXT file by accepting keyboard input and writing the lines of text to the workfile. The Filer may be used to erase the workfile, transfer it to another file, etc.

The Compiler takes text from the .TEXT workfile and translates it into P-code, which is written to the .CODE workfile. The Linker combines other P-code units with the SYSTEM.WRK.CODE file in preparation for the eXecutor program. (While the Pascal environment defaults to the workfile to process the commands we give it, we can also divert it to other files in the system.)

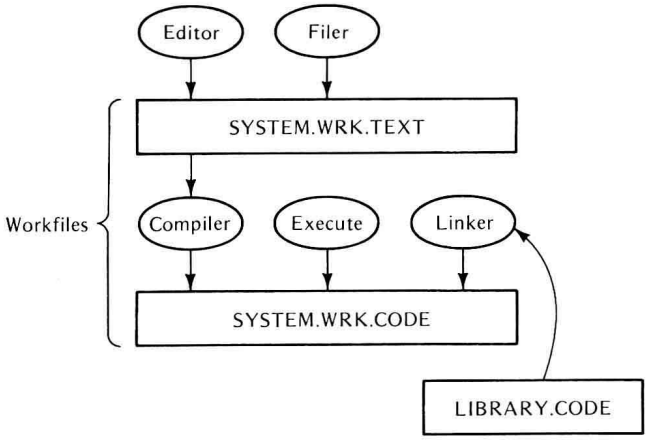


Figure 1.2. The Workfile: SYSTEM.WRK

The Pascal System is a language and a set of tools for developing programs, debugging them, and running them. The Pascal System is mostly independent of the **machine** environment used to run Pascal programs because of P-code. Therefore, we turn our attention to P-code before exploring the Pascal system in more detail in Chapter 2. Since knowledge of P-code is not essential to understanding Pascal, the reader may choose to skip the next section.

1.3 PASCAL THE MACHINE

One of Wirth's goals was to construct the Pascal System so that it could be easily implemented on a variety of computers. Clearly, the differences among computers are large and, if implemented *directly* on each machine, would consume years of intense labor. Instead, Wirth cleverly implemented the Pascal System on a hypothetical machine called the **P-code machine.**

The P-code machine did not exist when the first Pascal System was developed. Instead the Pascal System needs a P-code simulator before it can work. We must either turn every existing machine into a P-code machine or else build new machines to interpret the P-code produced by the Pascal System. The prospect of selling a P-code machine to everyone who wanted Pascal did not seem like an attractive alternative, so a series of **P-code machine simulators** was developed instead.

At University of California at San Diego, a group headed by Kenneth Bowles initially developed P-code simulators for LSI-11 microcomputers. Later, other simulators (8080, Z80) were developed so that the Pascal System could be

run on low-cost microcomputers. These simulators were mainly responsible for the widespread acceptance of Pascal.

The UCSD system developed by Kenneth Bowles is actually a dialect of the original Pascal System. Many features have been changed or eliminated because of the problems unique to such restricted machines. Thus the UCSD Pascal System is only an approximation of the full Pascal System as originally implemented on a large machine. Successive versions of UCSD Pascal are released under different numbers to indicate an increasingly powerful version.

The P-code simulator employed in the Apple computer uses a very different set of P-code "instructions" from the original. It is not necessary to understand P-code in order to use the Pascal System, but it may help to explain the internal behavior of the Pascal System if the general idea of P-code is understood.

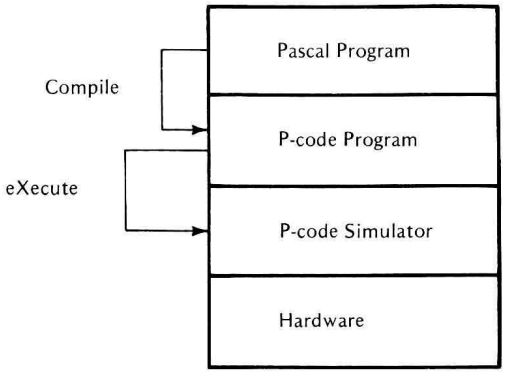


Figure 1.3. Relationship Between P-code and Pascal.

The P-code "machine" simulates a 16-bit word *stack machine*. A stack is a memory which allows push and pop operations as well as data processing operations like add, subtract, and move. A push loads (copies) a word from one location in the stack onto the top of the stack (tos), as shown in Figure 1.4. A pop copies a value from the tos (top of stack) word and places it in some other word in memory.

Figure 1.4 also shows a HEAP for storing lists (discussed later) and the relative locations in main memory of the P-code simulator and the Pascal system. Note that memory overflow occurs whenever the HEAP grows upward to meet the downward-growing stack.

We can demonstrate how a stack operates by example. Suppose a Pascal program for adding and subtracting is translated into stack operations:

```
X := ( A + B ) / ( C - D ) ;   (* Pascal statement *)
```

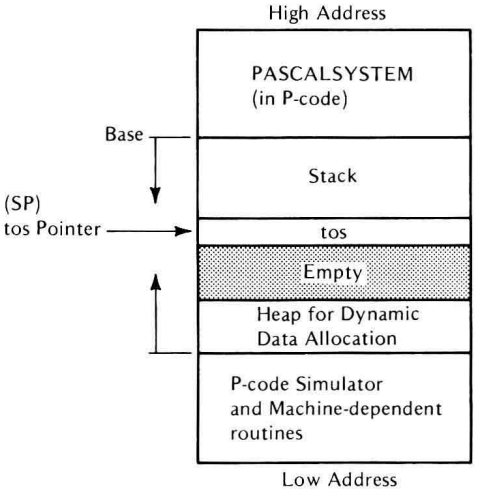


Figure 1.4. The P-code Stack

The corresponding stack operations are:

```
PUSH  A  ;  load (tos) with A
PUSH  B  ;  load (tos+1) with B
ADD      ;  (A+B) is put in (tos)
PUSH  C  ;  load (tos+1) with C
PUSH  D  ;  load (tos+2) with D
SUB      ;  (C-D) is put in (tos+1)
DIV      ;  (A+B)/(C-D) is put in (tos)
POP   X  ;  (tos) is put in X.
```

These operations are managed by a special P-code register called the SP (stack pointer). The SP **points** to the current tos (top of stack) word in the stack. Figure 1.5 shows what happens when the stack operations above are performed.

In the actual P-code machine there are several versions of the PUSH and POP instructions depending on the type of values manipulated and the location of the variables A,B,C,D, and X. We will translate X:=(A+B)/(C-D) into P-code by assuming **global** locations for integers A,B,C,D, and X. This allows us to use LDO to push a word onto the stack, SRO to pop a word, ADI (add integer), DVI (divide integer), and SBI (subtract integers). The Pascal example is then converted into the following P-code.

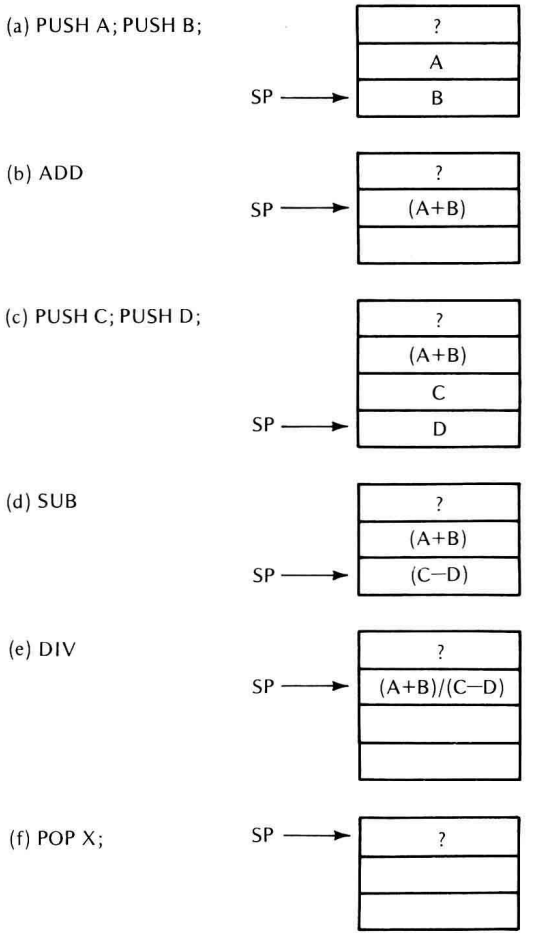


Figure 1.5. Interpret X:= (A + B) / (C - D)

Mnemonics	Operand		Comment
LDO	A	;	push A
LDO	B	;	push B
ADI		;	(A+B)
LDO	C	;	push C
LDO	D	;	push D
SBI		;	(C-D)
DVI		;	(A+B)/(C-D)
SRO	X	;	X:=

The simple example above disguises many of the elaborate instructions in P-code designed to address a variety of locations in the P-code memory, handle a variety of data types (integers, real, characters), and handle segments, procedures, and functions in Pascal. However, it does reveal an interesting substructure of the Pascal System.

Since all Pascal Systems can run on top of a P-code simulator, we can move a Pascal System from one machine to another by "implementing" the P-code simulator on any new machine. The simulator is much easier to implement than the entire Pascal environment. This is the main advantage of Wirth's (and subsequently Bowles') plan. The disadvantage, of course, is that some performance is lost due to the slow simulation process.

We are now equipped with enough knowledge to use the Pascal System. Each box at the Command level can now be opened and studied. This will allow us to enter, modify, examine, and run Pascal programs.

TWENTY QUESTIONS

Use this test to review your comprehension of the ideas and facts of Chapter 1.

1. Who invented Blaise Pascal?
2. Who invented the programming language Pascal?
3. Why did the inventor name the language after the person?
4. What was Jansenism?
5. What was the first purpose of the Pascal language?
6. What do we mean by the Pascal Environment?
7. What does the C(omp command do?
8. What is the purpose of the L(ink command?
9. Why is the E(dit command used?
10. What is the form of a file name?
11. What command is used to copy a file?
12. What three commands are combined by R(un?
13. What is the file name of the workfile?
14. Is P-code a machine or a program?
15. In Figure 1.3 the comments are enclosed in . . . . . ?
16. The push and pop operations use a . . . . . memory.
17. The abbreviation "tos" means . . . . . ?
18. What is the main advantage of P-code?