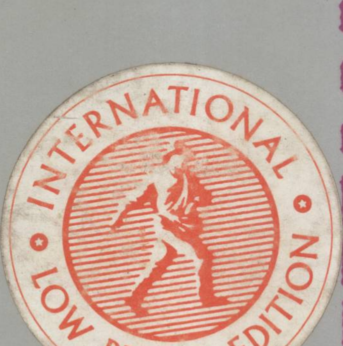
Brady

EDITORS OF DR. DOBB'S JOURNAL

DR. DOBB'S TOOLBOOK OF 68000 PROGRAMMING



TP31
D632

8763923

Dr. Dobb's Toolbook of 68000 Programming

The Editors of
Dr. Dobb's Journal of Software Tools

华南工学院
图书馆藏书

E8763923

A Brady Book
Published by Prentice-Hall Press
New York, New York 10023

Dr. Dobb's Toolbook of 68000 Programming

Copyright © 1986 by M & T Publishing, Inc.
All rights reserved
including the right of reproduction in whole or in part in any form.

A Brady Book
Published by Prentice Hall Press
A Division of Simon & Schuster, Inc.
Gulf + Western Plaza
New York, NY 10023

PRENTICE HALL PRESS is a trademark of Simon & Schuster, Inc.

Manufactured in the United States of America

1 2 3 4 5 6 7 8 9 10

Library of Congress Cataloging-in Publication Data

Dr. Dobb's toolbook of 68000 programming.

"A Brady Book."
1. Motorola 68000 (Microprocessor)—Programming.
I. Dr. Dobb's journal of software tools for the professional programmer. II. Title: Doctor Dobb's toolbook of 68000 programming.
QA76.8.M67D72 1986 005.265 86-25308
ISBN 0-13-216649-6 (case)
ISBN 0-13-216557-0 (paperback)

Special volume discounts are available by contacting the
Special Sales Department, Englewood Cliffs, NJ 07632.

For information about foreign rights,
contact M & T Publishing, 501 Galveston Drive, Redwood City, CA 94063.

Educational Computer Board, Tutor, MACSbug, 68000, 6800, 6500, 68008, 68010, 68020, 68881, 68851 and 68452 are trademarks of Motorola, Inc. 8080 and 8086 are trademarks of Intel Corp. Macintosh, Mac Plus and RMaker are trademarks of Apple Computer, Inc. Unix is a trademark of AT&T Bell Laboratories. CP/M is a trademark of Digital Research, Inc.

Software Availability

All the software listings in this book are available on disk in the following formats: Amiga, Atari 520ST, CP/M 8-inch, Macintosh, MS-DOS and Osborne.

In addition, the 68000 Cross-Assembler (Chapter 10) can be putchased as an executable program along with source code and documentation. Systems required to operate the assembler are one or more disk drives and either CP/M-80, CP/M 2.2 with 64K of memory or MS-DOS with 128K of memory.

Order Listings Disk or Assembler Disk by sending a check, or credit card number and expiration date, for $25.00 (each) to:

68000 Disk
c/o *Dr. Dobb's Journal of Software Tools*
501 Galveston Dr.
Redwood City, CA 94063
415/366-3600

Please remember to specify disk format. California residents must add appropriate sales tax.

Limits of Liability and Disclaimer of Warranty

The author(s) and publisher of this book have used their best efforts in preparing the book and the programs contained in it. These efforts include the development, research and testing of the theories and programs to determine their effectiveness. The author(s) and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author(s) and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance or use of these programs.

Dr. Dobb's Journal of Software Tools

Publisher	*Laird Foshay*
Editor-in-Chief	*Michael Swaine*
General Manager, M & T Books	*Ellen Ablow*
Technical Editor	*Nicholas Turner*
Book Editor	*Lisanne Abraham*
Design	*Michael Hollister*
	Joe Sikoryak
Production	*Frank Pollifrone*
Illustrations	*Al McCahon*

此为试读，需要完整PDF请访问：www.ertongbook.com

8763923

Table of Contents

Introduction

Many people in the microprocessor industry think that the assembly language programming community is divided into two groups: the sixers and the eighters. The "sixers" are people who like the CPU chips that start with the number 6—the 6500, 6800 and the 68000 families. The "eighters" like the chips starting with 8—the 8080, 8086, 8088, Z-80, 80286 and 80386. The sixers support instruction sets that are very general, with lots of addressing modes available and a lot of identical CPU registers that each support all of the addressing modes. Typical sixer machines use memory-mapped I/O and support a large, flat memory space. Conversely, the eighters seem to like CPUs with a few very specialized registers and a lot of powerful special-purpose instructions. Typical eighter machines use special instructions for I/O and support a segmented memory architecture.

There seems to be a deep-seated philosophical difference between the two groups, and great wars of words have taken place between the various factions of the sixers and the eighters. The pages of *Dr. Dobb's Journal of Software Tools (DDJ)* have long been a battleground for those wars. In this book, which contains articles originally printed in *DDJ,* along with some new material, we present information of interest to one of the most active factions in the Great Microprocessor Debates: the sixer folks who are fans of Motorola's 68000 family.

I'm an unabashed sixer from way back. I cut my microprocessor teeth on the 6502, and spent three years at Atari programming video games on the 6507 in the VCS game machine. When Motorola first introduced the 68000, I was delighted. Here at last was a machine with a huge address space, nice clean memory-mapped I/O, a lot of general-purpose registers, and an instruction set that applied equally (well, almost) to all the registers. What's more, the future seemed to promise more powerful chips in the same family with nearly identical instruction sets. A sixer's dream!

Since then my love for the 68000 family has grown steadily. Apple's Lisa and then the Macintosh started what was to become a whole wave of 68000 machines. Now we have the Amiga (what a glorious computer!), the Atari ST and a whole plethora of systems based on the VME bus, one of the most sensible and well-supported bus standards I've ever seen. The 68020, the latest chip in the family, has proven to be a real powerhouse. Combined with the 68881 floating-point coprocessor and the 68851 memory management unit, a 68020 system can easily outperform many of today's minicomputers and most of the mainframes of the 1970s.

Nicholas Turner
Technical Editor

I

Introduction to the 68000 Family

此为试读,需要完整PDF请访问: www.ertongbook.com

1

MC68000 Family History, Design and Philosophy

Daniel Appleman

In this introduction to the MC68000 family of microprocessor and support chips, Daniel Appleman describes its history, summarizes the features of the processors and examines the important differences between the 68000 family and the Intel family of processors.

Motorola's 68000 family, once little more than descriptive literature, has matured into a full selection of microprocessors and support chips. The original MC68000 has been followed by the 68008, 68010 and 68020, not to mention dozens of support and peripheral chips. Although these microprocessors differ in speed, memory addressing range and other details, they are based on common operating principles. These principles emerged in the late 1970s, with Motorola's introduction of the original 68000 microprocessor.

Gaining an understanding of the philosophy behind the 68000 family can shorten the learning time required to become an expert 68000 programmer. Whether or not you have had 68000 experience, the background provided in this chapter will help you understand the applications and examples in the rest of the book.

Of Space, Speed and Support

In 1979 the personal computer revolution was just beginning. Businesses were discovering that minicomputers could make some of their operations more efficient, but computer-aided design and large business applications were still confined to mainframes.

In the world of small computers, the 8-bit microprocessor was king. Because they were well-suited for single-user, small- and moderate-size applications, 8-bit processors were the most common processors in home computers. Their limitations—64-kilobyte address space, poor support for high-level languages and poor multiuser support—were largely irrelevant in the home market. Although these machines were initially successful in certain business applications, several factors soon sped them on their way to obsolescence in the business world.

The most obvious factor was the rapid advance of technology. Integrated circuit (IC) manufacturers were approaching VLSI (very large scale integration) technology—the ability to reliably place more than 100,000 transistors in a single IC. By itself, this ability would probably not have assured the development of 16-bit microprocessors. The big push came because semiconductor manufacturers identified a large potential market for the 16-bit chip—namely, the same businesses and laboratories that were using minicomputers and mainframes. The 8-bit processors could not compete with minicomputers in those areas; not only did the 8-bit machines suffer from the limitations already mentioned, but they were also too slow.

A 16-bit processor is not simply twice as fast as an 8-bit machine. Determining which computer would be faster in a certain application, a process called *benchmarking,* requires a fairly complex examination of cycle times, types of instructions and the actual program in question. For the purpose of demonstration, consider the example of adding one 16-bit word to another in memory. Say there are 4 bytes of memory called A, B, C and D. We'd like to add the 2-byte integer in A and B to the 2-byte integer stored in C and D and put the result in C and D.

On an 8-bit microprocessor like the 6800, the code to accomplish the task would look something like this:

```
ldaa D      ;Move D to Accumulator
adda B      ;Add B to Acc
staa D      ;Store Acc to D
ldaa A      ;Move A to Acc
adca C      ;Add C (with carry from D+B)
staa C      ;Store Acc to C
```

This code fragment would take 4+4+5+4+4+5=26 clock cycles to execute, which is the same as 13 microseconds if we assume a 2 megahertz (MHz) clock.

On a 68000 chip, the code would look like this:

```
mov.w       A,d0   ;Move A and B to register d0
add.w       d0,C   ;Add to C and D and store result
```

This requires 12+16=28 clock cycles, which means 3.5 microseconds if we assume an 8 MHz clock. The newer 68000 processors run at an even faster 12.5 MHz, which would allow the above operation to take place in 2.24 microseconds.

Differences in execution speed become even more profound when comparing more complex operations, such as 32-bit addition, multiplication and division. Although 8-bit machines will be around for years in low-end computing and many device-control applications, the 16-bit processor is taking over for high-end applications.

Unfortunately, increased speed is not the only consideration in designing a 16-bit microprocessor; the same applications that form the market for 16-bit chips also demand minicomputer-like features and architecture. Fortunately, the chip designers kept this fact in mind, and the results can be seen in most high-end microprocessors. The implementations vary from company to company, but the goals are largely the same for all. The ideal 16-bit microcomputer:

- offers a clear upgrade path to 32-bit processors and advanced superminicomputer architectures;
- provides good support for high-level languages and advanced multitasking operating systems;
- easily interfaces with a wide variety of peripheral chips and supports multiprocessor applications.

A Clear Upgrade Path

In the late 1970s, prospective manufacturers of advanced microprocessors sent marketing representatives to virtually any company that expressed interest in these products. Computer design engineers had to choose from among the various upcoming microprocessors without the benefit of having used them. And the technical considerations were not the only problems —engineers also had to consider the long-term future of each microprocessor. Would the company be around in five or ten years? Would it provide the advanced development tools needed to work with these devices? Would there be multiple sources for the product? Would future chips in the family be compatible with the current chip?

The last question was perhaps the most important of all. If a new microprocessor in a family was not hardware-compatible with the original, each new device—as well as its peripherals—would have to be redesigned to be incorporated into a system. (It is therefore no surprise that Motorola and Intel, the two main chip contenders, each specified its own system bus standards. The VME bus, specified by several manufacturers, including Motorola, supports the 68000 family; the Multibus II from Intel is closely tied to the 8086 family.)

Another issue raised during the design of the new processors asked to what degree the new 16-bit devices should be compatible with the existing 8-bit devices. Because several years would pass before there was a large family of 16-bit peripherals, all of the 16-bit processor manufacturers designed machines that supported 8-bit peripheral chips.

In designing the new 16-bit devices, Motorola and Intel took entirely different approaches to software compatibility with the 8-bit world. Intel attempted to expand the existing 8080 family (registers on the 8086 can be accessed either as 16-bit registers or as dual 8-bit registers), hoping that the ease of converting 8-bit software to the 8086 would win many new microcomputer designs. Motorola took a calculated risk. Since the new applications for the 16-bit devices were likely to be in markets that were not currently supported on 8-bit machines, the company decided to develop a software instruction set from scratch.

The results of those decisions have been mixed—the market seems to have split into two clearly divided factions. While a vast array of software has been written for the very successful IBM PC and the PC-compatible "clones" (the premier 8086 chipset machines), there have been very few truly new designs using the Intel chipset. At the same time, many new machines that use the 68000 chipset have emerged. Apple's Macintosh, the Commodore Amiga, the Atari ST, the Sun Microsystems and Apollo work stations and many others fall into this category. The supporters of the 68000 family tend to be quite fanatical. Why? Typically, the response from a programmer refers to the clean, uniform design of the chips in the family and to the ease of programming. Because the 68000 was designed from scratch, no compromises were necessary to ensure downward compatibility. At the same time, the family was designed with upward compatibility in mind, so that the family's later, more powerful chips would not be crippled by the necessary compatibility with earlier models.

Supporting High-Level Languages and Advanced Operating Systems

While hardware technology was advancing to the point at which 16-bit computers could be implemented on a single chip, software technology was also making strides. Newer languages such as C and Pascal were gaining in popularity for general programming, and specialized languages—APL, Lisp, Prolog and Forth, for example—were becoming more common. Most 8-bit machines, however, could only be programmed in machine language, BASIC and, occasionally, Forth and Pascal. The 8-bit 6500 series processors had only 256 bytes in their stack, hardly enough memory to support the sophisticated parameter-passing and local-variable capability provided by most high-level languages.

Why did high-level languages become so popular? As software development costs became significantly greater than hardware costs due to the increased sophistication of the applications demanded by the marketplace, any tools that enabled a programmer to be more efficient (generate more code in a shorter time) were quickly embraced by the computer industry and most programmers. Assembly language programming was relegated to extremely size- or speed-critical applications. Most major applications were written in the newer languages or in combinations of high- and low-level languages. With the advent of C, even operating systems could be written efficiently in a high-level language (Unix is the primary example).

The increasingly sophisticated applications required increasingly sophisticated operating systems to support them. New word processing programs demanded fast updates and the ability to handle enormous documents. Advanced databases, which had previously been restricted to large mainframes, were being rewritten for small computers. These applications demanded more memory and more speed than most 8-bit machines could provide. Furthermore, 8-bit machines are almost exclusively single-tasking systems meant for one user; sophisticated operating systems must support multiple users and multiple tasks.

Multiuser systems themselves introduce another demand that 8-bit machines can't handle: protection schemes. Imagine a system being used by two people—the first is running a crucial accounting application and the second is debugging a machine language program. In the course of this debugging, the second user accidentally runs the microprocessor's HALT instruction. In an unprotected system, the first user will probably lose all of his or her work. None of the major 8-bit processors implements a security scheme that protects against accidental or intentional operations that can stop the system or scramble data.

The new machines, however, support large address spaces and virtual memory schemes, and have instruction sets designed to run high-level languages, complex applications and operating systems efficiently. (See Chapter 2.)

Interfacing With Peripherals and Supporting Multiprocessors

Most microcomputer systems in the late 1970s were based on the original work by John Von Neumann. In the Von Neumann architecture, a central processing unit (CPU) was connected to peripheral chips and memory by way of a main computer bus. (See Figure 1.1.)

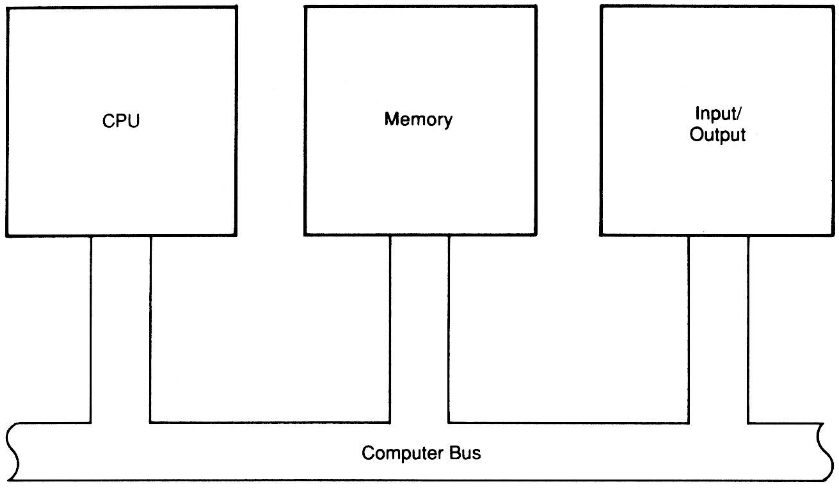


FIGURE 1.1 *The Von Neumann architecture.*

The processing speed possible with such a system is limited by the amount of data that can be transferred over the main computer bus (this limitation is often referred to as the "Von Neumann bottleneck"). A simple technique for improving system performance is to make the peripheral device intelligent by adding a microprocessor. This improvement allows some of the I/O processing to take place in the peripheral device or chip, thus freeing the main processor (and the bus) from this task. Another technique is to have a direct memory access (DMA) device take over the bus from the main CPU and transfer data at a much higher rate than was possible with the CPU alone. Both of these techniques are quite common on 8-bit systems.

A more sophisticated technique which is becoming more common is multiprocessing. At the simplest level, two or more CPUs share a bus with a bus arbitration mechanism. In more sophisticated schemes, each CPU has a local bus and local resources which can then tie in together to one or more main system buses. (See Figure 1.2.) Because multiple CPUs execute simultaneously, more work gets done in the same amount of time and the effective processing speed is greater.

Features of the 68000 Family

The original 68000 processor family provided by Motorola had four main members (others have since been added for specialized applications). As Table 1.1 shows, the 68000, 68008 and 68010 are almost the same chip.

The 68008 is identical to the 68000 except for the size of the data bus and address bus. It was designed for applications in which the system could use the power of a 16-bit machine, but could not justify the expense of the 16-bit support hardware. The 68010 is an improved version of the 68000 that supports virtual memory. From a programming point of view, the two chips are almost identical. The 68020, on the other hand, incorporates major improvements in both the hardware and the instruction set. And it maintains full compatibility with the other chips and will run their object code directly. Figure 1.3 shows the major structural differences among the processors within the family.

The 68000 doesn't look like a 16-bit processor. True, it has a 16-bit data bus and a 16-bit arithmetic logic unit (ALU), but all of the registers are 32 bits wide. This increased register size is one of the most important ways in

	68000	68008	68010	68020
Registers	17	17	20	23
ALU Width	16	16	16	32
Address Bus	24 bits	20 bits	24 bits	32 bits
Address Space	16 Mb	1 Mb	16 Mb	4 gigabytes
Data Bus	16 bits	8 bits	16 bits	8/16/32 bits
Data Registers	32 bits	32 bits	32 bits	32 bits

TABLE 1.1 *The 68000 family.*

此为试读，需要完整PDF请访问：www.ertongbook.com