

---

# Chapter 0

---

## Introduction

The microprocessor has brought about a revolution in computing. Whereas twenty or thirty years ago, computers were so expensive that only large organizations could afford them, the microprocessor has brought computers not just into the small company, but also into the home, the small office, and even into industrial and consumer equipment.

At this time, microprocessors are broken down into two major categories: CISC and RISC.

CISC microprocessors are *Complex Instruction Set Computers*. These are the traditional processors, which can execute a relatively large number of fairly complex instructions. RISC processors, on the other hand, are *Reduced Instruction Set Computers*, which can perform a smaller number of relatively simpler instructions. Right now, there is some disagreement about which are better. RISC processors, because they do less in each instruction, can be built so they do it faster. But this is outweighed to some extent by the fact that a larger number of instructions is needed to finish a particular job, and so it is not quite so clearcut which is faster on complex jobs.

In the CISC world, there are two major companies building microprocessors: Intel and Motorola. Intel was the very first microprocessor manufacturer, and Motorola was the second, and they have been the two major players ever since. Both make very capable products, and are forever playing leapfrog, trying to best each other. At any given time, one or the other may be out in front.

Although Intel is the more widely known microprocessor manufacturer - mainly because their 8088 and its followers were chosen by IBM for their PC line - many people prefer Motorola processors. Intel has attempted to make each processor they make somewhat compatible with earlier processors, with the result that even their newest processors have many features which seem archaic by today's standards. Motorola, on the other hand, made a major departure from earlier processors when they designed the

68000 - they tried to keep the "flavor" of their earlier machines, but they did not keep any of their limitations. Thus many users agree that the Motorola 68000 is easier to use - and certainly more pleasant to use - than the comparable Intel processors.

When higher level languages are used, it doesn't much matter to the programmer which processor they will eventually run on. But to get the utmost speed or compactness out of a program requires assembly language, and that is where programmers most often choose Motorola processors, which are much more orderly in their structure and easier to program on that level.

This book is about the 68000 Microprocessor and how it is used. We will cover many details of both its internal operation, the external circuitry it connects to, and the programs which it runs. Although Motorola makes more advanced 68000-family processors (such as the 68020 and 68030), almost all of the principles you will learn about the 68000 apply to those as well - they are merely extensions of the basic 68000 architecture.

When you look at the heavy-duty number-crunching applications of microprocessors, you will find the Motorola 68000 family used more often than any other. Even though the 68000 may be one of the slower microprocessors in the series, it is no slouch.

The 68000 is the microprocessor that gives the Atari ST, Commodore Amiga, and the Macintosh SE their power. It is also found inside many laser printers, as well as in industrial controllers and scientific workstations. The 68000 is roughly in the middle of what many call the '68K' family of processors - the 68008 is slightly below, the 68020 and 68030 are above. (A fifth processor, the 68010, is theoretically faster than the 68000, but the 68000 can be run at faster clock rates and so is just about equal in practical speed.)

Our method for teaching you about the 68000 is to have you build and use an actual system.

In Volume I, you will concentrate on hardware. You will start from the very beginning, mounting one part after another, until you have a complete system up and running. As we go, we will discuss each part of the system, see where it fits into the overall picture, build it, and then test it.

In Volume II, you will then use the hardware to learn about software. Using the HUMBUG ROM-based debugger, and the SK\*DOS disk operating system, we will progressively do some simple programming exercises, and ultimately look at various portions of the HUMBUG and SK\*DOS software code itself to examine how 68000 programs solve various programming problems, and how to interface the software with the hardware.

Above all, *don't try to skip ahead in the book*, because the treatment is logically organized in a progression. If you skip some part as you go, you will find yourself missing some crucial knowledge you would need later.

With that, let us continue to Chapter 1, to learn about the SK68K computer you will be building.