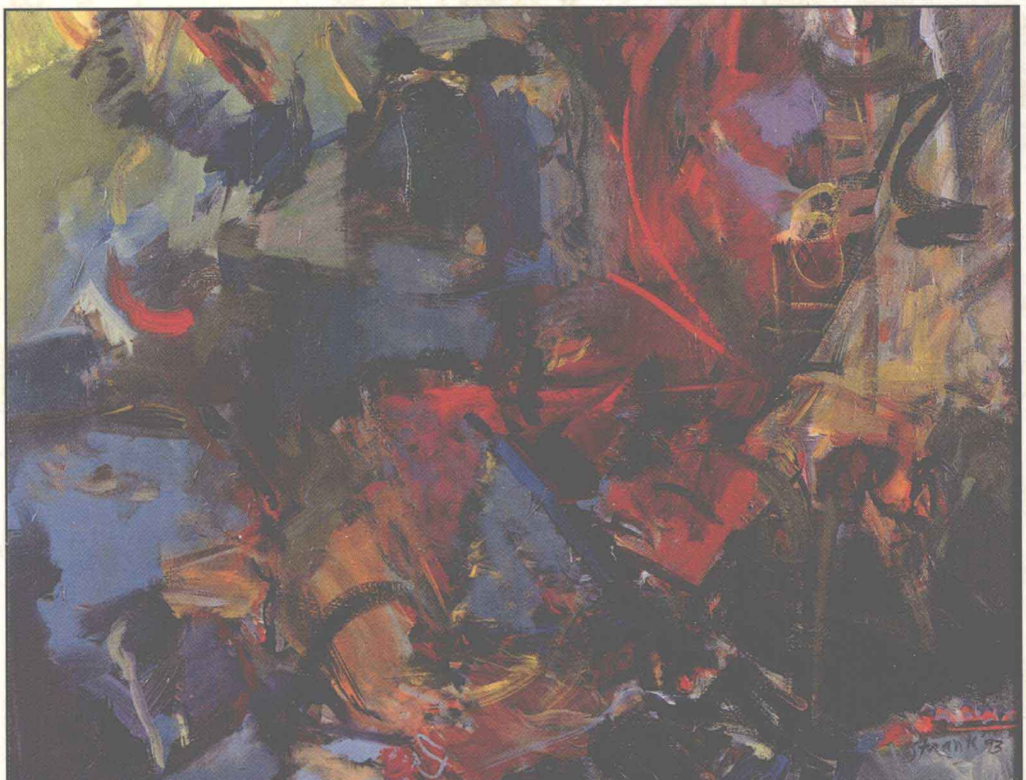
The 80x86 Family

Design, Programming, and Interfacing

SECOND EDITION



JOHN UFFENBECK

The 80x86 Family

Design, Programming, and Interfacing

Second Edition

JOHN UFFENBECK
Wisconsin Indianhead Technical College

Prentice Hall
Upper Saddle River, New Jersey   Columbus, Ohio

Library of Congress Cataloging-in-Publication Data

Uffenbeck, John E.
The 80x86 Family : design, programming, and interfacing / John Uffenbeck. — 2nd ed.
p. cm.
Rev. ed. of: The 8086/8088 Family : design, programming, and interfacing. c1987.
Includes index.
ISBN 0-13-362955-4
1. Intel 80x86 series microprocessor. I. Title.
QA76.8.I29U34 1998
004.165—dc21 97-3741
CIP

Editor: Charles E. Stewart, Jr.
Production Editor: Rex Davidson
Design Coordinator: Julia Zonneveld Van Hook
Cover Designer: Julia Zonneveld Van Hook
Cover Art: © Frank Shirley/SuperStock
Production Manager: Deidra M. Schwartz
Marketing Manager: Debbie Yarnell
Production Supervision: Custom Editorial Productions, Inc.

This book was set in Times Roman and Helvetica by Custom Editorial Productions, Inc., and was printed and bound by R.R.Donnelley & Sons. The cover was printed by Phoenix Color Corp.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-362955-4

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S. A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Preface

This is the second edition of a book previously titled *The 8086/8088 Family: Design, Programming, and Interfacing.* As the new title suggests, it has been expanded to include coverage of all of the 80x86 processors, from the 8-bit 8088 to the 16-bit 8086 and 80286 and the 32-bit 80386, 80486, Pentium, and Pentium Pro processors.

As in the first edition, this edition is more than a survey of Intel microprocessor chips. Think of these processors as the *vehicles* that will allow you to explore the real world of microcomputer technology. To appreciate the trip, you should be familiar with digital logic circuits and the binary and hexadecimal numbers systems. In addition, a familiarity with DOS and Windows will come in handy when the software features of the processors are explored.

Philosophy

If this book has an underlying philosophy, it is to stand back and observe the microprocessor as one component in a *microcomputer system*. In the case of the 80x86 processors, that system—thanks to IBM—has become known as the *PC* or *Personal Computer.* This philosophy is embedded throughout the book. Particular attention has been paid to developing examples based on the PC architecture. The software chapters, for example, use DEBUG, a utility commonly available with MS-DOS for program development. The program examples are all designed to run on a PC and utilize calls to the BIOS services and MS-DOS functions of the PC.

The hardware chapters focus on chips and circuits used in the PC. Chapter 8, for example, describes the PC's parallel printer interface. Software drivers for this circuit using programmed I/O and interrupt-driven I/O are also included. Chapter 9 includes a detailed description of the 8259A PIC (Programmable Interrupt Controller) still used by the PC today (in integrated form). Chapter 10 focuses on serial I/O and provides detailed coverage of the 16550 UART (Universal Asynchronous Receiver/Transmitter). Also included in this chapter is a description of the popular modem standards and the Hayes AT command set. Chapter 11 describes the architecture of the PC and the different bus systems, including ISA, EISA, MCA, VESA, PCI, SCSI, and USB.

Organization

Each chapter presents a consistent interface to the reader. The *Outline* lists the major sections of the chapter, followed by a list of *Objectives*. The chapter *Overview* gives a quick look at the chapter and tries to answer the question, "Why is this chapter important?" Each section is numbered (1.1, 1.2, 1.3, and so on) and includes a brief introduction that reiterates the objectives to be covered. To measure your understanding, a set of *Self-Review* questions (with answers) is also provided.

Each chapter ends with a *Self-Test,* typically twenty questions requiring short answers readily found in the text. More thought-provoking questions are included in a separate section entitled *Analysis and Design Questions*. These are keyed to the appropriate section in the chapter.

Changes from the Previous Edition

This new edition represents a major redesign of the original textbook. A brief summary of these changes follows.

- Chapter 1 has been split into two chapters. The first provides a history of computing from ENIAC to the present day. Also included is an overview of each processor in the 80x86 family. The second chapter provides a review of the binary and hexadecimal number systems and a description of various computer codes. Computer programming is also explained as well as computer operating systems (updated to include Windows).
- Chapter 2 of the first edition has also been split into two new chapters. One focuses on the specific architecture of each 80x86 processor, the other provides an overview of the 80x86 processor instruction sets. New in this chapter is a description of the MS-DOS BIOS services and function calls.
- The software chapters of the first edition have been completely redone. Two chapters are now provided. Chapter 5 illustrates 80x86 programming techniques. Seven program examples are provided in that chapter, each developed using a unique outline approach and coded using DEBUG. Chapter 6 explains 80x86 assembly language using Microsoft's Programmer's Workbench. Techniques for creating COM and EXE files are provided as well as the complete design of a simple game program.
- Chapter 7 on memory has been updated to include flash memory, SIMMs and DIMMs, EDO RAM, synchronous DRAMs and SRAMs, and PAL address decoders.
- The input/output coverage has been expanded to two chapters. Chapter 8 covers parallel I/O and programmed I/O techniques. Chapter 9 covers interrupt driven I/O and the 8259A PIC. DMA techniques are also included in this chapter.
- Chapter 10 on data communications now includes coverage of the 16550 UART and common modem standards. The AT command set is also covered.
- Chapter 11 of the first edition has been replaced with a new chapter that describes the architecture of the PC and the common bus systems. This includes the ISA and extended ISA buses, the microchannel architecture, and the EISA, VESA, and PCI buses. Two I/O buses are also covered, SCSI and USB.

Included Disk

In your book you will find a diskette that includes the assembly listings for all of the programs in the book. These are organized by chapter, with the figure name used as the program name. In addition, you will also find a copy of *DEBUG32* on this disk. This is an enhanced version of the popular DEBUG utility supplied with MS-DOS. It allows full access to the 32-bit registers and addressing capabilities of the 80x86 processors. In addition, it can also be used for debugging protected-mode programs. A special thanks to Rob Larson and Michael Schmit of Quantasm Corporation for their permission to include this program.

Acknowledgments

I would like to thank Charles Stewart and Kim Gundling of Prentice Hall, Inc., who encouraged me to resurrect this book nearly ten years after the first edition appeared. I would also like to thank Rex Davidson, also of Prentice Hall, and Jim Reidel of Custom Editorial Productions, Inc., the two editors who assisted me with the final production of this book. These are the people I spent the most time with, especially Jim, who had to put up with my last minute changes!

I am grateful to Intel Corp., Texas Instruments, Inc., Advanced Micro Devices, National Semiconductor Corp., Quantasm Corp., BYTE Publications, Hitachi America Ltd., and Addison Wesley Longman Ltd. for permission to include many of the data sheets and technical drawings presented throughout the book. I would also like to thank Al Subera of ADS Photography for many of the photographs that illustrate the book—and the Smithsonian Institution for the chapter-opening photographs that highlight significant computers from the past.

Finally, I want to give my special thanks to my students at Wisconsin Indianhead Technical College. Instead of using a "real" book, these folks have had to lug around a manuscript for the last several years. Their questions and comments have been available to me as I prepared this new edition.

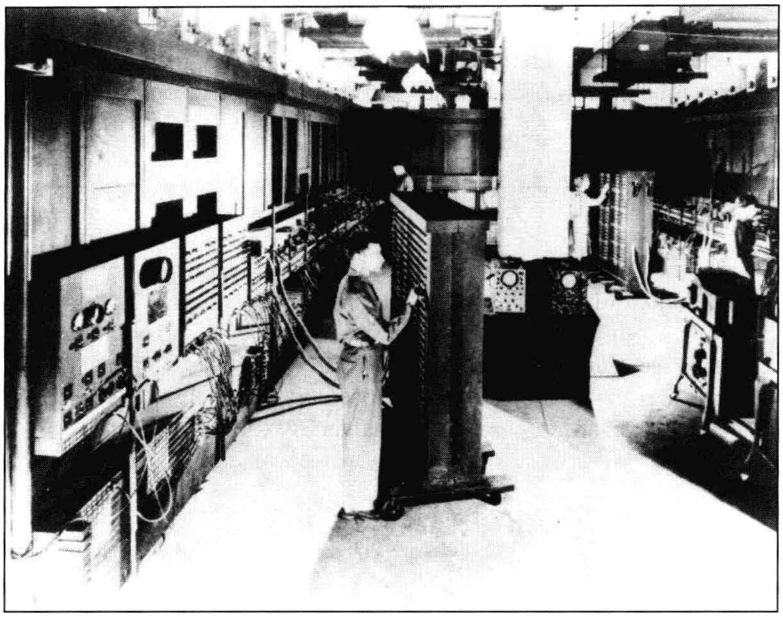
Final Thoughts

I first started teaching microprocessors at Hartnell College in 1976. We used a "briefcase" computer-trainer based on the 8080, called the MST-80. Microcomputer technology has come a long way since those days. To compare the Pentium Pro with the 8080 is to compare the Space Shuttle with the Wright Brothers' first powered airplane. Yet somehow we must come to grips with this incredible, rapidly accelerating technology. It is exciting, but frightening. Take a summer off and you may find yourself hopelessly behind. I hope you find this book useful as you attempt to hang on to the fast-moving technology train.

John Uffenbeck
Wisconsin Indianhead Technical College

Contents

1 Microcomputers and Microprocessors



One of the first practical computers was ENIAC (Electronic Numerical Integrator and Computer). Built in 1945, it contained nearly 17,000 vacuum tubes, weighed more than 30 tons, and required 1500 square feet of floor space. It is interesting to note that in 1949, *Popular Mechanics* magazine predicted that "computers in the future may perhaps only weigh 1.5 tons!" (Photo courtesy of Smithsonian)

Outline

Objectives

After completing this chapter you should be able to:

1. Draw a block diagram of a stored program computer.
2. Explain the fetch and execute processing cycle of a stored program computer.
3. Define the role of the data, address, and control buses in a stored program computer.

4. Trace the evolution of the computer from the vacuum tube era to the microprocessor.
5. Identify significant computers that have been built over the years.
6. Explain the difference between a microprocessor, a microcomputer, and a microcontroller.
7. Compare digital signal processors (DSPs) with conventional microprocessors.
8. Trace the evolution of Intel microprocessors from the 8086 through the Pentium Pro.
9. Compare the bus widths and internal register sizes for all of the processors in the 80x86 family.
10. Explain the difference between the 80x86 processor's Real, Protected, and Virtual 8086 modes of operation.
11. Explain the difference between a microprocessor second source and a clone.

Overview

This chapter presents a core of digital computer principles upon which the following chapters can build. In it you will find a brief introduction to much of the terminology you will be reading about in the later chapters. You will also learn about the evolution of the computer from the vacuum tube-based ENIAC to Cray supercomputers. The chapter concludes with brief descriptions of the microprocessors in the Intel 80x86 family. These descriptions will be expanded upon in later chapters.

1.1 The Stored Program Concept

Introduction

As complex as today's computer systems are, most are still based on a design principle first proposed by Dr. John Von Neumann in 1946. Now taken for granted by most computer users, Von Neumann's idea defined the architecture to be used by all computers for the next 50 years.

In this section we:

- Draw a block diagram of a stored program computer
- Explain the fetch and execute processing cycle of a stored program computer
- Define the role of the data, address, and control buses in a stored program computer

The Stored Program Concept Is Born

ENIAC. One of the first digital computers was a machine called ENIAC (Electronic Numerical Integrator and Computer). It was designed and built in 1946 at the Moore School of Electrical Engineering at the University of Pennsylvania. ENIAC measured over 18 ft. high and was 80 ft. long. It contained nearly 18,000 vacuum tubes, weighed more than 30 tons, and required 1500 square feet of floor space. More than 200,000 man-hours went into its construction (500,000 solder connections alone were required). It was programmed by setting up to 6000 switches and connecting cables between the various units of the computer.

While ENIAC was under construction, Dr. John von Neumann, also of the Moore School of Electrical Engineering, wrote a paper in collaboration with A.W. Burks and H.H. Goldstein that would define the architecture to be used by nearly all computers from that day on.[1] Now called the *stored program concept,* von Neumann suggested that rather than rewire the computer for each new task, the program instructions should be stored in a memory unit, just like the data. The resulting computer would then be *software programmable* rather than *hardware programmable*.

One of the first stored program computers to be built was called EDVAC (Electronic Discrete Variable Automatic Computer). Completed in 1952, it had a memory capacity of 1000 words of 10 decimal digits each. EDVAC was superior to ENIAC because it could be programmed much more efficiently and used a paper tape input device. At about this same time, the first random access core memory appeared. The first generation of computers was now well under way.

The Stored Program Processing Cycle

Fetch and Execute. Figure 1.1 is a block diagram of a basic stored program computer. There are three major parts to this system: (1) The *central processing unit (CPU),* which acts as the "brain" coordinating all activities within the computer; (2) the *memory unit,* where the program instructions and data are temporarily stored; and (3) the *input/output (I/O) devices,* which allow the computer to input information for processing and then output the result.

At one time, the CPU of a computer was constructed using many different logic circuits and several circuit boards. Today, all of this circuitry has been reduced to a tiny (typically 1/4 inch on a side) silicon chip, or integrated circuit (IC), called the *microprocessor*. The entire computer, including microprocessor, memory, and I/O, is called a *microcomputer*. The Intel microprocessors studied in this book derive their heritage from a chip whose part number was 8086. Subsequent versions of this chip have been numbered 80286, 80386, and 80486. The term *80x86* is therefore used to describe the *family* of compatible Intel microprocessors. (Section 1.2 describes the evolution of the microprocessor and microcomputer in more detail.)

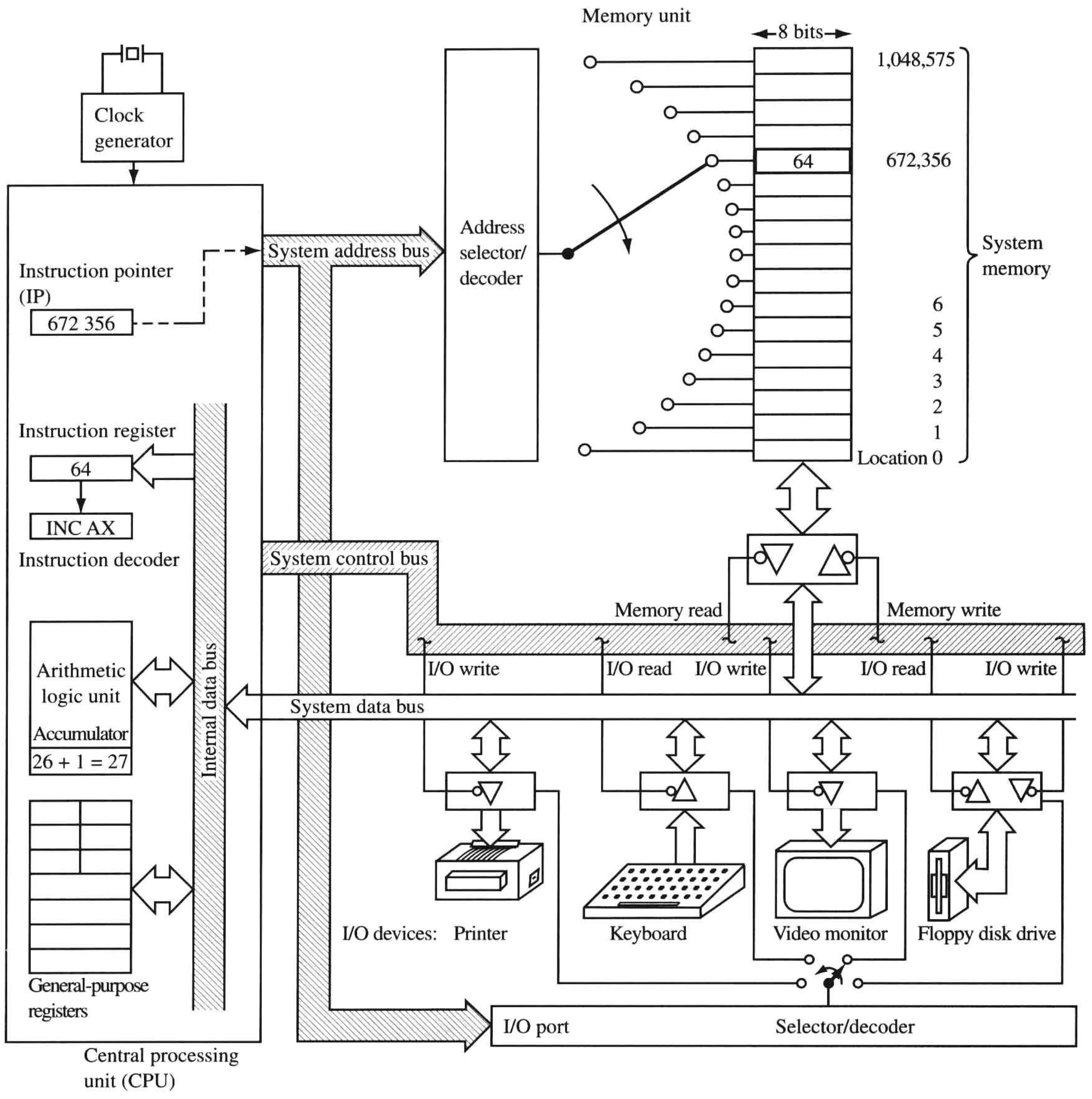
The basic timing of the computer is controlled by a square wave oscillator, or clock generator circuit. This signal is used to synchronize all activities within the computer, and determines how fast the program instructions can be fetched from memory and executed.

As shown in Figure 1.1, the CPU contains several *data registers* (flip-flops wired in series with each other). Some are general purpose and are used for storing temporary information. Others are special purpose. The accumulator, for example, is reserved for performing complex math operations such as multiply and divide. On 80x86 microprocessors, all data intended for the I/O devices must pass through this register.

The basic processing cycle begins with a memory fetch or read cycle. The *instruction pointer (IP)* register (also called the *program counter*) holds the address of the memory cell to be selected. In computerese, it "points" at the program instruction to be fetched. In this example, IP is storing the address 672,356, and the binary equivalent of this address is output onto the system address bus lines and routed to the memory unit.

[1]In Section 1.2 we discuss *parallel processors,* which offer an alternative to the von Neumann architecture.

Figure 1.1 The stored program computer consists of three units: the CPU, memory, and I/O devices.



The memory unit consists of a large number of storage locations, each with its own unique *address*. Because the CPU can randomly access any location in memory, the term *random access memory (RAM)* is often used. In this example, we assume each memory location is 8 bits wide, referred to as a *byte*. This memory organization is typical for most microprocessors today.[2]

[2]16-, 32-, and 64-bit microprocessors can fetch 2, 4, and 8 bytes, respectively, in one cycle. We still describe the memory capacity of these processors in bytes, however.

An important characteristic of RAM is its *volatility*. This means its contents will be lost when power is turned off. Because of this, a portion of the memory unit is often built using *read-only memory (ROM)* chips. The program stored by a ROM is permanent, and therefore not lost when power is removed. As the name implies, the data stored by a ROM can only be read, not written. A special program is required to write data into a ROM. (Chapter 2 explains the role of the boot ROM in a typical computer; Chapter 7 provides more detail on ROM programming.)

The memory unit's address selector/decoder circuit examines the binary number on the address lines and selects the proper memory location to be accessed. In this example, because the CPU is reading from memory, it activates its **MEMORY READ** control signal. This causes the selected data byte in memory to be placed onto the data lines and routed to the *instruction register* within the CPU.

Once in the CPU, the instruction is decoded and executed. In this example, the instruction has the decimal code 64, which (for an 80x86 microprocessor) is decoded to be INC AX—increment the accumulator register. The *arithmetic logic unit (ALU)* is therefore instructed to add 1 to the contents of the accumulator where the new result will be stored. In general, the ALU portion of the CPU performs all mathematical and Boolean logic functions.

With the instruction complete, the cycle repeats, beginning with a new instruction fetch cycle. The control logic in the CPU is wired so that register IP is always incremented after an instruction fetch; thus the next sequential instruction in memory will normally be accessed. The entire process of reading memory, incrementing the instruction pointer, and decoding the instruction is known as the *fetch and execute* principle of the stored program computer.

The Instruction Set. It is the job of the instruction decoder to recognize and activate the appropriate circuits in the CPU needed to carry out each new instruction as it is fetched from memory. The list of all such instructions recognizable by the decoder is called the *instruction set*. Microprocessors in the 80x86 family are known as *complex instruction set computers (CISC)* because of the large number of instructions in their instruction set (more than 3000 different forms). Some recent microprocessors have been designed to have only a small number of very fast executing instructions. Computers based on this concept are called *reduced instruction set computers (RISC)*. (RISC and CISC are discussed in more detail in Section 1.2.)

Modern CPUs. Most microprocessor chips today are designed to allow the fetch and execute cycles to overlap. This is done by dividing the CPU into an *execution unit (EU)* and a *bus interface unit (BIU)*. The BIU's job is to fetch instructions from memory as quickly as possible and store them in a special instruction *queue*. The EU then fetches instructions from this queue, not from memory. Because the fetch and execute cycles are allowed to overlap, the total processing time is reduced.

Some processors have a *pipelined* execution unit that allows the decoding and execution of instructions to be overlapped, further increasing processing performance. Intel's latest processor, the Pentium Pro, has a 12-stage pipeline with three engines: the Fetch/ Decode unit, the Dispatch/Execution unit, and the Retire unit. This architecture is referred to as *superscaler*. Superscaler microprocessors can process more than one instruction per clock cycle. (Chapter 3 discusses the architecture of the 80x86 family of processors in detail.)

Three-Bus System Architecture

8-, 16-, 32-, and 64-Bit Buses. A collection of electronic signal lines all dedicated to a particular task is called a *bus*. In Figure 1.1 there are three such buses: the *address, data,* and *control* buses. This three-bus system architecture is common to nearly all microcomputer systems.

The Data Bus. The width of the internal data bus in bits is usually used to classify a microprocessor. Thus, an 8-bit microprocessor has an 8-bit data bus, a 16-bit processor has a 16-bit data bus, etc. The width of the data bus determines how much data the processor can read or write in one memory, or I/O, cycle.

The width of the *internal* data bus (see Figure 1.1) is usually the same as the external bus—but not always. The 80386SX processor, for example, has a 32-bit internal data bus, but externally the bus is only 16 bits wide. This means the 80386SX will require two memory read operations to input the same information that the 80386 (with matching 32-bit internal and external data buses) inputs in one memory read cycle. The result is that the 80386SX operates less efficiently than the 80386.

The Pentium and Pentium Pro processors, on the other hand, have an external data bus width of 64 bits, but a 32-bit internal data bus. These chips are *data processing engines,* capable of executing two or three instructions per clock cycle, with clock rates greater than 200 MHz. The expanded data bus width is needed to keep these chips supplied with data.

Memory Banks. You may be wondering how a 64-bit (or 32-bit or 16-bit) processor can access an 8-bit-wide memory. The "trick" is to divide the memory into *banks*. The 64-bit Pentium and Pentium Pro, for example, require eight banks of memory, with each bank organized as one byte wide.[3] *Bank enable* signals are output by the processor to specify which bank (or banks) are to be accessed. (Chapter 7 discusses memory interfacing in detail.)

Example 1.1

An 80486 processor has its memory organized as shown in Figure 1.2. Determine the total amount of memory available to this processor.

Solution

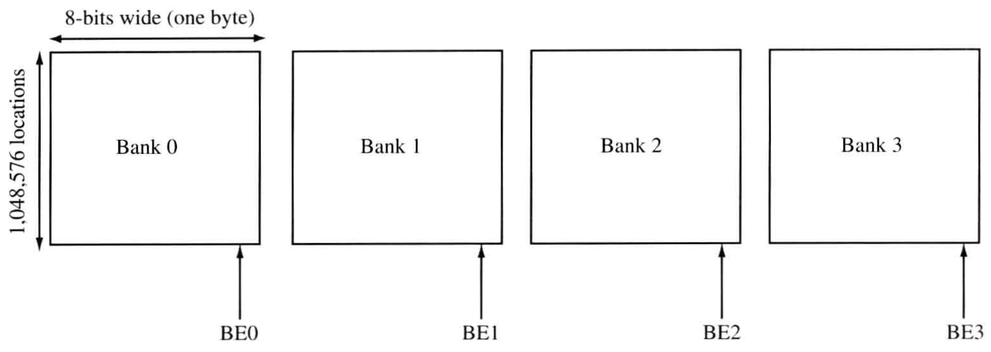
The 80486 has a 32-bit data bus, and therefore requires four banks of memory as shown. Note that each bank is organized as 1,048,576 8-bit bytes (1 megabyte or 1 MB). Four bank enable signals (BE0–BE3) are required to allow selection of each individual bank. The total amount of memory is:

$$4 \text{ banks} \times 1 \text{ MB/bank} = 4 \text{ MB}$$

The Address Bus. The *address bus* is used to identify the memory location or I/O device (also called I/O *port*) the processor intends to communicate with. For the 80x86 family of processors, the width of this bus ranges from 20 bits for the 8086 and 8088, to 32 bits for the 80386/486 and Pentium, and 36 bits for the Pentium Pro.

[3]These memory banks usually consist of memory chips soldered to small circuit boards called *SIMMs (single in-line memory modules)* or *DIMMs (dual in-line memory modules).* Thirty-pin SIMMs are 8 bits wide; 72-pin SIMMs are 32 bits wide. DIMMs are 64 bits wide.

Figure 1.2 Memory organization for the 80486 microprocessor. Four 8-bit (one-byte) banks are required. In this example, each bank stores 1 MB, thus providing a total of 4 MB of memory to the processor. Bank enable signals are used to select each memory bank.



Example 1.2

How many different memory addresses can an 8086 output? Repeat for a Pentium processor.

Solution
The 8086 has a 20-bit address bus and can therefore output all combinations of addresses from 0000 0000 0000 0000 0000 to 1111 1111 1111 1111 1111. This corresponds to 1,048,576 different addresses (2^{20}) or 1 MB (one megabyte).

The Pentium has a 32-bit address bus and can therefore access

$$2^{12} \times 2^{20} = 4096 \times 1 \text{ MB or } 4096 \text{ MB (four gigabytes)}$$

The Control Bus. How can we tell if the address on the bus is a memory address or an I/O port address? This is where the *control bus* comes in. Each time the processor outputs an address, it also activates one of four control bus signals. These are:

1. **MEMORY READ**
2. **MEMORY WRITE**
3. **I/O READ**
4. **I/O WRITE**

Thus, if the 8086 address bus holds 1010 0100 0010 0110 0100 ($672{,}356_{10}$) *and* the **MEMORY READ** signal is active, the data byte in memory location 672,356 will be selected to be read. The memory unit responds by outputting the contents of this location onto the data bus.

The control bus also identifies the *direction* of data flow on the data bus. When **MEMORY READ** or **I/O READ** is active, data is *input* to the processor. When **MEMORY WRITE** or **I/O WRITE** is active, data is *output* by the processor; that is, the control bus signals are defined from the processor's point of view.

Summary. The microprocessor manages the flow of data between itself, memory, and the I/O ports via the address, data, and control buses. The control and address buses are output lines (only) but the data bus is *bidirectional.*

Self Review 1.1 (Answers on page 32)

1.1.1 The three major blocks of a stored program computer are the __________, __________, and the __________.

1.1.2 The instruction pointer:
(a) Holds the address of the next memory location to be fetched into the instruction register.
(b) Is located within the processor.
(c) Is automatically incremented as part of the basic fetch and execute cycle.
(d) All of the above.

1.1.3 Complex arithmetic operations are performed in the __________ __________ __________, with results stored in the __________.

1.1.4 A microcomputer system with a 24-bit address bus could potentially access __________ memory locations.

1.1.5 The memory unit for an 80486 microprocessor requires __________ banks of memory.

1.1.6 When the processor writes data to the video terminal in Figure 1.1 the __________-__________ control bus signal will be activated.

1.1.7 Of the three system buses, only the __________ bus is bidirectional.

1.2 Types of Computers

Introduction

The first generation of computers were described in terms of hundreds of feet of floor space, tons of weight, and thousands of vacuum tubes. Reliability was often measured in hours or even minutes. Loading a new program required tens of man-hours to rewire the computer, and could only be done by trained engineers and technicians. Today we have *desktop* computers, *laptop* computers, and even *notebook* computers. We describe the processor in terms of submicron line spacings, millions of transistors, and onboard floating-point processors. Indeed, the *evolution* of the computer has occurred so rapidly that it is sometimes called a *revolution*.

In this section we:

- Trace the evolution of the computer from the vacuum tube era to the microprocessor.
- Identify significant computers that have been built over the years.
- Explain the difference between a microprocessor, a microcomputer, and a microcontroller.
- Compare digital signal processors (DSPs) with conventional microprocessors.

The Vacuum Tube Era

IBM Emerges. *First-generation computers* were massive machines based on vacuum tube technology. They occupied entire rooms and required an air-conditioned environment to operate reliably. In fact, because the average life of a vacuum tube was 3000 hours, and several thousand tubes were required to build a machine, some predicted that no useful work could ever be done—technicians would constantly be tracking down and replacing bad tubes! Nevertheless, in 1951, Remington-Rand delivered the first Univac I (UNIVersal Automatic Computer) to the Bureau of the Census. In 1952, CBS used a Univac I to predict the defeat of Adlai E. Stevenson by Dwight D. Eisenhower in the presidential election.

International Business Machines (IBM) reluctantly entered the computer field in 1952 with its Model 701 Data Processing System. IBM's founder, Thomas Watson, Sr., had to