3

Organization of the IBM Personal Computers

Overview

Chapter 1 described the organization of a typical microcomputer system. This chapter takes a closer look at the IBM personal computers. These machines are based on the Intel 8086 family of microprocessors.

After a brief survey of the 8086 family in section 3.1, section 3.2 concentrates on the architecture of the 8086. We introduce the registers and mention some of their special functions. In section 3.2.3, the important idea of segmented memory is discussed.

In section 3.3, we look at the overall structure of the IBM PC; the memory organization, I/O ports, and the DOS and BIOS routines.

3.1 The Intel 8086 Family of Microprocessors

The IBM personal computer family consists of the IBM PC, PC XT, PC AT, PS/1, and PS/2 models. They are all based on the Intel 8086 family of microprocessors, which includes the 8086, 8088, 80186, 80188, 80286, 80386, 80386SX, 80486, and 80486SX. The 8088 is used in the PC and PC XT; the 80286 is used in the PC AT and PS/1. The 80186 is used in some PC-compatible lap-top models. The PS/2 models use either the 8086, 80286, 80386, or 80486.

The 8086 and 8088 Microprocessors

Intel introduced the 8086 in 1978 as its first 16-bit microprocessor (a 16-bit processor can operate on 16 bits of data at a time). The 8088 was introduced in 1979. Internally, the 8088 is essentially the same as the 8086. Externally, the 8086 has a 16-bit data bus, while the 8088 has an 8-bit data bus. The 8086 also has a faster clock rate, and thus has better performance. IBM chose the 8088 over the 8086 for the original PC because it was less expensive to build a computer around the 8088.

The 8086 and 8088 have the same instruction set, and it forms the basic set of instructions for the other microprocessors in the family.

The 80186 and 80188 Microprocessors

The 80186 and 80188 are enhanced versions of the 8086 and 8088, respectively. Their advantage is that they incorporate all the functions of the 8086 and 8088 microprocessors plus those of some support chips. They can also execute some new instructions called the *extended instruction set*. However, these processors offered no significant advantage over the 8086 and 8088 and were soon overshadowed by the development of the 80286.

The 80286 Microprocessor

The 80286, introduced in 1982, is also a 16-bit microprocessor. However, it can operate faster than the 8086 (12.5 MHz versus 10 MHz) and offers the following important advances over its predecessors:

1. *Two modes of operation.* The 80286 can operate in either **real address mode** or **protected virtual address mode**. In real address mode, the 80286 behaves like the 8086, and programs for the 8086 can be executed in this mode without modification. In protected virtual address mode, also called **protected mode**, the 80286 supports **multitasking**, which is the ability to execute several programs (tasks) at the same time, and **memory protection**, which is the ability to protect the memory used by one program from the actions of another program.
2. *More addressable memory.* The 80286 in protected mode can address 16 megabytes of physical memory (as opposed to 1 megabyte for the 8086 and 8088).
3. *Virtual memory in protected mode.* This means that the 80286 can treat external storage (that is, a disk) as if it were physical memory, and therefore execute programs that are too large to be contained in physical memory; such programs can be up to 1 gigabyte (2^{30} bytes).

The 80386 and 80386SX Microprocessors

Intel introduced its first 32-bit microprocessor, the 80386 (or 386), in 1985. It is much faster than the 80286 because it has a 32-bit data path, high clock rate (up to 33 MHz), and the ability to execute instructions in fewer clock cycles than the 80286.

Like the 80286, the 386 can operate in either real or protected mode. In real mode, it behaves like an 8086. In protected mode, it can emulate the 80286. It also has a *virtual 8086 mode* designed to run multiple 8086 applications under memory protection. The 386, in protected mode, can address 4 gigabytes of physical memory, and 64 terabytes (2^{46} bytes) of virtual memory.

The 386SX has essentially the same internal structure as the 386, but it has only a 16-bit data bus.

The 80486 and 80486SX Microprocessors

Introduced in 1989, the 80486 (or 486), is another 32-bit microprocessor. It is the fastest and most powerful processor in the family. It incorporates the functions of the 386 together with those of other support chips, including the 80387 numeric processor, which performs floating-point number operations, and an 8-KB cache memory that serves as a fast memory area to buffer data coming from the slower memory unit. With its numeric processor, cache memory, and more advanced design, the 486 is three times faster than a 386 running at the same clock speed. The 486SX is similar to the 486 but without the floating-point processor.

3.2 Organization of the 8086/8088 Microprocessors

In the rest of this chapter we'll concentrate on the organization of the 8086 and 8088. These processors have the simplest structure, and most of the instructions we will study are 8086/8088 instructions. They also provide insight to the organization of the more advanced processors, discussed in Chapter 20.

Because the 8086 and 8088 have essentially the same internal structure, in the following, the name "8086" applies to both 8086 and 8088.

3.2.1 Registers

As noted in Chapter 1, information inside the microprocessor is stored in registers. The registers are classified according to the functions they perform. In general, *data registers* hold data for an operation, *address registers* hold the address of an instruction or data, and a *status register* keeps the current status of the processor.

The 8086 has four general data registers; the address registers are divided into *segment*, *pointer*, and *index registers*; and the status register is called the *FLAGS register*. In total, there are fourteen 16-bit registers, which we now briefly describe. See Figure 3.1. *Note:* You don't need to memorize the special functions of these registers at this time. They will become familiar with use.

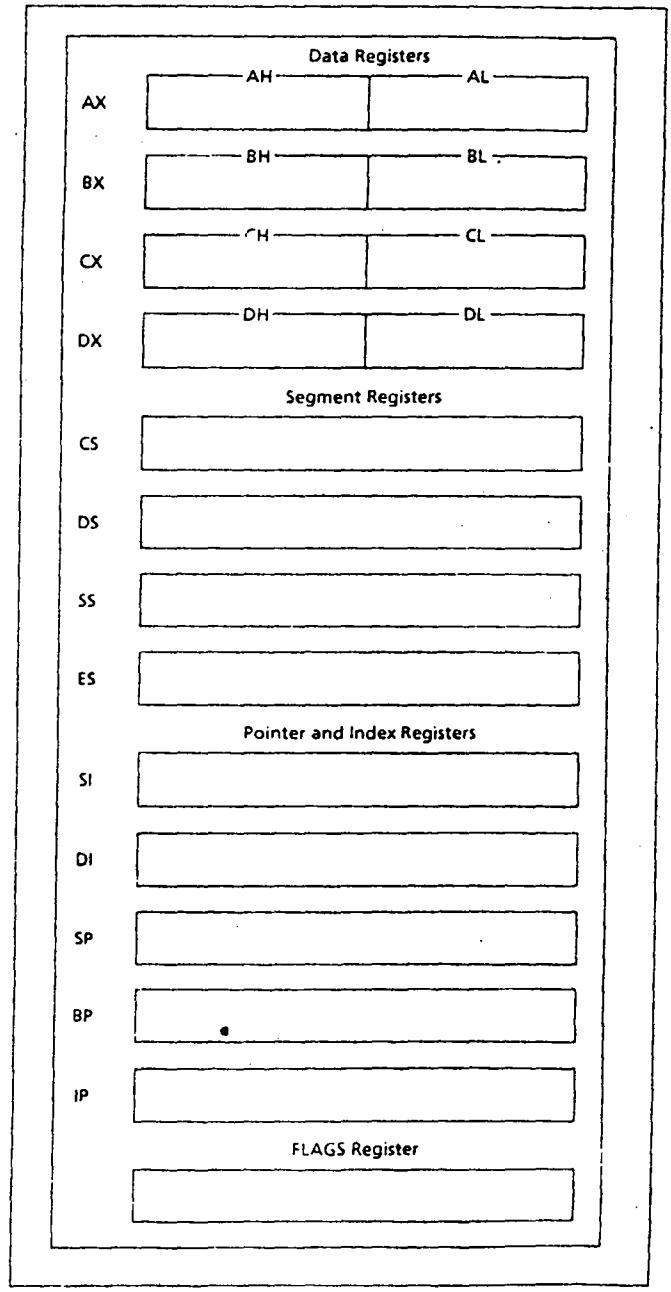
3.2.2 Data Registers: AX, BX, CX, DX

These four registers are available to the programmer for general data manipulation. Even though the processor can operate on data stored in memory, the same instruction is faster (requires fewer clock cycles) if the data are stored in registers. This is why modern processors tend to have a lot of registers.

The high and low bytes of the data registers can be accessed separately. The high byte of AX is called AH, and the low byte is AL. Similarly, the high and low bytes of BX, CX, and DX are BH and BL, CH and CL, DH and DL, respectively. This arrangement gives us more registers to use when dealing with byte-size data.

These four registers, in addition to being general-purpose registers, also perform special functions such as the following.

Figure 3.1 8086 Registers



AX (Accumulator Register)

AX is the preferred register to use in arithmetic, logic, and c
transfer instructions because its use generates the shortest machine co

In multiplication and division operations, one of the numbers involved must be in AX or AL. Input and output operations also require the use of AL and AX.

BX (Base Register)

BX also serves as an address register; an example is a table look-up instruction called XLAT (translate).

CX (Count Register)

Program loop constructions are facilitated by the use of CX, which serves as a loop counter. Another example of using CX as counter is REP (repeat), which controls a special class of instructions called *string operations*. CL is used as a count in instructions that shift and rotate bits.

DX (Data Register)

DX is used in multiplication and division. It is also used in I/O operations

3.2.3 Segment Registers: CS, DS, SS, ES

Address registers store addresses of instructions and data in memory. These values are used by the processor to access memory locations. We begin with the memory organization.

Chapter 1 explained that memory is a collection of bytes. Each memory byte has an address, starting with 0. The 8086 processor assigns a 20-bit **physical address** to its memory locations. Thus it is possible to address 2^{20} = 1,048,576 bytes (one megabyte) of memory. The first five bytes in memory have the following addresses:

00000000000000000000
00000000000000000001
00000000000000000010
00000000000000000011
00000000000000000100

Because addresses are so cumbersome to write in binary, we usually express them as five hex digits, thus

00000
00001
00002
.
.
.
00009
0000A
0000B
.
.
.

and so on. The highest address is FFFFFh.

In order to explain the function of the segment registers, we first need to introduce the idea of memory segments, which is a direct consequence of using a 20-bit address in a 16-bit processor. The addresses are too

big to fit in a 16-bit register or memory word. The 8086 gets around this problem by partitioning its memory into segments.

Memory Segment

A **memory segment** is a block of 2^{16} (or 64 K) consecutive memory bytes. Each segment is identified by a **segment number**, starting with 0. A segment number is 16 bits, so the highest segment number is FFFFh.

Within a segment, a memory location is specified by giving an **offset**. This is the number of bytes from the beginning of the segment. With a 64-KB segment, the offset can be given as a 16-bit number. The first byte in a segment has offset 0. The last offset in a segment is FFFFh.

Segment:Offset Address

A memory location may be specified by providing a segment number and an offset, written in the form *segment:offset;* this is known as a **logical address.** For example, A4FB:4872h means offset 4872h within segment A4FBh. To obtain a 20-bit physical address, the 8086 microprocessor first shifts the segment address 4 bits to the left (this is equivalent to multiplying by 10h), and then adds the offset. Thus the physical address for A4FB:4872 is

```
  A4FB0h
+ 4872h
-------
  A9822h   (20-bit physical address)
```

Location of Segments

It is instructive to see the layout of the segments in memory. Segment 0 starts at address 0000:0000 = 00000h and ends at 0000:FFFF = 0FFFFh. Segment 1 starts at address 0001:0000 = 00010h and ends at 0001:FFFF = 1000Fh. As we can see, there is a lot of overlapping between segments. Figure 3.2 shows the locations of the first three memory segments. The segments start every 10h = 16 bytes and the starting address of a segment always ends with a hex digit 0. We call 16 bytes a **paragraph**. We call an address that is divisible by 16 (ends with a hex digit 0) a **paragraph boundary**.

Because segments may overlap, the segment:offset form of an address is not unique, as the following example shows.

Example 3.1 For the memory location whose physical address is specified by 1256Ah, give the address in segment:offset form for segments 1256h and 1240h.

Solution: Let X be the offset in segment 1256h and Y the offset in segment 1240h. We have

$$1256Ah = 12560h + X \text{ and } 1256Ah = 12400h + Y$$

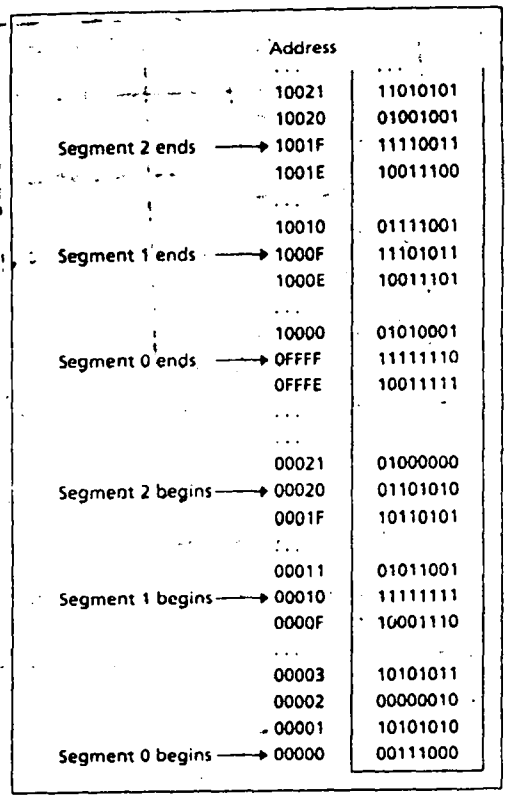
and so

$$X = 1256Ah - 12560h = Ah \text{ and } Y = 1256Ah - 12400h = 16Ah$$

thus

$$1256Ah = 1256:000A = 1240:016A$$

Figure 3.2 Location of Memory Segments



It is also possible to calculate the segment number when the physical address and the offset are given.

Example 3.2 A memory location has physical address 80FD2h. In what segment does it have offset BFD2h?

Solution: We know that

$$\text{physical address} = \text{segment} \times 10\text{h} + \text{offset}$$

Thus

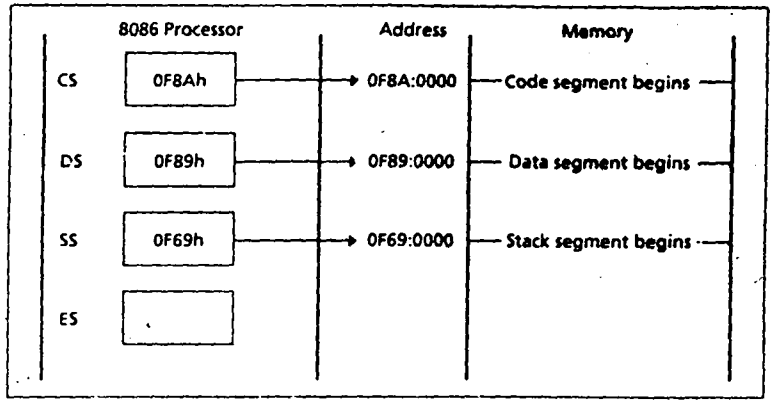
$$\text{segment} \times 10\text{h} = \text{physical address} - \text{offset}$$

in this example

$$\begin{array}{rl} \text{physical address} = & 80\text{FD2h} \\ -\ \text{offset} = & \text{BFD2h} \\ \hline \text{segment} \times 10\text{h} = & 75000\text{h} \end{array}$$

So the segment must be 7500h.

Figure 3.3 Segment Registers



Program Segments

Now let us talk about the registers CS, DS, SS, and ES. A typical machine language program consists of instructions (code) and data. There is also a data structure called the **stack** used by the processor to implement procedure calls. The program's code, data, and stack are loaded into different memory segments, we call them the **code segment**, **data segment**, and **stack segment**.

To keep track of the various program segments, the 8086 is equipped with four segment registers to hold segment numbers. The CS, DS, and SS registers contain the code, data, and stack segment numbers, respectively. If a program needs to access a second data segment, it can use the ES (extra segment) register.

A program segment need not occupy the entire 64 kilobytes in a memory segment. The overlapping nature of the memory segments permits program segments that are less than 64 KB to be placed close together. Figure 3.3 shows a typical layout of the program segments in memory (the segment numbers and the relative placement of the program segments shown are arbitrary).

At any given time, only those memory locations addressed by the four segment registers are accessible; that is, only four memory segments are *active*. However, the contents of a segment register can be modified by a program to address different segments.

3.2.4 *Pointer and Index Registers: SP, BP, SI, DI*

The registers SP, BP, SI, and DI normally point to (contain the offset addresses of) memory locations. Unlike segment registers, the pointer and index registers can be used in arithmetic and other operations.

SP (Stack Pointer)

The SP (stack pointer) register is used in conjunction with SS for accessing the stack segment. Operations of the stack are covered in Chapter 8.

BP (Base Pointer)

The BP (base pointer) register is used primarily to access data on the stack. However, unlike SP, we can also use BP to access data in the other segments.

SI (Source Index)

The SI (source index) register is used to point to memory locations in the data segment addressed by DS. By incrementing the contents of SI, we can easily access consecutive memory locations.

DI (Destination Index)

The DI (destination index) register performs the same functions as SI. There is a class of instructions, called *string operations*, that use DI to access memory locations addressed by ES.

3.2.5 Instruction Pointer: IP

The memory registers covered so far are for data access. To access instructions, the 8086 uses the registers CS and IP. The CS register contains the segment number of the next instruction, and the IP contains the offset. IP is updated each time an instruction is executed so that it will point to the next instruction. Unlike the other registers, the IP cannot be directly manipulated by an instruction; that is, an instruction may not contain IP as its operand.

3.2.6 FLAGS Register

The purpose of the FLAGS register is to indicate the status of the microprocessor. It does this by the setting of individual bits called **flags**. There are two kinds of flags: **status flags** and **control flags**. The status flags reflect the result of an instruction executed by the processor. For example, when a subtraction operation results in a 0, the ZF (zero flag) is set to 1 (true). A subsequent instruction can examine the ZF and branch to some code that handles a zero result.

The control flags enable or disable certain operations of the processor; for example, if the IF (interrupt flag) is cleared (set to 0), inputs from the keyboard are ignored by the processor. The status flags are covered in Chapter 5, and the control flags are discussed in Chapters 11 and 15.

3.3 Organization of the PC

A computer system is made up of both hardware and software. It is the software that controls the hardware operations. So, to fully understand the operations of the computer, you also study the software that controls the computer.

3.3.1 The Operating System

The most important piece of software for a computer is the **operating system.** The purpose of the operating system is to coordinate the operations of all the devices that make up the computer system. Some of the operating system functions are

1. reading and executing the commands typed by the user
2. performing I/O operations
3. generating error messages
4. managing memory and other resources

At present, the most popular operating system for the IBM PC is the **disk operating system (DOS),** also referred to as PC DOS or MS DOS. DOS was designed for the 8086/8088-based computers. Because of this, it can manage only 1 megabyte of memory and it does not support multitasking. However, it can be used on 80286, 80386, and 80486-based machines when they run in real address mode.

One of the many functions performed by DOS is reading and writing information on a disk. Programs and other information stored on a disk are organized into **files.** Each file has a **file name,** which is made up of one to eight characters followed by an optional **file extension** of a period followed by one to three characters. The extension is commonly used to identify the type of file. For example, COMMAND.COM has a file name COMMAND and an extension .COM.

There are several versions of DOS, with each new version having more capabilities. Most commercial programs require the use of version 2.1 or later. DOS is not just one program; it consists of a number of service routines. The user requests a service by typing a command. The latest version, DOS 5.0, also supports a **graphical user interface (gui),** allowing the use of a mouse.

The DOS routine that services user commands is called **COMMAND.COM.** It is responsible for generating the DOS prompt—that is, C>—and reading user commands. There are two types of user commands, **internal** and **external.**

Internal commands are performed by DOS routines that have been loaded into memory, external commands may refer to DOS routines that have not been loaded or to application programs. In normal operations, many DOS routines are not loaded into memory so as to save memory space.

Because DOS routines reside on disk, a program must be operating when the computer is powered up to read the disk. In Chapter 1 we mentioned that there are system routines stored in ROM that are not destroyed when the power is off. In the PC, they are called **BIOS (Basic Input/Output System)** routines.

BIOS

The BIOS routines perform I/O operations for the PC. Unlike the DOS routines, which operate over the entire PC family, the BIOS routines are machine specific. Each PC model has its own hardware configuration and its own BIOS routines, which invoke the machine's I/O port registers for input and output. The DOS I/O operations are ultimately carried out by the BIOS routines.

Other important functions performed by BIOS are circuit checking and loading of the DOS routines. In section 3.3.4, we discuss the loading of DOS routines.

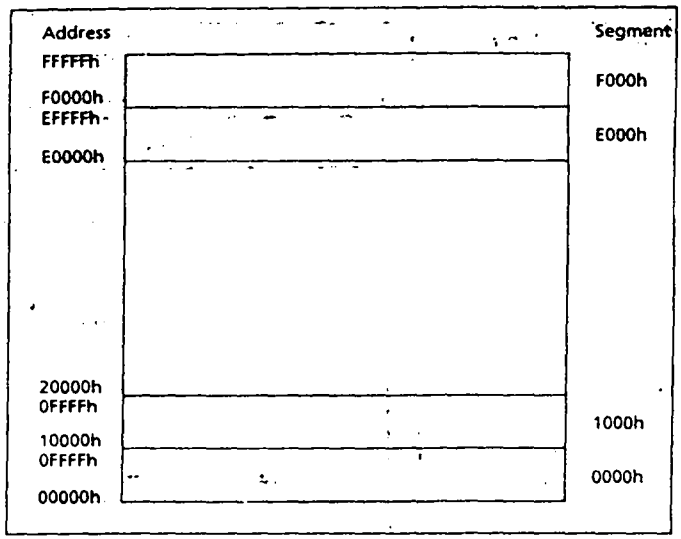


Figure 3.4 Memory Partitioned into Disjoint Segments

To let DOS and other programs use the BIOS routines, the addresses of the BIOS routines, called **interrupt vectors**, are placed in memory, starting at 00000h. Some DOS routines also have their addresses stored there.

Because IBM has copyrighted its BIOS routines, IBM compatibles use their own BIOS routines. The degree of compatibility has to do with how well their BIOS routines match the IBM BIOS.

3.3.2 Memory Organization of the PC

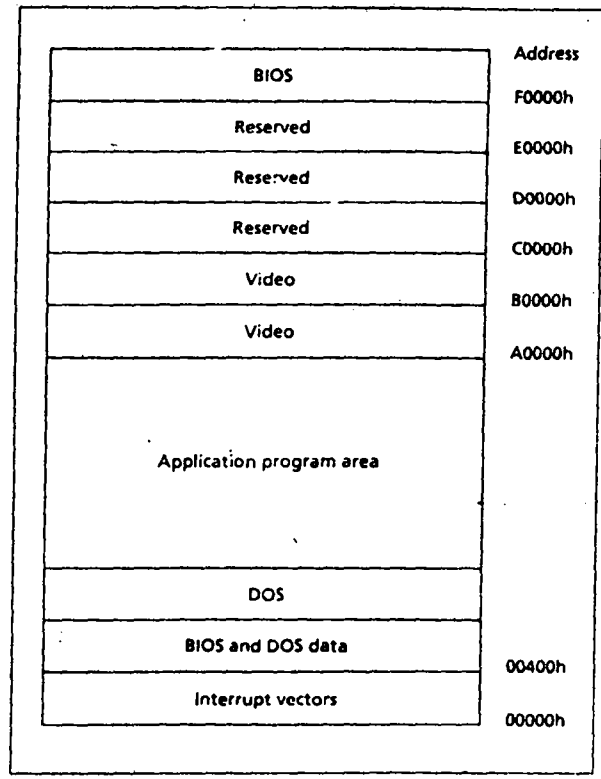
As indicated in section 3.2.3, the 8086/8088 processor is capable of addressing 1 megabyte of memory. However, not all the memory can be used by an application program. Some memory locations have special meaning for the processor. For example, the first kilobyte (00000 to 003FFh) is used for interrupt vectors.

Other memory locations are reserved by IBM for special purposes, such as for BIOS routines and **video display memory**. The display memory holds the data that are being displayed on the monitor.

To show the memory map of the IBM PC, it is useful to partition the memory into disjoint segments. We start with segment 0, which ends at location 0FFFFh, so the next disjoint segment would begin at 10000h = 1000:0000. Similarly, segment 1000h ends at 1FFFFh and the next disjoint segment begins at 20000h = 2000:0000. Therefore the disjoint segments are 0000h, 1000h, 2000h, . . . F000h, and so memory may be partitioned into 16 disjoint segments. See Figure 3.4.

Only the first 10 disjoint memory segments are used by DOS for loading and running application programs. These ten segments, 0000h to 9000h, give us 640 KB of memory. The memory sizes of 8086/8088-based PCs are given in terms of these memory segments. For example, a PC with a 512-KB memory has only eight of these memory segments.

Figure 3.5 Memory Map of the PC



Segments A000h and B000h are used for video display memory. Segments C000h to E000h are reserved. Segment F000h is a special segment because its circuits are ROM instead of RAM, and it contains the BIOS routines and ROM BASIC. Figure 3.5 shows the memory layout.

Table 3.1 Some Common I/O Ports for the PC

Port Address	*Description*
20h–21h	interrupt controller
60h–63h	keyboard controller
200h–20Fh	game controller
2F8h–2FFh	serial port (COM 2)
320h–32Fh	hard disk
378h–37Fh	parallel printer port 1
3C0h–3CFh	EGA
3D0h–3DFh	CGA
3F8h–3FFh	serial port (COM1)

3.3.3 I/O Port Addresses

The 8086/8088 supports 64 KB of I/O ports. Some common port addresses are given in Table 3.1. In general, direct programming of I/O ports is not recommended because I/O port address usage may vary among computer models.

3.3.4 Start-up Operation

When the PC is powered up, the 8086/8088 processor is put in a reset state, the CS register is set to FFFFh, and IP is set to 0000h. So the first instruction it executes is located at FFFF0h. This memory location is in ROM, and it contains an instruction that transfers control to the starting point of the BIOS routines.

The BIOS routines first check for system and memory errors, and then initialize the interrupt vectors and BIOS data area. Finally, BIOS loads the operating system from the system disk. This is done in two steps; first, the BIOS loads a small program, called the **boot program**, then the boot program loads the actual operating system routines. The boot program is so named because it is part of the operating system; having it load the operating system is like the computer pulling itself up by the bootstraps. Using the boot program isolates the BIOS from any changes made to the operating system and lets it be smaller in size. After the operating system is loaded into memory, COMMAND.COM is then given control.

Summary

- The IBM personal computer family consists of the PC, PC XT, PC AT, PS/1, and the PS/2 models. They use the Intel 8086 family of microprocessors.
- The 8086 family of microprocessors consists of the 8086, 8088, 80186, 80188, 80286, 80386, 80386SX, 80486, and 80486SX.
- The 8086 and 8088 have the same instruction set, and this forms the basic set of instructions for the other microprocessors.
- The 8086 microprocessor contains 14 registers. They may be classified as data registers, segment registers, pointer and index registers, and the FLAGS register.
- The data registers are AX, BX, CX, and DX. These registers may be used for general purposes, and they also perform special functions. The high and low bytes can be addressed separately.
- Each byte in memory has a 20-bit = 5 hex-digit address, starting with 00000h.
- A segment is a 64-KB block of memory. Addresses in memory may be given in segment:offset form. The physical address is obtained by multiplying the segment number by 10h, and adding the offset.
- The segment registers are CS, DS, SS, and ES. When a machine language program is executing, these registers contain the segment numbers of the code, data, stack, and extra data segments.

- The pointer and index registers are SP, BP, SI, DI, and IP. SP is used exclusively for the stack segment. BP can be used to access the stack segment. SI and DI may be used to access data in arrays.
- The IP contains the offset address of the next instruction to be executed.
- The FLAGS register contains the status and control flags. The status flags are set according to the result of an operation. The control flags may be used to enable or disable certain operations of the microprocessor.
- DOS is a collection of routines that coordinates the operations of the computer. The routine that executes user commands is COMMAND.COM.
- Information stored on disk is organized into files. A file has a name and an optional extension.
- The BIOS routines are used to perform I/O operations. The compatibility of PC clones with the IBM PC depends on how well their BIOS routines match those of the IBM PC.
- The BIOS routines are responsible for system testing and loading the operating system when the machine is turned on.

Glossary

basic input/output system, BIOS	Routines that handle input and output operations
boot program	The routine that loads the operating system during start-up
code segment	Memory segment containing a machine language program's instructions
COMMAND.COM	The command processor for DOS
control flags	Flags that enable or disable certain actions of the processor
data segment	Memory segment containing a machine language program's data
disk operating system, DOS	The operating system for the IBM PC
external commands	Commands that correspond to routines residing on disk.
file	An organized, named collection of data items treated as a single unit for storage on devices such as disks
file extension	A period followed by one to three characters; used to identify the kind of file
file name	A one- to eight-character name of a file
flags	Bits of the FLAGS register
graphical user interface, gui	A user interface with pointers and graphical symbols

internal commands	DOS commands that are executed by routines that are present in memory
interrupt vectors	Addresses of the BIOS and DOS routines
logical address	An address given in the form segment:offset
memory protection	The ability of a processor to protect the memory used by one program from being used by another running program
memory segment	A 64-KB block of memory
multitasking	The ability of a computer to execute several programs at the same time
offset (of a memory location)	The number of bytes of the location from the beginning of a segment
operating system	A collection of programs that coordinate the operations of the devices that make up a computer system
paragraph	16 bytes
paragraph boundary	A hex address ending in 0
physical address	Address of a memory location; 8086-based machines have 20-bit addresses
protected (virtual address) mode	A processor mode in which the memory used by one program is protected from the actions of another program
real address mode	A processor mode in which the addresses used in a program correspond to a physical memory address
segment number	Number that identifies a memory segment
stack	A data structure used by the processor to implement procedure calls
stack segment	Memory segment containing a machine language program's stack
status flags	Flags that reflect the actions of the processor
video display memory	Memory used for storing data for display on the monitor
virtual memory	The ability of the advanced processors to treat external storage as if it were real internal memory, and therefore execute programs that are too large to be contained in internal memory

Exercises

1. What are the main differences between the 80286 and the 8086 processors?
2. What are the differences between a register and a memory location?
3. List one special function for each of the data registers AX, BX, CX, and DX.

4. Determine the physical address of a memory location given by 0A51:CD90h.
5. A memory location has a physical address 4A37Bh. Compute
   a. the offset address if the segment number is 40FFh.
   b. the segment number if the offset address is 123Bh.
6. What is a paragraph boundary?
7. What determines how compatible an IBM PC clone is with an authentic IBM PC?
8. What is the maximum amount of memory that DOS allocates for loading run files? Assume that DOS occupies up to the byte OFFFFh.

   For the following exercises, refer to Appendix B.

9. Give DOS commands to do the following. Suppose that A is the logged drive.
   a. Copy FILE1 in the current directory to FILE1A on the disk in drive B.
   b. Copy all files with an .ASM extension to the disk in drive B.
   c. Erase all files with a .BAK extension
   d. List all file names in the current directory that begin with A.
   e. Set the date to September 21, 1991.
   f. Print the file FILE5.ASM on the printer.
10. Suppose that (a) the root directory has subdirectories A, B, and C; (b) A has subdirectories A1 and A2; (c) A1 has a subdirectory A1A. Give DOS commands to
    a. Create the preceding directory tree.
    b. Make A1A the current directory.
    c. Have DOS display the current directory.
    d. Remove the preceding directory tree.