Chapter 2 – Parts of a Computer System

A practical computer system consists of hardware and software.

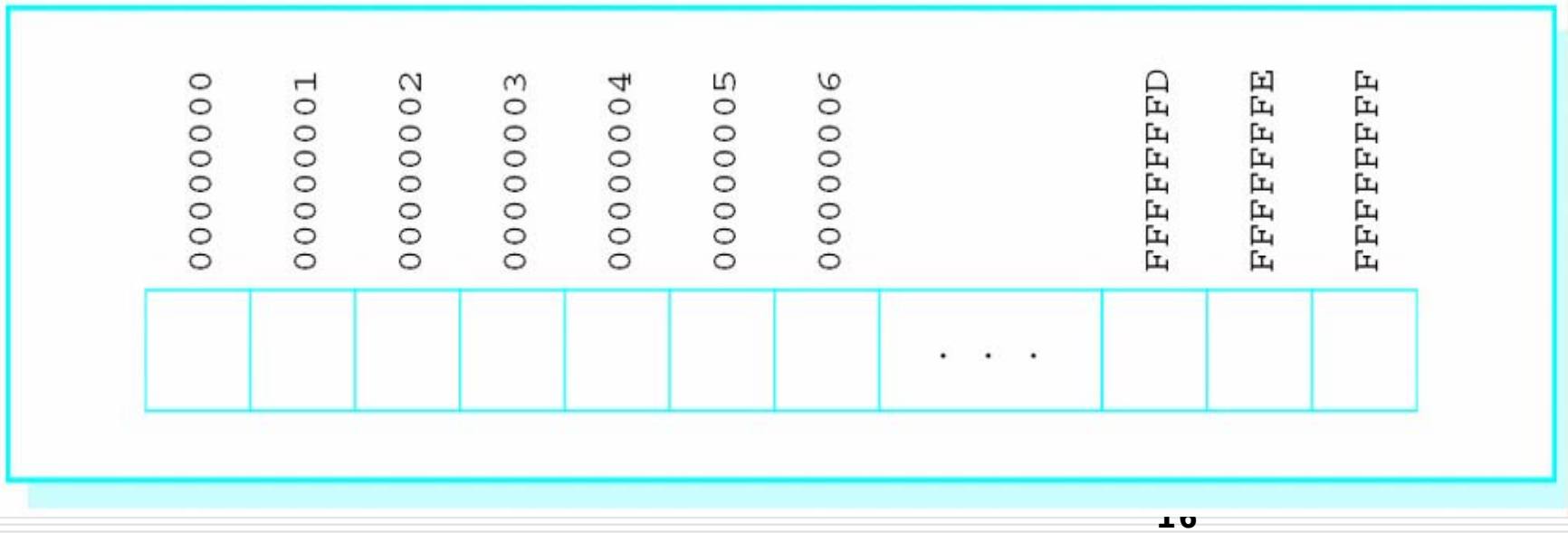
- ✓ The major hardware components of a typical microcomputer system are a central processing unit (CPU), memory circuits, a keyboard for input, a monitor or some other display device, specialized input/output devices like a mouse, a modem, or a sound card, and one or more disk drives to store programs and data.
- ✓ Software refers to the programs that the hardware executes, including system software and application software.

Chapter 2 – Parts of a Computer System

This chapter discusses how the memory and CPU look to the assembly language programmer for a particular class of microcomputers, the IBM PC and compatible systems. These computers have an Intel 80×86 CPU; that is, an 8086 or 8088, an 80286, an 80386, an 80486, or a Pentium processor.

We assumes a system that has an 80386 or higher processor and a 32-bit operating system such as Windows 95 or Windows NT.

2.1 PC Hardware: Memory



Since $FFFFFFFF_{16}$ = 4,294,967,295, a PC can contain up to 4,294,967,296 bytes of memory, or four gigabytes.

Chapter 2 – Parts of a Computer System

2.1 PC Hardware: Memory

Prior to the 80386 chip, the Intel 80×86 family of processors could only directly address 2^{20} bytes of memory. They used 20-bit physical addresses, often expressed as 5-hex-digit addresses ranging from 00000 to FFFFF.

The assembly language programs in Our class book will use **a flat memory model**. This means that the programs will actually encode 32-bit addresses to logically reference locations in a single memory space where data and instructions are stored.

The Intel 80×86 architecture also provides for a **segmented memory** model. In the original 8086/8088 CPU, this memory model was the only one available. With the 8086/8088, the PC's memory is visualized as a collection of segments, each segment 64 Kbytes long, starting on an address that is a multiple of 16.

Notice that the starting address of a segment ends in 0 when written in hex. The segment number of a segment consists of the first four hex digits of its physical address.

2.1 PC Hardware: Memory

A program written for the 8086/8088 does not encode a five-hex-digit address. Instead, each memory reference depends on its segment number and a 16-bit **offset** from the beginning of the segment. The offset is the distance from the first byte of the segment to the byte being addressed. In hex, an offset is between 0000 and $FFFF_{16}$. The notation for a segment-offset address is the four-hex-digit segment number followed by a colon (:) followed by the four-hex-digit offset.

As an example, **18A3:5B27** refers to the byte that is 5B27 bytes from the beginning of the segment starting at address 18A30. Add the starting address and the offset to get the five-hex-digit address.

```
  18A30   starting address of segment 18A3
+ 5B27    offset 1E557 five-hex-digit address
  1E557   five-hex-digit address
```

2.1 PC Hardware: Memory

From the 80386 on, 80×86 processors have had both 16-bit and 32-bit segmented memory models available.

Segment numbers are still 16-bits long, but they do not directly reference a segment in memory. Instead, a segment number is used as an index into a table that contains the actual 32-bit starting address of the segment. In the 32-bit segmented model, a 32-bit offset is added to that starting address to compute the actual address of the memory operand. Segments can be logically useful to a programmer: In the segmented Intel model, the programmer normally assigns different memory segments to code, data, and a system stack. The 80×86 flat memory model is really a 32-bit segmented model with all segment registers containing the same value.

2.1 PC Hardware: Memory

In reality, the 32-bit address generated by a program is not necessarily the physical address at which an operand is stored as the program executes. There is an additional layer of memory management performed by the operating system and the Intel 80×86 CPU.

A **paging** mechanism is used to map the program's 32-bit addresses into physical addresses. Paging is useful when a logical address generated by a program exceeds the physical memory actually installed in a computer.

It can also be used to swap parts of a program from disk as needed when the program is too large to fit into physical memory.

The paging mechanism will be transparent to us as we program in assembly language.

2.2 PC Hardware: The CPU

The original 8086/8088 CPU could execute over 200 different instructions.

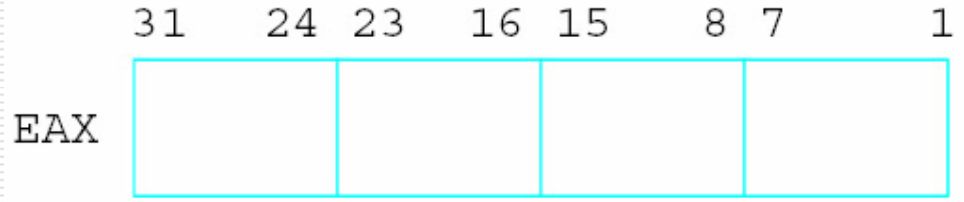
This instruction set has been extended as the 80×86 processor family has expanded to include the 80286, 80386, 80486, and Pentium processors.

Other manufacturers make CPUs that execute essentially the same instruction set, so that a program written for an Intel 80×86 runs without change on such CPUs.

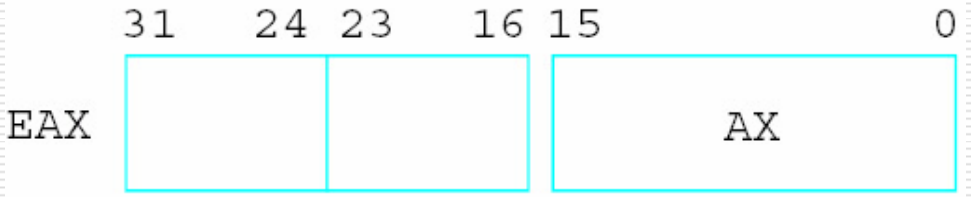
Many other processor families execute different instruction sets. However, most have a similar architecture, so that the basic principles you learn about the 80×86 CPUs also apply to these systems

2.2 PC Hardware: The CPU

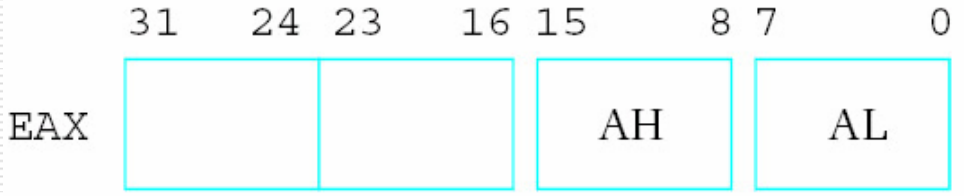
A CPU contains registers. The application registers are of most concern to the programmer. An 80×86 CPU (from 80386 on) has 16 application registers.



Parts of the EAX register can be addressed separately from the whole. The low-order word, bits 0-15, is known as AX.



Similarly, the low-order byte (bits 0-7) and the high-order byte (bits 8-15) of AX are known as AL and AH, respectively.



2.2 PC Hardware: The CPU

The instruction

```
sub ax, 10
```

subtracts 10 from the word stored in AX, without changing any of the high-order bits (16-31) of EAX.

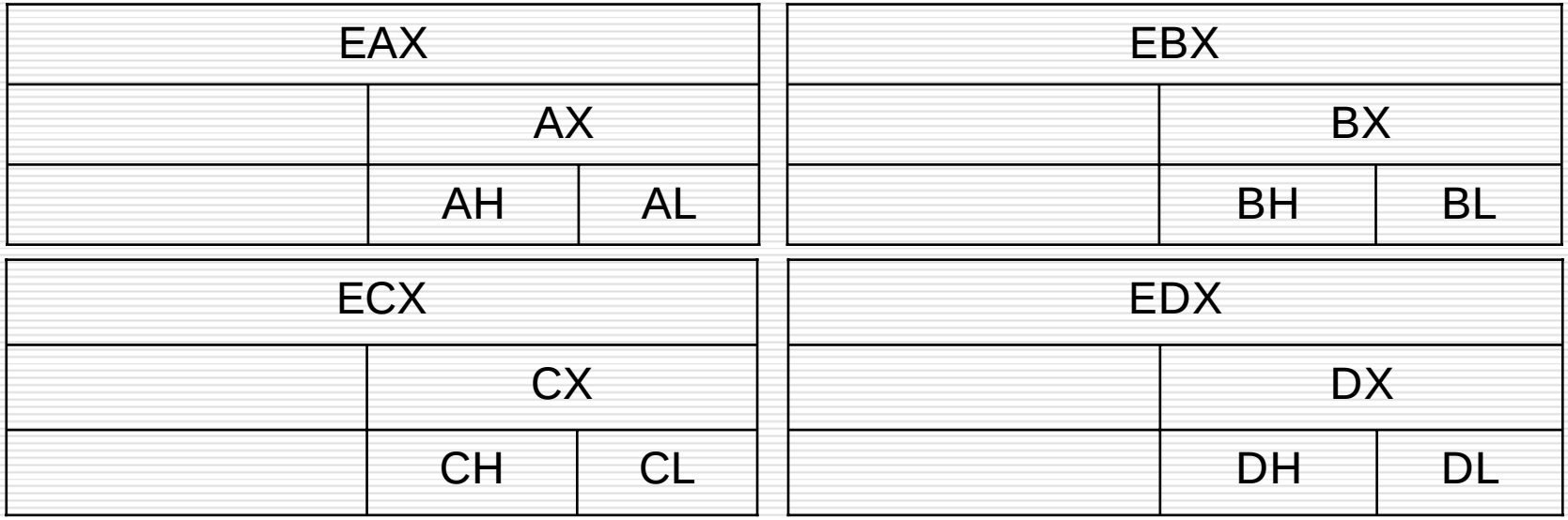
The instruction

```
mov ah, '*'
```

copies 2A, the ASCII code for an asterisk, to bits 8-15, without changing any of the other bits of EAX.

2.2 PC Hardware: The CPU

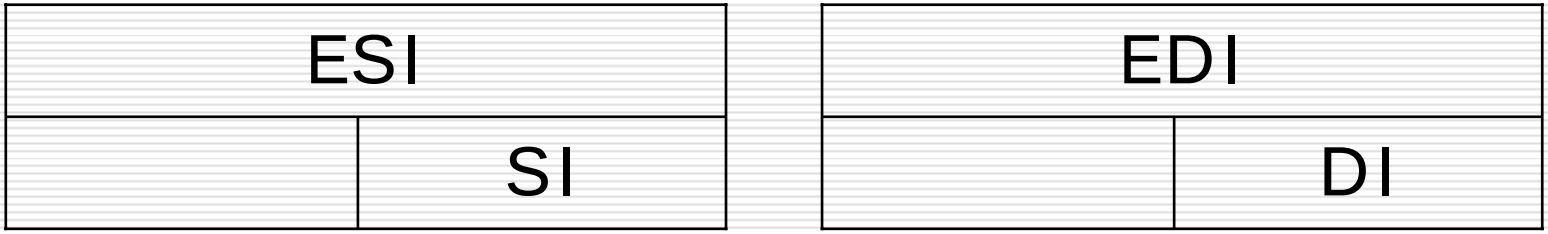
The EBX, ECX, and EDX registers also have low-order words BX, CX, and DX, which are divided into high-order and low-order bytes BH and BL, CH and CL, and DH and DL.



The 8086 through 80286 processors had four 16-bit general registers called AX, BX, CX, and DX. **The "E" was added for "extended" with the 32-bit 80386 registers**.

2.2 PC Hardware: The CPU

There are four additional 32-bit registers, ESI, EDI, ESP, and EBP. In fact, you can use these registers for operations like arithmetic, but normally you should save them for their special purposes. The ESI and EDI registers are index registers, where SI stands for source index and DI stands for destination index. The names SI and DI can be used for the low-order words of ESI and EDI, respectively.



2.2 PC Hardware: The CPU

The ESP register is the stack pointer for the system stack. It is rarely changed directly by a program, but is changed when data is pushed onto the stack or popped from the stack. One use for the stack is in procedure (subroutine) calls. The address of the instruction following the procedure call instruction is stored on the stack. When it is time to return, this address is retrieved from the stack. The name SP can be used for the low-order word of ESP.

The EBP register is the base pointer register. Normally the only data item accessed in the stack is the one that is at the top of the stack. However, the EBP register is often used to mark a fixed point in the stack other than the stack top, so that data near this point can be accessed. This is also used with procedure calls, particularly when parameters are involved.

2.2 PC Hardware: The CPU

- ✓ **There are six 16-bit segment registers: CS, DS, ES, FS, GS, and SS.**
- ✓ In the older 16-bit segmented memory model, the CS register contains the segment number of the code segment, the area of memory where the instructions currently being executed are stored. Since a segment is 64K long, the length of a program's collection of instructions is often limited to 64K; a longer program requires that the contents of CS be changed while the program is running.
- ✓ Similarly DS contains the segment number of the data segment, the area of memory where most data is stored.
- ✓ The SS register contains the segment number of the stack segment, where the stack is maintained.
- ✓ The ES register contains the segment number of the extra data segment that could have multiple uses.
- ✓ The FS and GS registers were added with the 80386 and make possible easy access to two additional data segments.

2.2 PC Hardware: The CPU

With the flat 32-bit memory model, the operating system will give each of CS, DS, ES, and SS the value. Recall that this is a pointer to table entry that includes the actual starting address of the segment. That table also includes the size of your program, so that the operating system can indicate an error if your program accidentally or deliberately attempts to write in another area.

The 32-bit instruction pointer, or EIP register, cannot be directly accessed by an assembly language programmer. The CPU has to fetch instructions to be executed from memory, and EIP keeps track of the address of the next instruction to be fetched.

If this were a older, simpler computer architecture, the next instruction to be fetched would also be the next instruction to be executed.

2.2 PC Hardware: The CPU

In addition to prefetching instructions, an 80×86 CPU actually starts execution of an instruction before it finishes execution of prior instructions. This use of pipelining increases effective processor speed.

The final register is called the flags register. The name EFLAGS refers to this register, but this mnemonic is not used in instructions. Some of the flag register's 32 bits are as follows:

Bit	Mnemonic	Usage
0	CF	Carry Flag
2	PF	Parity Flag
6	ZF	Zero Flag
7	SF	Sign Flag
10	DF	Direction Flag
11	OF	Overflow Flag

2.2 PC Hardware: The CPU

- ✓ Bit 11 is the overflow flag (OF). It is set to 0 following an addition in which no overflow occurred, and to 1 if overflow did occur.
- ✓ Similarly, bit 0, the carry flag (CF), indicates the absence or presence of a carry out from the sign position after an addition.
- ✓ Bit 7, the sign flag, contains the left bit of the result after some operations.
- ✓ Since the left bit is 0 for a nonnegative two's complement number and 1 for a negative number, SF indicates the sign.
- ✓ Bit 6, the zero flag (ZF) is set to 1 if the result of some operation is zero, and to 0 if the result is nonzero (positive or negative).
- ✓ Bit 2, the parity flag, is set to 1 if the number of 1 bits in a result is even and to 0 if the number of 1 bits in the result is odd.
- ✓ Other flags will be described later when their uses will be clearer.

2.2 PC Hardware: The CPU

As an example of how flags are set by instructions, consider again the instruction

```
add eax, 158
```

This instruction affects CF, OF, PF, SF, and ZF.

Suppose that EAX contains the word FF FF FF F3 prior to execution of the instruction.

Since 158_{10} corresponds to the word 00 00 00 9E, this instruction adds FF FF FF F3 and 00 00 00 9E, putting the sum 00 00 00 91 in the EAX register.

It sets the carry flag CF to 1 since there is a carry, the overflow flag OF to 0 since there is no overflow, the sign flag SF to 0 (the leftmost bit of the sum 00 00 00 91), and the zero flag ZF to 0 since the sum is not zero. The parity flag PF is set to 0 since 0000 0000 0000 0000 0000 0000 1001 0001 contains three 1 bits, an odd number.

2.2 PC Hardware: The CPU

80×86 registers

name	length (bit)	Use/comments
EAX	**32**	**accumulator, general use;** low-order-word AX, divided into bytes AH and AL
EBX	**32**	**general use;** low-order-word BX, divided into bytes BH and BL
ECX	**32**	**general use;** low-order-word CX, divided into bytes CH and CL
EDX	**32**	**general use;** low-order-word DX, divided into bytes DH and DL
ESI	**32**	source index; source address in string moves, array index
EDI	**32**	destination index; address of destination, array index
ESP	**32**	stack pointer; address of top of stack

2.2 PC Hardware: The CPU

80×86 registers contd.

name	length (bit)	Use/comments
EBP	32	base pointer; address of reference point in the stack
CS	16	holds selector for code segment
DS	16	holds selector for data segment
ES	16	holds selector for extra segment
SS	16	holds selector for stack segment
FS	16	holds selector for additional segment
GS	16	holds selector for additional segment
EIP	32	instruction pointer; address of next instruction to be
EFLAGS	32	collection of flags, or status bits

2.3 PC Hardware: Input/Output Devices

without input devices to get data or output devices to display data, the computer is not usable for many An assembly language programmer has multiple ways to look at I/O devices.

- ✓ At the lowest level, each device uses a collection of addresses or ports in the I/O address space. The 80×86 architecture has 64K port addresses, and a typical I/O device uses three to eight ports. These addresses are distinct from ordinary memory addresses.
- ✓ Instead of using separate port addresses, a computer system can be designed to use addresses in the regular memory address space for I/O device access. Such a design is said to use **memory-mapped input/output**.
- ✓ A common approach is to use procedures that do the busywork of communicating with the devices, while allowing the programmer a higher-level, more logical view of the devices. Many such routines are still fairly low-level.

2.4 PC Software

Without software, computer hardware is virtually useless. Software refers to the programs or procedures executed by the hardware.

different types of software:

✓ PC Software: The Operating System

A general-purpose computer system needs an operating system to enable it to run other programs. The DOS operating systems provide the user a command line interface. DOS displays a prompt (such as C:\>) and waits for the user to type a command. When the user presses the Enter (or Return) key, DOS interprets the command. The command may be to perform a function that DOS knows how to do or it may be the name of a program to be loaded and executed.

Many users prefer a graphical user interface that displays icons representing tasks or files, so that the user can make a selection by clicking on an icon with a mouse. Microsoft Windows provided a graphical user interface for PCs. The versions through Windows 3.1 enhanced the operating environment, but still required DOS to run. Windows 95 included a major revision of the operating system, which was no longer sold separately from the graphical user interface. In Windows 95 the graphical user interface became the primary user interface, although a command line interface was still available.

2.4 PC Software

different types of software:

✓ PC Software: Text Editors

text editor is a program that allows the user to create or modify text files that are stored on disk. A text file is a collection of ASCII codes. The text files of most interest in this book will be assembly language source code files, files that contain assembly language statements. An editor is sometimes useful to prepare a data file as well.

Later versions of MS-DOS and Windows 95 provide a text editor called **Edit**. Edit is invoked from the command line prompt. This full-screen editor uses all or part of the monitor display as a window into the file. The user can move the window up or down (or left or right) to display different portions of the file. To make changes to the file, cursor control keys or the mouse are used to move the cursor to the place to be modified, and the changes are entered.

Microsoft Windows includes a text editor called **Notepad**. It is also a full-screen editor. Either Edit or Notepad work well for writing assembly language source programs.

2.4 PC Software

different types of software:

✓ PC Software: Language Translators and the Linker

Language translators are programs that translate a programmer's source code into a form that can be executed by the computer. These are usually not provided with an operating system. Language translators can be classified as **interpreters**, **compilers**, or **assemblers**.

Interpreters directly decipher a source program. To execute a program, an interpreter looks at a line of source code and follows the instructions of that line. Basic or Lisp language programs are often executed by an interpreter.

Compilers start with source code and produce object code that consists mostly of instructions to be executed by the intended CPU. High-level languages such as Pascal, Fortran, Cobol, C, and C++ are commonly compiled. The object code produced by a compiler must often be linked or combined with other object code to make a program that can be loaded and executed. This requires a utility called a linker, usually provided with a compiler.

An assembler is used much like a compiler, but translates assembly language rather than a high-level language into machine code. The resulting files must normally be linked to prepare them for execution. Because assembly language is closer to machine code than a high-level language, the job of an assembler is somewhat simpler than the job of a compiler.

2.4 PC Software

A debugger allows a programmer to control execution of a program, pausing after each instruction or at a preset breakpoint. When the program is paused, the programmer can examine the contents of variables in a high-level language or registers or memory in assembly language. A debugger is useful both to find errors and to "see inside" a computer to find out how it executes programs

Integrated development environments use a single interface to access an editor, a compiler, and a linker. They also initiate execution of the program being developed and frequently provide other utilities, such as a debugger. An integrated development environment is convenient, but may not always be available for a particular programming language.

2.4 PC Software

Using again the assembly language instruction

```
add eax, 158
```

is translated by the assembler into the five bytes **05 00 00 00 9E**.

The first byte **05** is the **op code** (operation code), which says to add the number contained in **the next four bytes** to the doubleword already in **the EAX register**. The doubleword 00 00 00 9E is the 2's complement representation of 158_{10}.

Chapter Summary

This chapter has discussed the hardware and software components that make up a PC microcomputer system.

The major hardware components are the **CPU** and **memory**. The CPU executes instructions and uses its internal registers for instruction operands and results and to determine addresses of data and instructions stored in memory. Objects in memory can be addressed by 32-bit addresses. In a **flat memory model**, such addresses are effectively actual addresses. In a **segmented memory model**, addresses are calculated from a starting address determined from a segment number and an offset within the segment.

Input/output at the hardware level uses a separate collection of addresses called ports. Input/output is often done through operating systems utilities.

Chapter Summary

An **operating system** is a **vital software component**. Through a command line or a graphical user interface, it interprets the user's requests to carry out commands or to load and execute programs.

A **text editor**, **an assembler**, and **a linker** are necessary software tools for the assembly language programmer. These may be separate programs or available as part of an integrated development environment. A debugger is also a useful programmer's tool.

End Of Chapter 2