```
01010101
10001011 11101100
01001100
01001100
01010110
01010111
10111111 00000011 00000000
10111110 00000010 00000000
10001011 11000111
00000011 11000110
10001001 01000110 11111110
01011111
01011110
10001011 11100101
01011110
11000011
```

FIGURE 2-7
Machine language is the language of the microprocessor. This machine language program adds 3 + 2 and stores the result.

Analog vs. Digital

The early computers used gears, wheels, or other mechanical devices to represent numbers. These are called analog devices. Something is an ***analog*** *device if it uses quantities that are variable or exist in a range. For example, a second hand on a clock is an analog device because it represents a value with a continuously variable quantity.*

Electronics made digital devices the basis for computers. A ***digital*** *device uses switches in combination (or digits) to represent something in the real world. For example, rather than have a second hand rotate at a fixed speed to represent a value, a digital clock counts the seconds electronically.*

On the Net

In this section you learned about the base 2 number system called the binary number system. Another number system used frequently in computer programming is a base 16 number system called the hexadecimal number system. To learn how to use the hexadecimal system and to see the machine language program from Figure 2-7 represented in the hexadecimal system, go to http://www.ProgramCPP.com. See topic 2.1.1.

SECTION 2.1 QUESTIONS

1. Define data.
2. How many bits are in a byte?
3. What is the language called that is "understood" by the microprocessor?
4. How many combinations of bits are possible with three bits?
5. Looking at the table in Appendix A, add the decimal values of the ASCII characters that spell your first name. Remember to use a capital letter where necessary. Write each character, its decimal equivalent, and the sum of all the ASCII values.

Programming Languages

Supplying computers with instructions would be extremely difficult if machine language were the only option available to programmers. Fortunately, special languages have been developed that are more easily understood. These special languages, called ***programming languages,*** provide a way to program computers using instructions that can be understood by computers and people.

Like human languages, programming languages have their own vocabulary and rules of usage. Some programming languages are very technical, and others are made to be as similar to English as possible. The programming languages available today allow programming at many levels of complexity.

ASSEMBLY LANGUAGE

The programming language most like machine language is ***assembly language.*** Assembly language uses letters and numbers to represent machine language instructions (see Figure 2-8). However, assembly language is still difficult for novices to read.

Assembly language programming is accomplished using an assembler. An ***assembler*** is a program that reads the codes the programmer has written and assembles a machine language program based on those codes.

LOW-LEVEL VS. HIGH-LEVEL LANGUAGES

Machine language and assembly language are called ***low-level languages.*** In a low-level language, it is necessary for the programmer to know the instruction set of the microprocessor in order to program the computer. Each instruction in a low-level language corresponds to one or only a few microprocessor instructions. In the program in Figure 2-8 each assembly language instruction corresponds to one machine language instruction.

Most programming is done in ***high-level languages.*** In a high-level language, instructions do not necessarily correspond one-to-one with the instruction set of the microprocessor. One command in a high-level language may represent many microprocessor instructions. Therefore, high-level languages reduce the number of instructions that must be written. A program that might take hours to write in a low-level language can be done in minutes in a high-level language. Programming in a high-level language also reduces the number of errors because the programmer doesn't have to write as many instructions, and the instructions are easier to read. Figure 2-9 shows a program written in three popular high-level languages. Like the machine language and assembly language programs you saw earlier, these high level programs add the numbers 3 and 2 together.

Another advantage of programs written in a high-level language is that they are easier to move among computers with different microprocessors. For example, the microprocessors in Macintosh computers use a different instruction