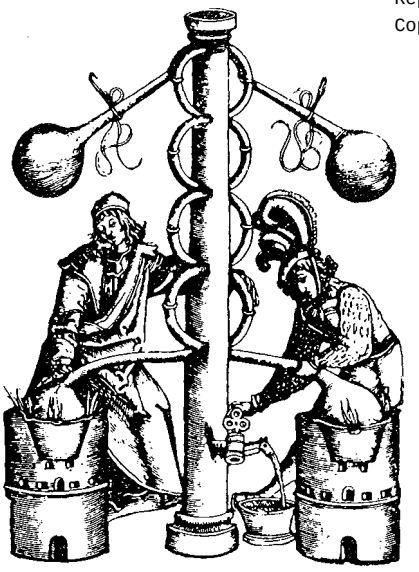
Reprinted with permission from American Laboratory, Vol. 10, No. 3, pp. 153-154.



Microcomputer interfacing

Integer addition and subtraction

By Peter R. Rony, Jonathan A. Titus, Christopher A. Titus, and David G. Larsen

MOST 8-BIT microprocessors, such as the 8080A, Z-80, 6800, 6502, and F-8, can add and subtract only 8-bit numbers, which can represent only decimal quantities between 0 and 255. This is not enough resolution or dynamic range for many applications. Consequently, *multiple-precision* and *jloating-point numbers* are used.

The term multiple precision refers to the use of two or more computer words to represent a numeric quantity. In the above-mentioned microprocessor chips, a computer word is called a *byte,* and is 8 bits long. A double-precision number, therefore, contains 2 bytes, or 16 bits, and can represent any unsigned integer number between 0 and 65,535. Similarly, a triple-precision number contains 3 bytes, or 24 bits, and can represent any unsigned integer number between 0 and 16,777,216. Despite this ability to represent very large numbers, multiple-precision numbers do have their limitations, especially when units such as picograms, liters per second, and kilograms all appear in a single equation.

Floating-point numbers are frequently used for scientific and engineering calculations because they can represent quantities that vary greatly in magnitude. The term floating-point number refers to a computer quantity that is usually composed of two parts, a *mantissa* and an *exponent.* For 8-bit microprocesses, a floating-point number is often represented by a 16-bit mantissa and an 8-bit exponent. Since the exponent and mantissa may be either positive or negative, one bit in each is used as a *sign bit.* This means that the 3-byte floating-point number (IS-bit mantissa plus sign bit, 7-bit exponent plus sign bit) can represent numbers between 32,767 x 2^{-127} and 32,767 x 2^{+127}, which correspond to the decimal number range 1.93 x 10^{-34} t05.58 x 10^{+}". It is quite common for the mantissa to contain an implied binary decimal point, and thus to represent binary numbers between O and 1.000 or between 0.500 and 1.000.

Unfortunately, a *jloating-point package,* which is a collection of subroutines that performs the addition, subtraction, multiplication, and division of floating-point numbers, is a complex program. The Intel 8080 floating-point package, [which was written by O.c. Juelich and had its origin in an earlier 8008 floating-point package written by C.E. Ohme, requires 865 bytes of read-only memory (ROM) and 24 bytes of read/write memory. Few programmers write floating-point packages because they are available from computer manufacturers or their respective user's groups.

Integer, or *jixed-point,* mathematical programs are relatively easy to write. For 8080A-based microcomputers, the add (ADD) and add-with-carry (ADC) instructions are used to write integer addition subroutines and programs. These instructions are used to add not only 8-bit numbers, but also 16-bit, 24-bit, and larger numbers. The add-with-carry instructions are particularly useful in this regard, since they add the content of the *carry bit* to the sum of two 8-bit bytes. The carry bit is also either set or cleared as a result of this addition.

Figure 1 *Triple-precision integer-addition subroutine.*

```
ADD3,   MVIC    /LOAD THE C REGISTER WITH THE
        003     INUMBER OF 8-BIT BYTES TO BE ADDED.
        LXIH    /LOAD REGISTER PAIR H WITH THE
        IACC    IMEMORY ADDRESS WHERE ONE OF THE
        O       IARGUMENTS IS STORED.
        XRAA    /CLEAR THE A REGISTER AND CARRY
ADDAGN, LDAXD   IGET ONE ARGUMENT INTO A
        ADCM    IADD THE OTHER ARGUMENT TO IT
        MOVMA   ISAVE THE RESULT BACK IN MEMORY
        INXD    IINCREMENT ONE MEMORY ADDRESS AND
        INXh    ITHEN INCREMENT THE OTHER.
        DCRC    IDECREMENT THE BYTE COUNT IN C
        JNZ     /IF THE COUNT IS NON-ZERO,
        ADDAGN  IPERFORM THE ADDITION AGAIN
        O
        RET     10THERWISE, RETURN FROM THE SUBROUTINE
```

Dr. Rony, Department of Chemical Engineering, and Mr. Larsen, Department of Chemistry, are with the Virginia Polytechnic Institute and State University, Mr. J. Tillis is *President and Mr. C. Tillis* is *Consultant, Tychon, Inc.*

```
SUB3,   MVIC    /LOAD THE C REGISTER WITH THE
        003     INUMBER OF 8-BIT BYTES TO BE SUBTRACTED
        I..XIH  /LOAD REGISTER PAIR H WITH THE
        IACC    IMEMORY ADDRESS WHERE ONE OF THE
        O       IARGUMENTS I S STORED.
        XRAA    /CLEAR THE A REGISTER AND CARRY
SUBAGN, LDAXD   IGET ONE ARGUMENT INTO A
        SBBM    ISUBTRACT THE OTHER ARGUMENT FROM IT
        MOVMA   ISAVE THE RESUI..T BACK IN MEMORY
        INXD    IINCREMENT ONE MEMORY ADDRESS AND
        INXH    ITHEN INCREMENT THE OTHER.
        DCRC    IDECREMENT THE BYTE COUNT IN C
        JNZ     IIF THE COUNT IS NON-ZERO,
        SUBAGN  IPERFQRM THE SUBTRACTION AGAIN
        O
        RET     /OTHERWISE, RETURN FROM THE SUBROUTINE
```

Figure 2 *Triple-precision integer-subtraction subroutine.*

A typical triple-precision integer-addition subroutine for an 8080A microcomputer is shown in *Figure 1.* The subroutine adds two 3-byte (24-bit) numbers that are stored in memory and returns the sum back to memory. When subroutine ADD3 is called, register pair D must contain the memory address where the least significant byte (LSBy) of one of the numbers is stored in memory. The more significant bytes of the 3-byte number must be stored in consecutive memory locations at the next two higher memory addresses. At location ADD3, the C register is loaded with the number of bytes that are to be added, in this case, three. Register pair H is then loaded with the memory address where the other 24-bit number is stored. The first of the three memory locations used for this storage is assigned the symbolic address lACC (Integer ACCcumulator). It should be noted that it is always possible to use a group of consecutive bytes in memory to create a multibyte accumulator, which contains one of the operands and in which the final result of an arithmetic or logical operation is stored.

The next instruction that the 8080A executes, XRAA, clears the carry to a logic zero. This instruction must be included in the subroutine because the state of the carry when the subroutine is called is unknown and the carry from some previous operation should not be added into the 24-bit result.

At ADDAGN (ADD AGaiN), a single byte is moved to the A register from the memory location addressed by register pair D. The content of the memory location addressed by register pair H is then added to the content of the A register, and the result of this addition is copied into the memory location addressed by register pair H. Both register pairs, D and H, are then incremented by one with the aid of the lNXD and lNXH instructions, respectively. The *byte count,* which is contained in the C register, is then decremented by one.

When the content of the C register is decremented to zero, the 8080A returns from the subroutine. If the content of the C register is not zero, the 8080A jumps back to ADDAGN and adds the next two bytes in sequence. Note that the XRAA instruction is used to clear the carry to a logic zero only when the subroutine is first called.

Subroutine ADD3 can be easily modified to add a 4-, 7-, or even a 200-byte number simply by changing the immediate data byte of the MVIC instruction. Of course, if 4-byte numbers are to be added, a 4-byte integer accumulator must be provided to store the *accumulated* result.

A triple-precision integer-subtraction subroutine for an 8080A microcomputer is listed in *Figure* 2. The program is almost identical to the integer addition program given in Figure I; the instruction ADCM in Figure I is replaced by SBBM in Figure 2. Note that the content of the integer accumulator is subtracted from the content of memory addressed by register pair D, and the result of the subtraction is stored in the integer accumulator.

In a subsequent column, we shall discuss integer multiplication and division subroutines. Another column will describe the application of these subroutines to the *smoothing* or *filtering* of data acquired from an analog-to-digital converter (ADC).

Reference

1. JUELlCH, O.C., "Elementary function package," lnsite" (Intel Corp., User's Library, Santa Clara, Calif. 1975).