A Fast, Ancient Method for Multiplication

Jostein Nyberg
Odv Solbergsv 100
Oslo 9, Norway

There are several ancient algorithms that adapt surprisingly well to the computer. One such example is the "Russian Peasant Method" for multiplication, which was discovered by Western visitors to Russia in the nineteenth century. However, the method is actually much older than that. It was used by Egyptian mathematicians as early as 1800 BC, although it was not stated as a completely systematic algorithm.

To explain this method, let A and B denote two numbers. A can be any number, while B must be a nonnegative integer. The problem is to calculate their product P. The method is:

1. Let $P = 0$
2. If B is odd, let $P = P + A$
3. Let $A = A + A$
4. Let B = integer part of $B/2$
5. If B is nonzero, repeat from step 2; otherwise the algorithm terminates.

An example will clarify how this works. Here are successive values of A and B, when their initial values are 175 and 18:

A:	175	350	700	1400	2800	5600
B:	18	9	4	2	1	0

Adding those As for which the corresponding Bs are odd, we have:

$$P = 350 + 2800 = 3150$$

which is the required result of 175 times 18. You may wish to try more examples to convince yourself that this procedure works correctly.

Notice that if A and B are unsigned integers expressed in binary, the doubling of A in step 3 can be performed by a left shift of A. Finding the integer part of $B/2$ in step 4 corresponds to a right shift of B. Furthermore, the B in step 2 is odd if its least-significant bit is 1.

Listing 1 shows a relocatable subroutine written in 6502 assembly language; also included is the hexadecimal object code. When the subroutine is entered, it is assumed that the low- and high-order bytes of A are found at memory locations 0000 and 0001 (hexadecimal), respectively. The low- and high-order bytes of B are found at locations 0002 and 0003, respectively. When the end of the subroutine is reached, locations 0004 and 0005 will contain the product P. If needed, the routine can be made shorter and faster by using the index registers (X and Y) for the product, instead of memory locations.

It is assumed here that P does not exceed 16 bits. If three or four bytes are required, it's relatively easy to expand the subroutine.

Multiplication routines similar to the one in listing 1 are found in arithmetic software and are coded in various languages. This does not mean that the routines' inventors were intentionally using the Russian Peasant Method. Probably, they were just imitating the familiar pencil-and-paper method for multiplication. As a matter of fact, when the numbers involved are binary and the algorithms are executed using the same instruction set, these two methods are identical.

A multiplication routine that looks slightly different, listing 1b, is often shown in microprocessor and microcomputer manuals. As a rule, this method should not be used. The loop starting at HALF is always entered sixteen times. Thus, the looping can continue to no purpose after B reaches 0.

The Russian Peasant Method can be modified to per-

form exponentiation. By setting *P* equal to 1 in step 1 and changing the addition in steps 2 and 3 to multiplication, the resulting value of *P* will be *A* raised to the power of *B*. Of course, steps 2 and 3 now assume that a multiplication routine is available. This method for exponentiation was stated by a Persian mathematician in the year 1414. ■

Reference

1. Knuth, D E. *The Art of Computer Programming*, Vol 2. Reading MA: Addison-Wesley, 1969. Pages 399 and 400.

Listing 1: *Relocatable subroutines for fast integer arithmetic on the MOS Technology 6502 microprocessor. Listing 1a shows a machine-language routine for multiplication by the Russian Peasant Method; listing 1b gives a version seen frequently in textbooks.*

(1a)

Object Code		Label	Mnemonic	
A9	00	MULT	LDA	#0
85	04		STA	PLOW
85	05		STA	PHIGH
46	03	HALF	LSR	BHIGH
66	02		ROR	BLOW
90	0D		BCC	DOUBLE
18			CLC	
A5	04		LDA	PLOW
65	00		ADC	ALOW
85	04		STA	PLOW
A5	05		LDA	PHIGH
65	01		ADC	AHIGH
85	05		STA	PHIGH
06	00	DOUBLE	ASL	ALOW
26	01		ROL	AHIGH
A5	02		LDA	BLOW
05	03		ORA	BHIGH
D0	E3		BNE	HALF
60			RTS	

(1b)

Object Code		Label	Mnemonic	
A9	00	MULT	LDA	#0
85	04		STA	PLOW
85	05		STA	PHIGH
A2	10		LDX	#$10
46	03	HALF	LSR	BHIGH
66	02		ROR	BLOW
90	0D		BCC	DOUBLE
18			CLC	
A5	04		LDA	PLOW
65	00		ADC	ALOW
85	04		STA	PLOW
A5	05		LDA	PHIGH
65	01		ADC	AHIGH
85	05		STA	PHIGH
06	00	DOUBLE	ASL	ALOW
26	01		ROL	AHIGH
CA			DEX	
D0	E6		BNE	HALF
60			RTS	