Popular Electronics®

WORLD'S LARGEST-SELLING ELECTRONICS MAGAZINE

SEPTEMBER 1978/$1

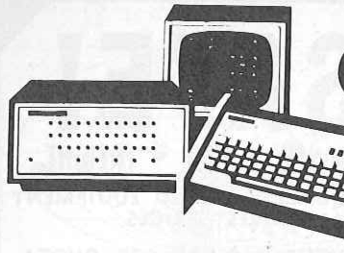
Analog-to-Digital with Two IC's
Converting Schematics to PC Patterns

Special Focus On Audio

NEW! METAL TAPE NEW! IHF AMPLIFIER STANDARD
NEW! HI-FI TV SOUND PLUS! HOME DISCO PREAMP/MIXER FOR $110

Tested In This Issue

Sony "Class D" Switching Amplifier
Electro-Voice "Series II" Speaker Systems
Panasonic 5-Band Portable Receiver



Computer Bits

By Hal Chamberlin

COMPUTER ARITHMETIC—MULTIPLY AND DIVIDE

IN OUR July column, the simpler multiple precision arithmetic operations were described. This month, we will discuss multiple precision multiplication and division. These two operations, which are invariably absent even in single precision form on 8-bit processors, make the difference between simple number shuffling and really sophisticated computing.

Unlike addition, subtraction, and the other operations described earlier, multiplication and division are *iterative* in nature. Although both could be done by successive addition and subtraction respectively, a very slow routine would be the result. For example, 16-bit by 16-bit multiplication could require over 30,000 iterations with straight successive addition and take well over a second even on a fast microprocessor. Much faster methods using addition/subtraction along with *shifting* can cut the same operation down to 16 somewhat more complex iterations requiring about 850 microseconds total. However, even this is too slow for some applications, so hardware multiplication/division boards, or newer 16-bit microprocessors with multiply and divide instructions built-in, become very attractive.

Another complication is that the result of, say, a 16-by-16-bit multiplication will very likely require more than 16 bits to represent. In fact, the product of two 16-bit integers can require as many as 32 bits. Division conversely always forms a quotient that will fit in half as many bits as the dividend and the divisor is restricted to half the length of the dividend. Thus a double precision (16-bit) multiply routine would take two 2-byte factors and produce a 4-byte product.

A division routine would accept a 4-byte dividend and 2-byte divisor and generate a 2-byte quotient and possibly a 2-byte remainder. Often 4-byte addition, subtraction, etc. is needed in a double precision package to facilitate handling 4-byte intermediate results without loosing accuracy.

Multiplication. The shift and add multiplication algorithm to be described is the fastest for software implementation on a microprocessor. For simplicity and generality, the routine is designed to multiply *unsigned* numbers. The more usual signed operands and result are handled by *correcting* the unsigned product.

The multiply routine uses two pseudo registers in memory. MPCD is two bytes long and holds the multiplicand. PROD must be 4 bytes long as discussed earlier. Before multiplication, the least significant two bytes of PROD hold the multiplier and the most significant two bytes are normally zero. If desired, a 2-byte number can be placed there and it will be automatically added to the product with no extra execution time required. After multiplication, PROD contains the 4-byte product; MPCD is unchanged.

The actual multiplication proceeds much like a decimal multiplication on paper. Each digit of the multiplier multiplies the entire multiplicand creating a series of partial products which are staggered left and added up to give the full product. For binary numbers however, each partial product is either zero for a zero multiplier bit or equal to the multiplicand for a one multiplier bit. Also, it is more efficient to add the partial products into a running total as they are formed. Bit-by-bit examination of the multiplier and staggered positioning of the partial products as they are added are both handled by quadruple shifting PROD alone.

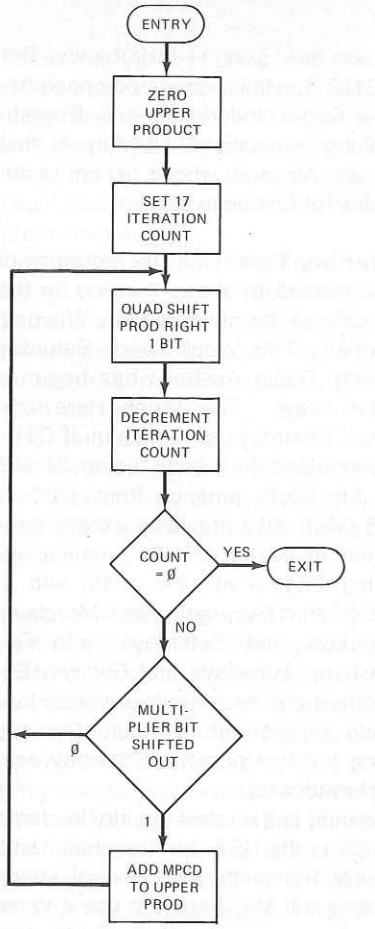


Fig. 2. Multiply flowchart.

A register diagram and a flowchart of the multiply routine are shown in Figs. 1 and 2. Sixteen full iterations are required to form the product, while a seventeenth iteration performs a final shift to position it properly. Each iteration starts with a right shift of all 4 bytes of PROD. This action simultaneously shifts the current sum of partial products right and puts the next multiplier bit to be examined into the carry flag where it is easily tested. After testing for completion, the carry flag (which holds the current multiplier bit) is tested. If it is a zero, the iteration is complete. If it is a one, then the multiplicand is *double-precision* added to the upper 2 bytes of PROD. Quad-precision addition of partial products is not needed which is one reason for the efficiency of this algorithm.

It is possible however for this addition to overflow. Fortunately, the overflow bit is retained in the carry flag and it will be shifted into the most significant bit of PROD during the next iteration. As the multiplication progresses, the multiplier is pushed off the right end of PROD while the product expands. There is never any interference between the two.

The correction needed for signed operands is actually quite simple. First, the numbers are multiplied together *as-is* with the above unsigned multiply subroutine. After multiplication, the sign bit of each factor is tested. If a factor is negative, the *other* factor is double-precision subtracted from the *upper half* of the product. The final result is a properly signed product. If signed fractional numbers were being multiplied, it is necessary to shift the product left one position after correction. The signed fractional result then is the *leftmost* two bytes of PROD.

Division. Division (Figs. 3 and 4) is the exact reverse of multiplication. Again, an unsigned algorithm will be used with correction applied for signed operands. For clarity, a different set of pseudo registers will be used but in practice they would occupy the same memory locations as those for multiply. DVND is 4 bytes long and holds the quad precision dividend. DVSR is two bytes and holds the divisor. After division, the quotient is in the low two bytes of DVND and the remainder is in the high two bytes.

A division iteration starts by subtracting the divisor from the high two bytes of the dividend and saving the difference. An underflow, evidenced by the carry flag being off after the subtraction, prevents updating of the upper divisor and a zero quotient bit to be recorded. If there was no underflow, the subtraction result is copied into the upper divisor and a quotient bit of one is produced. Following this the entire dividend is shifted left and the quotient bit, which is contained in the carry flag, is shifted in on the right. Only zeroes are ever shifted out at the left. As the division progresses, the divi-

EXAMPLES OF MULTIPLY AND DIVIDE SUBROUTINES

```
                       ;       16 X 16 UNSIGNED MULTIPLY SUBROUTINE
                       ;       ENTER WITH UNSIGNED MULTIPLIER IN PROD+2 AND PROD+3
                       ;       ENTER WITH UNSIGNED MULTIPLICAND IN MPCD AND MPCD+1
                       ;       RETURN WITH 16 BIT UNSIGNED PRODUCT IN PROD (HIGH) THROUGH
                       ;       PROD+3 (LOW)

0000 A900     UNSMPY:  LDA     #0            ; CLEAR UPPER PRODUCT
0002 8565              STA     PROD
0004 8566              STA     PROD+1
0006 A211     UNSMO:   LDX     #17           ; SET 17 MULTIPLY CYCLE COUNT
0008 18                CLC                   ; INITIALLY CLEAR CARRY
0009 204B00   UNSM1:   JSR     SRQL          ; SHIFT MULTIPLIER AND PRODUCT RIGHT 1
                                             ; PUTTING A MULTIPLIER BIT IN CARRY
000C CA                DEX                   ; DECREMENT AND CHECK CYCLE COUNT
000D F01A              BEQ     UNSM2         ; JUMP OUT IF DONE
000F 90FA              BCC     UNSM1         ; SKIP MULTIPLICAND ADD IF MULTIPLIER BIT
                                             ; IS ZERO
0011 A566              LDA     PROD+1        ; ADD MULTIPLICAND TO UPPER PRODUCT
0013 18                CLC
0014 656A              ADC     MPCD+1
0016 8566              STA     PROD+1
0018 A565              LDA     PROD
001A 6569              ADC     MPCD
001C 8565              STA     PROD
001E 4C0B00            JMP     UNSM1         ; GO FOR NEXT CYCLE
0021 60       UNSM2:   RTS                   ; RETURN

                       ;       DOUBLE PRECISION UNSIGNED DIVIDE SUBROUTINE
                       ;       ENTER WITH UNSIGNED DIVIDEND IN DVND (HIGH) THROUGH DVND+3 (LOW
                       ;       ENTER WITH UNSIGNED DIVISOR IN DVSR AND DVSR+1
                       ;       EXIT WITH UNSIGNED QUOTIENT IN DVND+2 AND DVND+3 AND UNSIGNED
                       ;       REMAINDER TIMES 2 IN DVND AND DVND+1 AND CARRY FLAG
                       ;       NO CHECK FOR OVERFLOW OR DIVIDE BY 0

0022 A211     UNSDIV:  LDX     #17           ; SET DIVIDE CYCLE COUNT
0024 18                CLC                   ; INITIALLY CLEAR CARRY
0025 A56C     UNSDV1:  LDA     DVND+1        ; SUBTRACT DIVISOR FROM HIGH DIVIDEND
0027 38                SEC                   ; AND SAVE DIFFERENCE IN Y AND A
0028 E570              SBC     DVSR+1
002A A8                TAY
002B A56B              LDA     DVND
002D E56F              SBC     DVSR
002F 9013              BCC     UNSDV2        ; SKIP IF UNDERFLOW
0031 856F              STA     DVSR          ; UPDATE HIGH DIVIDEND IF NOT
0033 98                TYA
0034 8570              STA     DVSR+1
0036 205800   UNSDV2:  JSR     RLQL          ; SHIFT DIVIDEND LEFT 1 BRINGING IN
                                             ; QUOTIENT BIT
0039 CA                DEX                   ; DECREMENT CYCLE COUNT
003A D0F1              BNE     UNSDV1        ; LOOP IF NOT DONE
003C 60                RTS                   ; RETURN

                       ;       QUAD SHIFT RIGHT SUBROUTINE
                       ;       ENTER AT RRQL TO SHIFT IN THE CARRY
                       ;       ENTER WITH QUAD PRECISION VALUE TO SHIFT IN PROD THROUGH PROD+3
                       ;       RETURNS BIT SHIFTED OUT IN CARRY

003D 6665     SRQL:    ROR     PROD          ; ROTATE RIGHT ENTRY
003F 6666              ROR     PROD+1
0041 6667              ROR     PROD+2
0043 6668              ROR     PROD+3
0045 60                RTS                   ; RETURN

                       ;       QUAD SHIFT LEFT SUBROUTINE
                       ;       ENTER AT RLQL TO SHIFT IN THE CARRY
                       ;       ENTER WITH QUAD PRECISION VALUE TO SHIFT IN DVND THROUGH DVND+3
                       ;       RETURNS BIT SHIFTED OUT IN CARRY

0046 2668     RLQL:    ROL     PROD+3        ; ROTATE LEFT ENTRY
0048 2667              ROL     PROD+2
004A 2666              ROL     PROD+1
004C 2665              ROL     PROD
004E 60                RTS                   ; RETURN

                       ;       STORAGE FOR THE MULTIPLY AND DIVIDE SUBROUTINES

004F 00000000 PROD:    .BYTE   0,0,0,0       ; 4 BYTES FOR MULTIPLIER AND PRODUCT
0053 0000     MPCD:    .BYTE   0,0           ; 2 BYTES FOR MULTIPLICAND
0055 00000000 DVND:    .BYTE   0,0,0,0       ; 4 BYTES FOR DIVIDEND, QUOTIENT, AND REM
0059 0000     DVSR:    .BYTE   0,0           ; 2 BYTES FOR DIVISOR
0000                   .END
```

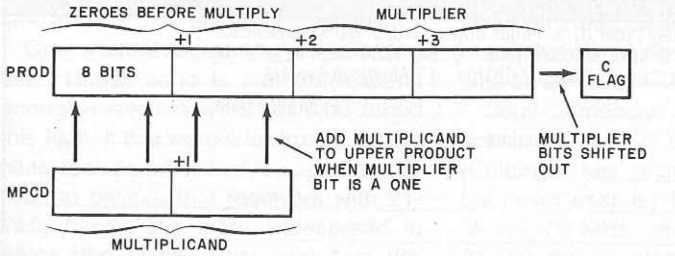

ZEROES BEFORE MULTIPLY
MULTIPLIER
+1
+2
+3
PROD
8 BITS
C FLAG
+1
MPCD
ADD MULTIPLICAND TO UPPER PRODUCT WHEN MULTIPLIER BIT IS A ONE
MULTIPLIER BITS SHIFTED OUT
MULTIPLICAND

Fig. 1. Multiply subroutine pseudo registers.

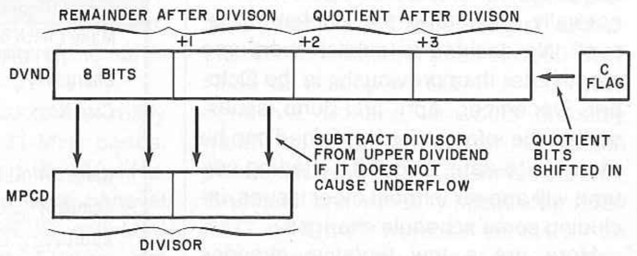


Fig. 3. Divide subroutine pseudo registers.

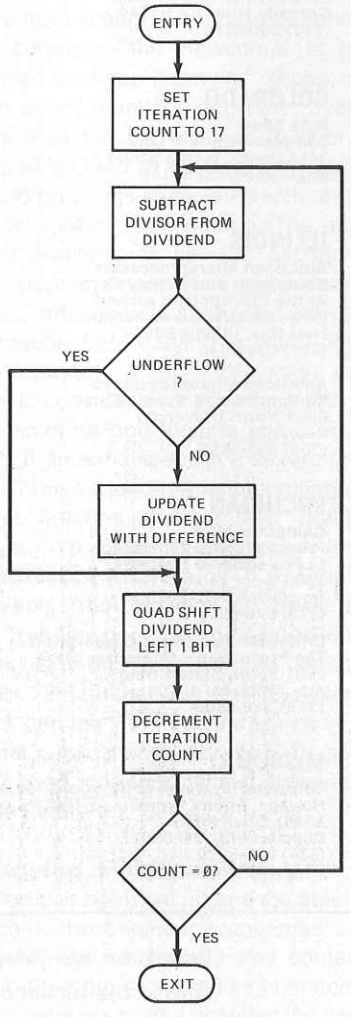

ENTRY
SET ITERATION COUNT TO 17
SUBTRACT DIVISOR FROM DIVIDEND
UNDERFLOW ?
YES
NO
UPDATE DIVIDEND WITH DIFFERENCE
QUAD SHIFT DIVIDEND LEFT 1 BIT
DECREMENT ITERATION COUNT
COUNT = Ø?
NO
YES
EXIT

Fig. 4. Divide flowchart.

dend is progressively eaten away as it shifts left and the quotient grows as additional bits are shifted in on the right. When complete, the quotient fills the right half of DVND and the remainder occupies the left half. As before there is no interference between the two numbers sharing DVND.

The most straightforward way of dividing signed operands is to compute their absolute values and record their signs. After dividing what are now signed *positive* numbers, the proper sign of the result is determined by applying the rules of algebra. Actually this amounts to nothing more than exclusive-OR'ing the sign bits of the original operands together to get the sign of the result. If this indicates that the quotient should be negative, then it is negated. When dividing signed fractions, it is necessary to shift the quotient right one position before doing the sign correction. Also, the fractional dividend is placed in the *upper* 2 bytes of the dividend and the lower 2 bytes are zeroed.

Unlike multiplication, it is possible for division to overflow. To avoid overflow, the upper two bytes of the dividend must be smaller than the divisor. This is easy to test for in the division routine itself and is often included. ◇

Software Sources

6800 and 6502 Calculator Programs. HUEY and HUEY LXVIII are calculator programs for the 6502 and 6800, respectively. Such functions as sine, exponent, log, arc tangent and others are pre-programmed, and the user can program other functions, Program area required is 2.25K. HUEY LXVIII, the 6800 version, resides at 1000 to 18FF hex. Hex listing with basic instruction is $10, and a manual containing commented disassembly and instructions on adding your own functions is $20. HUEY 6502 is available as a commented manual. with a zero-page location, for $20. Either version can also be custom-reassembled for any memory location for $5.00 above cost of the manual, or $25 total. The Bit Stop, P.O. Box 973, Mobile, AL 36601.

8080/Z80 Word Processing System. The Electric Pencil II is a character-oriented word processing system. Lines are not delineated, so any number of characters, words, lines or paragraphs may be inserted or deleted anywhere in the text, which opens up or closes as needed. Text lines are automatically formatted, with no typing of carriage returns required. Words partially completed when the end of a line is reached are shifted to the beginning of the following line. Text may be examined at will with variable-speed forward and reverse scrolling. Commands allow text strings to be located and/or replaced as desired. In printing, the Electric Pencil II automatically inserts carriage returns where they are needed with right-column justification, page numbering and page titling available. Diablo versions also include character spacing, bold face, multicolumn and bidirectional printing. Hardware requirements are: 8080 or Z80 processor, printer, video display, and disk or cassette interface. Base price, in Cuter or Tarbell cassette formula, for TTY or similar printers, is $100. Add $50 for Diablo Hy-term, $25 for North Star disk; CP/M compatible diskette systems are available for an additional $125, with such added features as file management, page-at-a-time scrolling, automatic word and record number tally, and others. From dealers or Michael Shrayer Software, 3901 Los Feliz Blvd., Los Angeles, CA 90027.

Amateur Radio

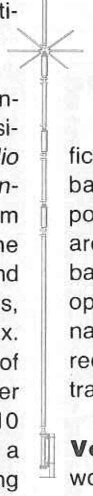
By Karl T. Thurber, Jr., W8FX

THE ANTENNA: GETTING OUT THE SIGNAL

OBTAINING the best station equipment is pointless if the signal isn't radiated properly. Fortunately, getting the signal out of the rig and into the air is *not* difficult at all, unless one lives under particularly restrictive conditions such as in a trailer park or an apartment where outside antennas are frowned upon or flatly prohibited. Assuming that you are not limited in this way, you would do well to stick with either a basic half-wave dipole antenna designed for single-band operation on your favorite band, a "trap" dipole for mulit-band work, or a multiband vertical.

Simple Dipoles. The proper dimensions for a half-wave dipole can be easily calculated or found in the *ARRL Radio Amateurs Handbook* or the *ARRL Antenna Book*, so we won't repeat them here. What bears repeating is that the antenna should be mounted as high and in the clear as possible, free of bends, and fed with good quality 75-ohm coax. At hf, RG-59/U is perfectly capable of handling more than 250 watts of power with very moderate line loss, even at 10 meters. Within the Novice subbands, a properly 'pruned' dipole (adjusted using an SWR bridge or directional wattmeter) should easily take power from the transmitter's pi output network without the aid of an antenna tuner.

Multi-band operation is possible by paralleling individual dipoles cut for the desired bands. The shorter dipoles can be suspended from the lowest-frequency (longest) one. You can use 4-wire rotator cable, cutting the longest conductors for the 80-meter band and cutting back the others to the proper length for resonant operation on the higher bands. All four wires are then joined on each side at the center point. The four dipoles will thus be fed with a single transmission line. This system generally works well, as the nonresonant dipoles are "not there" electrically speaking. However, one should be prepared to have to carefully prune dipole lengths after the antenna has been installed to get a low SWR because there is some electrical interaction between the dipoles. An antenna coupler is recommended for use with this antenna for two reasons: the antenna easily radiates unwanted harmonics of the fundamental frequency (which can lead to FCC citations known as "Pink Slips") and it is sometimes difficult to load up the transmitter on all bands. Commercially available trap dipoles generally work well unless they are physically very short for the lowest band in use (usually 80 meters). Unless operations are restricted to relatively narrow frequency ranges, trap dipoles require an antenna coupler for good transmitter loading.



Hustler's Model 4-BTV trap vertical covers 40 through 10 meters.

Verticals. The mulit-band vertical working against a good ground/radial system is an excellent radiator, and is well suited for DX chasing. It is very practical when space is at a premium, especially when the length of a 40- or 80-meter dipole would preclude its erection on a small city or suburban lot. All that's needed is vertical space and some room for radial wires, which can be fairly short if necessary. Hy-Gain, Mosley, and Gotham all make excellent verticals. They are generally mounted at ground level, using one or more long ground rods to get a low-resistance earth return in conjunction with four to a dozen or more buried radials. The radials may not be necessary in all cases. You may find the ground rod alone will provide good results, especially in areas with high ground conductivity, although theory dictates the installation of as many radials as possible to prevent the waste of r-f energy through ground losses. As in the case of the dipole, the vertical should be mounted in as clear an area as possible, away from TV antennas, power and telephone lines, and other signal obstacles. It should also be fenced-in or otherwise blocked off to prevent children from coming into contact with the antenna and suffering r-f burns. The Hy-Gain 18AVT, fed with 50-ohm coax, is a favorite among Novices and is well-known for excellent performance on 80 through 10 meters and its sturdy construction. The advanced Novice interested in chasing DX might consider Gotham's relatively inexpensive ($60) three-band (10/15/20 meters) quad which should work very well on 15 and 10 meters and come in handy once the license is upgraded.

Those able to erect full-size, single-band dipoles and verticals should certainly do so. Although there are literally hundreds of various hf antenna designs described in the amateur literature—more than enough to thoroughly confuse the beginner—the newcomer is best advised to "keep it simple." Zepps, random wires, rhombics and other exotica have their places, but lead to poor results when tried by the beginner, usually due to problems associated with matching and transmitter loading. The apartment dweller may have to experiment with loops, random wires and indoor dipoles, and should study the literature thoroughly before trying to pump power into a haywire antenna. He might even want to consider a window-mounted mobile antenna and loading coil arrangement if antenna space is a severe problem. I have found many interesting and novel antenna designs for "problem cases" in the paperback, *Ham Antenna Construction Projects* by J. A. Stanley, available from Howard W. Sams and Co. This little gem has some good ideas, particularly on "invisible" and restricted space antennas. The *ARRL Antenna Book* and the *73 Magazine* series of antenna publications also provide some very good ideas for difficult antenna installations.

Matching and Coupling. After the station has been set up and the antenna erected, the beginner sometimes finds that he just can't make any contacts. Most often the problem lies in matching the pi-network output circuit of the transmitter or transceiver to the antenna. If