The second in a series of tutorials for the beginner to intermediate machine language programmer

Machine Language Made BASIC

Part II: High Finances

By William P. Nee

First, let's review the SORT program from last month's article. (See Listing 1.) In the random number portion, we used LDY #$400 to indicate the upper left corner, but in the sort portion we used LDX #$400 for the same location. This was necessary because the random routine at $BF1F uses Register X for its own computations. We could have used Register X if we had saved it prior to executing $BF1F and recalled it afterwards; it was easier to use Register Y instead, since it was unchanged. It is a good idea to check any ROM routines for the registers they use prior to putting them in your program. If you have a choice between using Register X or Register Y, use Register X as it takes less memory and executes faster.

In the random portion of our program we checked to see if we had reached the end of the text screen, but in the sort portion we had to check to see if we were one space before the end. This was necessary because loading Register D with the contents of X actually loads Register A with the contents of X and loads Register B with the contents of X+1. If we allowed X to go to the end of the text screen, X+1 would move into the beginnings of graphics — and really start to make a mess!

Line 260 uses a branch (BLS) to see if one number is less than or the same as another number. Some branches compare signed numbers and some compare unsigned numbers. Figure 1 shows a comparison of branches for signed and unsigned numbers and what these branches check for.

So far, we've been using whole numbers generally between -32,000 and +32,000, but what about larger numbers or decimals? There is a way to input and save any number within the computer's range; however, it is only accurate to nine digits.

The routine at $A390 is the equivalent of LINE INPUT in BASIC. Whatever you input is stored in memory at $2DD in ASCII format. After executing $A390, $2DC will contain a zero, $2DD+ will be the ASCII numbers, and the end will be a zero. Register B will be the length of the input plus one, and Register X will be #$2DC. Any number you input can be preceded by +, -, &H (Hex), or O (Base 8).

The routine at $9F reads whatever is in a buffer whose location is stored in $A6/A7 and continues to read the buffer one byte at a time into Register A until a zero is reached.

Finally, the routine at $BD12 will change the ASCII numbers in Register A to floating point format in FP1. Putting all of these routines together gives us a SAVE subroutine. (See Listing 2.) Check your result by using the print subroutine from last month's article, Example 13A. (See Listing 3.)

Once a number is in FP1, it usually then has to be stored in some location. The easiest way to do this is to use the routine at $BC35 to transfer a number in FP1 to the location in Register X using either its name or location. It will take five bytes to completely store the number in floating point format, so reserve five bytes for each number you will be saving in your program.

Let's try the simple program shown in Listing 4 that will take any number, store it and then print it. Our print routine is good only for printing numbers, but BASIC has a PRINT USING command that gives you much more flexibility and lets you use the $, commas, +, -, etc. The routine at $BFA1 is the PRINT USING command for machine language; however, some setup is required.

First, determine the number of characters that will be to the right of the decimal, add one, and load this into Register A. Then determine the number of characters you will need to the left of the decimal (including the $ sign, commas, number signs, etc.) and load this into Register B. Register D is then stored in Location $D8/D9. The two numbers in $D8/D9 cannot total more than 17. If they do, you will get either

---

Bill Nee bucked the "snowbird" trend by retiring to Wisconsin from a banking career in Florida. He spends the long, cold winters writing programs for his CoCo.

a wrong answer or a Function Call error message.

Location $DA must contain a number indicating which format to use. The more common numbers are:

$DA	FORMAT
#$2	(-)number
#$4	number(-)
#$8	(+/-)number
#$C	number(+/-)
#$10	floating $
#$40	floating ,
#$50	floating $/,

Adding the numbers together will combine the results. Adding one to the number will print the result in exponential format.

If you need a PRINT @, load Register D with the @ location (+#$400) and store it in Location $88 (cursor location). Then you can load Register X with a message location minus one, and JSR $B99C will print the message. Try the program shown in Listing 5.

Note that there is a space before the actual message. This space does not appear when the message is printed at Location $420. Without the space we would have had to change the message location line to LDX #MSG-1. The message must end with a zero (FCB 0) to indicate the end of the message. Instead of FCB 0 we could have used FDB $0D00 and eliminated the JSR $B958, since either will print the carriage return (#$0D).

The comparison programs (listings 6 and 7) for this article are simple financial calculators. Each program asks for the annual interest rate, the number of months of the loan (term) and the amount borrowed (financed). The programs compute the monthly payment and print the answer in the PRINT USING "$#,###.##" format. You then have the option of inputting any new amount, term, or interest rate. If you run the machine language program from BASIC clear sufficient memory first (CLEAR 200, &H3000-1).

As a project, try to modify the program so it will compute the amount, term or monthly payments depending on what you input. Don't try to compute the rate — there is no exact formula for doing so. The basic formulas used in this program are:

rate = annual rate/1200
pv = ((1+r)**term)-1/r((1+r)**term)
monthly payment = amount/pv
(** is used as a symbol for exponential)

(Questions or comments concerning this tutorial may be directed to the author at Route 2, Box 216 C, Mason, WI 54846-9302. Please enclose an SASE when requesting a reply.) □

Shifts

BCC	Branch if carry clear (=0)
BCS	Branch if carry set (=1)

Unsigned Numbers

BHI	Branch if higher
BHS	Branch if higher or same
BLO	Branch if lower
BLS	Branch if lower or same
BEQ	Branch if equal (is 0)
BNE	Branch if not equal (is not 0)

Signed Numbers

BGE	Branch if greater than or equal (to 0)
BGT	Branch if greater (than 0)
BLE	Branch if less than or eual (to 0)
BLT	Branch if less (than 0)
BMI	Branch if minus
BPL	Branch if plus
BEQ	Branch if equal (is 0)
BNE	Branch if not equal (is not 0)

Figure 1: Assembly Language Branches

Listing 1:

```
            ORG   $3000
START       JSR   $A928     clear the text screen
            LDY   #$400     top left of text screen
LOOP1       LDD   #255      load register D with 255
            JSR   $B4F4     convert to a FP1 number
            JSR   $BF1F     get RND(255)
            JSR   $B3ED     put it in register D
            STB   ,Y+       put the CHR$ in register Y, move to next space
            CMPY  #$5FF     check to see if at bottom right of text screen
            BLS   LOOP1     if not, branch back to LOOP1
SORT        LDA   #1        create a
            STA   FLAG      test "flag"
            LDX   #$400     top left of text screen
LOOP2       LDD   ,X+       load register D with $400/401, move to $401
            PSHS  B         save the contents of $401
```

```
        CMPA ,S+      compare what's in $400 to what's in $401
        BLS  CONT     branch if it's less or equal to what's in $400
        EXG  A,B      if not, exchange the contents of $400 and $401
        STD  -1,X     put them back in $400 and $401
        CLR  FLAG     set the "flag" to zero
CONT    CMPX #$5FE    one away from bottom right of text screen?
        BLS  LOOP2    if not, branch back to LOOP2
        TST  FLAG     check the "flag"
        BEQ  SORT     if it's zero, sort again
        JSR  $ADFB    if not, wait for any input
        SWI           end the program (use RTS if in Basic)
FLAG    RMB  1        reserve one byte and call it "flag"
        END  START
```

Listing 2:

```
SAVE    JSR  $A390    input any number
        STX  $A6      put #$2DC in $A6/A7 (buffer location)
        JSR  $9F      increase the buffer location, store ASCII in "A"
        JSR  $BD12    make it a floating point number until reaches 0
        RTS           end the subroutine
```

Listing 3:

```
PRINT   JSR  $BDD9    transfer FP1 to buffer at $3DA
        LEAX -1,X     decrease location for sign
        JSR  $B99C    print buffer contents
        JSR  $B958    print a carriage return
```

Listing 4:

```
        ORG  $3000
SAVE    JSR  $A390    what's the number?
        STX  $A6      buffer starts at $2DC
        JSR  $9F      increase buffer, load "A" with first number
        JSR  $BD12    convert to floating point in FP1
        LDX  #NUMBER  where to store it
        JSR  $BC35    move the number in FP1 to (X)
PRINT   LDX  #NUMBER  where it is
        JSR  $BC14    move the number in (X) to FP1
        JSR  $BDD9    FP1 to ASCII format at $3DA
        LEAX -1,X     decrease buffer location
        JSR  $B99C    print buffer contents
        JSR  $B958    print a carriage return
        SWI           end of program
NUMBER  RMB  5
        END  SAVE
```

Listing 5:

```
        ORG  $3000
PRINT   LDD  #$420    print @ location $420 (second line down)
        STD  $88      store in cursor location
        LDX  #MSG     message location
        JSR  $B99C    print message
        JSR  $B958    print carriage return
        SWI
MSG     FCC  * THIS IS A SAMPLE MESSAGE*
        FCB  0
        END  PRINT
```

Listing 6: FINANBAS

```
1Ø CLS
2Ø INPUT"ANNUAL RATE";R:GOSUB 18
Ø
3Ø INPUT"MONTHLY TERM";T:GOSUB 2
ØØ
4Ø INPUT"AMOUNT FINANCED";AMOUNT
5Ø PMT=AMOUNT/PV
6Ø PRINT"MONTHLY PAYMENT IS - ";
7Ø PRINT USING"$#,###.##";PMT
8Ø PRINT"ANY NEW AMOUNT (Y/N)"
9Ø A$=INKEY$:IF A$="" THEN 9Ø
1ØØ IF A$="Y" THEN 4Ø
11Ø PRINT"ANY NEW TERM (Y/N)"
12Ø A$=INKEY$:IF A$="" THEN 12Ø
13Ø IF A$="Y" THEN 3Ø
14Ø PRINT"ANY NEW RATE (Y/N)"
15Ø A$=INKEY$:IF A$="" THEN 15Ø
16Ø IF A$="Y" THEN 2Ø
17Ø END
18Ø R=R/12ØØ
19Ø RETURN
2ØØ PV=((1+R)^T-1)/(R*(1+R)^T)
21Ø RETURN
```

Listing 7: FINANBIN

```
3ØØØ                ØØ1ØØ          ORG     $3ØØØ
3ØØØ BD   A928      ØØ11Ø START    JSR     $A928   CLEAR SCREEN
3ØØ3 8E   311Ø      ØØ12Ø INTR     LDX     #MSG1   FIND THE FIRST MESSAGE
3ØØ6 BD   B99C      ØØ13Ø          JSR     $B99C   PRINT IT
3ØØ9 17   ØØD8      ØØ14Ø          LBSR    SAVE
3ØØC 8E   3ØF7      ØØ15Ø          LDX     #RATE
3ØØF BD   BC35      ØØ16Ø          JSR     $BC35   SAVE THE RATE
3Ø12 8D   66        ØØ17Ø          BSR     CONV1
3Ø14 8E   312Ø      ØØ18Ø MONTHS   LDX     #MSG2   FIND MESSAGE 2
3Ø17 BD   B99C      ØØ19Ø          JSR     $B99C   PRINT IT
3Ø1A 17   ØØC7      ØØ2ØØ          LBSR    SAVE
3Ø1D 8E   3ØFC      ØØ21Ø          LDX     #TERM
3Ø2Ø BD   BC35      ØØ22Ø          JSR     $BC35   SAVE THE TERM
3Ø23 8D   68        ØØ23Ø          BSR     CONV2
3Ø25 8E   3131      ØØ24Ø AMOUNT   LDX     #MSG3   FIND MESSAGE 3
3Ø28 BD   B99C      ØØ25Ø          JSR     $B99C   PRINT IT
3Ø2B 17   ØØB6      ØØ26Ø          LBSR    SAVE
3Ø2E 8E   31ØB      ØØ27Ø          LDX     #AMNT
3Ø31 BD   BC35      ØØ28Ø          JSR     $BC35   SAVE THE AMOUNT
3Ø34 8E   3145      ØØ29Ø          LDX     #MSG4   FIND MESSAGE 4
3Ø37 BD   B99C      ØØ3ØØ          JSR     $B99C   PRINT IT
3Ø3A 8E   31Ø6      ØØ31Ø          LDX     #VARPV
3Ø3D BD   BC14      ØØ32Ø          JSR     $BC14   VARPV TO FP1
3Ø4Ø 8E   31ØB      ØØ33Ø          LDX     #AMNT
3Ø43 BD   BB8F      ØØ34Ø          JSR     $BB8F   AMOUNT*FP1
3Ø46 17   ØØ8B      ØØ35Ø          LBSR    PUSING
3Ø49 8E   315C      ØØ36Ø MORE     LDX     #MSG5   FIND MESSAGE 5
3Ø4C BD   B99C      ØØ37Ø          JSR     $B99C   PRINT IT
3Ø4F AD   9F AØØØ   ØØ38Ø LOOP5    JSR     [$AØØØ] WAIT FOR INPUT
3Ø53 27   FA        ØØ39Ø          BEQ     LOOP5
3Ø55 81   59        ØØ4ØØ          CMPA    #'Y
3Ø57 27   CC        ØØ41Ø          BEQ     AMOUNT
3Ø59 8E   3173      ØØ42Ø          LDX     #MSG6   FIND MESSAGE 6
3Ø5C BD   B99C      ØØ43Ø          JSR     $B99C   PRINT IT
3Ø5F AD   9F AØØØ   ØØ44Ø LOOP6    JSR     [$AØØØ] WAIT FOR INPUT
3Ø63 27   FA        ØØ45Ø          BEQ     LOOP6
3Ø65 81   59        ØØ46Ø          CMPA    #'Y
3Ø67 27   AB        ØØ47Ø          BEQ     MONTHS
3Ø69 8E   3188      ØØ48Ø          LDX     #MSG7   FIND MESSAGE 7
3Ø6C BD   B99C      ØØ49Ø          JSR     $B99C   PRINT IT
3Ø6F AD   9F AØØØ   ØØ5ØØ LOOP7    JSR     [$AØØØ] WAIT FOR INPUT
3Ø73 27   FA        ØØ51Ø          BEQ     LOOP7
3Ø75 81   59        ØØ52Ø          CMPA    #'Y
3Ø77 27   87        ØØ53Ø          BEQ     START
3Ø79 3F             ØØ54Ø          SWI             USE RTS IF RUN FROM BASIC
3Ø7A CC   Ø4BØ      ØØ55Ø CONV1    LDD     #12ØØ
3Ø7D BD   B4F4      ØØ56Ø          JSR     $B4F4   REGISTER D TO FP1
3Ø8Ø 8E   3ØF7      ØØ57Ø          LDX     #RATE
3Ø83 BD   BB8F      ØØ58Ø          JSR     $BB8F   RATE*FP1
```

```
3Ø86 8E   3ØF7    ØØ59Ø          LDX    #RATE
3Ø89 BD   BC35    ØØ6ØØ          JSR    $BC35   FP1 TO RATE
3Ø8C 39           ØØ61Ø          RTS
3Ø8D 8E   3ØF7    ØØ62Ø CONV2    LDX    #RATE
3Ø9Ø BD   BC14    ØØ63Ø          JSR    $BC14   RATE TO FP1
3Ø93 C6   Ø1      ØØ64Ø          LDB    #1
3Ø95 BD   BD99    ØØ65Ø          JSR    $BD99   REGISTER B+FP1
3Ø98 BD   8446    ØØ66Ø LOG      JSR    $8446   COMPUTE THE LOG
3Ø9B 8E   3ØFC    ØØ67Ø          LDX    #TERM
3Ø9E BD   BACA    ØØ68Ø          JSR    $BACA   TERM*FP1
3ØA1 BD   84F2    ØØ69Ø EXP      JSR    $84F2   COMPUTE THE EXPONENT
3ØA4 8E   31Ø1    ØØ7ØØ          LDX    #VARA
3ØA7 BD   BC35    ØØ71Ø          JSR    $BC35   FP1 TO VARA
3ØAA C6   FF      ØØ72Ø          LDB    #-1
3ØAC BD   BD99    ØØ73Ø          JSR    $BD99   FP1-1
3ØAF BD   BC5F    ØØ74Ø          JSR    $BC5F   FP1 TO FP2
3ØB2 8E   3ØF7    ØØ75Ø          LDX    #RATE
3ØB5 BD   BB88    ØØ76Ø          JSR    $BB88   FP2/RATE
3ØB8 BD   BC5F    ØØ77Ø          JSR    $BC5F   FP1 TO FP2
3ØBB 8E   31Ø1    ØØ78Ø          LDX    #VARA
3ØBE BD   BB88    ØØ79Ø          JSR    $BB88   FP2/VARA
3ØC1 8E   31Ø6    ØØ8ØØ          LDX    #VARPV
3ØC4 BD   BC35    ØØ81Ø          JSR    $BC35   FP1 TO VARPV
3ØC7 39           ØØ82Ø          RTS
3ØC8 BD   BDD9    ØØ83Ø PRINT    JSR    $BDD9   CHR$ TO BUFFER
3ØCB 3Ø   1F      ØØ84Ø          LEAX   -1,X   BUFFER LOCATION -1
3ØCD BD   B99C    ØØ85Ø          JSR    $B99C   PRINT BUFFER
3ØDØ BD   B958    ØØ86Ø          JSR    $B958   PRINT A CARRIAGE RETURN
3ØD3 39           ØØ87Ø          RTS
3ØD4 CC   Ø3Ø6    ØØ88Ø PUSING   LDD    #$Ø3Ø6  PRINT USING $#,###.##
3ØD7 DD   D8      ØØ89Ø          STD    $D8
3ØD9 86   5Ø      ØØ9ØØ          LDA    #$5Ø
3ØDB 97   DA      ØØ91Ø          STA    $DA
3ØDD BD   8FA1    ØØ92Ø          JSR    $8FA1   PRINT THE NUMBER
3ØEØ BD   B958    ØØ93Ø          JSR    $B958   PRINT A CARRIAGE RETURN
3ØE3 39           ØØ94Ø          RTS
3ØE4 9E   A6      ØØ95Ø SAVE     LDX    $A6     GET CURRENT POINTER
3ØE6 34   1Ø      ØØ96Ø          PSHS   X       SAVE IT
3ØE8 BD   A39Ø    ØØ97Ø          JSR    $A39Ø   GET INPUT (NO "," OR "$")
3ØEB 9F   A6      ØØ98Ø          STX    $A6     OUR NEW POINTER
3ØED 9D   9F      ØØ99Ø          JSR    $9F     GET NEXT CHR$
3ØEF BD   BD12    Ø1ØØØ          JSR    $BD12   CONVERT TO FP1
3ØF2 35   1Ø      Ø1Ø1Ø          PULS   X       GET OLD POINTER
3ØF4 9F   A6      Ø1Ø2Ø          STX    $A6     BACK IN LOCATION
3ØF6 39           Ø1Ø3Ø          RTS
3ØF7              Ø1Ø4Ø RATE     RMB    5
3ØFC              Ø1Ø5Ø TERM     RMB    5
31Ø1              Ø1Ø6Ø VARA     RMB    5
31Ø6              Ø1Ø7Ø VARPV    RMB    5
31ØB              Ø1Ø8Ø AMNT     RMB    5
311Ø              Ø1Ø9Ø MSG1     FCC    * ANNUAL RATE - *
311F      ØØ      Ø11ØØ          FCB    Ø
312Ø      2Ø      Ø111Ø MSG2     FCC    * MONTHLY TERM - *
313Ø      ØØ      Ø112Ø          FCB    Ø
3131      2Ø      Ø113Ø MSG3     FCC    * AMOUNT FINANCED - *
3144      ØØ      Ø114Ø          FCB    Ø
3145      2Ø      Ø115Ø MSG4     FCC    * MONTHLY PAYMENT IS - *
315B      ØØ      Ø116Ø          FCB    Ø
315C      2Ø      Ø117Ø MSG5     FCC    * ANY NEW AMOUNT (Y/N)*
3171      ØDØØ    Ø118Ø          FDB    $ØDØØ
3173      2Ø      Ø119Ø MSG6     FCC    * ANY NEW TERM (Y/N)*
3186      ØDØØ    Ø12ØØ          FDB    $ØDØØ
3188      2Ø      Ø121Ø MSG7     FCC    * ANY NEW RATE (Y/N)*
319B      ØDØØ    Ø122Ø          FDB    $ØDØØ
          3ØØØ    Ø123Ø          END    START
```