SLAC-TN-81-3
June 1981

A Floating Point Software Package for Use on LSI-11 Computers at SLAC*

Raymond G. Hendra
Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

Chapter 1: Introduction

A floating point software package has been devised to allow full use of the floating point hardware of the LSI-11 and MODEL40 computers. The procedures are written for the most part in the PL-11 language. The package may be run under the RT-11 operating system, or in RAM or EPROM as part of the KERNEL package (note 1). The current set of procedures has been run successfully in all three modes.

LSI-11 computers to be used with this package must be equiped with the EIS/FIS Integrated Circuit (ROM) supplied by Digital Equipment Corporation. The hardware floating point instructions FADD, FSUB, FMUL, and FDIV are available for use with this IC. This package of procedures allows the user to do the following:

1) Convert binary integers to DEC-compatible floating point format.

2) Read ASCII character strings from the console device, and convert the strings into floating point numbers.

3) Convert floating point numbers into ASCII character strings.

---

*Work supported by the Department of Energy, contract DE-AC03-76SF00515.

4) Convert floating point numbers into ASCII character strings, and print the strings at the console device.

5) Move a floating point number from one location to another.

6) Compare two floating point numbers and determine if the second number is greater than, less than, or equal to the first number.

7) Perform floating point arithmetic: add, subtract, multiply, and divide.

8) Convert floating point numbers into binary integers.

Chapter 2: Description of the Floating Point Package

The procedures contained in this package are:

```
FLOAT -- INTEGER TO FLOATING POINT CONVERTER
FPREAD -- READ FLOATING POINT NUMBER FROM CONSOLE
FPASC -- FLOATING POINT NUMBER TO ASCII STRING CONVERTER
FPPRINT -- FLOATING POINT NUMBER PRINTER
FPMOVE -- FLOATING POINT MOVE PROCEDURE
FPCOMPARE -- FLOATING POINT COMPARE
FPADD -- FLOATING POINT ADD PROCEDURE
FPSUB -- FLOATING POINT SUBTRACT PROCEDURE
FPMUL -- FLOATING POINT MULTIPLY PROCEDURE
FPDIV -- FLOATING POINT DIVIDE PROCEDURE
IFIX -- FLOATING POINT TO INTEGER CONVERTER
```

These procedures are described below, with information on how to call each procedure.

The procedure FLOAT converts an integer word into its floating point equivalent; the result occupies two words. The procedure is passed the reference of the integer in R0, and the reference of the resulting floating point number in R1. The registers are preserved.

Sample call:

```
EXTERNAL PROCEDURE FLOAT;
INTEGER I;
REAL X;
  .
  .
  .
REF(I) => R1;
REF(X) => R0;
FLOAT;
```

The procedure FPREAD reads a character string from the console device using the system stack for buffer space. The character string is converted to floating point by a call to the RCI routine. The reference of the 4-byte answer must be passed to FPREAD in R0.

The routine RCI is an entry point to KERMAC. It converts a string of characters to PDP-11 floating point format. The calling arguments and parameters are on the stack in the following order:

Top -- RETURN PC
2nd -- P FACTOR
3rd -- D FACTOR
4th -- FIELD WIDTH
5th -- OUTPUT BUFFER START ADDRESS

On return from RCI, R0, R1, and R2 are destroyed, R5 and R6 are preserved. R3 will be equal to zero if the conversion was successfully performed, and will not equal zero otherwise. The resultant double-precision floating point binary number is on the stack, with the most significant word at the top.

Since we are only interested in single-precision floating point representation at this time, only the top two words are of interest. Upon entry into FPREAD, these words are moved to the location pointed-at by the value of R0. On return from FPREAD, R0, R1, R2, and R3 are preserved.

If the conversion was not performed correctly, ERRFLAG is set upon return. CRFLAG is set if a carriage return was the only thing in the string; NFLAG is set if an 'N' is received; and XFLAG is set if an 'X' is received. If the conversion was successful, the result is in the two words pointed-at by R0.

The procedure FPASC converts a floating point number to an ASCII character string. The procedure is passed the reference of the floating point number in R0, and the reference of the holding buffer in R1. The registers are preserved.

The procedure performs the conversion by calling the routine GCO, which is an entry point of KERMAC. The routine produces the character string in the FORTRAN G12.6 format.

The procedure FPPRINT prints a floating point number in FORTRAN G12.6 format at the console device. The procedure is passed the reference of the floating point number to be printed in R0. The procedure prints a variable-length string whose length is determined by whether or not the exponent ('E-12' for example) is present. The string is printed with leading and trailing blanks.

The conversion from DEC floating point representation to an ASCII character string is performed by a call to the procedure FPASC.

Sample call:

```
EXTERNAL PROCEDURE FPPRINT;
REAL X;
  .
  .
  .
REF(X)=>R0; FPPRINT;
```

The procedure FPMOVE is passed references for floating point numbers in R0 and R1. It then moves the REAL number refered-to by R0 into the REAL number refered-to in R1. The registers are preserved.

Sample call:

```
EXTERNAL PROCEDURE FPMOVE;
REAL X,Y;
.
.
.
REF(X) => R0; REF(Y) => R1; FPMOVE;
```

The procedure FPCOMPARE compares two floating point numbers, determining whether one is greater than the other or whether they are equal. The method is accurate within the accuracy of the DEC floating point format, which is one part in (2**-22).

The procedure is passed the address of the first floating point number in R0, and of the second in R1. The first floating point number is compared to the second: if the first number is greater than the second, the GREATFLAG bit flag is set; if the first number is less than the second, the LESSFLAG bit flag is set; neither flag is set if the two numbers are equal. The flags are cleared upon entry; the register contents are preserved.

Sample call:

```
EXTERNAL PROCEDURE FPCOMPARE;
.
.
.
REF(<fl. pt. number 1>) => R0;
REF(<fl. pt. number 2>) => R1;
FPCOMPARE;
(results in bit flags GREATFLAG and LESSFLAG)
```

The procedures FPADD, FPSUB, FPMUL, and FPDIV call the hardware floating point operators located in the EIS/FIS ROM on the LSI-11 processor board. The REF (address) of the first operand is passed in R0, the REF of the second operand is passed in R1, and the REF of the result is passed in R2.

Sample calls:

```
EXTERNAL PROCEDURE FPADD, FPSUB, FPMUL, FPDIV;
REAL X,Y,ANS;
  .
  .
  .
REF(X)   =>  R0;
REF(Y)   =>  R1;
REF(ANS)   =>  R2;
'ANS = X+Y'
FPADD;
'ANS = X-Y'
FPSUB;
'ANS = X*Y'
FPMUL;
'ANS = X/Y'
FPDIV;
```

The procedure FPOPERATORS is called by each floating point operator procedure. R0 refers to the stack setup statements and to a case statement, selecting the hardware floating point operator.

The procedure FPADD performs a floating point addition of the contents of the addresses R0 and R1. The result is returned in the address pointed-at by R2.

The procedure FPSUB performs a floating point subtraction of the contents of the address in R1 from the contents of the address in R0. The result is returned in the address pointed-at by R2.

The procedure FPMUL performs a floating point multiplication of the contents of the addresses in R0 and R1. The result is returned in the address pointed-at by R2.

The procedure FPDIV performs a floating point division of the contents of the address in R0 by the contents of the address in R1. The result is returned in the address pointed-at by R2.

The procedure IFIX converts a floating point number into a binary number, truncating toward zero.

In case of numeric overflow, 32,767 will be returned if the overflow is positive, -32768 if negative. The OVERFLOW bit will be set in either case. R1 is used to store the exponent, R0 and R2 are used to rotate the fraction into the right position, and R3 is used to store the high word to determine if the number is negative. For more information, see the floating point representation used by DEC in the LSI-11.

Sample Call:

```
EXTERNAL PROCEDURE IFIX;
.
.
.
```

```
REF(<floating point number>) => R1;
REF(<integer answer>) => R0;
IFIX;
```

Chapter 3: Implementing the Floating Point Package

The floating point package is not a stand-alone module. The external procedures SAVREG, RSTREG, GCO, and RCI must be avaible at linking time. The procedures SAVREG and RSTREG are located in the PL-11 module WYL.PX.PLE.LSILIB#KERPL11. The procedures GCO and RCI are located in the MACRO-11 module WYL.PX.PLE.LSILIB#FPCONV.

Additionally, the external procedures READ and PRINT must be available at linking time. If the floating point package is linked under RT-11, use the module WYL.PX.PLE.LSILIB#FPPACKR. This module expects to find the procedures READ and PRINT as external calls and not as EMT calls. The EMT procedures READ and PRINT will be available if the module is linked for use in RAM or EPROM. Module WYL.PX.PLE.LSILIB#FPPACKR should be used in this case.

The global segment WYL.PX.PLE.LSILIB#FPSEGMNT must be used in every module that is linked with FPPACK and uses its procedures. The segment is shown below:

```
GLOBAL SEGMENT FPSEGMENT;
   BYTE BMEM SYN MEMORY;
   LOGICAL FLAGWORD;
   BIT KEYCOND SYN 0 OF FLAGWORD, OSACTIVE SYN 1 OF FLAGWORD,
       GREATFLAG SYN 2 OF FLAGWORD,
       LESSFLAG SYN 3 OF FLAGWORD, MILTFLAG SYN 5 OF FLAGWORD,
       OUTFLAG SYN 7 OF FLAGWORD,
       CRFLAG SYN 9 OF FLAGWORD, XFLAG SYN 10 OF FLAGWORD,
       NFLAG SYN 11 OF FLAGWORD, YFLAG SYN 12 OF FLAGWORD,
       RFLAG SYN 13 OF FLAGWORD, ERRFLAG SYN 14 OF FLAGWORD;
```

The global segment is readily used in the modules of an applications or systems program that is written in the PL-11 language. For FORTRAN or MACRO-11 modules, implementation as .GLOBL or COMMON elements is necessary.

The procedures of the floating point module are compiled on the SLAC TRIPLEX facility and downloaded to the LSI-11 computers as object modules. The exec file WYL.PX.PLE.LSILIB#X11 may be used to create a job to compile the PL-11 modules. The MACRO-11 module, FPCONV, must be assembled within the RT-11 environment.

The following is an example of the X11 EXEC file used to compile the FPPACKR module described, and of how to download the resulting load module to the computer. In the example, the user's replies to the prompts are shown in lower case:

```
exec from wyl.px.ple.lsilib#x11
```

```
X11 - Version 1.8 (17-Jul-79) - (<CR>="NO" ; "?" for more info)

COMPILE/ASM ACTIVE FILE ? Y/N/C c
MULTIPLE/SINGLE MODE ? (<CR>=SINGLE) s
ENTER THE NAME OF EACH MEMBER TO BE USED
COPY FROM wyl.px.ple.lsilib#fppackr
COPY FROM <carriage return>
ARE CM'S TO GO TO RT-11 (Y/N) y
    ***** new version for MILTEN V6.0+
DATA SET NAME FOR OBJ MODS (<CR>=NO SAVE) wyl.px.ple.fpobj
VERIFY OBJECT DSN : WYL.PX.PLE.FPOBJ Y/N y
ENTER VOLUME FOR NEW DSN (<CR> => Abort and reprompt for OM
LIB)
VOLUME? work01
-->READY TO RUN
-> EXEC PAUSE
> run h
.
.
.
>use wyl.px.ple.fpobj
><escape>tra fpobj.obj fro wyl
446 LINES WILL BE TRANSFERRED TO DK:FPOBJ.OBJ
>
```

At this point, the object module DK:FPOBJ.OBJ exists on the scratch disc. The MACRO-11 module, FPCCNV, must be downloaded and assembled. The following example shows how to download and assemble all three modules. Again, the user's responses are shown in lower case:

```
>use wyl.px.ple.lsilib#fpconv
><escape>tra fpconv.mac fro wyl
123 LINES WILL BE TRANSFERRED TO DK:CCNVF.MAC
.
.
.
><escape C: to get into RT-11>

.mac fpconv
```

All of the needed object modules now exist on the scratch disc. The easiest way to link these modules to the user's programs is to convert the modules into an object library, as in the following example:

```
.lib/cre fplib.obj fpobj,fpconv
```

The library is then linked to any other object module in the usual manner. The following example shows how the library FPLIB just created, may be linked to a dummy program called DUMMY, and to another library called OBJLIB:

```
LINK/T:1000/LIB:FPLIB.OBJ/LIB:OBJLIE.OEJ/MAP:DUMMY.MAP DUMMY
```

The command string also creates a .MAP file, which is always a good idea while developing programs.

Link, run, and library creation commands may always be combined into .CCM command files for quick application during program developement.

Chapter 4: Acknowledgments

Many thanks to Connie Logg for her work on the LSI-11 KERNEL, which is the background work of this package. My thanks also to Les Cottrell for his work on the CONVF (part of KERMAC) group of MACRO-11 routines.