PDP-7 PROGRAM LIBRARY

NUMBER: Digital 7-30-A

NAME: Floating Point Package

AUTHOR: D. Fellows - DEC

DATE: Revised December 22, 1965

ABSTRACT: A self-contained Scientific Programming System for the PDP-7 with Data and Results to 6 decimal digit accuracy, optionally 9 decimal digit accuracy. Instruction execution is interpretive. Arithmetic is double precision normalized floating point.

CONTENTS:

PDP 7 LIBRARY

1.0 GENERAL DESCRIPTION OF THE SYSTEM

1.1 The Interpreter

The heart of disc, the interpretive instruction processor uses double precision floating point binary arithmetic; data may be carried in one of two ways:

a) Sign plus packed 8 bit exponent plus 27-bit magnitude, requiring two storage words.

b) Signed exponent and signed 35-bit magnitude, requiring three words of storage.

The decimal precision possible with (a) is approximately 6 digits, with (b) 9 digits.

The interpreter is entered by the pseudo-instruction EIM (enter interpretive mode) which initializes the interpretive program counter. Instructions are executed sequentially until a transfer of program control or the pseudo-instruction LIM (leave interpretive mode) is encountered. (LIM initiates a return to the machine language program.) The floating accumulator is never implicitly saved and in particular is not preserved when I/O instructions are executed. All interpretive instructions except those which select data mode and the three indexing instructions, may be executed with one level of indirect addressing.

1.2 Input/Output

All entry of data is through a one-character-per-word buffer. Either teletype or paper tape may be used. Allowable input is limited to numeric characters but alphanumeric output is permitted. (see the HDG instruction.)

1.3 Iterative Operation

A primitive one level form of iterative operation is possible in the interpretive mode (see SIX, EXI). Any sequence of commands may be executed, iteratively, up to N times ($N \leq 8191$). An address modification instruction (ADM) is also provided which, when executed, increments the effective address of the instruction which precedes it in sequence by 2 or 3, depending on the current data mode (see DMD). The program must be re-initialized for a second run, since the instructions themselves are changed (ADM).

2.0 INTERPRETIVE COMMAND LIST

2.1 Mnemonics

DAC:	Deposit Floating Accumulator p. 6
JMS:	Subroutine JMP, Floating p. 5
INP:	Initiate Input p. 9
LAC:	Load Floating Accumulator p. 6
FCS:	Load Floating Accumulator with the Complement p. 6
ADD:	Add to the Floating Accumulator
FSB:	Subtract from the Floating Accumulator
FMP:	Multiply by the Floating Accumulator
FDV:	Divide into the Floating Accumulator
HDG:	Text Output p. 9
CAS:	Compare with Floating Accumulator p. 5
JMP:	Change Interpretive Program Counter p. 5
OUT:	Initiate Output p. 9
NUM:	Indicates a Numeric Argument p. 8
IOD:	I/O Device Indicator p. 7
DMD2:	Select two Word Data Mode p. 5
DMD3:	Select three Word Data Mode p. 5
SIX:	Start Iteration p. 8
EXI:	End Iteration p. 8
ADM:	Modify Preceding Address p. 8

2.2 Control Transfer Instructions

Floating JMS,JMP

These instructions are logically the same as the machine language JMS, JMP.

Compare Accumulator to Storage: CAS

The CAS instruction requires an operand address, say A. The logical transfer of control is to PC+1, PC+2, PC+3: where PC is the location of the CAS instruction and the condition of transfer is that the signum of the quantity (C(FLOACC) - C(A)) be -1,0, +1 respectively. One level of indirect addressing is allowable. For example, if the contents of the effective operand address were zero, then the following program:

```
CAS A
JMP B  (A > AC)
JMP C  (A= AC)
JMP D  (A < AC)
```

would execute "JMP B" when the floating accumulator is less than zero: "JMP C" when C(FLOACC) equals zero: "JMP D" when C(FLOACC) is greater than zero.

2.3 Internal Data Mode Instructions

DMD2, DMD3 set the mode for the interpreter with respect to data handling. The appropriate one should be used before any data is referenced. No operand is used.

Internal Data Mode Instructions cont'd

DMD2 designates two word operands. Considering the extended word as bits 0-35, the operand is packed:

bit 0 = sign
bits 1-8 = two's exponent + 128
bits9-35 = binary fraction (magnitude)

Approximate range of data, $10^{\pm 38}$. Precision 6 decimal digits.

DMD3 designates three word operands. Considering the extended word as bits 0-53:

bits 0-17 = signed two's exponent
bit 18 = sign
bits 19-53 = magnitude of binary fraction.

Usable range of data, $10^{\pm 99}$. Precision 9 decimal digits.

2.4 Data Move Instructions

LAC, DAC will load or store respectively in accordance with the current value set by the data mode instruction. FCS loads the floating accumulator with the complement of the effective operand.

2.5 Floating Point Arithmetic

Add, Subtract, Multiply, Divide are provided. Exponentiation is provided as a library subroutine.

Internal Arithmetic

The floating accumulator is a four register accumulator, one register for the sign of the magnitude, one register for the signed exponent in two's complement form and two registers for the positive 35 bit magnitude held as a binary normalized fraction. The floating point accumulator is the same for both two and three word data. The result of an arithmetic operation is normalized and unrounded. The uninitiated should consult the literature for an indication of the inherent pitfalls of this approach to floating point arithmetic. One excellent reference is "Numerical Methods for Scientists and Engineers" by R. W. Hamming; published by McGraw-Hill, 1962.

2.6 Red Tape Instructions

IOD indicates the I/O device referred to by an accompanying input or output control instruction. It is not interpreted by the interpreter but by the logical section of the I/O subprogram. Presently assigned device numbers are:

1 teletype (keyboard or teleprinter)

2 paper tape (reader or punch)

Numbers 3 thru 7 have not been assigned. Reference to an unassigned device will cause 765050 to appear in the AC lights and an irrecoverable halt to occur in the program. IOD+device number+100_8) may be used to control format. The occurence of the added (octal) hundred implies that input (or output) consists of (modulo) four-word strings.

Note that indirect addressing is not allowed with the IOD instruction.

Floating Point Arithmetic cont'd

NUM (numeric value)

This pseudo-instruction is used with the I/O commands and the iterative execution command SIX. When used with INP/OUT it indicates the number of values expected; with SIX, it indicates the number of times the following sequence of instructions is to be executed. Indirect addressing is allowable in the first use but not in the second.

ADM (address modification)

This instruction is used to modify, using a positive increment, the instruction which precedes it in sequence. In the two-word, (three word) data mode the increment is 2, (3). The effective operand address (of the modified instruction) is the address modified.

2.7 Iterative Execution Instructions

These instructions take no operand.

SIX initiates an iterative operation. It is always followed by a NUM command in the immediate address mode whose argument is the number of passes desired. The next sequential address (i.e., the contents of the floating mode program counter) is saved for use by the EXI instruction. Only one level of iteration is allowed.

EXI reduces by one the counter (number of passes) set by SIX and when it reaches zero falls through to the next instruction in sequence. If the counter is not zero, the program counter is reset and control is transferred to that location (note that all interpretive instructions, including this one, operate in floating mode only). Multiple exits are allowable, that is an iteration initiated by a UC command may have several exit paths.

2.8 I/O Instructions:

HDG (heading) is meaningful on output only and is used to initiate output of prestored text. (see for reference the assembler document with particular reference to the pseudo-instruction text.) Formats are predetermined by the particular I/O device selected. The effective address is the location of the first word of the desired output. The program sequence is:

```
HDG (I) A
IOD     N
```

INP/OUT are used to initiate data transfers only. The legal characters are (for teletype and paper tape)

SPACE	/plus, the +sign is not legal
DASH	/or minus sign
PERIOD	/decimal point
DECIMAL DIGIT	/0-9
NULL/IDLE	/ignored
LINE FEED	/ignored
CR	/item delimiters
TAB	/item delimiters

If an illegal character is used "X" is typed and the buffer is cleared. The entire input item must be retyped to enter the number correctly.

Input format is flexible, output format rigid. The normal form for both input and output is:

$$\pm .DD...D \pm EE$$

where the D's are the decimal digits of a (decimally normalized) fraction and the E's represent a two digit decimal exponent. Any meaningful variation of the normal form is valid on input. If the decimal point is omitted, the input value is assumed to be integral; if the sign is omitted, it is assumed to be positive; if the (decimal) exponent is omitted, it is assumed to be zero. (e.g. 767.12-1 → 76.712; 76712 → 76712 etc). If more than ten digits are input, the

resulting value will not be true. In the two-word mode, a converted 35 bit value is truncated to 27 bits on input, rounded to six digits on output. The effective address is the first pick-up or store address and indexing is in 2 or 3 (storage) word increments in accordance with the current data mode.

3.0 SUBROUTINE LIBRARY

Subroutines are entered in floating mode with the floating JMS and the argument in the accumulator. They exit in floating mode with the result left in the accumulator. The following functions are an integral part of the disc package. The non-appearance of a version accurate to only 6 decimal digits indicates that there was no essential temporal or spatial advantage in providing a routine other than that accurate to 9 decimal digits. All of the functions except GXP (q.v.) require the following program sequence:

```
LAC (I) A       /argument to floating
                /accumulator
JMS     BBB     /enter with floating jms
XX              /return to program se-
                /quence
```

3.1 Trigonometric Functions.

9 decimal digit accuracy with 6 decimal digits optionally available.

Sine/Cosine (routine has optional entry, common exit).

Expected argument is in radians. The quoted accuracy falls off markedly when the argument is (in absolute value) greater than 2^{π}. Call SIN_{UC}(COS).

Arctangent 9 decimal digits with 6 digits optionally available. Result is ± radians. Call ATN.

3.2 Natural Logarithm

The routine finds the logarithm to the base e of the absolute value of the argument. Accuracy is 9 decimal digits. To find the logarithm to a different base, multiply the natural log by the log of e to the new base. Call LOG

3.3 Exponential Functions

Normal Exponential Function.

Result is e raised to the argument as a power. 9 decimal digit accuracy with 6 decimal digits optionally available. Call EXP.

General Exponentiation

A compound function of EXP and LOG and hence relatively slow. Accuracy depends on the version of EXP in use. To calculate A↑B call:

```
LAC (I) A
JMS     GXP
LAC     B     /direct addressing expected
```

3.4 Square Root

9 decimal digit accuracy if the value for which the root was taken was exact. (e.g. 2). In general, the precision of the result will be (necessarily) only one-half the precision of the argument. Call SQR.

4.0 OPERATIONAL DESCRIPTION OF THE SYSTEM

4.1 Input/Output calling sequence

```
INP (I) A          /(out(i)a)
IOD     X
NUM     Y          /(num i b, c(b)=y)
```

'A' (or its contents) is the initial data address. 'X' is the device number (1 or 2). 'Y' is the number of items to be processed. If 100 (octal) has been used (101 or 102) with IOD, the data will occur four words per line. (the last line need not.) Each data item must be terminated by a tab or (line feed) carriage return.

4.2 Symbolic Tapes

1) A definitions tape, defining for the assembler the special symbols of the system.

2) The disc system itself in four parts. There are two versions of part IV, one for normal arithmetic, the other for use with machines having an extended arithmetic element.

3) Two library subroutine tapes with 6 and 9 digit accuracies.

4) A 'PUNDEF' request tape.

4.3 Binary Tapes

No address assignment exists on the symbolic tapes. The library version was assembled at 12000 (octal) for use in 8K machines. The system itself (with library) occupies approximately 3000 (decimal) core locations. A PUNDEF tape is supplied. Before assembling a program using disc, load this tape through address 4 immediately after the assembler is loaded.

4.4 Assembling from the symbolic tapes

1) Supply a title tape with the desired address assignment. In lieu of this, the system will be assembled at core locations 22 and ff.

2) Assemble together (and in order)

   a) the title tape
   b) the definitions tape
   c) disc parts I, II, III
   d) the desired part IV
   e) the desired library

3) After punching the binary tape, get a symbol print if you wish.

4) without restoring the assembler (start at 20) assemble the PUNDEF request tape.

4.5 Assembling a Program to be used with Disc

Load the PUNDEF tape binary before assembling, i.e. START 17770 to load assembler, START 4 to load PUNDEF tape then assemble in the normal way. To use, load the using program and the binary Disc and start at the using programs beginning location. If you are making use of the automatic start through the loader, load the binary Disc first.

5.0 ILLUSTRATIVE PROGRAM

```
DISC TEST
/PROBLEM TO CALCULATE THETA=EXP(-X) AND SIN (THETA)
/FOR X=A(B)C WHERE A IS STARTING VALUE, B IS INCREMENT
/AND C IS FINAL VALUE
BAR 2
BEGIN,    EIM                    /ENTER INTERPRETIVE MODE
          DMD2                   /DATA MODE TO 2 WORD
          INP ALPHA#X            /INITIAL STORE ADDRESS
          IOD 1                  /FROM TELETYPE
          NUM 3                  /3 ITEMS
/PRECEDING INSTRUCTIONS READ IN A, B, C, TO ALPHAX, +1, +2
          JMS INITAL
GAMMA,    LAC ALPHAX
          DAC START#X            /SET STARTING VALUE
          FCS STARTX             /-X
          DAC MINUS#X
          JMS EXP
          DAC EXP#ONX
          JMS SIN
          DAC SIN#EX
          JMS STOREX
          LIM
          ISZ COUNT
          EIM
          LAC STARTX
          CAS GAMMAX
          JMP ALPHA
          JMP BETA
          JMP BETA
ALPHA,    LAC ALPHAX
          ADD BETAX
          DAC ALPHAX
          JMP GAMMA
BETA,     HDG MESS
          IOD 1
          LIM
          LAC COUNT
          RTL
          ADD (NUM
          DAC DELTA
          EIM
          OUT DATA
          IOD 101
DELTA,    XX
          JMP BEGIN
```

```
INITAL,   Ø
          LIM
          LAC (DAC DATA
          DAC MODEX
          DZM COUNT
          EIM
          JMP I INITAL
STOREX,   Ø
          LIM
          LAC (LAC STARTX
          DAC INITEX
          EIM
          SIX
          NUM 4
INITEX,   XX
          ADM
MODEX,    XX
          ADM
          EXI
          JMP I STOREX
ALPHAX,   Ø
          Ø
BETAX,    Ø
          Ø
GAMMAX,   Ø
          Ø
STARTX,   Ø
          Ø
MINUSX,   Ø
          Ø
EXPONX,   Ø
          Ø
SINEX,    Ø
          Ø
/OUTPUT FORMAT 15 CHARS
MESS,     TEXT/
X         -X        EXP(-X)              SINE
/
DATA,     DATA+1750/
START BEGIN
```

LINE 5: Bar 2 tells the assembler to assign 2 words to multi-word variables (#). Later the program overrides the variable assignment by providing explicit storage, (because the programmer decided to order his variable storage in a definite way).

LINE 11: Comment '+1, +2', refers to variable storage, not core locations.

LINE 12: This program stores four values for each value of 'x' and then transfers the four values to array 'data'. Subroutine 'inital' zeros the item count (for the output routine) and initializes the data storage address. Note the shift from interpretive to machine language and back again.

LINE 13: 'gamma' calculates and/or stores x, -x, exp (-x), sin(exp(-x)).

LINE 21: 'storex' initializes the pick-up address, which has been modified by 'ADM' then transfers the current tabular values to the array 'data'.

LINE 30: 'alpha' increments x and continues.

LINE 34: 'beta' initiates printout on the teletype. The item count is multiplied by four and set with 'NUM' in the I/O sequence (delta). Printout will be four items per line (IOD 101).

The material reproduced below is one run through the program with a=Ø, b=1, c=5. Note that the programmer misjudged slightly when he prepared his text message.

```
0.
1.
5.
X                -X      EXP(-X)              SINE
 .000000 00 -.000000 00          .100000 01          .841471 00
 .100000 01 -.100000 01          .367879-00          .359638-00
 .200000 01 -.200000 01          .135335-00          .134921-00
 .300000 01 -.300000 01          .497871-01          .497667-01
 .400000 01 -.400000 01          .183156-01          .183145-01
 .500000 01 -.500000 01          .673795-02          .673791-02
```