DECUS
PROGRAM LIBRARY

DECUS NO.	8-155
TITLE	HEP
AUTHOR	Dr. A. K. Head
COMPANY	C.S.I.R.O. University of Melbourne Melbourne, Australia
DATE	April 10, 1968
SOURCE LANGUAGE	

DECUS Program Library Write-up

DECUS No. 8-155

HEP is a package for the PDP-8 and PDP-8/S which gives both calculating machine type operation and stored program operation. It is based on, but not identical to, DEC Floating Point Package 8-5-S-D and DECUS Floating Point Controller 8-44. Calculations have an accuracy of just over six decimal digits, and printout is rounded to six decimal digits. Besides the usual arithmetic operations and mathematical functions, there are facilities for format control, program control and tests, subroutines, and an array of variables. Although it was designed mainly for quick results from small calculations, it also has facilities and space for quite large and elaborate programs. HEP occupies 0003 to 7577.

Start

Read through the repertoire of instructions. Load in the HEP tape using the binary loader. While it is going in, read through the instructions again.

Load address 0200 and start. You are now in ASUGO mode. As you type an instruction, HEP does it and replies with the resulting contents of FAC, the central arithmetic register. Try the following:

```
<1.5      +Ø.15ØØØØ:+Ø1
*6       +Ø.9ØØØØØ:+Ø1
>A      +Ø.9ØØØØØ:+Ø1
#2@      +Ø.3ØØØØØ:+Ø1
/A      +Ø.333333:+ØØ
#1ØG      +Ø.333333
>X      +Ø.333333
<A      +9.ØØØØØØ
*X      +3.ØØØØØØ
-A      -6.ØØØØØØ
```

Remember that in instructions like the first two you must signal the end of the number with a space. Go on and try other instructions.

When you feel you understand ASUGO mode, try to store a program. Load address 0201 and start. You are now in STORE mode. As you type instructions, they are stored. Here are three sample programs, printed across the page to save space.

Program 1 - Print running average of numbers typed in

```
<Ø  >T  >N  #13A  <@  +T  >T  #15N  /N  >@  #9@  #14A
```

Program 2 - Calculate (1+1/N) ↑ N by N multiplications

```
#13S  <@  >N  *-1  >C  <1  /N  +1  >F  <1  *F  #15C  &-2  >@  #9@  #14S
```

Program 3 - Prime factors of N (999999 or less)

#1ØF #11@ #13S #9N #12H <@ >N #9@ #13B <N #2@ >R <1 >F

#13A #15F <F -R #12F #14C <N /F >Q #8D -Q #12B #14A <Q

>N <F >@ #9@ #14B #13C <N >@ #9@ #9@ #14S

When you have typed in a program, you then want to EXECUTE it. Load address 0202 and start. The program in store will now be executed. In each of the three samples, it will come to <@and wait for a number to be typed. Remember to terminate this number with a space.

From here on it is up to you. Do not be afraid to experiment; there is nothing you can do from the teleprinter keyboard that will destroy HEP. The worst that can happen is that you will be sent to LOCKUP.

Starting Addresses

0200	ASUGO mode. Instructions are executed as you go, and the resulting contents of FAC printed.
0201	STORE mode. Instructions are stored.
0202	EXECUTE mode. The program in store is executed.
0203	EXTEND mode. Store more instructions at the end of the program already in store.
0204	PUNCH mode. The program in store is punched as a binary tape with leader, checksum, and trailer for subsequent loading with the binary loader.

The only interaction between ASUGO mode and STORED PROGRAM mode is that they share the same named variables.

Instructions

There is a central arithmetic register (called FAC) through which all numbers are transferred and in which arithmetic is done.

There are 31 named locations in which numbers can be stored and brought back later. These are called the letters of the alphabet A through Z and the five characters [\ ] ↑ ← which are typed by shift KLMNO.

The character @ is a special name.

Numbers can be typed in any of the following forms:

17 +17 -17 17.32 173.2:-1

The last number means 173.2 multiplied by 10 to the power -1. The end of a number always must be indicated by typing space (or any character). If you type 17, the computer will not know if that is the end; you must tell it by typing space.

In the following instructions, Z is used as a typical letter and 17.32 as a typical number.

The different uses made of the letters of the alphabet are similar to the different meanings of 26 in the FORTRAN statement:

IF(J(26) - 26) 26, 26, 27

Arithmetic Instructions

Instruction	Result
+Z or +17.32	FAC = FAC + Z or = FAC + 17.32
-Z or -17.32	FAC = FAC - Z or = FAC - 17.32
*Z or *17.32	FAC = FAC * Z or = FAC * 17.32
/Z or /17.32	FAC = FAC / Z or = FAC / 17.32
<Z or <17.32	FAC = Z or = 17.32
>Z	Z = FAC

When typing these instructions, think of FAC as being off to the left of the paper. The last two instructions can be thought of as arrow heads indicating the direction of transfer. All these instructions leave the originating number unchanged; in particular, >Z does not change FAC.

Mathematical Functions

Instruction	Result	
#1@	FAC = $(FAC)^2$	
#2@	FAC = $\sqrt{\lvert FAC \rvert}$	
#3@	FAC = SIN (FAC)	Angles in Radiances
#4@	FAC = COS (FAC)	
#5@	FAC = ARCTAN (FAC)	
#6@	FAC = EXP (FAC)	
#7@	FAC = LOG |FAC|	
#8A	FAC = |FAC|	
#8B	FAC = -FAC (faster than *-1)	
#8C	FAC = 1/FAC	
#8D	FAC = INTEGER PART OF FAC	
#8E to ←	NO OPERATION	
#8@	Read bits 4-11 of SWITCH REGISTER as an integer (0-255) and float into FAC.	

INTEGER PART OF: If already an integer, leave alone; otherwise, take the next more negative integer, e.g., 3→3 3.1 → 3 0.9 → 0 0 → 0 -3 → -3 -3.1 → -4 -0.9 → -1

Input, Output, and Format Control

<@ Read in a number into FAC. Any text preceding a number will be echoed but otherwise ignored.

>@ Print the number in FAC. No CRLF. This does not change FAC.

#9@ Print CRLF.

#9Z Print the letter Z. No CRLF.

#10@ Print future numbers in format

#10Z Print future numbers in fixed point format with a total of 26 digits. A = 1 digit, B = 2 digits, up to ← = 31 digits maximum.

#11@ If fixed point printing, then print as integer.

#11Z If fixed point printing, then print 26 (≡Z) digits after the decimal point.

#12H to ← Print spaces. H = 1 up to ← = 24 spaces.

Numbers will automatically be printed in format unless changed by a #10 instruction. If a number is too big to fit in a fixed point format then ± (((((((will be printed.

Program Control

& Jump and continue program from there.

Eg.
- &3 Jump on 3 instructions.
- &-3 Jump back 3 instructions.
- &Ø Jump to here (forever).
- &Z Take the integer part of the number in Z and jump on that number of instructions (back if negative).

#12@ Pause. Press CONT to continue.

#12A Skip the next instruction (faster than &2).

#12B to G Compare the number in FAC with zero and skip the next instruction if test is true.

Tests:
- B FAC = Ø
- C FAC ≠ Ø
- D FAC < Ø
- E FAC ≥ Ø
- F FAC ≤ Ø
- G FAC > Ø

#13Z Marker Z, no operation

#14Z Go to marker Z and continue program from there.

#15Z Increase the number in the variable Z by one, and if the new value of Z is exactly zero, then skip the next instruction (not just zero integer part).

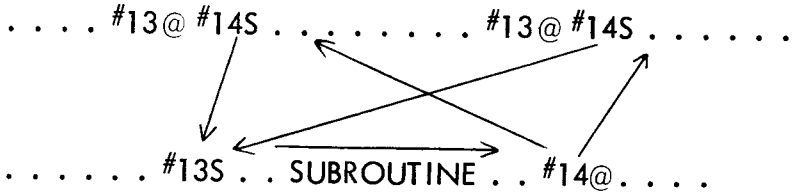
Subroutines

HEP Language Subroutines

#13@ Remember this position in program at the top of the LINK list. The next instruction after #13@ must be a GO TO the start of the subroutine.

#14@ Subroutine exit. Take top LINK from list and continue program two instructions beyond LINK.

Eg.



There are three positions in the pushdown LINK list so subroutines may be three deep. Within this limit, subroutine calls may be self- or mutually recursive.

One number can be transmitted to or from a subroutine in FAC. More numbers must be left in the variables.

Machine Language Subroutine

#15@ Do your machine language subroutine. See Advanced Programming for details.

Besides the eight symbols, + - * / < > & # , HEP will accept two others at the start of a line. These are:

% Delete the last instruction of the program in store. This can be repeated to delete the new last instruction and so on. On EXTEND, you may start by deleting back into the the program already in store.

" Program comment. Anything typed between " and the next CRLF will be echoed but nothing is stored.

Any character other than these ten will be ignored and not even echoed.

Errors

Incomprehensible Instruction

A string of periods is output, and the "instruction" is ignored completely. You do not have to type % to delete the "instruction;" typing % will delete only the last legal instruction.

Lockup

When faced with an impossible situation, HEP will output a string of exclamation points and lockup on a running jump. The only exit from lockup is to load an address and restart from the switches.

Lockup can be caused by the following: too large a program, jumping outside the program with &, go to a nonexisting marker, no link in list at subroutine exit, variable array indexing outside limits, or just running off the end of your program.

(If an error is made when typing a number, then type (and retype the whole of the number. This is mainly of use with keyboard input at <@ but can be used to correct some mistyped instructions, e.g., <7.8 (8.7 is stored as <8.7, #13 (14S is stored as #14S. However, for most mistyped instructions, you will need to use % to delete the whole instruction.

Size of Program

There are two separate limits on the size of your HEP program which can be stored. If N is the number of instructions and C the number of different constants in your program, then it is necessary that $C \leq 127$ and $N + 3C \leq 1365$. If either limit is exceeded, you will end up in lockup. Both limits are in decimal.

In ASUGO mode, neither instructions nor constants are stored, so there is no limit on the number of instructions you can do.

Program on Tape

If the punch is on while you are typing a program into store, it will create a tape for future use. The program printup has a double CRLF after every ten instructions and some blank tape is punched at the same place. This is an aid to editing the program tape.

Switch Options

All switch options can be altered at any time.

BIT Ø On STORE or EXTEND, pause after each 10 instructions?

Ø = NO 1 = YES

Press CONT to continue.

BIT 1	Input from	1 = high speed tape reader Ø = Teletype
BIT 2	Output on Teletype?	1 = NO Ø = YES
BIT 3	Output on high speed punch?	1 = YES Ø = NO

Output can be on either Teletype and/or high speed punch. No output is necessary when reading in a program tape of which you do not want a printup.

Remember to reset switches before loading a starting address.

Advanced Programming

Array of Variables

There is storage set aside for a one dimensional array of 127 numbers. These can be referenced by the symbol ? after the operators + - * / < > & . For example, * ? Z means multiply FAC, not by Z, but by the variable whose serial number is in Z. If the number in Z is not an integer, then the integer part of it is taken. A serial number outside 1 to 127 will cause lockup. The named variables A through ← are the first 31 numbers of this array and can be referenced both directly by name and indirectly by ?. The other 96 can only be individually referenced by *?127, for example.

More Variables

The storage of constants is economized, i.e., if a program contained the instructions +3 -3 *3, then the number 3 is stored only once, and the 3 in these instructions is really just the name of the location where the number 3 was put when the program was stored. The instruction >3 overwrites this location with the number in FAC so that subsequent references to 3 will no longer bring out the number 3 but what was put there by >3. The 3 has become just the name of a variable like Z and can be used as such with all the arithmetic operators and ?; but it cannot extend the array of 127 variables, nor can it be used in the increment and skip instruction #15. Choose only numbers that you do not want to use as genuine constants since they cannot be used both ways.

Machine Language Subroutine #15@

Your machine language subroutine should be in the standard form

```
SUB, Ø
. . . . . .
JMP I SUB
```

and loaded into store with the binary loader. Your binary tape must also load into location Ø177 the number SUB, the starting address of your subroutine.

The instruction #15@ leads to JMS I Ø177 and so to your subroutine. AC will be clear on entry but need not be cleared on exit. A one word integer (±2047) can be floated into FAC by DCA ØØ45, JMS I Ø175. Some of the facilities of the floating point system are also available to you as explained under "Subroutines" on pages 3-6 of the Floating Point Manual.

The subroutine can be located either on the last page or in the middle of HEP, in any spare space between your HEP program and your constants. HEP is stored at 1232. Constants are stored downwards using three words per constant, the first constant going in 3754-3756. Locations 0, 1, and 2 on page zero are also available.

The HEP tape comes with 6171 in location Ø177 which leads to a dummy no-operation subroutine.

The variables are stored in 3762-4556 with A in 4554-4556, B in 4551-4553, etc.

Integer Arithmetic

There is no separate treatment of integers by HEP. They are handled like other numbers as three-word floating point numbers. However, arithmetic is exact for integers in the range ±999999. Specifically, for integers in this range, the following instructions will be exact: input, output, + - * / < > &, #1@, #2@, #8A, #8B, #12 tests for zero and #15 counting. For divide and square root this means that if the result should be exactly an integer, then it will be. The square root routine used is not that of DEC-8-5. The #8D will correctly give the integer part of numbers in this range.