DECUS
PROGRAM LIBRARY

DECUS NO.	8-122A & B
TITLE	SNAP (Simplified Numerical Analysis
AUTHOR	
COMPANY	Harvard Medical School Boston, Massachusetts
DATE	1967
SOURCE LANGUAGE	

ATTENTION

This is a USER program. Other than requiring that it conform to submittal and review standards, no quality control has been imposed upon this program by DECUS.

The DECUS Program Library is a clearing house only; it does not generate or test programs. No warranty, express or implied, is made by the contributor, Digital Equipment Computer Users Society or Digital Equipment Corporation as to the accuracy or functioning of the program or related material, and no responsibility is assumed by these parties in connection therewith.

SNAP

A Real-Time Computer Language

SNAP was developed at Harvard Medical School under grant number 5PD 7 FR 00246-02 from the Special Research Resources Branch of the Division of Research Facilities and Resources of the National Institutes of Health.

PROGRAM INSTRUCTIONS

(five characters, any letter except X, I, or T)

Arithmetic Manipulations:	A=B+C	add
	A=B-C	subtract
	A=B*C	multiply
	A=B/C	divide
	A=LGB	$\log_e B$
	A=EXR	exponentiation (e^R)
	A=SRB	square root of B
	A=SIB	sinB
	A=CSB	CosB
	A=ATB	arctanB
	A=NEB	A=-B
Input-Output:	TYPEB	type value of B, do line feed, and carriage return
	T>PEB	type value of B, no carriage return
	PLOTB	plot B against X
	DISTB	display table values on scope
	L=AN4	L = channel 4 voltage input x 1000, also sets T=real-time.
Conditional:	BP13U	go to 13 if U is positive (incl. zero)
	BM12K	go to 12 if K is negative
	BZ13U	go to 13 if U=0
	BRSUL	branch setup L
	BR12L	go to 12, L-1 number of times
	BS12X	branch to subroutine beginning at line 12
	RETUR	return from subroutine to line following last executed BS instruction
Table lookup:	C=TBA	table A lookup
	M=TBB	table B lookup
	N=TBD	N as a function of D
	RESET	reset table A and B lookups
	GITBD	get D indirectly from the table (I pointer)
	PITBC	put C indirectly into the table (I pointer)
	RUBTB	inserts the value zero into one hundred table locations
Special Instructions:	GOPL2	perform machine language subroutine which is already in memory. Number 1-5 indicates which of 5 possible routines is to be executed.

CONTROL COMMANDS

PROgram	load program
CONstants	enter numerical constants
TBLE	load 100 numbers in A and B tables
AXIS	origin for plotter output
STArt at	change starting point of program
CALculate	run the program
INSert	insert an instruction
DELete	delete an instruction
CHAnge	change an instruction
LIST	type-punch program and constants
TYPE	type a list of constants which have been set by the program
EXTend	extend the number of entries in the table

SNAP Keyboard Controls

To return to SNAP Control mode, depress ALT MODE key on teletype. (Red Key)
To repeat execution of the program, depress the button marked , (comma). (Green key)
SNAP's starting address is 1200.

S N A P

A Real-Time Computer Language

A simple SNAP program. Find the hypotenuse of a right triangle given the two sides.

From plane geometry

$$H = \sqrt{A^2 + B^2}$$

Choose two other letters, say J and K to represent A^2 and B^2. The SNAP program is as follows:

```
J=AxA
K=BxB
L=J+K
H=SRL        (SR means square root)
TYPEH
ENDPT        (All SNAP programs end with ENDPT)
```

Note that there are six steps to the program. These are called "instructions." In SNAP all instructions have five characters

Examples,	H=SRL,	PLOTY,	BP10U
	12345	12345	12345

This is important because it helps to distinguish in the beginner's mind the difference between instructions and "control commands." Control commands are used to load programs and numbers into the computer and to edit these. They operate at a management level above the program. They are not the program. The distinction between program and control commands (usually called metacommands) is probably the most difficult concept a beginner encounters. Therefore, to clarify this point in SNAP all control commands have three or four letters, all instructions have five characters.

Initially you need to know the following control commands.

PRO	ACCEPT PROGRAM
CAL	CALCULATE
LIST	TYPE OUT PROGRAM
INS	INSERT AN INSTRUCTION
CHA	CHANGE AN INSTRUCTION
DEL	DELETE AN INSTRUCTION
CON	ENTER CONSTANTS (or change constants)
TBLE	ENTER A TABLE

The symbol ↵ means a carriage return. The control command is not executed until the carriage return is performed.

Now let us see how we enter and operate a SNAP program by means of the control commands. SNAP starts from the phase called "CONTROL."

Start Computer

```
CONTROL     PRO↵   (Enter Program)
0010        J=A*A
0011        K=B*B
0012        L=J+K
0013        H=SRL
0014        TYPEH
0015        ENDPT
0016        Return to Control (Red Key of teletype)*

CONTROL     CON↵   (Enter Constants)
A=5

B=6
Return to Control (red key)
CONTROL     CAL↵   (Calculate)
H=+0.7810249E+01(which is read as .78 x 10^1 or 7.810..)
```

The ENDPT instruction now stops the computer and allows the operator to change any or all constants. Suppose he does the following

```
A=27.2
B=16.4
```

He then strikes the single key called "repeat calc" (the green key)** and the program is run again. The computer answers .H=+0.3176161E+02 (read as .31 x 10^2 or 31.76161)

Note that after ENDPT it is not necessary to return to CONTROL to change a constant. However if a change in the program is to be made the red key returns to CONTROL.

*This is the key marked "ALT MODE" on ASR 33 teletype
**Comma key

Suppose we want the ratio of A/B. Call this R. We want to insert two instructions:

```
R=A/B
TYPER
```

To do this, return to control.

```
CONTROL        INS↵
LINE NUMBER    15
TYPER
CONTROL        INS↵
LINE NUMBER    15
R=A/B
```

(Note these are inserted in reverse order so that the line numbering comes out right)

```
CONTROL      LIST↵
0010    J=A*A
0011    K=B*B
0012    L=J+K
0013    H=SRL
0014    TYPEH
0015    R=A/B
0016    TYPER
0017    ENDPT
CON

A+0.27200E+02
B+0.163999E+02
      .
      .
      .
      .
CONTROL      CAL↵
```

(This types out the program and the last values of all the constants used by the program)

Partial List of SNAP Instructions

The variables used here are illustrative; any letter except X,I,T may be used for any variable. The use of these will be described later.

```
A=B+C      Sets A=B+C
A=B*C      Sets A=BxC
E=J/K      Sets E=J/K
F=R-S      Sets F=R-S
```

TYPEB	Types out the value of B
R=LGP	Sets R=$\log_e P$
Q=EXR	Sets Q=e^R
M=SRN	Sets M=$\sqrt{N}$
U=SIL	Sets U=SinL
U=CSL	Sets U=CosL
P=ATY	Sets P=ArcTanY

DO PROBLEM ONE NOW.

On to Bigger Things

Let us type out the sum of the series of odd numbers beginning with 5.

i.e.
```
  5  =5          (Type 5)
7+5  =12         (Type 12)
5+7+9  =21       (Type 21)

5+7+9+11+13+15 =60        (Type 60)

        etc.
```

Let C be the current odd number. We will initially set C=5. Let S be the sum of numbers so far. Let T=2. The SNAP Program follows:

```
S=S+C
C=C+T
TYPES
ENDPT
```

After entering the program, set S=0
C=5
T=2

Now calculate. SNAP will print out S=5.
The second instruction will have changed C from 5 to 7 so that when we hit repeat calculation, the next pass through the program will add 7 to 5 and print

```
S=12
  etc.
```

Programs of this type occur frequently. They are characterized by a repetitive calculation in which one or more variables takes on new values at each execution of the program. In the above case we accomplished this by means of SNAP'S rept. calc. key. There is another faster way using Branch Instructions. SNAP has three simple branch instructions. The first is of the form BP12U which

means Branch on Positive value of U to instruction 12. When the program reaches this instruction, SNAP checks the value of the variable U. If U is positive (zero is included as a positive number), instruction number 0012 will be executed next and the program will continue from there. If U is found to be negative, the branch to 0012 will not occur and the program will continue in sequence. If we wish the previous program to continue indefinitely we modify it as follows:

```
0010      S=S+C      (new value of S = C+ old value of S)
0011      C=C+T      (new value of C = T+ old value of C)
0012      TYPES
0013      BP10T
0014      ENDPT
```

Since T is always 2, it is positive and the branch to instruction 10 will occur repeatedly. The program will therefore loop indefinitely. The sequence of operations will be as follows:

```
                    (Initially S=0,C=5,T=2)

S=S+C               (new S= 0+5)
C=C+T               (new C=5 +2)
TYPES               (types S= 5)

T positive, go through loop again

S=S+C               (new S= 5+7)
C=C+T               (new C= 7+2)
TYPES               (types S= 12)

T positive, continue looping

S=S+C               (new S= 12+9)
C=C+T               (new C= 9+ 2)
TYPES               (types S= 21)

T positive, loop
.
.
.
.
etc.
```

DO PROBLEM 2 NOW.

Suppose we do not want a looping program to continue to loop indefinitely. We have two ways of terminating this action. One is to insert a variable which is changed on each pass through the

loop. For example we can let H=7 at the beginning of a program and add -1 to it during each pass through the loop. A branch on positive H instruction (BP10H) will cause a return to instruction 0010 seven times. Then on the eighth pass through the loop, H will have the value -1, a branch will not occur and the program will continue in sequence. This kind of operation is particularly useful when the program must do a branch which depends upon a variable it is calculating. For example:

```
Find the smallest integer greater than 2 for which
Y = 14x(log_eN)-N is a negative number.
0010      N=U+U      Sets N=2.
0011      N=N+U      Add one which makes N=3.
0012      B=LGN      Calculates log_eN
0013      C=A*B      Sets C=14x(log_eN)
0014      D=C-N      Sets D=14x(log_eN)-N
                     When N=3, D will be positive.  (log_e3=1.1).
0015      BP11D
0016      TYPEN
0017      ENDPT
      Set U = 1 and A = 14.
```

At instruction 0015 the program returns to instruction 0011 and N is increased by one again. This loop continues until N=57 at which point D is negative and the value of N is typed. The program then stops at ENDPT.

The other technique for terminating a looping sequence requires two instructions, a BRANCH SET UP instruction and a BRANCH RETURN instruction. The branch setup must precede the branch return and all other instructions in the repeating loop. Any variable may be chosen to specify how many times the loop is to be repeated. As an example let us choose K as the variable and let the loop repeat 4 times. The program looks like this:

```
0010
0011      BRSUK      Branch set up K
0012
0013
0014
0015      BR13K      Branch return to 13, K-1 times
```

The constant K must be set equal to 4 before instruction 0011 is encountered. It may be assigned or computed in some other part of the program. The value of K is not affected by either of these instructions.

Example. Add up the first one hundred odd integers and type out the result.

```
Let M=100, U=1, T=2, Z=0
0010      S=Z+Z
0011      N=Z+U
0012      BRSUM
0013      S=S+N
0014      N=N+T
0015      BR13M
0016      TYPES
0017      ENDPT
```

DO PROBLEM 3 NOW.

PLOT INSTRUCTION

Up to this point all the communication with the computer has been via the teletype. For many problems it is desirable to have a pictorial or graphic output from the computer. This is possible by means of a special cathode ray oscilloscope which is connected to the computer. In effecting a display the horizontal component is always represented by the value of X; the vertical component may be represented by any variable other than X. For example, PLOTK will cause a display of a single point using the value of X as the horizontal coordinate of the point and K as the vertical coordinate. X=0 represents the left side of the oscilloscope; X=1000, the right side. Vertically, -500 is the bottom of the scope and +500 is the top.

Let us plot the curve, $K=AX^2$.
for X=0 to 1000 and A=.0005

We will plot the values of:
X=0, 10, 20, etc. to 990, 1000.

```
0010    X=Z+Z
0011    BRSUM
0012    V=X*X
0013    K=A*V
0014    PLOTK
0015    X=X+P
0016    BR12M
0017    BP10P
0018    ENDPT
          Set Z=0    A=.005    M=101     P=10
```

DO PROBLEM 4 NOW.

ANALOGUE AND TIME INPUTS

The most unique feature of SNAP is its ability to interact on-line with other laboratory instruments. Most computers and computing languages are designed to accept input information only from punched cards or teletypewriters. For computer interaction with a laboratory experiment, a language is needed which can accept electrical inputs directly and which can read inputs from a time measuring device, known in the computer trade as a "real-time clock." Both of these functions are incorporated in a single SNAP instruction.

The computer has seven channels of voltage input, numbered 1 to 7. For example the instruction:

P=AN5

sets the value of P=1000 x (Voltage on channel 5) and simultaneously reads the time (in seconds) which has elapsed since the program was first run. Any letter except T can be used in the analogue input instructions (for example B=AN2, R=AN7) but the time is <u>always</u> inserted into T. T is read to six significant figures so that very small time intervals can be measured.**

The input voltages are limited to the range -1 to +1 volt so that the variable set by this voltage will be between -1000 and +1000. The analogue input voltages can be fed into the computer from the outside or can be generated within the computer by means of knobs on its front panel.

Let us try a problem involving reading an analogue value from the knobs. In one of our earlier problems we generated a parabola $K=AX^2$ using the value A=.0005. Let us instead read the value of A from knob 7 and use it to change the shape of the parabola. Since a variable read from the knobs is in the range -1000 to +1000 we will have to multiply it by a small number to make it suitable for use as A. For Example:

```
0010    B=AN7
0011    A=N*B
0012    X=Z*Z
0013    BRSUM
0014    V=X*X
0015    K=A*V
0016    PLOTK
0017    X=X+P
```

**Program running time is sometimes important. In SNAP, each instruction takes about one and one-half milliseconds.

```
0018    BR14M
0019    BP10P
0020    ENDPT
```

Z=0 N=.000001 M=100 P=10

DO PROBLEM 5 NOW.

ANALOG INPUT FROM CLOCK

A very common problem involves measuring the time between successive electrical pulse inputs. Suppose we have a series of positive pulses occurring at a rate of about one per second. The easy way to measure this interval is to set the knobs on the front of the computer so that a negative value is read in the absence of a pulse and a positive value is read when the pulse occurs. Let V be the variable used to measure the voltage.

The pair of instructions,

```
0012    V=AN4
0013    BM12V
0014
```

will remain in a loop until V goes positive since the branch back to 0012 will occur repeatedly for negative V. When V goes positive, instruction 0013 will not branch and 0014 will be executed next. Each time 0012 is executed the time will be inserted into T. Therefore when 0014 is executed T will have the time of occurrence of the pulse.

Suppose we want the interval between two pulses. This program will measure it and type out the interval.

```
0010    V=AN4
0011    BM10V    Sets T equal to time of first pulse
0012    U=T+Z    Sets U=T first
0013    V=AN4
0014    BM13V    Sets T equal to time of second pulse
0015    C=T-U    Sets C equal to difference
0016    TYPEC    Types interval in seconds
0017    ENDPT
        Z=0
```

Be sure that you understand this before proceeding.

SNAP TABLE INSTRUCTIONS

SNAP has another feature particularly useful for biological problems, Table Instructions. A list of one hundred numbers may be entered from the keyboard or from punched paper tape. To do this, return to "CONTROL" and type TBLE↵ , and then the list of one hundred numbers. SNAP will return to "CONTROL" when the maximum number of entries has been entered. If fewer points are to be entered the red key will return to "CONTROL."

The table entries may be referred to in a variety of ways. Choose any variable, say G, and let

G=TBA (G equal table A)

The first time this instruction is executed following "CAL," G is set equal to the first table entry. The next time this instruction is executed G is set equal to the second table entry. The next time to the third, etc. For example:

Let the table entries be---

```
 1)    1098
 2)    1096
 3)    1094
 4)    1092
         .
         .
         .
99)     900
100)    898
```

Then the program---

```
0010    G=TBA
0011    TYPEG
0012    BP10Q
0013    ENDPT     Q=1
```

Will type out---

```
G=.1098E+4
G=.1096E+4
G=.1094E+4
```

etc., until the end of the table has been reached (100 values).

Note that this sequential step reading of the table works only if the table is referred to by

Something=TBA (Table A)

If instead you refer to the table as TBB (Table B), a similar action occurs except that the first execution of the instruction reads the fifty-first entry, the next the fifty-second, etc.

If at some point in the program you wish to start the table reference from the beginning again (i.e. entry 1 for TBA, entry 51 for TBB), a RESET instruction is used.

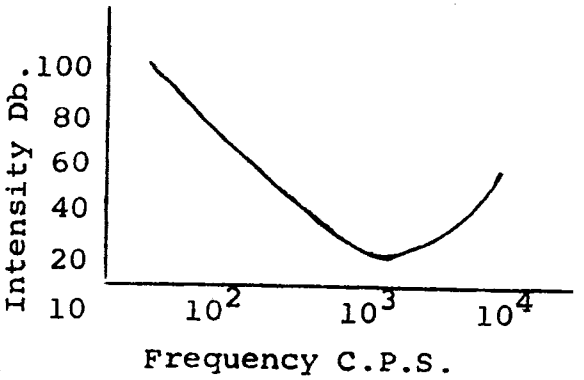
Suppose you wish to display a table of 40 points continuously. Enter the X coordinates in the first 40 locations of the table. Fill the next ten locations with zeros, then enter the Y coordinates as the 51st to 90th entries.

The following program will display the forty points.

```
0010    BRSUD
0011    Q=TBA
0012    X=K*Q
0013    R=TBB
0014    Y=L*R
0015    PLOTY
0016    BR11D
0017    RESET
0018    BP10D
```

D = 40, L and K are scale factors which depend upon the size of the numbers in the tables. The program will loop 40 times, displaying successive points. Then on the reset instruction, the table pointers will be re-initialized and the successive points will be displayed again.

The SNAP tables can be used in another very useful way. Frequently biological problems involve an intermediate step of a graphical lookup. This occurs when one quantity is an empirical function of another. For example a plot of the threshold of hearing as a function of frequency looks like this:



If we wish to incorporate this graph into the solution of a problem the procedure is as follows. Choose any number (up to 50), of arbitrary points along the horizontal axis. Enter these as Table A, i.e. the first fifty points of the SNAP table. Then enter as Table B (i.e. points 50 to 100) the corresponding points from the graphed line.

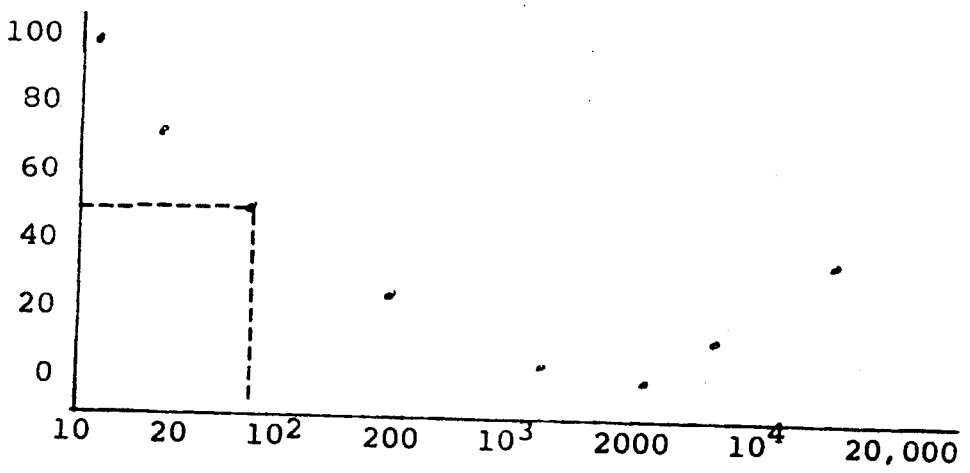


Table A *	Table B
10	100
20	75
100	58
200	36
1000	22
2000	15
10,000	8
20,000	25

The instruction P=TBC will, when given a value for C, look C up in Table A and set P equal to the corresponding value from Table B, interpolating if necessary. Suppose C=80, then P=62.25. Any letter except A or B may be used following TB. Thus the above instruction can be written V=TBW. If W = 3000 then V = 14. If W = 200, V=36, etc.

The density of leaves under my maple tree as a function of distance from the center of the trunk is given by the following table:

*must be in increasing order

r distance ft.	leaves/sq. ft.
1	2
2	2
3	7
5	10
7	12
10	15
15	12
20	10
30	7
40	3
50	1
60	0
70	0

The area of a ring 1/10 ft. wide of radius r ft. is $\frac{1}{10}\ 2\pi r$. By adding up rings find the total number of leaves out to 60 ft. Start from r=1 ft.

```
N=Z+Z
R=U+Z
A=Q*R
D=TBR
M=A*D
N=N+M
R=R+L
B=R-P
BM12B
TYPEN
ENDPT
```

$Q=\frac{2}{10}=$ .62832

P=60 ft.
L=0.1 ft.
Z=0
U=1

Note: The SNAP program which uses the D=TB instructions must have checks so that the entry referenced is not out of range of the table.

Indirect Table Instructions

The instruction PITB- and GITB- allow the SNAP programmer to use the table in another way. By setting I to some number between 1 and 100, the program may Put information into table position "I" or may Get the value out of position "I."

PITBE will put the value of E into table entry I.
GITBE will get table entry I and set E equal to its value.

Example: If E = 2.7145 and I = 5, the instruction PITBE would put the value 2.7145 into table position 5. The instruction GITBK would then set K = 2.7145, if I were 5 and the table is left unchanged.

Use of Indirect Instructions is illustrated in the following program.

Pulse Interval Histogram

```
10    RUBTB      Clear Table
11    U=T+Z      Sets U equal to time of last pulse
12    Y=AN1      Look for positive pulse
13    BM12Y      If no pulse go back to 12
14    I=T-U      Interval equals time of new pulse-time of old
15    I=I*K      Scale I to between 1 and 100
16    GITBC      Get value of C from table place I
17    C=C+D      Increase C
18    PITBC      Put new C into table place I
19    DISTB      Display the table
20    BZ11Z      Branch back to 11
21    ENDPT

Z=0
D=1.0
```

Problem

Change the above to a signal amplitude histogram taking a sample every 0.1 sec.

Response Averaging

```
10    N=Z+Z      Set N=0
11    RUBTB      Clear table
12    Y=AN6
13    DISTB      Check for trigger, If no trig, display and sample
14    BM12Y      again. If trig go to 15
15    L=N+U      Set L=N-1
16    BRSUM      Set up loop counter for 50 loops
17    I=Z+U      Set table pointer to 1
18    V=T+D      Set V equal to time of last sample + delay
19    Y=AN7
20    F=T-V      Sample channel 7 until time is greater than old
21    BM19F      time plus delay
22    GITBP      Get P from table location I
```

```
23      P=P*N
24      P=P+Y       Compute average for Ith entry
25      P=P/L       Average (new)=/Average(old)*N/7+Y//N+17
26      PITBP       Put new average into location I
27      I=I+U
28      BR18M       Increment I until 50th point is done
29      N=N+U
30      BP12M       Increment N and do another fifty points
31      ENDPT
```

```
M=50
D=time between points in sec
U=1
Z=0
```

Subroutine Instructions

The BS__X and RETUR Instructions permit a sequence of SNAP instructions to be executed at several points in a program without the actual instructions being repeated in the program.

For example, the program on page 11 could be rewritten:

```
0010    BS16X
0011    U=T+Z
0012    BS16X
0013    C=T-U
0014    TYPEC
0015    ENDPT
0016    V=AN4
0017    BM16V
0018    RETUR

        Z=0
```

The first time RETUR is executed, it acts like a BZ11Z; the second time, it acts like BZ13Z.

How to Use GOPL

The instruction GOPL# calls a subroutine which is residing in core memory of the computer, performs the subroutine and returns to perform the next SNAP instruction.

There may be up to 5 such subroutines available for use with SNAP. They may be keyed into memory, if short, or written in

symbolic language and assembled by PAL. The resulting binary tape may then be read into memory.

The spaces available for binary subroutines are locations 4145-4571. The starting addresses of such routines should be stored in locations 1765-1771, where location 1765 contains the starting address of the first subroutine, 1766 the second,...

The SNAP variable list begins at location 2725. The variables are in floating point format, each requiring 3 memory cells. Some sample addresses are:

		exp.	mantissa
A	loc.	2725,	2726, 2727
B		2730,	2731, 2732
.			
.			
.			
I		2755,	2756, 2757
.			
.			
.			
T		3016,	3017, 3020
.			
.			
.			
X		3032,	3033, 3034

The 100 entry table begins at location 3470 and uses the same floating point format.

To manipulate floating point numbers, the "floating-point package" in SNAP may be used. For arithmetic manipulations, the FP interpreter is entered by the instruction

JMS 1 Z 7

All instructions following this are interpreted as floating point until a FEXT (0000) instruction is encountered. A return is then made to the next sequential instruction. The following are FP instructions:

FGET Y	get Y into FP accumulator
FPUT Y	put contents of FP accumulator in loc. Y
FADD Y	add Y to FP accumulator
FSUB Y	subtract Y from FP accumulator
FMPY Y	multiply FP accumulator by Y
FDIV Y	divide FP accumulator by Y

Indirect addressing may be used.

The resultant answer in each case is left in the FP accumulator (loc. 0044-0046). Other floating point instructions may be entered by using their numerical codes.

0002	Square Root
0003	Sine
0004	Cosine
0005	Arctan
0006	Exponent
0007	Logarithm (base e)

The instruction JMS I Z 5 allows a number to be typed into the computer from the keyboard in floating-point format. The number remains in the floating-point accumulator.

The instruction JMS I Z 6 will type out the number contained in the floating-point accumulator.

A sample sequence of instructions using the floating-point interpreter might be:

$X = \sqrt{A^2 + B^2}$, where A and B are SNAP variables, but X is not.

```
JMS  I Z 7        enter floating point interpreter
FGET  I  RA       get A
FMPY  I  RA       get A^2
FPUT  X           store temporarily
FGET  I  RB       get B
FMPY  I  RB       get B^2
FADD  TEMP        add A^2
0002              take  sqrt(A^2 + B^2)
FPUT  X
FEXT
  .
  .
  .
RA, 2725
RB, 2730
X, 0
0
0
.
.
.
```

Control Instruction EXTEND

This control instruction allows the 100_{10} point table to be extended to a maximun of 210_{10} points. The table will be divided into tables A and B, with one half the total number of points in each. The number of table entries may also be decreased to less than 100 points with this instruction.

After gaining CONTROL, type **EXTEND**,
Computer types TO . User inserts number of table entries desired.

Each time PRO is typed to enter a new program, the number of table entries will return to 100.

Note: DISTB will still display only 100 points.

Important: The table is extended to locations above 4145. Therefore, if GOPL is being used, be sure binary routines are located above the table extension.

SNAP Checkout Routine

SNAP contains a checkout routine which is read into core along with the program. To check that SNAP has been properly loaded into the computer, proceed as follows:

1. If SNAP was read in from magnetic tape.

   a) after computer types CONTROL, user types "CAL ".

   b) if system is working properly, "I'M OKAY" will be typed, then a SINE wave will appear on the display. The X- and Y-coordinates of the curve may be changed by adjusting analogue inputs 1 and 2.

   c) depress RED KEY (ALT MODE) to return to control and begin to enter program.

   d) if anything occurs other than the message and display described above, SNAP is not properly loaded into core.

2. If SNAP was read in from paper tape.

   a) set switches to 1200.

   b) depress LOAD ADDRESS key.

c) depress START key.

d) follow instructions above, starting at 1.a).

PROBLEM 1:

Find $Y = A\lfloor\log_e(N-B)\rfloor 7$

Type the value of Y for

B=34 N=36, 40, 50

_____

_____ Fill these in

_____

TYPEY
ENDPT

Think about how you would enter this program. You will be asked to run it on the computer.

Now repeat the calculation for

B=60 N=70, 80, 90

What additional steps will be necessary?

PROBLEM 2:

Calculate the value of C=CosA and type out A and C for the following values of A: 0, .1, .2, .3, .4 etc. Use a branch instruction so that the program runs indefinitely.

_____

_____

_____

_____

_____

ENDPT

PROBLEM 3:

The factorial of a number means the product of all the integers up to and including that number. Thus

4 Factorial = 4 x 3 x 2 x 1 = 24
6 Factorial = 6 x 5 x 4 x 3 x 2 x 1 = 720

If N is the number write a SNAP program using BRSUN and BR??N to compute and type out N Factorial. U=1, Z=0, P=Product, M= Each Factor

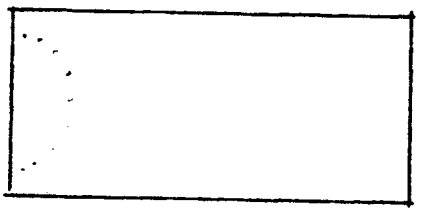
```
0010    P=U+Z
0011    M=U+Z     Sets these initially to one
0012    BRSUN
0013    _____
0014    _____
0015    _____
0016    TYPEP
0017    ENDPT
```

PROBLEM 4:

To Plot a Circle.

```
C=CSA       (C=cos angle A)
S=SIA       (S=sin Angle A)
X=R*C
Y=R*S
PLOTY
A=A+I
BP10I
ENDPT       N=200     1=0.1     A=0
```

Since the point X=0, Y=0 is at the left center of the scope, the circle will appear as a half circle.



Problem: Modify the above program to put the center of the circle in the center of the scope.

PROBLEM 5:

Draw a circle as before but with the radius controlled by knob 6.

PROBLEM 6:

Repeat problem on page 14 but find the radius which contains half the leaves.