INTERPROGRAM

MANUAL
IN ENGLISH LANGUAGE
FOR AUTOMATIC PROGRAMMING

CSIRAC

COMPUTATION LABORATORY
UNIVERSITY OF MELBOURNE

G. W. HILL
DIVISION OF MATHEMATICAL STATISTICS
Commonwealth Scientific and Industrial Research Organization

INTERPROGRAM

ENGLISH LANGUAGE

AUTOMATIC PROGRAMMING

FOR

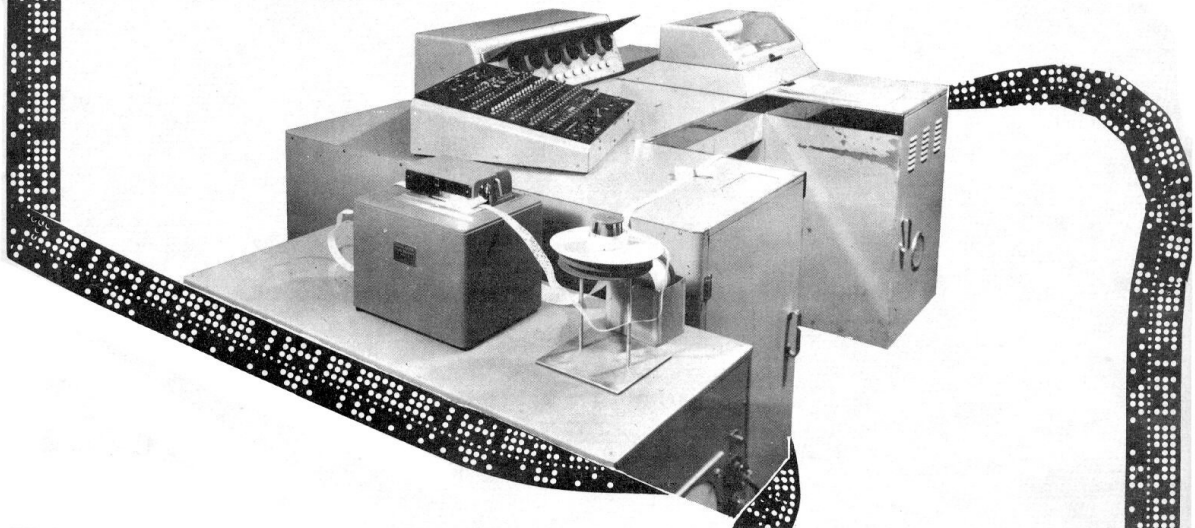
CSIRAC

G. W. HILL, 1960

PROBLEM

```
(1)  TITLE  ALPHA = BETA - 27.394 + A(3).(X/Y)
(2)  SYMBOLS FOR INTEGERS   J
(3)  MAXIMUM SUBSCRIPTS   A(3)
(4)  COMPILE THE FOLLOWING INTERPROGRAM

*1   INPUT J
*2   IF THIS IS ZERO, GO TO *7
*3   INPUT BETA, & A(3), & X, & Y
*4   TAKE X, DIVIDE BY Y
     MULTIPLY BY A(3), SUBTRACT 27.394
     ADD BETA, REPLACE ALPHA
*5   TAKE J, OUTPUT, OUTPUT ALPHA
*6   GO TO *1
*7   END OF INTERPROGRAM
```



DATA

J	BETA	A(3)	X	Y
1	25.382	2.754	7	0.92
2	31.542	-1.852	9	1.24
3	22.379	-0.023	11	2.57
4	23.456	+0.001	23	0.17
5	27.356	+1.113	3	-95.76
0				

RESULT

```
CSIRAC INTERPROGRAM COMPILED 1/10/60
ALPHA = BETA - 27.394 + A(3).(X/Y)
  1   18.9424
  2   -9.14884
  3   -5.11335
  4   -3.80275
  5   -.0727539
```

INTERPROGRAM

MANUAL
IN ENGLISH LANGUAGE
FOR AUTOMATIC PROGRAMMING

CSIRAC

COMPUTATION LABORATORY
UNIVERSITY OF MELBOURNE

G. W. HILL
DIVISION OF MATHEMATICAL STATISTICS
Commonwealth Scientific and Industrial Research Organization

Table of Contents

INTERPROGRAM

AUTOMATIC PROGRAMMING SYSTEM FOR CSIRAC

1 Introduction

The time required to learn to prepare a calculation for execution by the electronic computer, CSIRAC, is greatly reduced by the INTERPROGRAM system. The user writes steps of the calculation as commands in English language, using any convenient names for values arising in the calculation. The flexible INTERPROGRAM vocabulary can be used with little training and understood with even less. The computer translates INTERPROGRAM language into its own language and executes the required calculation. Use of the computer to construct its own program of operations is known as "automatic programming".

The BASIC INTERPROGRAM system caters for most calculations arising in practice. The system can be extended by increasing the vocabulary or by replacing the arithmetic system by one providing greater precision or for treating complex numbers. Such extensions will be described in separate appendices to the BASIC INTERPROGRAM MANUAL.

It should be noted that automatic programming is most suited to "once-only" calculations, such as arise in exploratory investigations. For extensive repeated calculations the human programmer can take advantage of special aspects of the computer and the problem. Use of computer language increases the task of preparation, but may substantially reduce the cost of execution of a calculation. The INTERPROGRAM system may be useful for investigating the method of calculation; incidentally providing an unambiguous specification for a computer language program.

2 Strategy

This manual defines the computer operation corresponding to each operation-word in the current vocabulary. The user prepares a "programme" of these operations for his computation by:-

(1) breaking the calculation into steps expressed in terms of the current vocabulary,
(2) organizing the sequence in which they are to be executed; catering for all possibilities, such as denominators becoming zero etc.,
(3) preparing data in the order required by the computer, or arranging the program to accept data already prepared,
(4) choosing a desired layout of results and arranging the program accordingly,
(5) typing the INTERPROGRAM and data onto punched paper tape and operating the computer to process these tapes.

3 Elements of an INTERPROGRAM

An INTERPROGRAM is typed on a tape-punching typewriter, such as the Flexowriter located in the CSIRAC Computation Laboratory. Each key operation produces a conventional typewriter response and at the same time punches a row of holes on paper tape. The typewritten record is a clear statement of the required calculation and the punched tape is ready for insertion into CSIRAC.

A tape "character" is punched for each of the typewriter operations:

Letters: `A B C D E F G H I J K L H N O P Q R S T U V W X Y Z`

Figure.: `0 1 2 3 4 5 6 7 8 9 + - ( ) . , # &` Stop Tab
`* ’ £ / x =` (Ignored during translation)

Operations: Space (denoted by "`SP`"), shift to letters case (`SL`), shift to figures case (`SF`), line feed and carriage return (`CR`), blank tape (`Tape Feed` key), erase (`LF` key).

Erase characters are ignored during translation. Errors detected while typing may be over-punched by erase characters and followed by corrections. Using these "characters", an INTERPROGRAM is typed as a set of statements, consisting of:-

(a) preliminary statements classifying symbols to be used,
(b) operation statements outlining steps of the calculation,
(c) the concluding statement "`END OF INTERPROGRAM`".

Operation statements are separated by 'comma' or CR characters, enabling operations to be typed on the same line or on separate lines. For ease of reference any line may be "labeled" with an asterisked integer up to “*150”. Operations are executed in the order in which they are typed, unless explicitly stated otherwise, e.g., "GO TO *7". Explanatory comments following the character "# " at the end of a line are ignored by the translation process. An operation statement could consist of;

*label	operation	operand# comment	‘comma’ or CR
integer	word(s)	name or constant (ignored)	character

For example:-

```
*1    TAKE PRESSURE    # POUNDS/SQ.IN.
*2    MULTIPLY BY AREA # SQ. FT.
      MULTIPLY BY 144, REPLACE FORCE
```

Labels are generally used only when needed for internal references within an INTERPROGRAM, but will also be used in examples in this manual to simplify references in the text.

EXAMPLE 1

Suppose it is required to tabulate the value of

$\alpha. \quad = \beta - 27.394 + a_3(x/y)$

for each set of values of β, a_3, x and y punched on a "data tape" as follows;

```
J     BETA       A(3)        X     Y
1     25.382      2.754      7     0.92
2     31.542     -1.832      9     1.24
3     22.379     -0.023     11     2.57
4     23.456     +0.001     23     0.17
5     27.356     +1.113      3    -95.76
0
```

The symbol J is used for the serial number of each set of data: the end being indicated by the value J = 0. The symbol A(3) will be used to illustrate subscripted variables, although A(0), A(1), A(2), A(4), etc., do not occur in this formula. An appropriate INTERPROGRAM is typed as follows:

```
(1)   TITLE ALPHA = BETA - 27.394 + A(3).(X/Y)
(2)   SYMBOLS FOR INTEGERS  J
(3)   MAXIMUM SUBSCRIPTS    A(3)
(4)   COMPILE THE FOLLOWING INTERPROGRAM

*1    INPUT J
*2    IF  THIS IS ZERO, GO TO *7       # END WHEN J = 0
*3    INPUT BETA, &  A(3), &  X, & Y
*4    TAKE X, DIVIDE BY Y, MULTIPLY BY A(3)
      SUBTRACT 27.394, ADD BETA, REPLACE ALPHA
*5    TAKE  J, OUTPUT, OUTPUT ALPHA
*6    GO TO *1
*7    END OF INTERPROGRAM
```

The punched paper tape produced by typing this INTERPROGRAM is translated by the computer and then executed using the data tape above. The results punched on tape by the computer are printed on the Flexowriter as follows:

```
CSIRAC INTERPROGRAM COMPILED 1/1/60
ALPHA = BETA - 27.394 + A(3).(X/Y)
1     18.9424
2     -9.14884
3     -5.11335
4     -3.80275
5     -.0727539
```

The operation statements *4 concern the calculation proper, while statements *1, *3 and *5 arrange input of data and output of results and statements *2, *6 and *7 control the repetitions required. Preliminary statement (2) enables the computer to treat listed symbols as integer variables; all others being treated in a more general way. Preliminary statement (3) reserves space for the four variables A(0), A(1), A(2) and A(3).

4 Preliminary Statements

The user begins punching an INTERPROGRAM by inserting appropriate characters into a copy of a standard punched tape available at the Computation Laboratory.

(1) TITLE The user inserts an appropriate title terminated by blank tape rows; to be later recorded at the head of results produced by the INTERPROGRAM.

```
(2)  SYMBOLS FOR INTEGERS  J K BETA MU, L, N, MAXIMUM
```

Up to 8 integer symbols, selected to suit the user, may be reserved by this type of preliminary statement.

```
(3)  MAXIMUM SUBSCRIPTS  ZZ(4),  A(3),  X(37)
```

This statement reserves working space for the subscripted variables chosen by the user; 5 spaces for ZZ variables, 4 spaces for A(0), A(1), A(2), A(3) and 38 for X(0) to X(37). To cater for negative values of subscripts, a "dummy" symbol such as ZZ(4) reserves space for 5 values A(-5), A(-4) .... A(-1), so that the subscript may range from -5 to +3.

```
(4)  COMPILE THE FOLLOWING INTERPROGRAM
```

This statement satisfies the requirement that the preliminary statements must end with the letters "GRAM".

If no integer symbols are required, add the word "NONE" in statement (2). If no subscripted variables are required, omit statement (3). Subscripted integers are listed in statement (2). E.g.:

```
(2)  SYMBOLS FOR INTEGERS J K(0) K(1) K(2) K(3) MU L(0) L(1)
```

Statement (3) may be repeated if all the subscripted symbols do not fit in one line. The title is a sequence of tape-codes, which may represent letters, figures-case or operation characters such as CR, SF, etc. Such a sequence will be called an "alphameric" sequence and must be terminated by blank tape (tape feed key). Statements (2), (3), and (4) end with a "CR" character.

5 Representation of Variables

Values arising in a calculation may be named to suit the user. Any sequence of upper-case letters may be used to represent a variable subject to the following rules:-

1. The first four letters are used and the rest ignored;
   i.e. `VARIABLE` and `VARIANCE` are treated as the same name.
2. A name ends with a type space, carriage return or figure shift;
   i.e. `MARKETVALUE` is a valid name but `MARKET VALUE` is not.
3. The names `IS` and `THIS` are used for special purposes; `THIS` is the name of the current result of calculation (*2 of example 1).
4. Integer subscripts may be used with one or two letter names;
   e.g. `A(123), CR(-31), MU(+7), T(0)`, but not `CREDIT(-31)`.
5. The subscript may include the name of an integer variable;
   e.g. `A(J), X(J+7)` or `MU(MAXIMUM-400)`.

6 Representation of Numbers

The basic INTERPROGRAM system automatically scales numbers to conserve accuracy. Scaled numbers may have any magnitude needed in practice, generally accurate to at least one part in 100,000. Provision is made for manipulation of integers taking exact values from -512 to + 511, including zero. (Integer arithmetic modulo 1024; i.e.+510+5 = +515 = 1024-509 = -509). Any value, named as an integer listed in the preliminary reservation statement occurs in "integer mode" whereas the value of any other variable occurs in "scaled mode".

Numerical data, results and constants in operation statements are typed in the conventional decimal notation as is illustrated in example 1. Results of one INTERPROGRAM may be used as data for another because the conventions for recording results and data are compatible. Negative numbers are preceded by the minus sign, but for positive numbers the plus sign may be replaced by SP. The "full-stop" is used as a decimal point, which is omitted when typing integers. The end of a number is indicated by SP or CR or the "TAB" character, which corresponds to the conventional typewriter tabulation operation.

Extremely large numbers or extremely small numbers are more conveniently represented with the "decimal index" typed in brackets as in:-

1.23456(13)	=	1.23456×10^{13}	=	12,345,600,000,000
-1.23456(-9)	=	-1.23456×10^{-9}	=	-0.000,000,001,234,56
0.5(-3)	=	0.5×10^{-3}	=	0.0005
-127.(8)	=	$-127 \times 10^{+8}$	=	-12,700,000,000

7 Calculation Statements

Statement *4 of example 1 illustrates that the object of ADD, SUBTRACT, MULTIPLY BY, DIVIDE BY or TAKE may be a constant or a variable (i.e number or a name) in either integer or scaled mode. TAKE X replaces any previous value of "THIS", the current result of calculation, by the current value of X. DIVIDE BY Y proceeds to form the quotient THIS/Y, which becomes the new value of "THIS". After ADD BETA, the value of the required result has been obtained as "THIS" and the statement REPLACE ALPHA copies the value of THIS into the "pigeon-hole" in store reserved for ALPHA. As a new name for a variable is introduced in preliminary and operation statements, the system reserves a new "pigeon-hole" in store for the value corresponding to that name. Subsequent statements such as ADD BETA or OUTPUT ALPHA will refer to the value in the appropriate "Pigeon-hole".

8 Initial Values for Variables

A scaled variable may be assigned a value by:-

```
SET X    = 23              # THIS = X = 23.0000, SCALED
SET X(1) = -499.999        # THIS = X(1) = -499.999, SCALED
SET TOTAL = 0              # THIS = TOTAL = 0.000000476837
```

where the latter is the value assigned to a scaled "zero".

An integer variable may be (a) assigned an integral value, (b) increased or decreased by an integral constant or (c) given the current value of another integer variable or its complement:-

```
(a)   SET J = 23           SET K = -499
(b)   SET J = J + 5        SET K = K - 99
(c)   SET J = K            SET J = -K
```

The value of THIS is unaffected by SET commands concerning integer variables, but with scaled variables THIS takes the scaled value mentioned.

9 Repetition Using "&"

"&" may be used instead of repeating the operation in the preceding statement as is illustrated by *3 of example 1. "&" may be used for SET, provided that integer variables are not SET to new values after setting values for scaled variables.

```
      TAKE 3, MULTIPLY BY X, & J, & X(1), & MU, & SUM
Thus:- SET J = 3, & MU = 5, & X = 27.35, & X(1) = -35.27
      SET K =  K +5, & M = -K, &  SUM = 0,  SET L = 15
```

where J, MU, K, M and L are integer variables, while X, X(1) and SUM are scaled.

10 Scaled and Integer Values

The INTERPROGRAM system provides automatic scaling of numbers to simplify calculations and also provides for integer arithmetic required for counting repetitions, constructing subscripts etc. The user may combine operations with integers and scaled numbers without restriction, so that the user is rarely concerned with the "mode" of the result of any calculation, i.e., integer or scaled. The mode is relevant only in the case of `OUTPUT` and `FORM` statements. The diverse formats of printed integers and scaled numbers will be detailed on page 8, and the requirement of `FORM` statements, that the argument be scaled, will be noted on page 9.

The mode of a constant used in a statement is "scaled" if a decimal point occurs or "integer" otherwise; except in the special case of SET statements already treated. The mode of a variable is "integer" if the variable is listed in preliminary statement (2):, otherwise it is a "scaled mode" variable. However, THIS, the current result of calculation, may occur in either scaled or integer mode, as determined by the following rules:-

1. After `TAKE`, `INPUT` or `REPLACE` the value and mode of THIS will be the same as the value and mode of the operand.
2. After `ADD`, `SUBTRACT` or `MULTIPLY BY` the mode of `THIS` will be
   (A) integral if both `THIS` and the operand were integers
   (B) scaled if either `THIS` or the operand were scaled.
3. After `DIVIDE BY`, `THIS` will be in scaled mode, even if both operands were integers.
4. After "`SET X =` constant", where X is a scaled variable, `THIS` will be in scaled mode, even if the constant is an integer.
5. All other operations leave the value and mode of `THIS` unchanged.

If the user wishes to convert `THIS` from integer mode to scaled mode, a `REPLACE` statement with scaled variable operand is used, e.g., `REPLACE THIS`. The integral part of a scaled number is obtained by `REPLACE J`, (or any other integer variable not currently needed,) e.g., `THIS = 187.644` becomes
`THIS = J = 187`

EXAMPLE 2

If X is an amount of money (e.g., 12.0000 pounds) and J is an interest rate percent (e.g., J = 5), the interest can be calculated and converted to shillings and pence as follows:-

```
TAKE J, DIVIDE BY 100               # SCALED - RULE 3
MULTIPLY BY X, REPLACE INTEREST     # SCALED - RULES 2(B), 1
MULTIPLY BY 20, REPLACE J           # INTEGER - RULE 1
OUTPUT, SUBTRACT INTEREST           # -VE FRACTION OF SHILLING
MULTIPLY BY -12, REPLACE J, OUTPUT  # THIS = INTEGER PENCE
```

11 Input of Numbers

The statement "INPUT X" replaces THIS and X by the number next available on the data tape in the tape reader. If the operand is an integer variable, as in "INPUT J" the number must be punched as an integer, but for scaled operands or for the statement "INPUT" the result is scaled whether the datum is punched as an integer or as a scaled number.

Normally statements are executed in the order in which they are typed, but INPUT of the characters "END" or # causes the statement following the input statement to be ignored. This may be used to control a program "switch" as illustrated subsequently in example 4.

12 Output of Numbers

Numbers punched on the result tape by OUTPUT statements are subsequently printed on the Flexowriter. The layout and format are determined in the following way:-

(1) The terminating character of an OUTPUT statement may be a comma or CR character, which causes the punched result to end with two spaces or CR respectively. In example 4, each OUTPUT statement except the last ends with "comma", so that results are printed along the line until CR at the end of the last result ends the line.

(2) When the operand is explicitly mentioned in an output statement, such as OUTPUT X(7) or OUTPUT J, the number of type-spaces for the result may vary depending on the magnitude of the result. A more regular format is obtained by implicit reference to the value of THIS by the statement OUTPUT. For example;-

```
TAKE X,OUTPUT, OUTPUT X          TAKE J, OUTPUT, OUTPUT J

-.000888   -.000888888           -444       -444
888.888    888.888                44        44
-.000008   -.00000888888        -  4        -4
-888888888 -888888888               0
```

In example 4, implicit references by the statements "OUTPUT", produce neat columns; scaled numbers less than 10^5 always occupy eight type spaces. In example 1, "TAKE J, OUTPUT" ensures that integers occupy four type spaces, enabling alignment of the result of OUTPUT ALPHA, whose explicit reference to ALPHA ensures that small numbers will be printed to six significant figures.

Note that OUTPUT statements do not change THIS, i.e., TAKE Y, OUTPUT X or TAKE Y, OUTPUT, leave THIS = Y.

13 Headings and Layout of Results

A sequence of tape characters terminated by one or more blank tape rows is refered to as "alphameric". Such sequences may be required for headings to identify numbers or for layout of printed results. An alphameric sequence punched following the SP, CR or SF character,which ends "PUNCH THE FOLLOWING CHARACTERS", is stored during translation and punched on the result tape whenever the statement is executed. For identification purposes an alphameric sequence may be copied from the data tape to the result tape by the statement "COPY TAPE". 10-20 erase characters punched before alphameric headings on data tapes make it easier to position the tape in the reader at a "non-blank-row".

Numbers output to the result tape may be input as data in another calculation. Alphameric sequences on result tapes end with blank tape and may be ignored on re-reading by using "IGNORE TAPE", which merely searches for the next blank tape row.

14 Mathematical Functions of "THIS"

The basic vocabulary includes statements for forming the square root of x, sin x, cos x, tan x, arctan x, exp (x) or $\log_e$ x. The statement "FORM SQUARE ROOT" replaces the scaled value THIS by its square root as a scaled value. The statements FORM SINE and FORM COSINE and FORM TANGENT and FORM ARCTAN treat angles in units of 180° , i.e., 1.75000 = 315° = -45° . The result of FORM ARCTAN is in the range (-0.5, +0.5) (-90° , +90°), which may be extended to the full range by the programmer. The statement FORM NATURAL LOG produces the logarithm to base *e* of the absolute value of THIS (*e* = 2.71828) and FORM EXPONENTIAL replaces THIS by exp(THIS). In all cases THIS should be a scaled value and the result will be in scaled mode.

Using these statements it is possible to construct more elaborate operations such as log (x+ iy) = u+ iv, where u = log $(x^2+ y^2)^{1/2}$ and v = arctan y/x.

EXAMPLE 3.

```
      TAKE X, MULTIPLY BY X, REPLACE XSQUARED
      TAKE Y, MULTIPLY BY Y, ADD XSQUARED
      FORM SQUARE ROOT, FORM NATURAL LOG, REPLACE U
      TAKE Y, DIVIDE BY X, FORM ARCTAN
      IF X IS POSITIVE, GO TO *37                # EXTEND  ARCTAN
      ADD 1, IF THIS IS GREATER THAN 1, ADD -2   # TO RANGE (-1,1)
*37   MULTIPLY BY 3.14159, REPLACE V             # V IN RADIANS
```

EXAMPLE 4. Tabulation of functions using FORM-statements.

Interprogram typed on Flexowriter.

```
      (1)   TITLE  EXAMPLE OF OUTPUT, LAYOUT AND FUNCTION FORMATION
bbbbb[1](2)   SYMBOLS FOR INTEGERS  NONE
      (4)   COMPILE THE FOLLOWING INTERPROGRAM

      PUNCH THE FOLLOWING CHARACTERS
      DATUM       LOG   EXP(LOG)    SINE  TAN   ARCTAN(TAN)
      ------------------------------------------------
bbbbbbCOPY TAPE                                  # DATA TAPE IDENTIFICATION
      *2    INPUT DATUM, GO TO *3,  GO TO *4     # WHEN 'END' IS READ
      *3    OUTPUT, FORM NATURAL LOG, OUTPUT,
            FORM EXPONENTIAL, OUTPUT,            # EXP(LOG(DATUM))
            TAKE DATUM, FORM SINE, OUTPUT,
            TAKE DATUM, FORM TANGENT, OUTPUT,
            FORM ARCTAN, OUTPUT                  # END PRINTED ROW
            GO TO *2                             # REPEAT FOR EACH DATUM
VALUE
      *4    END OF INTERPROGRAM
```

Data Tape;- Alphameric heading, numbers and "END".

```
eeeeeSAMPLE SET OF DATA
bbbbbee0.0  0.166667    0.25  0.333333    0.5         0.666667    1.0
2.71828     -0.166667   -0.25 -0.5        123.456     -4567.89    END
```

Result tape:- Printed on the Flexowriter.

CSIRAC INTERPROGRAM COMPILED 1/1/60

EXAMPLE OF OUTPUT, LAYOUT AND FUNCTION FORMATION

DATUM	LOG	EXP(LOG)	SINE	TAN	ARCTAN(TAN)
SAMPLE SET OF DATA					
.000000	-14.5580	.000000	.000001	.000001	.000001
.166667	-1.79174	.166669	.499998	.577352	.166666
.250000	-1.38628	.250000	.707105	1.00000	.250000
.333333	-1.09861	.333335	.866020	1.73204	.333333
.500000	-.693138	.500000	.999998	524297	.500000
.666666	-.405462	.666667	.866024	-1.73204	-.333335
1.00000	.000000	1.00000	.000001	.000001	.000001
2.71829	.999982	2.71821	.773952	-1.22220	-.281726
-.166667	-1.79174	.166669	-.500003	-.577354	-.166666
-.250000	-1.38628	.250000	-.707107	-.999998	-.249996
-.500000	-.693138	.500000	-.999998	524297	.500000
123.456	4.81582	123.443	-.990377	7.15692	.455810
-4567.94	8.42670	4566.89	.336889	.357805	.109375

1 Editorial note, not in the original document: The “b” characters are tape blank symbols. The “e” that follows is an erase (LF) symbol. These are used as separators in the program and data.

15 Condition Statements

The second statement of a pair of statements such as:-

```
IF THIS IS NEGATIVE, GO TO *7
```

is executed only if the condition in the first statement is satisfied. Such condition statements may refer to any scaled or integer variable, including THIS, and may require that its value be positive, negative or zero, e.g.,

```
IF X IS POSITIVE      IF J IS POSITIVE      IF THIS IS POSITIVE
IF X IS NEGATIVE      IF J IS NEGATIVE      IF THIS IS NEGATIVE
IF X IS ZERO          IF J IS ZERO          IF THIS IS ZERO
```

The integer,0, is considered both positive and zero. For scaled numbers "zero" is approximately z = 0.000000 953675 (i.e., 2^{-20}) or less. A scaled value is the range (-z, + z) is considered zero, being positive in the range (0,z) or negative in the range (-z,0).

In comparisons of values by condition statements such as:-

```
IF THIS IS GREATER THAN X(J)     IF THIS IS SMALLER THAN 23
```

the operand may be integral or scaled, constant or variable. These conditions are not satisfied by equality of values and positive values are considered to exceed negative values; i.e., + 2 is greater than -200, but not greater than +2. On the other hand, signs are disregarded in magnitude comparisons such as;

```
IF  THIS IS MUCH GREATER THAN X(J)
IF  THIS IS MUCH SMALLER THAN 23
```

which would be satisfied if `THIS` is over a million times the magnitude of X(J) or, in the second case, if `THIS` is less than about 23 millionths.*

1. `THIS` remains unaffected by any condition statement.
2. For scaled values of `THIS` the statement "`IF THIS IS ZERO`" is equivalent to "`IF THIS IS MUCH SMALLER THAN 1`".
3. Magnitude comparisons may be achieved by a group of statements, whose effect is more predictable in the marginal cases; e.g.,

```
IF THIS IS NEGATIVE, MULTIPLY BY -1   # FORMS MODULUS
MULTIPLY BY 1000000, IF THIS IS SMALLER THAN 23
```

---

* Numbers are represented in "floating binary" form; e.g., 6 = 0.75 x 2^3 ; 3 being called the "binary index". The precise condition involved in order of magnitude comparisons is that the binary indices differ by more than 20. This implies that the values are in the ratio 1/1000000 or less.

16 Repetition of Operations

It is often useful to have a group of operations repeated a number of times; e.g.-

```
      SET J = 5, & TOTAL = 0
*22   INPUT A(J), ADD TOTAL, REPLACE TOTAL
      REPEAT FROM *22 J TIMES     # J DECREASES TO ZERO
      SET J  = -5
*23   OUTPUT A(J + 5), REPEAT FROM *23 J TIMES
      OUTPUT TOTAL
```

Because the "REPEAT" statements decrease the magnitude of J, values are read from tape in the order; A(5) ,A(4) ,A(3),...A(0) and their total accumulated: Then the values are punched in the reverse order; A(0), A(1), A(2),... A(5), followed by the total. When finally J = 0, the "REPEAT" operation is complete and the calculation proceeds with statements following the REPEAT statement.

1. Any integer variable may be used in place of J, and any label is permitted.
2. A value must be assigned to the integer before the loop is entered.
3. For J = 5 or -3 initially the loop is executed once before repetitions begin, so that the loop is executed 6 times with J= 5,4,3,2,1,0 or 4 times with J = -3, -2,-1, 0.
4. The "REPEAT" statement must terminate with CR.

17 Routine Processes

The vocabulary may be extended by defining a frequently required group of operations as a "routine process", preceded by a label and ended with the statement "END OF PROCESS DEFINITION".

For example, arccosine (y) may be derived from arctan (x), where $x= (1-y^2)^{1/2}/y$;-

```
*100  # PROCESS DEFINED FOR  ARC-COSINE(THIS)
      REPLACE ARGUMENT, MULTIPLY BY THIS, & -1 , ADD 1
      FORM SQUARE ROOT, DIVIDE BY ARGUMENT, FORM ARCTAN
      END OF PROCESS DEFINITION  # THIS = ARC-COS(ARGUMENT)
```

A process defined in this way may be invoked at any stage in the INTERPROGRAM by the statement;-

```
      EXECUTE PROCESS *100
```

When the process has been executed, the calculation proceeds with statements following the EXECUTE statement.

1. Labels and variables of a process must not conflict with those used elsewhere in the INTERPROGRAM.
2. A routine process may invoke any subordinate process, which does not invoke further subordinate processes.

18 Miscellaneous Operations

The sequence of execution of labeled statements and the selection of routine processes may be controlled by values of integer variables, e.g.,-

```
GO TO *J   GO TO *J + 3    EXECUTE PROCESS *J - 25
```

This technique may be combined with multiple labeling of statements.

EXAMPLE 5. Suppose business transactions are recorded in random order, with each account number (0 to 9) followed by the corresponding amount and that it is desired to accumulate amounts for accounts 1,3,5,8 as CREDIT, accounts 2,6,9 as DEBIT and to ignore accounts 0,4,7 until the data ends with account number 10.

```
                SET CREDIT =  0, & DEBIT = 0
*1  *50 *54 *57 INPUT J, INPUT AMOUNT, GO TO *J + 50
*51 *53 *55 *58 ADD CREDIT, REPLACE CREDIT, GO TO *1
*52 *56 *59     ADD DEBIT, REPLACE DEBIT, GO TO *1
*60             OUTPUT CREDIT, OUTPUT DEBIT, PAUSE 23
```

It is convenient to be able to stop the computer for such purposes as placing a new tape in the reader. The statement PAUSE 23 or PAUSE (J) stops the computer until the "restart " button is pressed. Because the value of the integer mentioned is displayed in binary form on the "interpreter neons", diverse numbers in PAUSE statements enable the operator to identify the stage reached in executing an INTERPROGRAM,

The tape-code for Flexowriter "stop" is used in typed INTERPROGRAMS to stop the computer during translation so that a further INTERPROGRAM tape may be placed in the reader. Long INTERPROGRAMS may be produced as "page-length" tapes ending with a stop-code. This reduces the effort required to modify an INTERPROGRAM. At a PAUSE in execution or a stop-code in translation, the command "S T" stops the computer, i.e.; neons marked S and T are lit.

19 Operation of the Computer

INTERPROGRAMS can be written without reference to details of computer operations. Instruction in operation of the computer switch-board will be available at the CSIRAC laboratory. The system causes the monitor printer to type instructions to the operator to ensure that switches are correctly set. Certain logical or spelling errors in INTERPROGRAMS are automatically detected by the system and noted on the console printer. Thus "UNASSIGNED VARIABLE" is typed if the INTERPROGRAM calls for a variable, to which a value has not been assigned by a statement of the type; REPLACE, INPUT or SET. If an "OPERATION NOT IN DIRECTORY" or "NON INTEGER SUBSCRIPT" is encountered, the user should mark the tape row last read and print the preceding few inches of tape to check for errors. Assistance will be available at the laboratory for diagnosis of errors or for reorganization of an INTERPROGRAM, which produces the response;- "STORE CAPACITY EXCEEDED".

20 Summary of INTERPROGRAM Rules

These Operations applied to	these operands change	THIS to;-	see p
TAKE ...	X, J, 0.3, 3 (a)	Operand	a
ADD ...	X, J, 0.3, 3, THIS	Sum	g
SUBTRACT ...	X, J, 0.3, 3	Difference	e
MULTIPLY BY ...	X, J, 0.3, 3, THIS	Product	
DIVIDE BY ... (b)	X, J, 0.3, 3	Quotient	6
IF THIS IS GREATER THAN.	X, J, 0.3, 3		
" SMALLER THAN.	X, J, 0.3, 3	THIS	
" MUCH GREATER THAN ...	X, J, 0.3, 3	is	11
" MUCH SMALLER THAN ...	X, J, 0.3, 3	not	
IF ... IS ZERO	X, J, THIS	changed	
IF ... IS POSITIVE	X, J, THIS		
IF ... IS NEGATIVE	X, J, THIS		
REPLACE ...	X, J, THIS	integer J or	7
INPUT ... (c)	X, J, or THIS	Scaled X	8
OUTPUT ... (d)	X, J, implied	Unchanged	8
SET ... (scaled variable) (e)	X = 0.3, X = 3	Scaled X	6
SET ... (integer variable)	J=3, J=J+3, J=-K		6
GO TO ...	*3, *J, *J + 3		13
EXECUTE PROCESS ...	*3, *J, *J - 3		12
REPEAT FROM .....TIMES (f)	*3, J		12
PAUSE ...	3, (J)	THIS	13
PUNCH THE FOLLOWING	Alphameric	is	
CHARACTERS ... (g)	Sequence	not	9
		changed	
COPY TAPE	Alphameric sequence		
IGNORE TAPE	on data tape		9
END OF PROCESS DEFINITION	No operand		13
END OF INTERPROGRAM	is required		2

(a) Typical operands;- X represents any scaled variable; C(12), DEBIT, KR(J-23). J represents any integer variable listed in preliminary statements. 0.3 and 3 represent typical scaled and integer constants. Alphameric sequence; any characters ending with blank tape.

(b) Scaled mode quotient even for the ratio of integers (see page 7.)

(c) For INPUT J the tape datum must be punched as an integer.

(d) End with CR or comma to end printed number with CR or SP, SP.

(e) If repeating SET using &, treat integer before scaled variables.

(f) REPEAT statement must terminate with CR.

(g) In this case the alphameric sequence must end in letters case before blank-tape.

PRELIMINARY STATEMENTS (see page 4) Copy the standard heading tape**.** inserting a title followed by blank tape; symbols for up to 8 integers (or NONE); maximum subscripts for one or two-letter variables (omit statement (3) if no subscripted variables); and finally the word INTERPROGRAM.

"FORM-FUNCTION" STATEMENTS (see page 9) `FORM SQUARE ROOT`, `FORM SINE`, `FORM COSINE`, `FORM TANGENT`, `FORM ARCTAN`, `FORM NATURAL LOG`, `FORM EXPONENTIAL`. `THIS` is scaled before and after an operation. Angles are fractions of 180° ; arctan is in the range (-0.5, +0.5).

21 Terminating Characters

Preliminary statements; end (1) with blank tape, (2)&(3) with CR, (4) with "`GRAM`".
Operation statement; comma or CR (`REPEAT` statements with CR).
Numbers (constants and data); SP, CR or TAB.
Alphameric sequence; blank tape.
Comment; begins with "#", ends with CR.
Interprogram; "`END OF INTERPROGRAM`" (this must end with SP or CR or ,)
Process defined; "`END OF PROCESS DEFINITION`".
Data blocks on tape; "`END`" or "#".

22 Additional Notes

(1) On completion of an INTERPROGRAM, the computer "hoots". To repeat the calculation for another data tape, clear the sequence register and restart.
(2) Labels are valid from *1 to *150; the asterisk is ignored by the computer.
(3) Blank tape, essential after alphameric sequences, is not copied by the CSIRAC Flexowriter operating in "print-mode", but is copied in "non-print mode".
(4) Leave about 8 inches of tape before INTERPROGRAM or data to enable the tape to be placed in the tape reader.
(5) The character following the letter "S" of "`PUNCH THE FOLLOWING CHARACTERS`" should be either, CR or SP and is not included on the result tape.
(8) An alphameric sequence is assumed to begin and end in letters case.

---

23 Acknowledgments

J.E. Meiss, Organization of typography, courtesy, Data Control.
Xerography, Department of Supply, Central Drawing Office, Maribyrnong.
Offset printing of text, University of Melbourne.