P A C T   P R I M E R

INTRODUCTION

PACT, "Project for the Advancement of Coding Techniques",
is a system intended to make utilization of large scale
digital computers easier for experienced programmers as
well as possible for the non-experienced. The aim of
this PRIMER is to make PACT I (as coded for the IBM Type
701) available to and usable by the latter group. In
addition, it can serve as an introduction to the more
detailed manual for those more experienced users.

AUGUST 1955

In the following discussion, we are going to treat PACT I not as a system for use on a machine, but rather as the machine itself. We are, in a certain sense, going to design a machine which will enable us to do the problems which arise in the course of our work. Our machine, like all machines, will not include all of the desirable features that one might ask for. We think, however, that it will be better than most.

In order to design a machine to do our problems, we must see what kind of problems we have. Let's begin with a simple one:

EXAMPLE:

EG-1   Compute $Y = \dfrac{AZ^2 + BZ + D}{CZ - E}$ for one value of Z

What do we need in our machine to do this computation? First, we need registers to hold the quantities involved. We shall call these registers and their contents "FACTORS".* We must have an instruction ("OPERATION") which allows us to "TAKE" quantities from the Factors. We must have the ability to "ADD", "SUBTRACT", "MULTIPLY", and "DIVIDE". We should be able to put our final answer into one of the Factors (i.e. we want to make a Factor "EQUAL" to something). We must be able to "HALT" when our problem is completed. We must have the ability to do the "STEPS" involved in the computation in a particular order.

Now, let us write down a "CODE" to carry out the computa-

---

* When capital letters and quotation marks are used, the word is being defined.

tion we wish to do:

STEP	OPERATION	FACTOR
1	TAKE	Z
2	MULTIPLY (by)	C
3	SUBTRACT	E
4	EQUALS	DENOMINATOR
5	TAKE	Z
6	MULTIPLY (by)	A
7	ADD	B
8	MULTIPLY (by)	Z
9	ADD	D
10	DIVIDE (by)	DENOMINATOR
11	EQUALS	Y
12	HALT	

1A

Note that in our Code, the result of one Step is available for use as an operand for the succeeding Step only, and that we therefore had to create the additional Factor "DENOMINATOR". In general, we will have many such "intermediate" quantities. Let us, therefore, create an additional convenience on our machine; namely, that the result of step n (denoted by "Rn") shall be available on all succeeding steps. In order to differentiate between ordinary Factors and these new ones we will "CLUE" the machine by placing the R in a special place.

Note also that we write many characters to specify our operations. Let us therefore, again for convenience, make our machine accept "SYMBOLS" in place of "spelled out" operations:

OPERATION	SYMBOL
TAKE	(no symbol)
ADD	+
SUBTRACT	-
MULTIPLY	x
DIVIDE	/
EQUALS	EQ
HALT	HALT

We shall now rewrite the Code of our preceding example:

(1B)

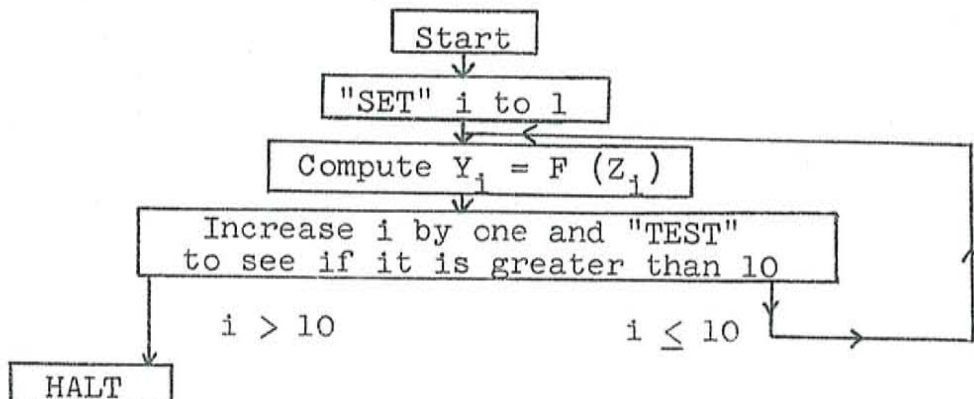| STEP	OPERATION	CLUE	FACTOR	NOTES
1			Z	
2	x		C	
3	-		E	CZ-E
4			Z	
5	x		A	
6	+		B	ZA+B
7	x		Z	
8	+		D	NUMERATOR
9	/	R	3	÷ Results of 3
10	EQ		Y	
11	HALT			

Let us now change our problem somewhat and see what additional features we should like to have on our machine.

EXAMPLE:

EG-2   Compute $Y_i = \dfrac{AZ_i^2 + BZ_i + D}{CZ_i - E} = F(Z_i)$

for $i = 1$ to 10 in steps of 1 $[i = 1(1)10]$

First we shall write a "FLOW DIAGRAM" to indicate how we are going to do our problem:

We need to add to our machine, to do this problem, the ability to specify subscripts and to "SET" and "TEST" them. As a part of our testing, we also need the ability to go back to a particular step in our code. We shall put our subscripts in a special place and also allow each of our Factors to have two subscripts. (Eventually, we should like to deal with matrices.)

STEP	OP	CLUE	FACTOR	S_1	S_2
1	SET			i	1
2			Z	i	
3	X		C		
4	-		E		
5			Z	i	
6	X		A		
7	+		B		
8	X		Z	i	
9	+		D		
10	/	R	4		
11	EQ		Y	i	
12	TEST		2	i	10
13	HALT				

EXPLANATION:

Step (1): Set i equal to 1 wherever it appears as a subscript between the Set and Test Operations (steps 2, 5, 8, 11).

Steps (2) through (11): These steps are essentially the same as steps (1) through (10) of (1B).

Step (12): Increase i by 1 wherever it appears as a sub-script between the Set and Test operations (steps 2, 5, 8, 11). If the new i is greater than 10,

Note: Our machine shall have the following restriction: when-ever an operation (such as Test) requires that the next step to be carried out is not the next step as numbered, then the step to be carried out must be within a certain class which we define as "the first step in a sequence of steps" and which will be more fully explained later.

proceed to the next step [step (13)]; if it is less than or equal to 10, go ("TRANSFER") to the step specified as the Factor step (2).

As a further extension of the problem we have been doing, suppose we wanted to do the following:
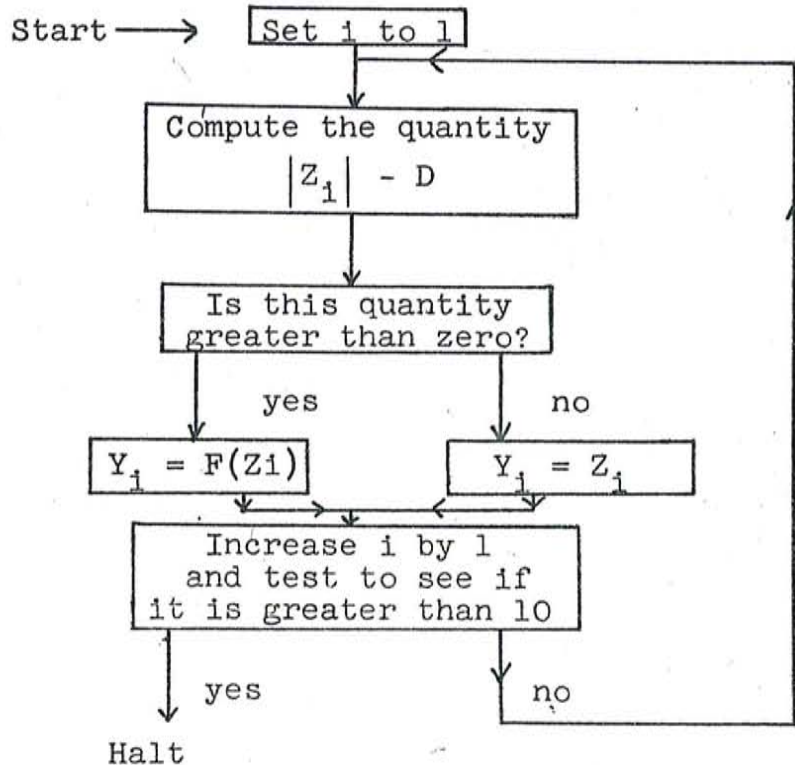
EG-3   Compute: $Y_i = \dfrac{A Z_i^2 + B Z_i + D}{C Z_i - E} = F(Zi)$

$\underline{if} \quad \left| Z_i \right| > D \qquad\qquad i = 1(1)10$

$Y_i = Z_i$

$\underline{if} \quad \left| Z_i \right| \leq D$

Again, we shall begin by making a Flow Diagram

Start $\longrightarrow$ Set i to 1

Compute the quantity
$\left| Z_i \right| - D$

Is this quantity greater than zero?

yes   no

$Y_i = F(Zi)$ $Y_i = Z_i$

Increase i by 1
and test to see if
it is greater than 10

yes   no

Halt

This poses two additional problems; we should like to be able to work easily with the absolute value of numbers; we must be able to make comparisons and as a result of these comparisons "TRANSFER" to a given Step.

Operation	Symbol	Description
ABSOLUTE	ABS	"Take" the absolute value of the Factor.
TRANSFER ON POSITIVE	TP	If the result of the previous Step is greater than zero, go to the Step noted as the Factor; otherwise proceed in sequence.
TRANSFER	T	Go to the step noted as the Factor.

Code for EG-3

STEP	OP	CLUE	FACTOR	S_1	S_2
1	SET			i	1
2	ABS		Z	i	
3	-		D		
4	TP		8		
5			Z	i	
6	EQ		Y	i	
7	T		18		
8			Z	i	
9	x		C		
10	-		E		
11			Z	i	
12	x		A		
13	+		B		
14	x		Z	i	
15	+		D		
16	/	R	10		
17	EQ		Y	i	
18	TEST		2	i	10
19	HALT				

Note that it is essential that all steps which use the subscript i must occur between the Set and Test.

EXAMPLE 4: Let us refer back to EG-1, replacing Y by F(Z):

$$F(Z) = \frac{A\,Z^2 + B\,Z + D}{C\,Z - E}$$

Now suppose we wished to compute:

$$W = P \cdot F\,(Q) + T \cdot F\,(R)$$

Note that with our machine as it now stands, we would be obliged to write the code of (1B) twice in order to accomplish our purpose (unless we wished to use a trick which is left as an exercise for the reader). But we don't like tricks and we are too lazy to write excessively, so we'll improve our machine. We will call the Steps associated with a given computation a "REGION", and we shall have our machine able to "DO" a Region and then return to the Step following the Do. The code of (1B) then becomes

REGION	STEP	OP	CLUE	FACTOR	S_1	S_2
1	1			Z		
1	2	x		C		
1	3	-		E		
1	4			Z		
1	5	x		A		
1	6	+		B		
1	7	x		Z		
1	8	+		D		
1,	9	/	R	3		
1	10.	EQ		Y		

The Halt is omitted because we no longer wish to stop at the end of this particular computation.

Our new operation is "DO". Do the Region specified as the Factor and return to the Step following the Do.

The Code for computing w

REGION	STEP	OP	CLUE	FACTOR	S_1	S_2
2	1			Q		
2	2	EQ		Z		
2	3	DO		1		
2	4			Y		
2	5	x		P		
2	6	EQ		W		
2	7			R		
2	8	EQ		Z		
2	9	DO		1		
2	10			Y		
2	11	x		T		
2	12	+		W		
2	13	EQ		W		

Note: Step (6) was needed for the reason described below; W was chosen as the factor in order to save factor space.

Our machine shall have the following restriction:

The results of a step in one region shall not be available as a factor in any other region unless defined as a factor by an Equals; in addition, they will not be available to any steps in the given region if a "DO" has occurred between the step whose result is wanted and the step which uses this result.

SCALING

In all of our preceding examples, we have been working with numbers whose magnitude has been ignored. Unfortunately, the magnitude of numbers cannot be overlooked when one is doing computation work (if any kind of significance is to be retained). Therefore, we shall be required to specify for each of our arithmetic steps, the number of possible digits to the left of the point resulting from the computation being done on that step (or, if the result is less than 1.0, the number of lead zeros in the result). We shall differentiate between whole numbers and lead zeros by putting a minus sign on the "Q" that we use to specify the latter.

For example:

12·345 would have Q = 2 to indicate 2 whole numbers

·01234 would have Q =-1 to indicate 1 lead zero.

There is one additional complication we must mention before we proceed to make life somewhat easier: our machine is a BINARY computer so that the whole numbers or lead zeros must be expressed in terms of powers of 2. (A table of powers of 2 is included in the appendix.) Thus for the numbers in the preceding example:

12·345 has Q = 4 $(2^3 < 12\cdot345 < 2^4)$

·01234 has Q = -6 $(2^{-7} < \cdot01234 < 2^{-6})$

Now to make life somewhat easier: we first note that every Factor and the result of every step has a Q associated with it. We may then define a "normal" specification of Q as follows:

1. In addition and subtraction, the Q of the result
    is the Q of the larger of the two operands.

2. In multiplication, the Q of the result is the
    sum of the Q's of the two operands.

3. In division, the Q of the result is the difference
    of the Q of the Dividend and that of the Divisor.

(Note that 2 and 3 are merely exponential arithmetic.)

We shall now give our machine an additional capability:
With but one exception, where Q is not specified, our machine
shall assume a "normal" Q for arithmetic operations. The ex-
ception is that every EQUALS shall have a Q specified.

Since it is not impossible to make errors in scaling,
we shall also put some checks into our machine:

1.) If the Q ("normal" or specified) for a division
    is not large enough to accommodate the quotient,
    the machine will halt and turn a special (DIVIDE
    CHECK) light on.

2.) If the Q ("normal" or specified) for any other
    operation is not large enough to accommodate the
    result of that operation, a special "overflow
    indicator" shall be turned on. (Note that in
    multiplication this may only occur from the
    specified Q being wrong.) We shall also supply
    an operation to "test" the overflow indicator.

Transfer on Overflow (TF). If the overflow indicator is on,
turn it off and then go to the Step noted as
the Factor; otherwise, proceed in Sequence.

Let us now return to example 2, giving values to our letters;

Compute
$$Y_i = \frac{AZ_i^2 + BZ_i + D}{CZ_i - E} \qquad i = 1(1)10$$

where A = 2
      B = 3
      C = .56
      D = 7.5
      E = 16
      $.6 \leq Z_i \leq 3.5$

Let us assume that the numbers are in the machine with Q's as noted below (the method of entering numbers into the machine and getting answers out of it, "Input-Output", will be discussed later).

Factor	Q
A	2
B	2
C	0
D	3
E	5
Z_i	2

Notes 1. The variable Z_i must be scaled for the maximum value it will take on,

    2. Although $2^4 = 16$, there are 5 digits in the binary representation of 16 (10000); thus the scaling for e.

We shall now proceed to analyze our computation to find the <u>maximum</u> magnitude the various steps may give us. Note that in order to evaluate the maximum Y_i, we must also obtain the minimum of the denominator

	Specified	"Normal"
$AZ_i \leq 7.0$	$Q = 3$	$Q = 4$
$AZ_i + B \leq 10.$	$Q = 4$	$Q = 4$
$(AZ_i + B)Z_i \leq 35$	$Q = 6$	$Q = 6$
$(AZ_i + B)Z_i + D \leq 42.5$	$Q = 6$	$Q = 6$
$.336 \leq CZ_i \leq 1.96$	$Q = 1$	$Q = 2$
$14 \cdot 04 \leq |CZ_i - E| \leq 15.664$	$Q = 4$	$Q = 5$
$|Y_i| < 3.03$	$Q = 2$	$Q = 1$

Note that in the computation of the numerator, we gain nothing by specifying Q; in that of the denominator we may gain some significance; in addition, if we specified Q for the denominator, then the "normal" Q for the Quotient would be correct. One additional point is that it doesn't really pay to specify Q for the first term of the denominator, since this is changed immediately by subtraction.

REGION	STEP	OP	CLUE	FACTOR	S_1	S_2	Q
1	1	SET			i	1	
1	2			Z	i		
1	3	x		C			
1	4	-		E			4
1	5			Z	i		
1	6	x		A			
1	7	+		B			
1	8	x		Z	i		
1	9	+		D			
1	10	/	R	4			
1	11	EQ		Y	i		2
1	12	TEST		2	i	10	
1	13	HALT					

<u>Numbers and the coding sheet</u>:

A. In our preceding example, A, B, C, D, and E were not really Factors in the sense of being variables. They were merely numbers which were needed to carry out the computation. To make life easier, we shall provide more convenient ways for using numbers.

1)   Any integer, n, where $-999 \leq n \leq 999$ may be written in the space on the coding sheet called "Factor" (the minus sign is written in the "Clue" space). In order to differentiate between integers and the normal factors we have been discussing (such as Z), we shall require the latter to contain at least one alphabetic character.

2)   Any number, Z, where $|Z| < 34,359,738,368$ may be written in the space provided on the coding sheet and, provided the Factor space is blank, will be used as the Factor.

B. Two additional conveniences will be added:

1)   The negative of the result of a previous step may be obtained by putting an N in the Clue column. If R or N is used, Q must be specified on the steps to which R or N refers.

2)   The negative of any other factor may be obtained by placing a minus sign in the clue column.

Explanation of the coding sheet

1)   The numbers such as "9", "11" written under the title of each column indicate card columns (Input to PACT is from cards).

Cols. 9(1)11 - Any mixture of alphabetic or numeric.

Cols. 12(1)15 - All numeric.(Col. 15 is usually left blank to allow corrections to be inserted.)

Cols. 16(1)19 - (See operation list.) It is essential to start from the left; thus Equals (EQ) would be in Cols. 16 and 17.

Col. 20 - This is the Clue column R, N, or - .

Cols. 21(1)23 - Either a three digit integer or at least one alphabetic symbol. Important: the factor: --X (where - means blank) is completely different from -X- or X--. In other words, it is essential to be consistent as to where a symbol is placed, if that symbol is to have the same meaning in different steps.

Col. 24 - Used only when referring to a step which does not have Col. 15 blank.

Cols. 25(1)27 - Any combination of alphabetic and/or numeric. Same restriction as on Cols. 21(1)23.

Cols. 28(1)30 - Same as above.

Cols. 31(1)32 - 2 digit Q; negative Q is indicated by placing a minus sign over the left-most digit.

Col. 33 - Sign of number in Cols. 34(1)44.

Cols. 34(1)44 - 10 digit number with decimal point or 11 digit integer (see A2 above).

The sample coding sheet contains the example of page 12 using numbers in place of A, B, C, D, E.

We have given our machine one additional capability: if the start button is pressed after the machine has stopped as

the result of a HALT, we will go to the step noted as the factor; the factor on a HALT step may not be left blank.

PROBLEM REG.	STEP	OP.	FACTOR	S_1	S_2	$\pm Q$	$\pm$	NUMBER	NOTES
A.B.C001		SET	E	I	1				
	2	X		H				$+.56000000$	
	3	—	16		04				
	4	X	N	H					
	5	+	2						
	6	X	3	H					
	7	+	E					$+7.50000000$	
	8	X		H					
	9	/	R 4						
	10	EQ	Y	H		0.2			
	11	TEST	2	H	1.0				
	12	HALT	I						
	13								

At this point in the design of our machine, we have pro-
vided ourselves with a reasonable ability to do computations
but no way of getting numbers into or out of the machine. The
latter point will be discussed first (because it's easier!).

All output from our machine shall be on printed "lists"
with a maximum of six 11 digit numbers printed across the
page. We require an operation to "LIST" our Factors and a
means of identifying the Factors (ID). For example: to
print the 6X6 matrix

$A = (a_{ij})$, we could code as follows:

Reg	Step	OP	Clue	Factor	S_1	S_2	Q
20	1	Set			i	1	
20	2	LIST					
20	3	ID		A	i	1	
20	4	ID		A	i	2	
20	5	ID		A	i	3	
20	6	ID		A	i	4	
20	7	ID		A	i	5	
20	8	ID		A	i	6	
20	9	TEST			i	6	
20	10	HALT		1			

The resulting output will be:

$$a_{11} \quad a_{12} \quad \cdot \quad \cdot \quad \cdot \quad a_{16}$$
$$a_{21} \quad \cdot \quad \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$a_{61} \quad \cdot \quad \quad \cdot \quad \cdot \quad \cdot \quad a_{66}$$

Note that in step 9 we did not specify a factor on the
TEST. Where a factor is not specified in a TEST operation,
the transfer will be to the step immediately following the
Set for the same subscript.

The same example could have been coded as follows:

Reg	Step	OP	Clue	Factor	S_1	S_2	Q
21	1	SET			i	1	
21	2	SET			J	1	
21	3	LIST					
21	4	ID		A	i	J	
21	5	TEST			J	6	
21	6	TEST			i	6	
21	7	HALT		1			

The resulting output would have been

$$a_{11}$$
$$a_{12}$$
$$\cdot$$
$$a_{16}$$
$$a_{21}$$
$$a_{22}$$
$$\cdot$$
$$\cdot$$
$$\cdot$$
$$a_{66}$$

Note that step 5 transfers to step 3; step 6 transfers to step 2.

Input:

Warning: This section is probably the most difficult one in the entire Primer.

We shall begin by defining two terms:

1) Array - any factor which has subscripts (one or two).

2) Scalar - any factor which has no subscripts. The variable definition sheet

Any scalar for which we desire to enter (input) initial values must be defined on the variable definition sheet.

All arrays must be defined on the variable defini-
tion sheet.

Examples: In example (1), all the quantities involved (A, B,
C, D, E, Y, Z) are scalars; all but Y are ones for which
we want to input values.   All but Y must be defined on
the variable definition sheet.

In example (2), A, B, C, D, E are scalars; Y_1
and Z_1 are arrays.   All must be defined on the variable
definition sheet.

In the example on the sample coding sheet, Y_1 and
Z_1 are arrays and must be defined.

Layout of the Variable Definition Sheet

Card Cols. 9 - 11	12 - 14	15 - 17	18 - 20	21 - 23	24 - 26
FACTOR	S_1	S_2	Q	D_1	D_2
(3)	(3)	(3)	(3)	(3)	(3)

The numbers in parentheses indicate the number of digits.
Insofar as the PRIMER is concerned, S_1 and S_2 will
always be left blank.   Q must always be specified on
the Variable Definition Sheet ($\pm$XX). D_1 contains the
maximum value taken on by S_1 for the given Factor,
similarly D_2 and S_2.

The Variable Definition Sheet for the example of page
16 would be

FACTOR	S_1	S_2	Q	D_1	D_2
Z			+02	010	
Y			+02	010	

Assume for the moment that compiling has been done.   The
first page of output contains the "Variable List".   For this

example it might well be in part as follows:

TAG	REL 1ST	LOC. LAST	VARIABLE FACT.	S_1	S_2	Q	XXXXX
V	2	20	Z		2^2	2	
V	22	40	Y		2	2	

The only things of interest to us are: 1ST, LAST, FACT, S_1, S_2.

We shall define "LOC" only as something that we have to know and then write the equations:

(EQ 1)   LOC (of X_{ij}) = 1ST + (i-1) S_1 + (j-1) S_2

(EQ 2)   LOC (of W_i) = 1ST + (i-1) 2

We shall also define "P" as the number of digits to the left of the decimal point.

We are now ready to fill out the input sheet. Assume for the preceding example that

$$Z_1 = .6 \qquad Z_6 = 1.5$$
$$Z_2 = .7 \qquad Z_7 = 2.0$$
$$Z_3 = .8 \qquad Z_8 = 2.5$$
$$Z_4 = .9 \qquad Z_9 = 3.0$$
$$Z_5 = 1.0 \qquad Z_{10} = 3.5$$

Then the P_i are, in order: 0, 0, 0, 0, 1, 1, 1, 1, 1, 1

The LOC are, in order (according to EQ 2): 2, 4, 6, 8, 10, 12, 14, 16, 18, 20.

The following restrictions apply to the input sheet $-5 \leq p \leq 15$; $-18 \leq Q \leq 52$. All minus signs (Factor, P, Q) are indicated by writing a minus over the left-most digit.

A filled-out input sheet is on the next page. We now need an operation to "READ" input cards.

READ: Input the Factors from cards. When a "12" punch in card col. 80 is encountered, go on to the next step.

PACT I INPUT SHEET

Loc	Factor	P	Q	Loc	Factor	P	Q	Loc	Factor	P	Q
000	26.000	00000	00	002	0047.0000	00000	00	000	0006.8000	00000	00 02
0010	0.000	00000	01	0020	012.5000	00000	01	0020	0019.2000	00000	01 02
0018	30.000	00000 01 02		002 0020	35.000	00000 01 02		0020 0016.25000	00000	01	02

Loc	Factor	P	Q
000	0003.9000	00000 00	02

Partial Summary and some details

The steps involved in doing a problem on our machine (assuming the analysis and flow diagram have been completed) are as follows:

1. Do the coding and have the cards key-punched.

    a. The last code card must have a "12" punch in card column 80.

2. Make up the variable definition sheet and have the cards key-punched.

    a. The last variable definition card must have a "12" punch in card column 80.

    b. There must be at least one variable definition card (this card may contain only the "12" punch in card column 80).

3. Arrange a deck as follows:

    a. PACT load card.

    b. Code cards. [The first code card input will contain the first instruction which is eventually executed.]

    c. Variable Definition cards.

    d. Two blank cards.

4. Put the PACT Compiler Tape on unit 403 of the IBM Type 701.

5. Put the PACT Compiler Board in the printer.

6. Have tape units 400(1)403 ready; have printer ready, have punch ready.

7. Load the deck of (3) at zero.

8. The cards generated in the punch are the running deck; the order of this deck is critical.

9. The first page(s) of printed matter is the "Variable List". Subsequent pages are the compiled code.

10. By use of the Variable List and EQ 1 and 2, prepare the input sheet and have the cards key-punched.

    a. The last input card of a block must have a "12" punch in card column 80. (See the READ operation.)

11. Make up a deck as follows:

    a. The running deck of (8)

    b. The input cards of (10)

12. Put the PACT running board in the printer and have the printer ready.

13. Load the deck of (11) at zero.

14. Hope that the answers come out correctly.

Thus far, in designing our machine, we have been concerned with giving it the ability to do operations which are either necessary or extremely useful. However, there are many things we may add to the machine for nothing more than convenience or general usefulness. This we have done to a certain extent. Let us, therefore, enumerate and describe the "complete" set of PACT operations. (The word "complete" is true only insofar as this Primer is concerned.)

OPERATION	SYMBOL	TYPE	DESCRIPTION
Take	(Blank)	1	Take the factor as the first operand of a sequence of operations.
Add	+		Add the factor to the result of the previous step.
Subtract	-		Subtract the factor from the result of the previous step.
Multiply	x		Multiply the result of the previous step by the factor.
Divide	/		Divide the result of the previous step by the factor and get the quotient.
Equals	EQ	2	Put the result of the previous step into the factor.
Absolute	ABS	1	Take the absolute value of the factor as the first operand in a sequence of operations.
Add Absolute	+ ABS		Add the absolute value of the factor to the result of the previous step.
Subtract Absolute	- ABS		Subtract the absolute value of the factor from the result of the previous step.
Transfer	T	1	Go to the step noted as the factor.

OPERATION	SYMBOL	TYPE	DESCRIPTION
Transfer on Zero	TZ		Go to the step noted as the factor if the result of the previous step is zero. Otherwise, proceed in sequence.
Transfer on Plus	TP		Go to the step noted as the factor if the result of the previous step is greater than zero. Otherwise, proceed in sequence.
Transfer on Negative	TN		Go to the step noted as the factor if the result of the previous step is less than zero. Otherwise, proceed in sequence.
Transfer on Overflow	TF		Test the overflow switch: if it is on, shut it off and then go to the step noted as the factor; if it is off, proceed in sequence. [The overflow switch is turned on when a Q is insufficient to accommodate the result of its associated step.]
Halt	HALT	1	Halt; if the start button is pushed go to the step noted as the factor.
Do Region and Return	DO	1	Do the region noted as the factor and return to the step following the DO.
Clear	CL	1	Clear the factor to zero.
Set	SET	1	Set the subscript in S_1 to the value noted or represented in S_2. [Note: a subscript may not be set to zero.]
Test	TEST	1	Increase the subscript noted in S_1 by one. If this results in a value which is greater than the value noted or represented in S_2, proceed in sequence. Otherwise, go to the step noted as the factor or, if the factor is blank, to the step immediately following the SET for the same subscript.
Sine	SIN		Compute the sine of the factor. The angle must be in radians. Q of result is 1.
Cosine	COS		Compute the cosine of the factor. The angle must be in radians. Q of result is 1.

OPERATION	SYMBOL	TYPE	DESCRIPTION
Arctangent	ARCT		Compute the Arctangent of the factor. The result is in radians. Q of result is 1.
Square Root	SQRT	2	Compute the square root of the factor.
Logarithm	LOG		Compute the natural logarithm of the factor. Q of result is 6.
Exponential	EXP	2	Compute e^x where x is the factor.
Read	READ		Input factors from cards until a "12" punch in card column 80 is encountered. Then proceed to the next step.
Print	LIST ID		Print the factor(s) specified by the ID which occur immediately after the LIST operation.

Type 1 operations are the ones to which a transfer may be made. (See the Note on the bottom of page 4.)

Type 2 operations are ones for which Q must be specified. The HALT operation and the transfer operations must have Q blank.

Summary

The essential aim of this Primer has been not only to explain PACT to the inexperienced (or non-experienced) coder, but also to make it a system which he could use with a minimum of extra explanation. To accomplish this end, we have, at various times: omitted information, oversimplified, uttered half-truths, and on occasion, have just plain lied. The experienced programmer has no doubt recognized many of these instances and we apologize if he has been confused by them.

Appendix A

Some pointers and additional restrictions

1. Every SET operation must be followed by a TEST operation within the same region.

2. If a DO operation is performed between a SET and TEST, the SET and TEST will have no effect in the region specified by the DO, and vice-versa.

3. A subscript need not be SET to a number; it may be SET to a variable. If so, the variable must have $Q = 17$.

4. When doing a LIST operation, the last ID cannot be followed by a HALT. (If this is desired, put in a "phony" TAKE before the HALT.)

5. The HALT operation cannot have a blank factor. [This is because the Halt acts as a "stop" then "transfer" when the start button is depressed.]