7-19-55

PACT PRIMER

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:

EG-1 Compute $y = \frac{az^2 + bz + d}{cz - e}$ 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 and the final answer. We shall call these registers "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 the "CODE" to carry out the computation we wish to do:

1A

Step #	Operation	Factor
1	TAKE	Z
2	MULTIPLY (by)	C
3	SUBTRACT	e
4	EQUALS	DEMONINATOR
5	TAKE	Z
6	MULTIPLY (by)	a
7	ADD	b
8	MULTIPLY (by)	Z
9	ADD	d
10	DIVIDE (by)	DEMONINATOR
11	EQUALS	y
12	HALT	

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 (R n) 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)

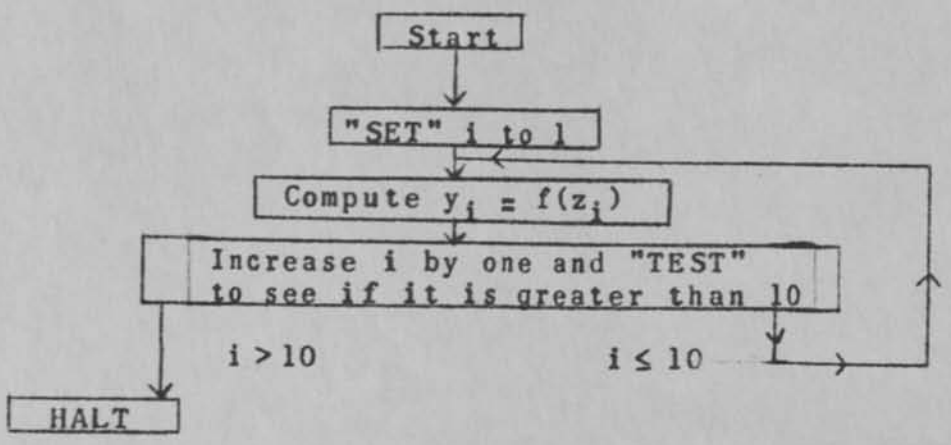
STEP	OPERATION	CLUE	FACTOR
1			z
2	x		c
3	-		e
4			z
5	x		a
6	+		b
7	x		z
8	+		d
9	/	R	3
10	EQ		y
11	HALT		

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

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 $\overset{a}{\wedge}$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 subscript between the Set and Test operations (steps 2, 5, 8, 11). If the new i is greater than 10, 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) .

Note: Our machine shall have the following restriction: whenever 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.

As a further extension of the problem we have been doing, suppose we wanted to do the following:

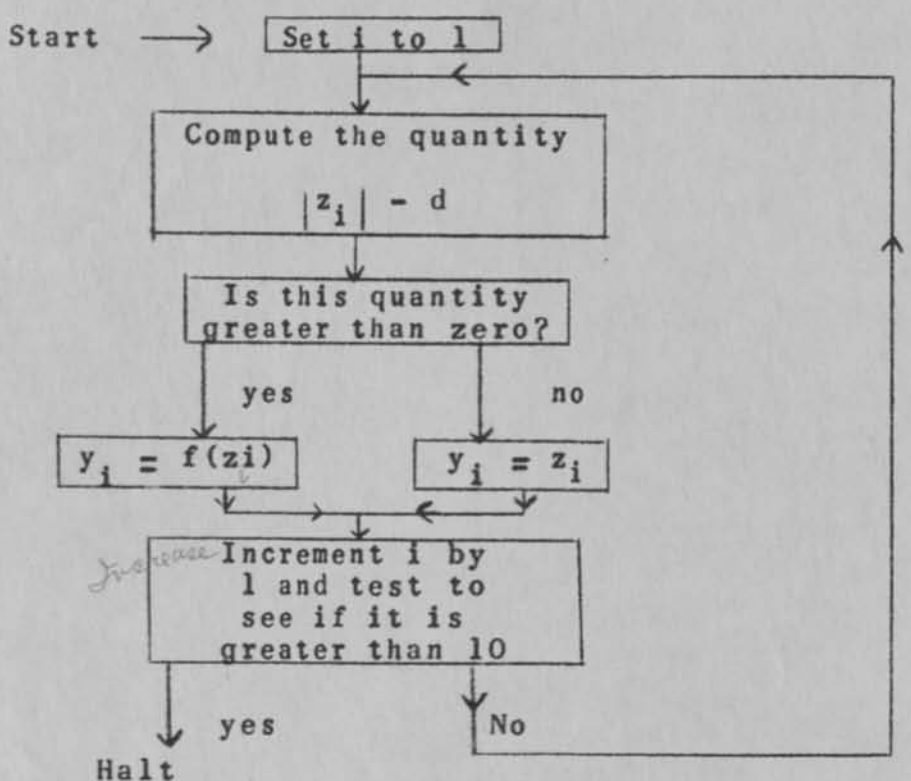
EG-3 Compute: $$y_i = \frac{a z_i^2 + b z_i + d}{c z_i - e} = f(z_i)$$

$$\underline{\text{if}} \quad |z_i| > d \qquad\qquad i = 1(1)10$$

$$y_i = z_i$$

$$\underline{\text{if}} \quad |z_i| \leq d$$

Again, we shall begin by making a Flow Diagram



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, proceeding 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.

EG-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		c		
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; 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 can not be overlooked when one is doing computational work. Therefore, we shall be required to specify for each of our arithmetic steps, the number of 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 $(2^3 < 12.345 < 2^4)$

•01234 has Q = -6 $(2^{-7} < .01234 < 2^{-6})$

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 exception 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 that 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
$14.04 \leq \lvert cz_i-e \rvert \leq 15.664$	Q = 4	Q = 5
$\lvert y_i \rvert < 3.03$	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 specify Q for the denominator, the "normal" Q for the Quotient will 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 the 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					

Numbers and the coding sheet:

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 normal factors, 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.

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 in the REG space 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.

Col. 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.

JOB NO. _____ PROG. NO. _____ CARD COLOR _____

ANALYST _____ CHECKED BY _____ DATE _____ PAGE ___ OF ___

PROBLEM _____

REG. (9–11)	STEP (12–15)	OP. (16–19)	FACTOR (20–24)	S_1 (25–27)	S_2 (28–30)	±Q (31–32)	± (33)	NUMBER (34–44)	NOTES
A.B.C	0.0.1	S.E.T.		I	1				
	2		Z	I					
	3	X						+.5600000000	
	4	−	16			04			
	5		Z	I					
	6	X	2						
	7	+	3						
	8	X	Z	I					
	9	+						+7.5000000000	
	10	/	R 4						
	11	E.Q.	Y	I		02			
	12	T.E.S.T.	2	I	10				
	13	H.A.L.T.	1						

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

All output from our machine shall be on printed "lists" with a maximum of six 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

$A = (a_{ij})$, where each a_{ij} has a Q = 3 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	3
20	4	ID		a	i	2	3
20	5	ID		a	i	3	3
20	6	ID		a	i	4	3
20	7	ID		a	i	5	3
20	8	ID		a	i	6	3
20	9	TEST			i	6	
20	10	HALT					

The resulting output will be:

$$\begin{matrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & a_{16} \\ a_{21} & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{61} & \cdot & \cdot & \cdot & \cdot & a_{66} \end{matrix}$$

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.

~~An additional restriction: every ID shall have a Q specified~~.

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	3
21	5	TEST			j	6	
21	6	TEST			i	6	
21	7	HALT					

The resulting output would have been

a_{11}
a_{12}
.
a_{16}
a_{21}
a_{22}
.
.
.
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 definition 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_i and z_i are arrays. All must be defined on the variable definition sheet.

In the example on the sample coding sheet, y_i and z_i 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 (±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

15 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 as follows:

TAG	REL 1st	LOC. LAST	VARIABLE FACT.	S_1	S_2	Q	XXXXX
V	2	20	Z	2^1	2^2	2	
V	22	40	Y	2	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 <u>decimal</u> point.

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

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

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	LOC	FACTOR	P	Q
0 0 0 0 0 0 0 0 1 2 3 4 5 6 7 8	0 1 1 1 9 0 1 2	1 1 1 1 1 1 1 1 2 2 2 2 3 4 5 6 7 8 9 0 1 2 2	2 2 3 4	2 2 5 6	2 2 2 3 7 8 9 0	3 3 3 3 3 3 3 3 3 4 1 2 3 4 5 6 7 8 9 0	4 4 1 2	4 4 3 4	4 4 4 4 5 6 7 8	4 5 5 5 5 5 5 5 9 0 1 3 4 5 6 7 8	5 6 9 0	6 6 0 1 2	6 6 6 6 3 4 5 6	6 6 6 7 7 7 7 7 7 7 8 9 0 1 2 3 4 5 6	7 7 7 8	7 8 9 0
	0002	60000 00000	00	02	0004	70000 00000	00	02	0006	80000 00000	00	02	0008	90000 00000	00	02
	0010	10000 00000	01	02	0012	15000 00000	01	02	0014	20000 00000	01	02	0016	25000 00000	01	02
	0018	30000 00000	01	02	0020	35000 00000	01	02								

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0

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
   c. Variable Definition cards.
   b. Code cards
   c. 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, p 19 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	~~1~~	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	~~1~~	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 ~~4~~ 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 ~~the~~ 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.

OPERATION	SYMBOL	TYPE	DESCRIPTION
Sine	SIN	~~2~~	Compute the sine of the factor. The angle must be in radians. $Q_e = 1$
Cosine	COS	~~2~~	Compute the cosine of the factor. The angle must be in radians. $Q_e = 1$
Arctangent	ARCT	~~2~~	Compute the Arctangent of the factor. The result is in radians. $Q_e = 1$
Square Root	SQRT	2	Compute the square root of the factor.
Logarithm	LOG	~~2~~	Compute the natural logarithm of the factor. $Q_e = 6$
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	~~2~~	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.

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.
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.]
6. IF THE RESULT OF A STEP IS CALLED FOR BY ANOTHER STEP WITHIN A REGION, THE FIRST STEP MUST HAVE "Q" SPECIFIED