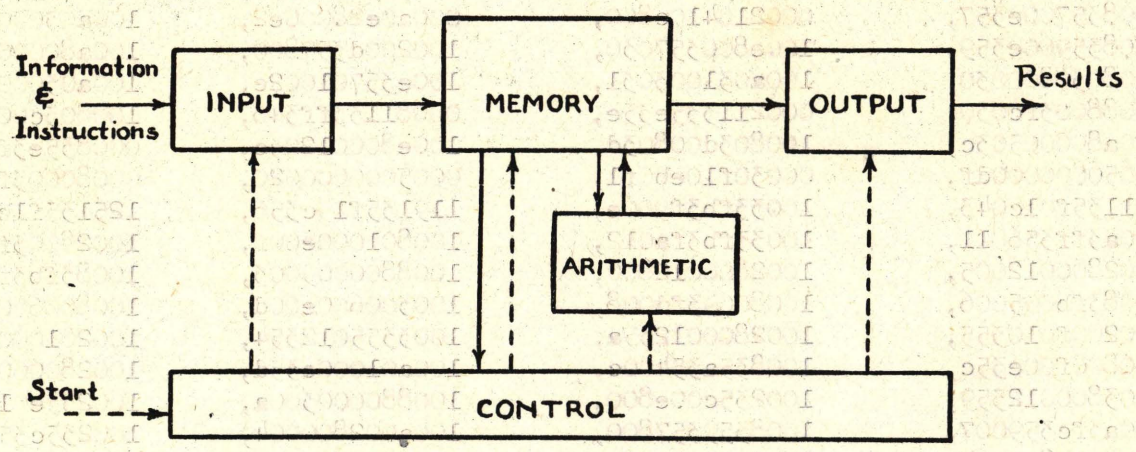

Figure 1--Basic Components of an Automatic Computer

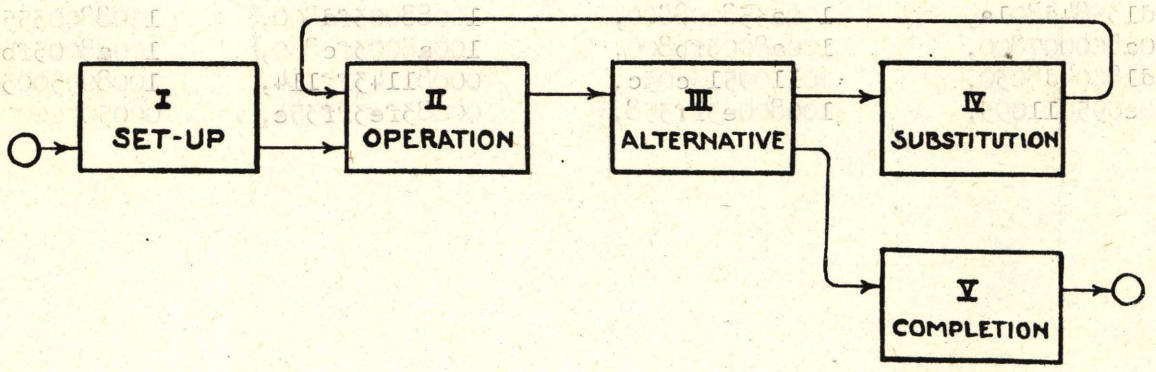

Figure 2--Flow Diagram

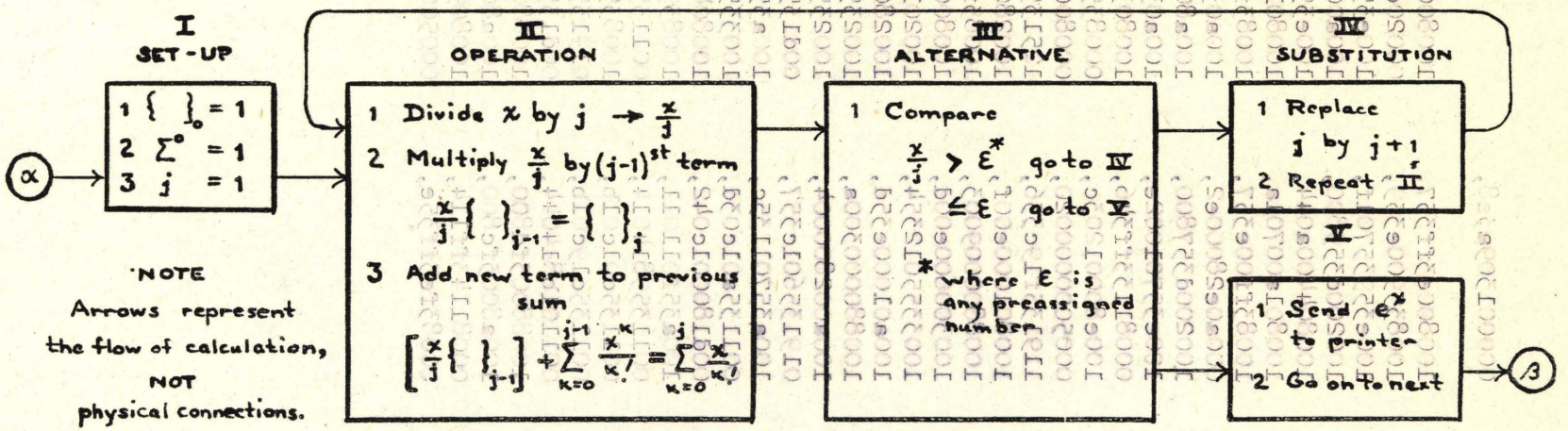

Figure 3 — Flow Diagram for Generating e^x

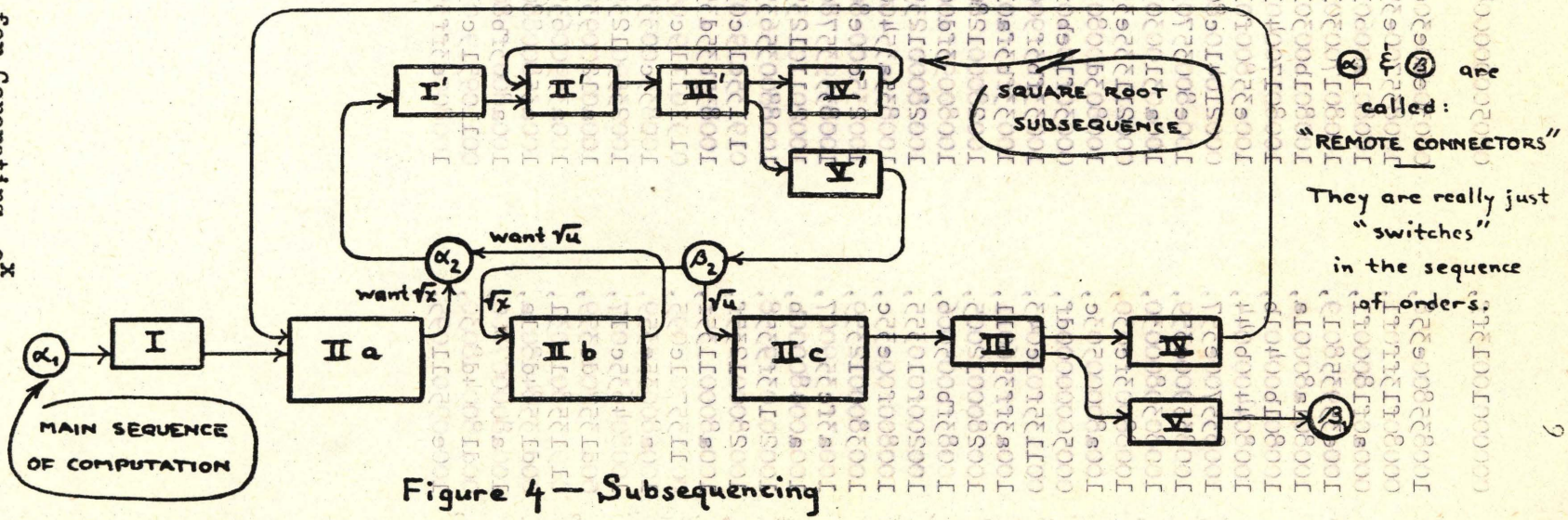

Figure 4 — Subsequencing

Figure 3—Flow Diagram for Generating e^x
Figure 4—Subsequencing

b. Digit Assignment

A "word" in the WISC is either an order or a number, and consists of fifty-five binary digit spaces. Of these, fifty spaces have bits assigned to them, and the remaining five are left blank to provide switching time. Figure 5 shows the current assignment of digits in both numbers and orders.

The direction of writing words, as illustrated in Figure 5, has been chosen so that the most significant bit of a number is the bit to the extreme left, just to the right of the binary point. Since in this machine numbers must pass through arithmetical operations starting with the least significant bit, this places bit 1 at the extreme right, and it is this bit which is always the first to appear.

Note that the eight bits 41 to 48 are not used in any order except an Extract order. In this order the B and C addresses are the same, so that the 12 B bits together with the 6 bits 41 to 46 are used for three 6-bit groups of additional information:

x Taking from operand A starting with bit x , and

y writing into operand C starting with bit position y,

z extract z bits.

c. Coding Representation

Now we can itemize the operations mentioned in A2 above, in terms of specific orders. Since it turns out to require twelve storage locations for the instructions, and an additional seven for data, constants, and intermediate storage, we will reserve the first nineteen storage locations for this subsequence. The last seven of these will be assigned thus:

13: 1

14: ϵ

15: x

16: $\sum^{j}$ reserved for the summation of j terms, which will eventually be e^{x}

In the case of an E order, an additional section must be added to give the extract specifications:

N [A , B , C , TYPE (x , y , z)]

For instance,

7 [16 , 14 , 14 , E (41 , 9 , 41)]

is the coding abbreviation for order No. 7.

There follows the WISCoding notation for the entire procedure above, labelled Figure 6.

d. Short Memory Coding

One difficulty has been ignored in the above coding presentation, and that is the availability of information when wanted. On five different occasions, one of the two operands called for is the result of the immediately preceding operation. Now a study of Figure 45, Appendix 1, shows that in a normal sequence of orders, a result is not delivered back to memory until the third cycle after its processing has been initiated. However, a way around this obstacle was pointed out by the author in the early days of logical planning for the WISC.*

As explained in Appendix 1, a short memory is used to act as a buffer between standard memory and arithmetic, both for input of operands and delivery of results. In the normal course of orders a result has been arrived at by at least the last MA cycle of arithmetic operation, and during the first MM cycle of the following, or delivery, major cycle it is transferred to short memory to await storage in standard memory during the proper MM cycle. (In the event its storage location is reached during the first MM cycle, the result passes directly from arithmetic to standard memory during the first MM cycle of the delivery major cycle.) With this procedure, if the result is wanted for the next operation, the coder would have to instruct the machine to kill time for two complete cycles before starting operations on the next order. Similarly, if the result is wanted as part of the second following order, one cycle must be waited before it is available (see Figure 45).

Now the second of these cases turns out to be taken care of automatically, that is, if it is desired to use a result for the second following operation, the coder need make no special arrangements. But for the first case, that of immediately successive operation, a special coding procedure must be used. If the coder desires to use a result for the immediately following operation, he instructs the machine to take operand A, or operand B, or both, from short memory where they are awaiting delivery, and send them directly back to the arithmetic section for the next operation. This by the way does not interfere

---

* Reference 1, page 29.

with the results being written at the same time into standard memory in the usual manner.

This instruction is given by means of two binary digits in the 31-40 group reserved for this purpose, which have the following significance:

n = 0	normal operation
1	take operand A from short memory
2	take operand B from short memory
3	take both from short memory (eg, in squaring)

The coding for such an operation looks like this:

N[A , B , C (n) TYPE]

Once more recoding our exponential subsequence to include this (the last) modification, we get the set of orders shown in Figure 7.

2. Single Problems

Now that the symbolism of WISCoding has been established, we will proceed to apply it to the preparation of several progressively more difficult problems for presentation to the machine.

a. Open Loop

At the very simplest level is the problem which involves purely a series of arithmetical steps, with no repetitions. As an example see Figure 8 which lists the steps necessary to calculate

$$v = \frac{ax^2 + bx + c}{dx + e}$$

given x and the five constants. Note that two steps are combined into one by first forming $(ax + b)$ and then multiplying by x. It will be seen that this involves only an operation box--no alternative, substitution, etc.

b. Single Closed Loop

(1) Change of Variable

This can be turned into a repetition problem if we now ask for the calculation of v from the same expression for a whole series of values of x. It is found convenient on such problems involving more than one block of orders to lay out the coding first in a Flow Diagram,* assigning temporary numbers to such storage

---

* The practice of making flow diagrams before final coding is to be found at Princeton, Aberdeen, and several other places (see references 1 and 2), but I have modified the procedure considerably in the interests of efficienty and convenience for application to the WISC.

Storage Number	Order Code or Contents	Meaning
1	[13, 16, 16 (o) E (1, 50, 1)]	$1 \rightarrow \sum^j$
2	[13, 17, 17 (o) E (1, 50, 1)]	$1 \rightarrow \{\ \}_j$
3	[13, 18, 18 (o) E (1, 50, 1)]	$1 \rightarrow j$
4	[15, 18, 19 (2) D]	$\frac{x}{j}$
5	[19, 17, 17 (1) M]	$\frac{x}{j}\{\ \}_{j-1} \rightarrow \{\ \}_j$
6	[17, 16, 16 (1) A]	$\{\ \}_j + \sum^{j-1} \rightarrow \sum^j$
7	[16, 14, 14 (1) E (1, 9, 41)]	exp of $\sum^j$ with ε
8	[17, 14, 11 (2) CA]	$\{\ \}_j < \varepsilon$?
9	[13, 18, 18 (o) A]	$1 + j \rightarrow j$
10	[—, —, 4 (—) T]	
11	[17, =, = (o) E (1, 50, 1)]	store e^x
12	[—, —, 20 (—) T]	
13	j	
14	ε	
15	x	
16	$\sum^j$	
17	$\{\ \}_j$	
18	j	
19	ODSTO	
20	Next part of the calculation	

Figure 7--WISCoding for e^x with Short Memory

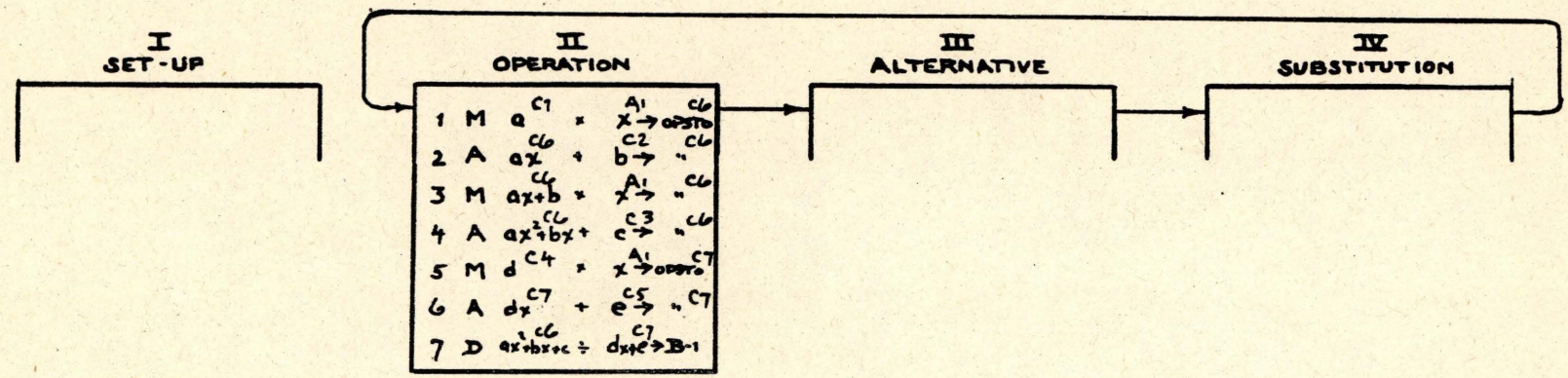

Figure 8

STORAGE

	A-1	x
	B-1	v
	C-1	a
	2	b
	3	c
	4	d
	5	e
1016	6	OPSTO
1017	7	OPSTO

FLOW DIAGRAM
for
$v = \frac{ax^2+bx+c}{dx+e}$

ORDERS 7
STORAGE 5
12

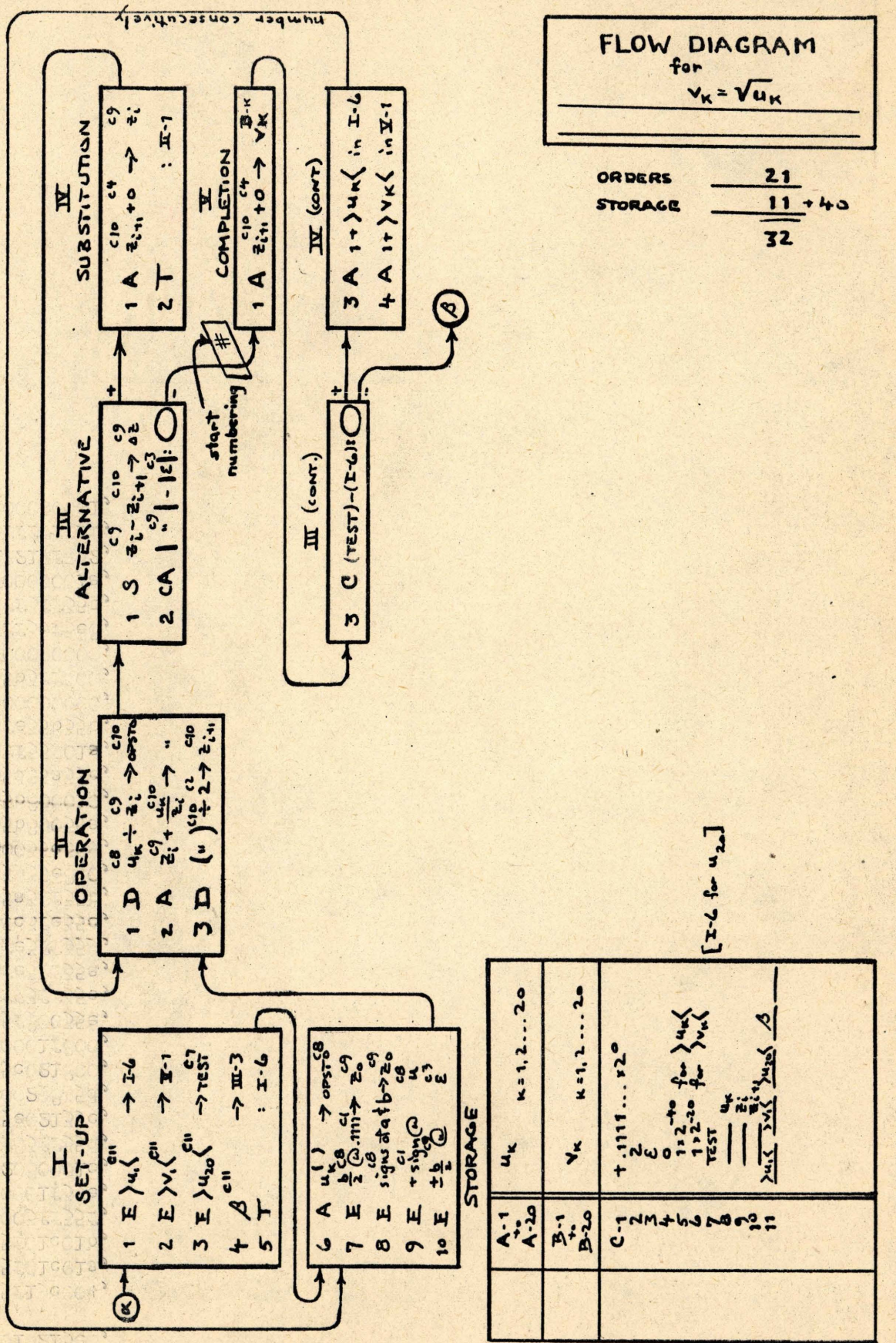
FLOW DIAGRAM
for
$v_K = \sqrt{u_K}$
ORDERS 21
STORAGE 11 + 40
32
number consecutively
I SET-UP
II OPERATION
III ALTERNATIVE
III (cont.)
IV SUBSTITUTION
IV (cont.)
V COMPLETION
start numbering
STORAGE
[I-6 for u_{20}]

Figure 17

The problem is that of determining the probability, after $j+1$ drawings from a population of n members among which there are n different types, of not getting two alike. The probability is given by

$$P_k = \prod_{j=1}^{k-1} \frac{n-j}{n}$$

where the first drawing ($j = 0$) of course gives $P_0 = 1$, and the calculation starts with the second drawing ($j = 1$). Consider the three progressively more involved tasks:

(a) Print a table of m values of P_k for a given n.

(b) Find the value of $j = k$ for which $P_k = \varepsilon$ for given n.

(c) Find a value of $j = k$ for which $P_k = \varepsilon$ for a series of values of n.

Figures 19 to 24 show the coding for all three parts to this problem, and Figure 25 tabulates the comparison of time and effort needed to solve them on the two machines.

3. Compound Problems

In the sections above have been illustrated some techniques which can be applied to minor problems, problems which are complete in themselves, or which are encountered as parts of major problems. But what about the larger problems? How do we tackle those? There are two basic attitudes we can take.

a. Unique Coding

We can get as much experience as possible coding problems of all types: single, multiple, compound. We can then use the knowledge thereby acquired to tackle each complex problem as a unique case, utilizing techniques and short cuts where they apply.

b. Prefabrication of Subroutines

Or we can recognize that most of the big problems are made up of combinations of the smaller ones, and plan the smaller ones to be incorporated directly as building blocks of the larger. The square root sequence coded above is a good example; once its coding has been worked out, we can file it for further use. Then when a problem calls for a square root at some point, the subsequence to calculate it can be dug up, modified at beginning and end points, and fitted into the whole.

It is this latter approach that we plan to use with the WISC, where the subsequences are not only listed and filed away, but can actually be coded onto tape and stored as a lending library. These libraries will be discussed in greater detail in D below.

WISCoding for $P_k = \prod_{j=1}^{k-1} \frac{n-j}{n}$

Part (a)

Flow No.	Storage No.	Order Code or Contents	Meaning
II-1	1	[9, 1001, 9(0)S (, ,)]	[n-(j-1)] -1 → n-j
2	2	[9, 10, 1016(1)D (, ,)]	(n-j) ÷ n
3	3	[(11), 1016, (12)(2)M (, ,)]	$\frac{n-j}{n} P_{j-1} \rightarrow P_j$
III-1	4	[9, 7, 3(0)C (, ,)]	(n-j) - (n-k)
III-1	5	[8, 3, 3(0)A (, ,)]	1+)P_{j-1} ≠ 1+)P_j (in (3)
2	6	[—, —, 1(-)T (, ,)]	
C-1	7	n-k	
2	8	$1 \times 2^{-20} + 1 \times 2^{-40}$	
3	9	n n-j	
A-1	10	n	
B-1	11	1	
B-2	12	P_j	
+0	+0		
B-k	10+k		

Figure 20

WISCoding for $P_K = \prod_{j=1}^{K-1} \frac{n-j}{n}$

Part (b)

Flow No.	Storage No.	Order Code or Contents	Meaning
II-1	1	[10, 1001, 10(0) S (, ,)]	[n-(j-1)] - 1 → n-j
2	2	[10, 8, 1016(1) D (, ,)]	n-j ÷ n
3	3	[1016, 11, 11(1) M (, ,)]	$\frac{n-j}{n}$ $P_{j-1} \rightarrow P_j$
III-1	4	[11, 9, 6(1) C (, ,)]	$P_j < \varepsilon$?
IV-1	5	[—, —, 1(-) T (, ,)]	
V-1	6	[8, 10, 1016(0) S (, ,)]	n-(n-j) = j
2	7	[1001, 1016, 12(2) A (, ,)]	1+j → K
A-1	8	n	
C-1	9	ε	
3	10	$\frac{n}{n}$ n-j	
5	11	$\frac{1}{K}$ P_{j-1}	
B-1	12		

Figure 22