PLANNING AND CODING OF PROBLEMS
FOR AN
ELECTRONIC COMPUTING INSTRUMENT

By

Herman H. Goldstine   John von Neumann

Report on the Mathematical and Logical Aspects
of an Electronic Computing
Instrument

Part II, Volume III

The Institute for Advanced Study
Princeton, New Jersey
1948

PREFACE

This report was prepared in accordance with the terms of Contract No. W-36-034-ORD-7481 between the Research and Development Service, U. S. Army Ordnance Department, and The Institute for Advanced Study. It is another in a series of reports entitled, "Planning and Coding of Problems for an Electronic Computing Instrument", and it constitutes Volume III of Part II of that sequence. It is expected that Volume IV of this series will appear in the near future.

Herman H. Goldstine
John von Neumann

The Institute for Advanced Study
16 August 1948

TABLE OF CONTENTS

12.0 COMBINING ROUTINES

12.1 Each one of the problems that we have coded in the past Chapters 8-11 had the following properties: The problem was complete in the sense, that it led from certain unambiguously stated assumptions to a clearly defined result. It was incomplete, however, in another sense: It was certain in some cases and very likely in others, that the problem in question would in actual practice not occur by itself, as an isolated entity, but rather as one of the constituents of a larger and more complex problem. It is, of course, justified and even necessary from a didactical point of view, to treat such partial problems, problem fragments -- especially in the earlier stages of the instruction in the use of a code, or of coding per se. As the discussion advances, however, it becomes increasingly desirable to turn one's attention more and more from the fragments, the constituent parts, to the whole, In our present discussion, in particular, we have now reached a point where this change of emphasis is indicated, and we proceed therefore accordingly.

There are, in principle, two ways to effect this shift of emphasis from the parts to the whole.

The first way is to utilize the experience gained in the coding of simpler (partial) problems when one is coding more complicated (more complete) problems, but nevertheless to code all the parts of the complicated problem explicitly, even if equivalent simple problems have been coded before.

The second way is to code simple (partial) problems first, irrespective of the contexts (more complete problems) in which they may occur subsequently, and then to insert these coded sequences as wholes, when a complicated problem occurs of which they are parts.

We should illustrate both procedures with examples: This is not easy for the first one, because its use is so frequent that it is difficult to circumscribe its occurrences with any precision. Thus, if we had coded the calculation of the general third order polynomial, then any subsequent calculation involving (as a part) a third order polynomial, would offer such an example. Also, in view of Problem 1, any calculation involving a quotient of a second order and a first order polynomial would be an example. Problem 3, where the whole of Problem 1 is recoded as a part of the new coded sequence (but not Problem 2, where this is not done) is a specific instance.

Examples of the second procedure are more clearly identifiable. Problem 12 was used in this sense as a part of Problem 13.a and of Problem 13.b, and also, after some modifications, as a part of Problem 13.c. Problem 14 was used as a part of Problem 15. In addition it is fairly clear that all of the Problems 4-11 and 13-15 must be intended as parts of more complicated problems, and that it would be very convenient not to have to recode any one of them when it is to be used as part of another problem, but to be able to use it more or less unchanged, as a single entity.

12.2 The last remark defines the objective of this chapter: We wish to develop here methods that will permit us to use the coded sequence of a problem, when that problem occurs as part of a more complicated one, as a single entity, as a whole, and avoid the need for recoding it each time when it occurs as a part in a new context, i.e. in a new problem.

The importance of being able to do this is very great. It is likely to have a decisive influence on the ease and the efficiency with which a computing automat of the type that we contemplate will be operable. This possibility should, more than anything else, remove a bottleneck at the preparing, setting up, and coding of problems, which might otherwise be quite dangerous.

This principle must, of course, be applied with certain common sense limitations: There will be "problems" whose coded sequences are so simple and so short, that it is easier to recode them each time when they occur as parts of another problem, than to substitute them as single entities -- i.e. where the work of recoding the whole sequence is not significantly more than the work necessitated by the preparations and adjustments that are required when the sequence is substituted as a single entity. (These preparations and adjustments constitute one of the main topics of this chapter, cf. 12.3-12.5.) Thus the examples of the first procedure discussed in 12.1 above are instances of problems that are "simple" and "short" in this sense.

For problems of medium or higher complexity, however, the principle applies. It is not easy to name a precise lower limit for the complexity, say in terms of the number of words that make up the coded sequence of the problem in question. Indeed, this lower limit cannot fail to depend on the precise characteristics of the computing device under consideration, and quite particularly on the properties of its input organ. Also, it can hardly be viewed as a quite precisely defined quantity under any conditions. As far as we can tell at this moment, it is probably of the order of 15-20 words for a device of the type that we are contemplating (cf. the fourth remark in 12.11).

These things being understood, we may state that the possibility of substituting the coded sequence of a simple (partial) problem as a single entity, a whole, into that one of a more complicated (more complete) problem, is of basic importance for the ease and efficiency of running an automatic, high speed computing establishment in the way that seems reasonable to us. We are therefore going to investigate the methods by which this can be done.

12.3 We call the coded sequence of a problem a *routine*, and one which is formed with the purpose of possible substitution into other routines, a *subroutine*. As mentioned above, we envisage that a properly organized automatic, high speed establishment will include an extensive collection of such subroutines, of lengths ranging from about 15-20 words upwards. I.e. a "library" of records in the form of the external memory medium, presumably magnetic wire or tape. The character of the problems which can thus be disposed of in advance by means of such subroutines will vary over a very wide spectrum -- indeed a much wider one than is now generally appreciated. Some instances of this will appear in the subsequent Chapters 13 and 14. The discussions in those chapters will, in particular, give a more specific idea of what the possibilities are and what aims possess, in our opinion, the proper proportions.

Let us now see what the requirements and the difficulties of a general technique are, if this technique is to be adequate to effect the substitution of subroutines into routines in the typical situations.

12.4 The discussion of the precise way in which subroutines can be used, i.e. substituted into other routines, centers around the changes which have to be applied to a subroutine when it is used as a substituent.

These changes can be classified as follows:

Some characteristics of the subroutine change from one substitution of the subroutine (into a certain routine) to another one (into another routine), but they remain fixed throughout all uses of the subroutine within the same substitution (i.e. in connection with one, fixed routine). These are the changes of the *first kind*. Other characteristics of the subroutines may even vary in the course of the successive uses of the subroutine within the same substitution. These are the changes of the *second kind*.

Thus the order position at which a subroutine begins is constant throughout one substitution (i.e. routine, or, equivalently, one larger problem of which the subroutine's problem is part), but it may have to vary from one such substitution or problem to another. The first assertion is obviously in accord with what will be considered normal usage, the second assertion, however, needs some further elaboration.

If a given subroutine could only be used with its beginning at one particular position in the memory, which must be chosen in advance of all its applications, then its usefulness would be seriously limited. In particular, the use of several subroutines within one routine would be subject to very severe limitations. Indeed, two subroutines could only be used together, if the preassigned regions that they occupy in the memory do not intersect. In any extensive "library" of subroutines it would be impossible to observe this for all combinations of subroutines simultaneously. On the other hand, it will be hard to predict with what other subroutines it may be desirable to combine a given subroutine in some future problem. Furthermore, it will probably be very important to develop an extensive "library" of subroutines, and to be able to use it with great freedom. All solutions of this dilemma that are based on fixed positioning of subroutines are likely to be clumsy and of insufficient flexibility.

Hence we should postulate the variability of the initial order position of a subroutine from one substitution to another. Consequently this is an example of a change of the first kind. This requires corresponding adjustments of all references made in orders of the subroutine to definite (order or storage) positions within the subroutine, as they occur in the final form (the final enumeration) of its coding. These adjustments are, therefore, changes of the first kind.

The parameters or free variables of the problem that is represented by the subroutine (cf. 7.5) will, on the other hand, usually change from one use of the subroutine (within the same substitution, i.e. the same main routine or problem) to another. The same is true for the order position in the main routine, from which the control has to continue after the completion (of each particular use) of the subroutine. Since the subroutine sends the control after its completion to e (this is the notation that we have used in all our codings up to now, and we propose to continue using it in all subsequent codings), this observation can also be put as follows: The actual value of e will usually change from one use of the subroutine to another.

These remarks imply, that the parameters of the subroutines problem, as well as the actual value of its e, will usually undergo changes of the second kind.

12.5 All the changes that the use of a subroutine in a given substitution requires can be effected by the routine into which it is being substituted, i.e. by including appropriate coded instructions into that routine. For changes of the second kind this is the only possible way. For changes of the first kind, however, it is not necessary to put this additional load on the main routine. In this case the changes can be effected as preparatory steps, before the main routine itself is set in motion. Such preparations might be effected outside the machine (possibly by manual procedures, and possibly by more or less automatic, special, subsidiary equipment). It seems, however, much preferable to let the machine itself do it by means of an extra routine, which we call the *preparatory routine*. We will speak accordingly of an *internal preparation* of subroutines, in contradistinction to the first mentioned outside process, the *external preparation* of subroutines. We have no doubt that the internal preparation is preferable to the external one in all but the very simplest cases.

Thus changes of the first kind are to be effected by preparatory routines, which will be discussed further below. Changes of the second kind, as we have pointed out already, have to be effected by the main routine itself (into which the subroutine is being substituted): Before each use that the routine makes of the subroutine, it must appropriately substitute the quantities that undergo changes of the second kind (the parameters of the subroutines problem and the actual value of its e, cf. the discussion in 12.4), and then send the control to the beginning of the subroutine (usually by an unconditional transfer order). It may happen that some of these quantities remain unchanged throughout a sequence of successive uses of the subroutine. In this case the corresponding substitutions need, of course, be effected once, jointly for the entire sequence. If this sequence includes all uses of the subroutine within the routine, then the substitutions in question need only be performed once in the entire routine, at any sufficiently early point in it. In this last case we are, of course, really dealing with changes of the first kind, and the quantities in question could be dealt with outside the main routine, by a preparatory routine. It is, however, sometimes preferable to view this case as an extreme, degenerate form of a change of the second kind, or at any rate to treat it in that way.

This discussion should, for the time being, suffice to clarify the principles of the classification of subroutine changes, and of the effect which they (specifically: the changes of the second kind) have on the arrangements in the main routine (into which the subroutine is being substituted). We now pass to the discussion of the preparatory routine, which effects the essential changes of the first kind: The adjustments that are required in the subroutine by the variability of its initial order position.

12.6 Assume that a given subroutine has been coded under the assumption that it will begin at the order position a. (I.e. at the left-hand order of the word a. To simplify matters, we disregard the possibility that it may begin at the right-hand order of the word a. In our past codings we had usually a = 0, excepting Problem 2 where a = 100, Problems 13,a-c where a = 52, and Problem 15 where a = 42.) The orders that are contained in this subroutine can now be classified as follows:

First: The order contains no reference to a memory position x. It is then one of the orders 10, 20, 21 of Table II.

Second: The order contains a reference to a memory position x, but the place of this x is irrelevantly occupied in the actual code of the subroutine. In this case the subroutine itself must substitute appropriately for x, before the control gets to the order in question. I.e. some earlier part of the subroutine must form the substitution value for x, and substitute it into the order.

Third: The order contains a reference to a memory position x, the place of this x is relevantly occupied in the actual code of the subroutine, and this actual value of x corresponds to a memory position not in the subroutine.

Fourth: Same as the third case, except that the actual value of x corresponds to a memory position in the subroutine.

Fifth: One of the preceding cases, but at some point the subroutine treats the order or its x as if it were irrelevantly occupied, i.e. it substitutes there something else. ----

Assume next, that the subroutine, although coded as if it began at a, is actually to be used beginning at $\bar{a}$. This necessitates certain changes, which are just the ones that the preparatory routine has to effect, in the sense of the concluding remark of 12.5. Our above classification of the orders of the subroutine permits us to give now an exact listing of these changes.

Orders of the first and of the third kind require clearly no change. The same is true of the orders of the second kind if they produce x's which correspond to memory positions not in the subroutine. And even if x's are produced which correspond to memory positions in the subroutine, no change is necessary if the following rule has been observed in coding the subroutine: a was used explicitly in forming the x that corresponds to positions in the subroutine, and it was stored not as the actual quantity a, but as a parameter of the subroutine's problem. If it is then understood, that this parameter should have the value $\bar{a}$, then it will be adequately treated as a parameter in the sense of 12.5. Indeed, it represents that degenerate form of a change of the second kind, which can also be viewed as a change of the first kind, as discussed in 12.5. Thus it might be treated by a special step in the preparatory routine, but we prefer to assume, in order to simplify the present discussion, that it is handled as a parameter (of the subroutine) by the main routine. In this way the orders of the second kind require no change either (by the preparatory routine).

Orders of the fourth kind clearly require increasing their x by $\bar{a} - a$.

Orders of the fifth kind behave like a combination of an order of one of the four first kinds with an order of the second kind. Since all of these orders are covered by the measures that emerged from our discussion of the four first kinds of orders, it ensues that the orders of the fifth kind are automatically covered, too, by those measures.

Thus the preparatory routine has precisely one task: To add $\bar{a} - a$ to the x of every order of the fourth kind in the subroutine.

12.7 The next question is this: By what criteria can the preparatory routine recognize the orders of the fourth kind in the subroutine?

Let I be the length of the subroutine. By this we mean the number of words, both orders and storage, that make it up. We include in this count all those words which have to be moved together when the final code (final enumeration) of the subroutine is moved (i.e. when its initial order position is moved from a to $\bar{a}$), and no others. The count is, of course, made on the final enumeration. In this sense a word counts fully, even if it contains a dummy order (e.g. 14 in Problem 6, and 6 in Problem 10 or 74, 91 in Problem 13.b). On the other hand storage positions which are being referred to, but which are supposed to be parts of some other routine, already in the machine (i.e. of the main routine, or of another subroutine) do not count (e.g. the storage area A in Problem 3 or the storage area A in Problem 10).

For an order of the fourth kind x must have one of the values a, ..., $a + I - 1$, i.e. it must fulfill the condition

$$(1) \qquad a \leqq x < a + I .$$

For an order of the third kind x will not fulfill this condition. For orders of the first and of the second kind the place of x is inessentially occupied. Concerning its relation to condition (1) we can make the two following remarks:

First: We can stipulate, that in all orders where the position of x is inessentially occupied, x should actually be put in with a value x^0 that violates (1). This is a perfectly possible convention. The simplest ways to carry it into effect are these:

Let x^0 always have the smallest value or always have the largest value that is compatible with its 12-digit character. (Regarding the latter, cf. section 6.2.) I.e. $x^0 = 0$ or $x^0 = L-1$, where L-1 is the largest 12-digit integer: $L = 2^{12} = 4,096$. Then all subroutines must fulfill $a \neq 0$ or $a + I \neq L$, respectively (in order that (1) be violated).

These rules are easy to observe. We chose a = 0 in most of our codes, hence we might prefer the second rule, but this is a quite unimportant preference.

Second: If an order of the first or of the second kind has an x which fulfills (1), and the order is thereupon mistakenly taken (by the preparatory routine) for one of the fourth kind, and its x is increased by $\bar{a} - a$, this need not matter either. Indeed: The place x is irrelevantly occupied, hence changes which take place there before the subroutine begins to operate do not matter. There is, however, one possible complication: Adding $\bar{a} - a$ to this (inessential) x may produce a carry beyond the 12 digits that are assigned to x. (Regarding these 12 digits cf. above, and also orders 18, 19 of Table II and the second remark among the Introductory Remarks to Chapter 10. The carry in question will occur if $\bar{a} - a > 0$ and $x \geqq L - (\bar{a} - a)$, or if $\bar{a} - a < 0$ and $x < -(\bar{a} - a)$; $L = 2^{12}$, cf. above.) Such a carry affects the other digits of the order, and thereby modifies its meaning in an undesirable way. This complication can be averted by special measures that paralyze carries of the type in question, but we will not discuss this here. No precautions are needed, if we see to it that no such carries occur. (I.e. if we observe $-(\bar{a} - a) \leqq x \leqq L - (\bar{a} - a)$ for the inessential x, cf. above.) ----

In view of these observations we may accept (1) as the criterium defining the orders of the fourth kind. We will therefore proceed on this basis.

12.8 We have to derive the preparatory routines which are needed to make subroutines effective. For didactical reasons, we begin with a preparatory routine which can only be used in conjunction with a single (but arbitrary) subroutine. Having derived such a *single subroutine preparatory routine*, we can then pass to the more general case of a preparatory routine which can be used in conjunction with any number of subroutines. This is a *general*, or *multiple subroutine preparatory routine*. The point in all of this is, of course, that both kinds of preparatory routines need only be coded once and in advance -- they can then be used in conjunction with arbitrary subroutines.

We state now the problem of a single subroutine preparatory routine. This includes a description of the subroutine, in which we assume that the subroutine has been coded in conformity with the (not a priori necessary) conventions that we found convenient to observe in our codings in these reports. It does not seem necessary to discuss at this place possible deviations from these conventions, and the rather simple ways of dealing with them.

PROBLEM 16.

A subroutine Σ consisting of I consecutive words, of which the k first ones are (two) order words, is given. (Concerning the definition of the length of a subroutine, cf. the beginning of 12.7. The subroutine under consideration may also make use of stored quantities, or of available storage capacity, outside this sequence of I words -- or rather of the $I - k$ last ones among them. We need not pay any attention to such outside positions in this problem.) This subroutine is coded as if it began at the memory position a. Actually, however, it is stored in the memory, beginning at the position $\bar{a}$. It is desired to modify it so that its coding conform with its actual position in the memory. ----

Our task consists in scanning the words from $\bar{a}$ to $\bar{a}+k-1$, to inspect in each one of these words the two orders that it contains, to decide for each order whether its x fulfills the condition (1) of 12.7; and in that (and only in that) case increase this x by $\kappa = \bar{a} - a$.

Let the memory position u be occupied by the word w_u. w_u is then an aggregate of 40 binary digits:

$$w_u = \{ w_u(1), w_u(2), \ldots, w_u(40) \} .$$

The two orders of which it consists are the two 20 digit aggregates

$$\{w_u(1), \ldots, w_u(20)\} , \quad \{w_u(21), \ldots, w_u(40)\} ,$$

the two x in these orders are the two 12 digit aggregates

$$w_u^I = \{w_u(9), \ldots, w_u(20)\}, \; w_u^{II} = \{w_u(29), \ldots, w_u(40)\}$$

(cf. orders 18, 19 of Table II).

Reading w_u^I, w_u^{II} as binary numbers with the binary point at the extreme left (and an extra sign digit 0), the condition (1) of 12.7 becomes

(1) $$2^{-12}a \leq w_u^I < 2^{-12} (a + \ell),$$

and

(2) $$2^{-12}a \leq w_u^{II} < 2^{-12} (a + \ell).$$

Reading w_u as we ordinarily read binary aggregates, i.e. as a binary number with the binary point between the first and second digits from the left (the first digit being the sign digit) we can now express our task as follows: If (1) or (2) holds, we must increase w_u by $2^{-19}\kappa$ or $2^{-39}\kappa$, respectively. I.e.

(3) $$w_u' \begin{cases} = w_u + 2^{-19}\kappa & \text{if (1) holds,} \\ = w_u & \text{otherwise,} \end{cases}$$

(4) $$w_u'' \begin{cases} = w_u' + 2^{-39}\kappa & \text{if (2) holds,} \\ = w_u' & \text{otherwise.} \end{cases}$$

Note, that we may replace (2), for its use in (4), by

(2') $$2^{-12}a \leq w_u'^{II} < 2^{-12} (a + \ell) ,$$

where

$$w'_u = \{ w'_u(1),\ w'_u(2),\ \ldots,\ w'_u(40) \} ,$$

$$w'^{II}_u = \{ w'_u(29),\ \ldots,\ w'_u(40) \} ,$$

since w_u and w'_u have by (3) the same digits with the positional values $2^{-20}, \ldots, 2^{-39}$, i.e. with the numbers 21, ..., 40.

The words w_u with which we have to deal are in the interval of memory locations $a + \kappa, \ldots, a + \kappa + k - 1$ (i.e. $\bar{a}, \ldots, \bar{a} + k - 1$, cf. above). Let this be the storage area O, we will index it with a $u = a + \kappa, \ldots, a + \kappa + k - 1$, so that O.u corresponds to u, and stores w_u. This u has the character of an induction index.

Further storage capacities are required as follows: u (as u_0) in A, the w_u under consideration (and w'_u after it) in B, the given data of the problem, a, ℓ, k, κ, in C. (It will be convenient to store them as $2^{-19}a$, $2^{-19}\ell$, $2^{-19}k$, $2^{-19}\kappa$. Regarding these quantities cf. also further below.) Storage will also have to be provided for various other fixed quantities (-1, 1_0, 2^{-7}, 2^{-12}, 2^{-32}), these too will be accomodated in C.

We can now draw the flow diagram, as shown in Figure 12.1. In coding it, we will encounter some deviations and complications which should be commented on.

a, ℓ, k, κ must occasionally be manipulated along the lines discussed in connection with the coding of Problem 13.a. Thus in the case of κ transitions to $2^{-39}\kappa$ and to κ_0 occur, and these would be rendered more difficult if we had to allow for the possibility $\kappa < 0$. In order to avoid this rather irrelevant complication, we assume

(5) $$\kappa \geqq 0, \quad \text{i.e. } \bar{a} \geqq a.$$

This has the further consequence, that the difficulties referred to in 12.7 can be avoided by giving every irrelevant x the value 0 (because of the second remark in 12.7) or the value L - 1 (because of the first remark in 12.7). We also note this: (5) can be secured by putting all a = 0, i.e. by coding every subroutine as if it began at 0, but we will not insist here that this convention be made.

The conditions (1), (2') can be tested by testing the signs of the quantities

$$w^I_u - 2^{-12}a,\ w^I_u - 2^{-12}(a+\ell),\ w'^{II}_u - 2^{-12}a,\ w'^{II}_u - 2^{-12}(a+\ell)$$

or, equivalently, the signs of the quantities

$$2^{-7}w^I_u - 2^{-19}a,\ 2^{-7}w^I_u - 2^{-19}(a+\ell),\ 2^{-7}w'^{II}_u - 2^{-19}a,\ 2^{-7}w'^{II}_u - 2^{-19}(a+\ell) .$$

It is easily seen that replacing $2^{-19}a$, $2^{-19}(a+\ell)$ by a_0, $(a+\ell)_0$ vitiates these sign-criteria, but replacing

$$w_u^I = \{w_u(9), \ldots, w_u(20)\}$$

by

$$w_u^I + \varepsilon_u = \{w_u(9), \ldots, w_u(20), w_u(21), \ldots, w_u(40)\}$$

does not have this effect. It is convenient to use $w_u^I + \varepsilon_u$ in place of w_u^I.

Both quantities $w_u^I + \varepsilon_u$ and $w_u'^{II}$ must be read as binary numbers with the binary point at the extreme left, with an extra sign digit 0. Indicating this sign digit, too, we have

$$(6)\qquad \begin{cases} w_u^I + \varepsilon_u = \{0, w_u(9), \ldots, w_u(40)\}, \\ w_u'^{II} \qquad = \{0, w_u'(29), \ldots, w_u'(40)\}. \end{cases}$$

With the same notations

$$(7)\qquad \begin{cases} w_u = \{w_u(1), w_u(2), \ldots, w_u(40)\}, \\ w_u' = \{w_u(1), w_u(2), \ldots, w_u'(40)\}. \end{cases}$$

In order to get the $w_u^I + \varepsilon_u$, $w_u'^{II}$ of (6) from the w_u, w_u' of (7), it seems simplest to multiply w_u, w_u' by 2^{-32}, 2^{-12}, respectively, and to pick up $w_u^I + \varepsilon_u$ in the register (cf. order 11 of Table II). There is, however, one minor complication at this point: The register contains not the $w_u^I + \varepsilon_u$, $w_u'^{II}$ of (6), but the aggregates

$$(6')\qquad \begin{cases} \{w_u(9), w_u(9), w_u(10), \ldots, w_u(40)\}, \\ \{w_u'(29), w_u'(29), w_u'(30), \ldots, w_u'(40)\}. \end{cases}$$

The simplest way to get from (6') to (6) is to sense the sign of each quantity of (6), and to add -1 to it if it proves to be negative (i.e. if the sign digit is 1). We will do this; it requires an additional conditional transfer in connection with each one of the two boxes III and VI. For this reason two boxes III.1 and VI.1, not shown in the flow diagram, will appear in our coding.

To conclude, it is convenient to change the position of IX somewhat (it follows upon VIII and absorbs part of it) and to absorb XI into X (it is replaced by X,9).

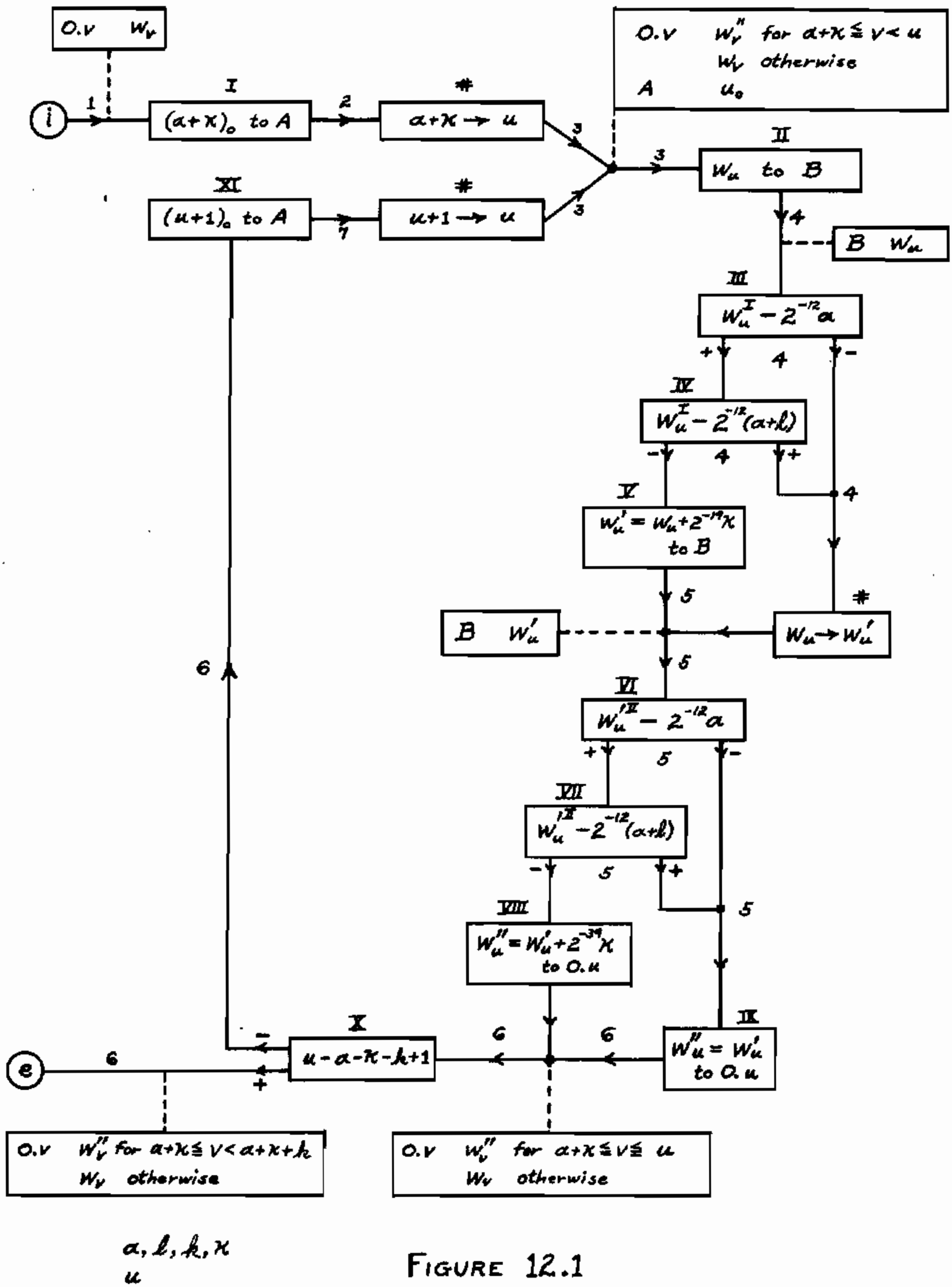

O.v W_v
(a+κ)_o to A
a+κ → u
O.v W''_v for a+κ ≤ v < u
W_v otherwise
A u_o
W_u to B
(u+1)_o to A
u+1 → u
B W_u
W_u^I − 2^-12 a
W_u^I − 2^-12 (a+l)
W'_u = W_u + 2^-19 κ to B
B W'_u
W_u → W'_u
W'_u^II − 2^-12 a
W'_u^II − 2^-12 (a+l)
W''_u = W'_u + 2^-39 κ to O.u
W''_u = W'_u to O.u
u − a − κ − k + 1
O.v W''_v for a+κ ≤ v < a+κ+k
W_v otherwise
O.v W''_v for a+κ ≤ v ≤ u
W_v otherwise
a, l, k, κ
u

FIGURE 12.1

The static coding of the boxes I-XI follows:

C.1	$2^{-19}a$			
C.2	$2^{-19}\kappa$			
I,1	C.1		Ac	$2^{-19}a$
2	C.2	h	Ac	$2^{-19}(a+\kappa)$
3	s.1	Sp'	s.1	$2^{-39}(a+\kappa)$
4	s.1	h	Ac	$(a+\kappa)_0$
5	A	S	A	$(a+\kappa)_0$
(to II,1)				
II,1	A		Ac	u_0
2	II,3	Sp	II,3	u
3	-			
[	u	]	Ac	w_u
4	B	S	B	w_u
(to III,1)				
C.3	2^{-32}			
III,1	C.3	R	R	2^{-32}
2	B	x	R	$\{w_u(9), w_u(9), w_u(10), \ldots, w_u(40)\} =$ $= w_u(9) + w_u^I + \varepsilon_u$
3		A	Ac	$w_u(9) + w_u^I + \varepsilon_u$
4	III.1,1	Cc		
C.4	-1			
III,5	C.4	h	Ac	$w_u^I + \varepsilon_u$
(to III.1,1)				
C.5	2^{-7}			
III.1,1	C.5	R	R	2^{-7}
2	s.1	S	s.1	$w_u^I + \varepsilon_u$
3	s.1	x	Ac	$2^{-7}(w_u^I + \varepsilon_u)$
4	C.1	h-	Ac	$2^{-7}(w_u^I + \varepsilon_u) - 2^{-19}a$
5	IV,1	Cc		
(to VI,1)				
C.6	$2^{-19}\chi$			
IV,1	C.6	h-	Ac	$2^{-7}(w_u^I + \varepsilon_u) - 2^{-19}(a+\chi)$
2	VI,1	Cc		
(to V,1)				
V,1	C.2		Ac	$2^{-19}\kappa$
2	B	h	Ac	$w_u + 2^{-19}\kappa = w_u'$
3	B	S	B	w_u'
(to VI,1)				
C.7	2^{-12}			

VI,1	C.7	R	R	2^{-12}
2	B	x	R	$\{w'_u(29), w'_u(29), w'_u(30), \ldots, w'_u(40)\} =$ $= w'_u(29) + w_u'^{II}$
3		A	Ac	$w'_u(29) + w_u'^{II}$
4	VI.1,1	Cc		
5	C.4	h	Ac	$w_u'^{II}$
(to VI.1,1)				
VI,1.1	C.5	R	R	2^{-7}
2	s.1	S	s.1	$w_u'^{II}$
3	s.1	x	Ac	$2^{-7}\, w_u'^{II}$
4	C.1	h-	Ac	$2^{-7}\, w_u'^{II} - 2^{-19}a$
5	VII,1	Cc		
(to IX,1)				
VII,1	C.6	h-	Ac	$2^{-7}\, w_u'^{II} - 2^{-19}(a+\ell)$
2	IX,1	Cc		
(to VIII,1)				
VIII,1	C.2		Ac	$2^{-19}\kappa$
2	s.1	Sp'	s.1	$2^{-39}\kappa$
3	s.1		Ac	$2^{-39}\kappa$
4	B	h	Ac	$w'_u + 2^{-39}\kappa = w''_u$
5	B	S	B	w''_u
(to IX,1)				
IX,1	A		Ac	u_0
2	IX,4	Sp	IX,4	u Sp
3	B		Ac	w''_u
4	-	S		
[	u	S]	0.u	w''_u
(to X,1)				
X,1	C.1		Ac	$2^{-19}a$
2	C.2	h	Ac	$2^{-19}(a+\kappa)$
C.8	$2^{-19}k$			
X,3	C.8	h	Ac	$2^{-19}(a+\kappa+k)$
4	s.1	Sp'	s.1	$2^{-39}(a+\kappa+k)$
5	s.1	h	Ac	$(a+\kappa+k)_0$
6	s.1	S	s.1	$(a+\kappa+k)_0$
7	A		Ac	u_0
C.9	1_0			
X,8	C.9	h	Ac	$(u+1)_0$
9	A	S	A	$(u+1)_0$
10	s.1	h-	Ac	$(u-a-\kappa-k+1)_0$
11	e	Cc		
(to XI,1)				
XI,1	-			
(to II,1)				

Note, that the box XI required no coding, hence its immediate successor (II) must follow directly upon its immediate predecessor (X).

The ordering of the boxes is I, II, III, III.1, VI, VI.1, IX, X; IV, V; VII, VIII, and VI, IX, II must also be the immediate successors of V, VIII, X, respectively. This necessitates the extra orders

V,4	VI,1	C
VIII,6	IX,1	C
X,12	II,1	C

We must now assign A, B, C.1-9, s.1 their actual values, pair the 59 orders I,1-5, II,1-4, III,1-5, III.1,1-5, IV,1-2, V,1-4, VI,1-5, VI.1,1-5, VII,1-2, VIII,1-6, IX,1-4, X,1-12 to 30 words, and then assign I,1-X,12 their actual values. We wish to place this code at the end of the memory, so that it should interfere as little as possible with the memory space that is normally occupied by other subroutines and routines. Let us therefore consider the words in the memory backwards (beginning with the last word), and designate their numbers (in the reverse order referred to) by $\overline{1}$, $\overline{2}$, ... In this way we obtain the following table:

I,1-5	$\overline{42}$ - $\overline{40}$	VI.1,1-5	$\overline{30}$ - $\overline{28}$	VII,1-2	$\overline{17}'$-$\overline{16}$
II,1-4	$\overline{40}'$-$\overline{38}$	IX,1-4	$\overline{28}'$-$\overline{26}$	VIII,1-6	$\overline{16}'$-$\overline{13}$
III,1-5	$\overline{38}'$-$\overline{36}'$	X,1-12	$\overline{26}'$-$\overline{20}$	A	$\overline{12}$
III.1,1-5	$\overline{35}$ - $\overline{33}$	IV,1-2	$\overline{20}'$-$\overline{19}$	B	$\overline{11}$
VI,1-5	$\overline{33}'$-$\overline{31}'$	V,1-4	$\overline{19}'$-$\overline{17}$	C,1-9	$\overline{10}$ - $\overline{2}$
				s.1	$\overline{1}$

Now we obtain this coded sequence:

$\overline{42}$	$\overline{10}$,	$\overline{9}$ h	$\overline{28}$	$\overline{17}$ Cc',	$\overline{12}$	$\overline{14}$	$\overline{11}$ h ,	$\overline{11}$ S
$\overline{41}$	$\overline{1}$ Sp',	$\overline{1}$ h	$\overline{27}$	$\overline{26}$ Sp,	$\overline{11}$	$\overline{13}$	$\overline{28}$ C',	- -
$\overline{40}$	$\overline{12}$ S ,	$\overline{12}$	$\overline{26}$	- S ,	$\overline{10}$	$\overline{12}$	-	
$\overline{39}$	$\overline{39}$ Sp',	-	$\overline{25}$	$\overline{9}$ h ,	$\overline{3}$ h	$\overline{11}$	-	
$\overline{38}$	$\overline{11}$ S ,	$\overline{8}$ R	$\overline{24}$	$\overline{1}$ Sp',	$\overline{1}$ h	$\overline{10}$	$2^{-19}a$	
$\overline{37}$	$\overline{11}$ x ,	A	$\overline{23}$	$\overline{1}$ S ,	$\overline{12}$	$\overline{9}$	$2^{-19}\kappa$	
$\overline{36}$	$\overline{35}$ Cc,	$\overline{7}$ h	$\overline{22}$	$\overline{2}$ h ,	$\overline{12}$ S	$\overline{8}$	2^{-32}	
$\overline{35}$	$\overline{6}$ R ,	$\overline{1}$ S	$\overline{21}$	$\overline{1}$ h-,	e Cc	$\overline{7}$	-1	
$\overline{34}$	$\overline{1}$ x ,	$\overline{10}$ h-	$\overline{20}$	$\overline{40}$ C',	$\overline{5}$ h-	$\overline{6}$	2^{-7}	
$\overline{33}$	$\overline{20}$ Cc',	$\overline{4}$ R	$\overline{19}$	$\overline{33}$ Cc',	$\overline{9}$	$\overline{5}$	$2^{-19}\chi$	
$\overline{32}$	$\overline{11}$ x ,	A	$\overline{18}$	$\overline{11}$ h ,	$\overline{11}$ S	$\overline{4}$	2^{-12}	
$\overline{31}$	$\overline{30}$ Cc,	$\overline{7}$ h	$\overline{17}$	$\overline{33}$ C',	$\overline{5}$ h-	$\overline{3}$	$2^{-19}k$	
$\overline{30}$	$\overline{6}$ R ,	$\overline{1}$ S	$\overline{16}$	$\overline{28}$ Cc',	$\overline{9}$	$\overline{2}$	1_0	
$\overline{29}$	$\overline{1}$ x ,	$\overline{10}$ h-	$\overline{15}$	$\overline{1}$ Sp',	$\overline{1}$	$\overline{1}$	-	

The durations may be estimated as follows:

I: 200 μ, II: 150 μ, III: 250 μ, III.1: 270 μ, IV: 75 μ, V: 150 μ, VI: 250 μ, VI.1: 270 μ, VII: 75 μ, VIII: 225 μ, IX: 150 μ, X: 450 μ.

$$\text{Total:}\quad I + \left[\begin{array}{l} II + III + III.1 + (\theta^{*)} \text{ or } IV \text{ or } IV + V) \\ \quad + VI + VI.1 + (\theta^{*)} \text{ or } VII \text{ or } VII + VIII) \\ \quad + IX + X \end{array}\right] \times k =$$

$$\text{maximum} = \left(200 + \left[\begin{array}{l} 150 + 250 + 270 + 75 + 150 \\ \quad + 250 + 270 + 75 + 225 \\ \quad + 150 + 450 \end{array}\right] k\right) \mu =$$

$$= (2{,}315\ k + 200)\ \mu \approx (2.3\ k + .2)\ m.$$

12.9 We now pass to the multiple subroutine preparatory routine. The requirements for such a routine allow several variants. We will consider only the basic and simplest one. It is actually quite adequate to take care of most situations involving the use of several subroutines -- even of very complicated ones. (Examples will occur in our future codings, in particular in Chapters 13 and 14.)

PROBLEM 17

Same as Problem 16 with this change: The modification is desired for I subroutines $\Sigma_1, \ldots, \Sigma_I$. The characteristic data for Σ_i (in the sense of Problem 16) are a_i, ℓ_i, k_i, κ_i. Each Σ_i is stored as its a_i, κ_i, and ℓ_i indicate (cf. Problem 16), the a_i, ℓ_i, k_i, κ_i $(i = 1, \ldots, I)$ are stored at 4I suitable, consecutive memory locations. ----

In order to be able to use the treatment of Problem 16 in 12.8, we assume in conformity with (5) in 12.8

(1) $$\kappa_i \geq 0 \quad \text{for all } i = 1, \ldots, I.$$

(Cf. also the other pertinent remarks loc. cit.)

i is the induction index, running from 1 to I. For each value of i we have to solve Problem 16. We can do this with the help of our coding of that problem, but we must substitute the data of the problem, a_i, ℓ_i, k_i, κ_i, into the appropriate places. Inspection of the coded sequence shows that these places are $\overline{10}$, $\overline{5}$, $\overline{3}$, $\overline{9}$, respectively.

We propose to place the coded sequence that we are going to develop immediately before that one of Problem 16, i.e. immediately before $\overline{42}, \ldots, \overline{1}$. Let P be the number of the memory location immediately before the coded sequence that we are going to develop -- i.e., if the length of that sequence is ℓ' and the total memory capacity is L' (both in terms of words), then

(2) $$P = \overline{43 + \ell'} = L' - 43' - \ell' .$$

---

*) θ represents the possibility of going directly from III via 4, 5 to VI, or from VI via 5 to IX, respectively, with no other boxes intervening. We will use this same notation in similar situations in the future.

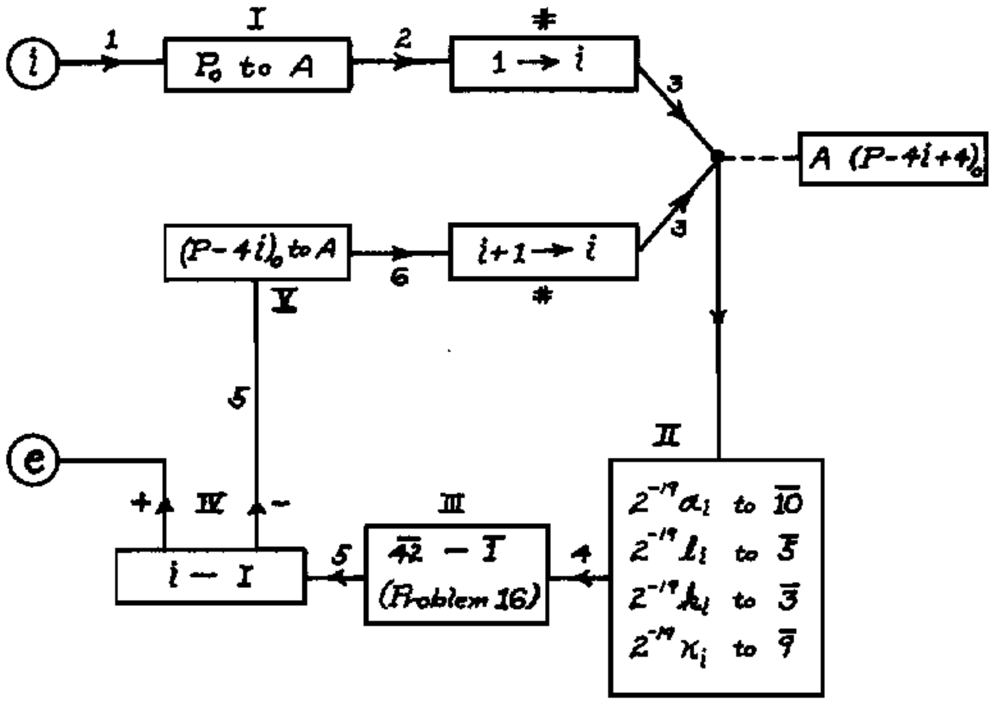
We will place the a_i, ℓ_i, k_i, κ_i ($i = 1, \ldots, I$), in the form $2^{-19}a_i$, $2^{-19}\ell_i$, $2^{-19}k_i$, $2^{-19}\kappa_i$, immediately before these coded sequences, and in inverse order, i.e. at P-4i+4, P-4i+3, P-4i+2, P-4i+1, respectively.

The induction index i will be stored in the form $(P\text{-}4i+4)_0$ in the storage area A. The quantities P, I will be stored in the form P_0, $(P\text{-}4I)_0$ in the storage area B (1_0 will be needed and taken from $\bar{2}$).

We can now draw the flow diagram, as shown in Figure 12.2. The actual coding obtains from this quite directly, box V is absorbed into box II (it is replaced by II,10).

The static coding of the boxes I-V follows:

B,1	P_0			
I,1	B,1		Ac	P_0
2	A	S	A	P_0
(to II,1)				
II,1	A		Ac	$(P\text{-}4i+4)_0$
2	II,11	Sp	II,11	P-4i+4
3	$\bar{2}$	h-	Ac	$(P\text{-}4i+3)_0$
4	II,13	Sp	II,13	P-4i+3
5	$\bar{2}$	h-	Ac	$(P\text{-}4i+2)_0$
6	II,15	Sp	II,15	P-4i+2
7	$\bar{2}$	h-	Ac	$(P\text{-}4i+1)_0$
8	II,17	Sp	II,17	P-4i+1
9	$\bar{2}$	h-	Ac	$(P\text{-}4i)_0$
10	A	S	A	$(P\text{-}4i)_0$
11	- [P-4i+4]		Ac	$2^{-19}a_i$
12	$\overline{10}$	S	$\overline{10}$	$2^{-19}a_i$
13	- [P-4i+3]		Ac	$2^{-19}\ell_i$
14	$\bar{5}$	S	$\bar{5}$	$2^{-19}\ell_i$
15	- [P-4i+2]		Ac	$2^{-19}k_i$
16	$\bar{3}$	S	$\bar{3}$	$2^{-19}k_i$
17	- [P-4i+1]		Ac	$2^{-19}\kappa_i$
18	$\bar{9}$	S	$\bar{9}$	$2^{-19}\kappa_i$
(to III,1)				



0 . $P-4i+4$	$2^{-19}a_i$
$P-4i+3$	$2^{-19}l_i$
$P-4i+2$	$2^{-19}k_i$
$P-4i+1$	$2^{-19}\kappa_i$

I
i

Note: Numbers
$\overline{42}-\overline{1}$
refer to
$\overline{42}-\overline{1}$
in 12.7

FIGURE 12.2

III (Problem 16: $\overline{42}$ - $\overline{1}$.)

(to IV,1)				
B.2	$(P-4I)_0$			
IV,1	B.2		A_c	$(P-4I)_0$
2	A	h-	A_c	$(4(i-I))_0$
3	e	C_c		
(to V,1)				
V	----			
(to II,1)				

Note, that box V required no coding, hence its immediate successor (II) must follow directly upon its immediate predecessor (IV).

The ordering of the boxes is I, II, III, IV, and II must also be the immediate successor of IV. In addition, III cannot be placed immediately after II, since III,1 is 42 but III must nevertheless be the immediate successor of II. All this necessitates the extra orders

II,19	III,1	C
IV,4	II,1	C

Finally, in order that IV be the immediate successor of III, the e of $\overline{42}$ - $\overline{1}$ (in 21) must be equal to IV,1.

We must now assign A, B.1-2 their actual values, pair the 25 orders I,1-2, II,1-19, IV,1-4 to 13 words, and then assign I,1-IV,4 their actual values. (III is omitted, since it is contained in $\overline{42}$ - $\overline{1}$.) We wish to do this as a (backward) continuation of the code of 12.8. In this way we obtain the following table:

I,1-2	$\overline{58}$ - $\overline{58}'$	II,1-19	$\overline{57}$ - $\overline{48}$	A	$\overline{45}$
		IV,1-4	$\overline{48}'$ - $\overline{46}$	B.1-2	$\overline{44}$ - $\overline{43}$

Thus, in terms of equation (2), $I' = 16$ and

(2') $$P = \overline{59} = L' - 59 \quad .$$

Now we obtain this coded sequence:

$\overline{58}$	$\overline{44}$, $\overline{45}$ S	$\overline{53}$	$\overline{2}$ h-, $\overline{45}$ S	$\overline{48}$	$\overline{42}$ C , $\overline{43}$
$\overline{57}$	$\overline{45}$, $\overline{52}$ S_p	$\overline{52}$	- , $\overline{10}$ S	$\overline{47}$	$\overline{45}$ h-, e C_c
$\overline{56}$	$\overline{2}$ h-, $\overline{51}$ S_p	$\overline{51}$	- , $\overline{5}$ S	$\overline{46}$	$\overline{57}$ C , - -
$\overline{55}$	$\overline{2}$ h-, $\overline{50}$ S_p	$\overline{50}$	- , $\overline{3}$ S	$\overline{45}$	-----
$\overline{54}$	$\overline{2}$ h-, $\overline{49}$ S_p	$\overline{49}$	- , $\overline{9}$ S	$\overline{44}$	P_0
				$\overline{43}$	$(P-4I)_0$

In addition $\overline{21}$ in $\overline{42}$ - $\overline{1}$ of 12.8 must read

$\overline{21}$ $\overline{1}$ h-, $\overline{48}$ Cc'

The durations may be estimated as follows:

I: 75 μ, II: 725 μ, IV: 150 μ.

III: The precise estimate at the end of 12.7 is

$$\text{maximum} = (2,315\ k_i + 200)\ \mu.$$

$$\text{Total:}\quad I + \sum_{i=1}^{I} (II + III + IV) =$$

$$\text{maximum} = (75 + \sum_{i=1}^{I} (725 + 2,315\ k_i + 200 + 150))\ \mu =$$

$$= (2,315 \sum_{i=1}^{I} k_i + 1,075\ I + 75)\ \mu \approx$$

$$\approx (2.3 \sum_{i=1}^{I} k_i + 1.1\ I)\ m\ .$$

$\sum_{i=1}^{I} k_i$ is the total length of all subroutines. Hence it is necessarily $\leq L' \leq L = 2^{12} = 4,096$. Actually it is unlikely to exceed, even in very complicated problems, the order $\approx \frac{1}{4} L \approx 1,000$. 1.1 I is negligible compared to $2.3 \sum_{i=1}^{I} k_i$. Hence

$$2,300\ m = 2.3 \text{ seconds}$$

is a high estimate for the duration of this routine.

12.10 Having derived two typical coded sequences for preparatory routines, it is appropriate to say a few words as to how their actual use can be contemplated. We do not propose to present a discussion of this subject to any degree of completeness -- we only wish to point out some of the most essential aspects. Since the routine of Problem 17 is more general than that one of Problem 16, we will base our discussion on the former.

The discussion of the use of preparatory subroutines is necessarily a discussion of a certain use of the input organs of the machine. We have refrained in our reports, so far, from making very detailed and specific assumptions regarding the characteristics of the input (as well as of the output) organs. At the present juncture, however, certain assumptions regarding these organs are necessary. On the basis of engineering developments up to date, such assumptions are possible on a realistic basis. Those that we are going to formulate represent a very conservative, minimum program.

Our assumptions are these:

As indicated in section 4.5, we incline towards the use of magnetic wire (soundtrack) as input (and output) medium. We expect to use it at pulse rates of about 25,000 pulses (i.e. binary digits) per second. We will certainly use several input (and output) channels, but for the purposes of the present discussion we assume a single one.

We assume that the contents of a wire can be "fed" into the machine, i.e. transferred into its inner, selectron memory, under manual control. We assume that we can then describe by manual settings

1) to which memory position the first word from the wire should go (the subsequent words on the wire should then go in linear order into the successive, following memory positions),

2) how many words from the wire should be so "fed".*)

We assume finally, that single words can also be "fed" directly into the machine by typing them with an appropriate "typewriter". (This "typewriter" will produce electrical pulses, and will be nearly the same as the one used to "write" on the magnetic tape.) We assume that we can determine the memory position to which the typed word goes by manual settings.

The last assumption, i.e. the possibility of typing directly into the memory, is not absolutely necessary. It is alternative to the previously mentioned "feeding" of words from a magnetic wire. When longer sequences of words have to be fed, the wire is preferable to direct typing. When single words, irregularly distributed, have to be fed, however, then feeding from appropriate wires would still be feasible, but definitely more awkward than direct typing. In addition, the possibility of direct typing into the memory is probably very desirable in connection with testing procedures for the machine. We are therefore assuming its availability.

These things being understood, we can describe the procedure of placing a, presumably composite, routine into the machine. It consists of the following steps:

First: There are one or more constituent routines: The main routine and the subroutines, where it is perfectly possible that the subroutines bear further subordination relationships to each other, i.e. are given as a hierarchy. All of these are coded and stored on separate pieces of magnetic wire.

These are successively fed into the machine, i.e. into the inner memory. The desired positions in the memory are defined by manual settings.

---

*) These operations should also be feasible under the "inner", electronic control of the machine. We will discuss this aspect of the input-output organ, and the logical, code orders which circumscribe it, in a subsequent paper (Part III in this series).

Second: The multiple subroutine preparatory routine (Problem 17) is also coded and stored on an individual magnetic wire. (It is, of course, assumed to be part of the library of wires mentioned in section 4.6.)

This is also fed into the machine, like the "constituent" routines of the first operation, described above. As observed in 12.8 and 12.9 it has to be positioned at the end of the memory.

Third: The constants of the multiple subroutine preparatory routine, i.e. of Problem 17, are typed directly into the memory. They are the following ones:

1) The number of subroutines, I, which is put in the form $(P\text{-}4I)_0$ into the position

$$\overline{43} = L' - 43 = P + 16.$$

2) The 4I data which characterize the I subroutines: a_i, ℓ_i, k_i, κ_i $(i = 1, \ldots, I)$, which are put in the form $2^{-19}a_i$, $2^{-19}\ell_i$, $2^{-19}k_i$, $2^{-19}\kappa_i$ into the positions $\overline{P\text{-}4i\text{+}4}$, $\overline{P\text{-}4i\text{+}3}$, $\overline{P\text{-}4i\text{+}2}$, $\overline{P\text{-}4i\text{+}1}$, respectively.

Fourth: The machine is set going, with the control set manually at the beginning of the preparatory routine ($\overline{58} = L'\text{-}48 = P + 1$), and the adjustment of all subroutines to their actual positions (in the sense of 12.5-12.7) is thus effected.

Fifth: Any further adjustments which are necessitated by the relationships of the subroutines to each other (cf. the first operation, as described above, and the fifth remark in 12.11) are made by typing directly into the memory. ----

After all these operations have been carried out, the machine is ready to be set going on the (composite) routine of the problem itself The preparatory routine (in the 58 last memory positions) and its data (in the 4I preceding positions) are now no longer needed. I.e. these positions may now be viewed, from the point of view of the problem itself, as irrelevantly occupied. They are accordingly available for use in this sense.

12.11 We can now draw some conclusions concerning the setting-up procedure for a machine of the type contemplated, on the basis of the discussion of 12.10. We state these in the form of five successive remarks.

First: The pure machine time required to feed the (main and subsidiary) routines of the problem into the machine may be estimated as follows: A word consists of 40 pulses. For checking and marking purposes it will probably have to contain some additional pulses. With the systems that we are envisaging, a total of, say, 60 pulses per word will not be exceeded. With the speed of 25,000 pulses per second, as assumed in 12.10, this gives 2.4 m per word.

Thus the pure machine time of the first and second operations of 12.10 is 2.4 m per word.

The pure machine time of the third operation of 12.10 is, as we saw at the end of 12.9, 2.3 m per word.

We also observed at the end of 12.9, that even in very complicated problems the number of words thus involved is not likely to exceed 1000. This puts on the total time requirement on these counts an upper limit of 2.4 + 2.3 = 4.7 seconds, i.e. of less than 5 seconds.

Second: These time requirements are obviously negligible compared to the time consumed by the attendant manual operations: The placing of the magnetic wires into the machine, the setting of the (memory) position definitions, etc.

It follows therefore, that there is no need and no justification for any special routine-preparing equipment (other than the typing devices already discussed) to complement a machine of the type that we contemplate.

Third: Assuming a composite routine made up of ten parts, i.e. of a routine and nine subroutines, we have I = 9. This represents already a very high level of complication. The preparatory routine requires 58 + 4I words, i.e. for I = 9 94 words. This represents 2.3% of the total (selectron) memory capacity, if we assume that the latter is $L' = 2^{-12} = 4,096$.

Fourth: Each subroutine requires the direct, manual typing of four words into the machine (for a_i, l_i, k_i, κ_i), as well as one for all subroutines together (for I). In addition the changes of the second kind in the sense of 12.4, i.e. those which the main routine must effect on the subroutine, require several words. Indeed, the sending of the control to the subroutine requires one order, i.e. half a word. Any number is sent there at the price of two orders (bringing the number to be substituted into the subroutine into the accumulator, substituting it into the subroutine) and of possibly one storage word (for the number to be substituted), i.e. of a total of one or two words. There will usually be three or more such number substitutions (the e of the subroutine, i.e. the memory position in the main routine from where the control is to continue after the completion of the subroutine, and two or more data for the subroutine). Thus five words for these changes is a conservative estimate.

A subroutine consumes therefore ten words in extra instructions, by a conservative estimate. It seems therefore, that the storage of a subroutine in a library of wires, in the sense of section 4.6, and its corresponding treatment as an individual entity becomes justified when its length in words is significantly larger than 10. A minimum length of 15-20 words would therefore seem reasonable.

To conclude this discussion, we observe that in making these estimates, we disregarded all operations other than the actual, manual typing of words (on wire or into the machine). This is legitimate, because the time and the memory requirements of the automatic operations that are involved are negligible, as we saw in our three first remarks.

Fifth: We pointed out in our description of the first operation in 12.10, that the various subroutines used in connection with a main routine, may bear further subordination relationships to each other. In this case they will also contain actual references to each other, and these will have to be adjusted to the actual positions of the subroutines in question in the memory. These adjustments may be made as changes of the second kind, in the sense of 12.4, by the routines involved. They may also be handled by special preparatory routines. We expect, however, that it will be simplest in most cases to take care of them by direct typing into the memory, as indicated in the description of the fifth operation in 12.10.

The adjusting of the references of a subroutine to itself, to the actual position of the subroutine in the memory, might also have been made by direct typing into the machine. We chose to do it automatically, however, by means of a preparatory routine, because these references are very frequent: The great majority of all orders in a subroutine contain references to this same subroutine. References to another subroutine, on the other hand, are likely to be rare and irregularly distributed. They are therefore less well suited to automatic treatment, by a special preparatory routine, than to ad hoc, manual treatment, by direct typing into the machine.

Actual examples of such situations in which it will also be seen that the proportions of the various factors involved are of the nature that we have anticipated here, will occur in the subsequent chapters, and in particular in Chapters 13 and 14.