CHAPTER 6

SCALING

With the convention adopted for SILLIAC (see Chapter 2), only numbers which lie in the range $-1 \leq x < 1$ can be held in the registers. Since most problems require numbers outside this range, some scaling process is usually needed to fit a problem to the machine. It is necessary that <u>each number</u> at <u>every stage</u> of a calculation lie within the capacity of the machine. The organization required to assure this is sometimes trivial, but in many instances it is the very essence of the problem.

6.1 <u>SCALING BY SHIFTING.</u> Although the number 2 lies outside the range of SILLIAC numbers, we can multiply and divide numbers by powers of 2 by shifting. Thus the left shift order 00 10F will cause (AQ) to be multiplied by $2^{10} = 1024$. Similarly, the right shift order 10 9F will divide AQ by $2^9 = 512$. A knowledge of the use of the shift orders is essential to an understanding of scaling.

6.2 <u>NUMBERS WITH THE BINARY POINT SHIFTED.</u> Let us consider the problem of computing with numbers in which the binary point has been moved 10 places to the right of its SILLIAC position. We are then dealing with numbers in the range $-1024 \leq y \leq 1024 - 2^{-29}$. Let such a number, when in location m, be designated by $N_{10}(m)$. Then we have $N_{10}(m) = 2^{10}(m)$.

We can formulate rules for doing arithmetic with the numbers N_{10}. Addition and subtraction are simple. If $(q) = (m) + (n)$, then

$$2^{10}(q) = 2^{10}(m) + 2^{10}(n)$$

and

$$N_{10}(q) = N_{10}(m) + N_{10}(n).$$

Thus, the SILLIAC addition rules hold.

Multiplication requires a shift to the left of 10 places. We want

$$N_{10}(q) = N_{10}(m) \times N_{10}(n).$$

Thus, we require $2^{10}(q)$ to represent the product of $2^{10}(m)$, and $2^{10}(n)$. This requirement is satisfied if

$$(q) = 2^{10} \left[(m) \times (n)\right] .$$

We may consider several simple routines to carry out the multiplication. The method given by (b) is probably the best.

(a) The shortest method merely multiplies and shifts. Notice that a 75 order must be used rather than a 74 order because the least significant digit of $N_{10}(p)$ is now 2^{-29} and not 2^{-39}. A bias of -2^{-30} is introduced by the absence of a roundoff. The program is given in Table 6.1.

(b) This method rounds off by adding 2^{-30} to $N_{10}(p)$, giving an unbiased result. The program is given in Table 6.2.

```
50 mF    (m) to Q
75 nF    (m) x (n) to AQ
00 10F   2^10(m) x (n)
40 pF    to p.
```

Table 6.1

Multiplication with Binary Point Shifted

19	10F	2^{-11} to A
50	nF	(n) to Q
74	mF	(m) x (n) + 2^{-50} to AQ
00	10F	2^{10}(m) x (n) + 2^{-40} to A
40	pF	to p.

Table 6.2
Unbiased Multiplication with
Binary Point Shifted

(c) A roundoff similar to that of division (i.e. in which the last digit is forced equal to 1) is obtained with the program given in Table 6.3.

t	50 mF	(m) to Q
	75 nF	(m) x (n) to AQ
	00 9F	2^{9}(m) x (n) to AQ
	50 tF	$q_1 = 1$
	00 1F	This makes 2^{-39} A = 1; (A) = 2^{10}(m) x (n)
	40 pF	to p.

Table 6.3
Division-Type Roundoff in Multiplication
with Binary Point Shifted

In Table 6.3 the order 50 tF is used simply because the word at location t has a 1 in the appropriate position.

In division we also need an extra shift to restore the quotient to the proper range. But here the shift precedes the divide order and no special arrangements for roundoff are necessary. We want

$$N_{10}(q) = N_{10}(m)/N_{10}(n).$$

Hence we wish $2^{10}(q)$ to represent (m)/(n). The operation which gives this representation is:

$$(q) = 2^{-10}(m)/(n),$$

the shift being carried out before the division.

The following program in Table 6.4 will carry out the required operations:

L5	mF	(m) to A
10	10F	2^{-10}(m)
66	nF	2^{-10}(m)/(n)
NO	pF	to p.

Table 6.4

Division with Binary Point Shifted

It can be noted that by using such relations as

$$10^p = 2^n \times 10^p/2^n$$

where $10^p < 2^n$ we can use decimal scaling although it will be slower and clumsier to handle because of the factors $10^p/2^n$. Decimal scaling however, has the **decided** advantage that any print-out done with standard routines will be easier to interpret than if binary scaling were to be used.

6.3 SCALING A FULL PROBLEM. There are conceptually two ways in which we can approach the scaling of a problem; both give the same program.

(a) We can alter the problem using such substitutions as x' = 100x or p' = 32p, so that the modified problem has all its variables less than one but retaining full significance.

(b) We can use scaled numbers inside the machine

to represent the variables. Thus, we can use x/1000 instead of x and p/32 instead of p.

The final result must be such that the variables lie within machine range and retain sufficiant accuracy. Constants greater than one can be represented by numbers less than one in conjunction with scaling factors. For example, multiplication by 5.63 can be done by multiplying by 5.63/8 and shifting left 3 places.

Let us consider the following simple problem:

Example 1. Program the SILLIAC to compute the quantities

$$y = \frac{e^x}{1 + x^2}$$

at intervals of 0.01 from x = 0 to x = 6.

We note that we must scale x. Since the largest value of x is 6, let us use x/8 inside the computer because this will minimize the loss of significant figures.

What scaling factor is needed for y? A rough estimate shows that y does not exceed $e^6/(1 + 36) \sim 11$, so that we shall compute and store y/16.

Library Routine S2, the exponential routine, will give valid results only if x is negative. We therefore write

$$e^{x/8} = e \times e^{-1 + x/8}$$

so that

$$e^x = (e^{x/8})^8.$$

Now

$$y/16 = \frac{e^x}{16 \times 64} \times \frac{1}{1/64 + x^2/64},$$

and

$$\frac{e^x}{16 \times 64} = \left\{\left[(e^{-1 + x/8} \times e/4)^2 \times 2\right]^2 \times 2\right\}^2.$$

We shall store the constants e/4 and 1/16. Each part of the

computation is within machine capacity. We proceed as follows:

(a) Square x/8 and add 1/64,
(b) call the result R and store it,
(c) evaluate $e^{-1 + x/8}$ with Library Routine S2,
(d) multiply by e/4 and square,
(e) double and square,
(f) double and square again. Call the result P,
(g) form y/16 by dividing P by R.

Accuracy in the Result. Let us now consider how much accuracy we obtain. If x/8 is near unity, numbers remain large during the calculation and we do not lose significance by subtracting nearly equal numbers or by other ill-conditioning.

However, when x/8 is small we notice that we form $e^x/(16 \times 64)$ quite accurately and then divide by $R \sim 1/64$ which is about equivalent to a left shift of 6 places and loses 6 binary digits on the right. We can prevent this if we perform the division while the double-length P is in AQ.

Accuracy in the Argument. We should also consider the accuracy of the argument x/8. It can be formed either by successively adding the increment 0.01/8 = 0.00125 or by counting and multiplying. The adding method is not so good because the quantity 0.00125 will not be stored exactly and the accumulated roundoff error obtained by the time x is 6 (which required 600 additions) may be troublesome.

The counting method avoids this trouble. Any x/8 is n x 0.01/8. Hence we have

$$x/8 = n \quad 0.01/8$$
$$= (n \quad 2^{-39}) \quad 0.64 \quad 2^{30} .$$

If we store 0.64 and count to get $n \times 2^{-39}$, we can use the

following set of orders to get x/8 (where we have used arguments in place of addresses):

```
50    n x 2^-39
75    0.64
00    30
40    x/8   .
```

The program for the entire calculation of y/16 is given in Table 6.5 on page 6-8 where we have again used arguments instead of addresses.

Example 2: Solve the pair of simultaneous equations

$$ax + by + c = 0$$
$$dx + ey + f = 0$$

where the coefficients are in absolute values less than 1/2 and where the answers are known to lie within machine range. Retain as much accuracy as is reasonably possible.

We can distinguish two cases:

(a) If $|a| \geq |d|$, then

$$y = - \frac{dc/a - f}{bd/a - e} ,$$

$$x = - \frac{c+by}{a}$$

(b) If $|d| > |a|$, then

$$y = - \frac{c - fa/d}{b - ae/d}$$

$$x = - \frac{f + ey}{d}$$

In this program we shall follow the conventions of the Decimal Order Input (see Chapter 5). Notice that divisions

ORDER	ARGUMENT
F5	$n \times 2^{-39}$
40	$(n+1) \times 2^{-39}$
50	$(n+1) \times 2^{-39}$
75	.64
00	30
40	x/8
50	x/8
7J	x/8
L4	1/64
40	$1/64 + x^2/64$
L5	x/8
L4	-1
50	
26	to S2
40	$e^{-1 + x/8}$

ORDER	ARGUMENT
50	$e^{-1 + x/8}$
7J	$e \times 2^{-2}$
40	$e^{2/8}/4$
50	$e^{x/8}/4$
75	$e^{x/8}/4$
00	1
40	$e^{x/4}/8$
50	$e^{x/4}/8$
75	$e^{x/4}/8$
00	1
40	$e^{x/2}/32$
50	$e^{x/2}/32$
75	$e^{x/2}/32$
66	$1/64 + x^2/64$
NO	y/16

Table 6.5
Calculation of $e^x/(1 + x^2)$

are always made using a full 78 digit dividend to retain as much accuracy as possible. The program treats the two cases (a) and (b) separately, distinguishing with the 36 order at 1L. Location 0 in the memory are used as temporary storage. The program is given in Table 6.6 on pages 6-11 and 6-12.

6.4 ADJUSTABLE SCALING FACTORS. It is not always possible to arrange a program so that a single scaling factor can be used throughout the calculation. Then it is necessary to make tests at appropriate places to discover when variables are becoming too large or too small and to make proper adjustments in the scaling factors. For many problems it is advantageous to have the variable less than 1/2. Then two numbers can be added or multiplied without exceeding capacity. By using the LL n order we place $1/2 + |(n)|$ in A. Hence A is positive if $|(n)| < 1/2$ and A is negative if $|(n)| \geq 1/2$.

6.5 CONTINUOUS SCALING. FLOATING POINT ROUTINES.

For calculations in which continual tests are required to maintain accuracy floating point routines (see also Section 4.7) may be used. These routines represent numbers as $y = a \times 10^b$ and store a and b. Thus they can represent accurately the numbers in some large range such as, for example, $10^{-64} \leq |y| < 10^{64}$ where y has 30 significant binary digits. There are three such routines in the Program Library. The first, Library Routine A1 is as described above. The second, Library Routine A4 is a multiple precision floating point program in which numbers lie in the range $10^{-153} < |y| < 10^{153}$ with y having

20 significant decimal places. The third floating point scheme, embodied in Library Routine A7, is designed so that it provides many facilities provided by A1, but is somewhat faster. With this routine, it is somewhat easier to carry out instructions in normal machine code than with A1, and its use is recommended in cases where a balance must be struck between the ease of coding and speed of final operation.

Floating point routines are slow because numbers are scaled at each step of the calculation. Certain conveniences have been programmed in, however, and these to some extent compensate for the extra time required and also simplify the programming.

For small ad hoc calculations, A1 has been combined with a number of convenient auxiliary routines as Library Routine A9. This simplified coding scheme, which is described in Chapter 11, is very easy to learn, and can be used for small arithmetical calculations by users with no knowledge of the basic SILLIAC code.

0	L7	21L	|a| - |d|
	L2	24L	
1	32	12L	
	50	26L	|a| < |d|; f to Q
2	75	21L	
	66	24L	fa/d
3	-1	F	
	L4	23L	
4	40	F	-fa/d + c to 0
	50	21L	
5	75	25L	
	66	24L	ae/d
6	-1	F	
	L4	21L	
7	40	1F	-ae/d + b to 1
	L5	F	
8	66	1F	(-fa/d + c)/(-ae/d + b)
			= -y
	-1	F	
9	40	28L	y to 29L
	50	28L	
10	75	25L	
	L4	26L	(ey + f)
11	66	24L	-(ey + f)/d = x
	NO	27L	x to 28L
12	OF	F	Waste Order
	50	22L	|a| ≥ |d|; b to Q
13	75	24L	
	66	21L	bd/a

cont..

Table 6.6

Solution of Two Simultaneous Equations

```
14   -5   F
     LO   25L
15   40   F      bd/a - e to O
     50   24L
16   75   23L
     66   21L    dc/a
17   -1   F
     L4   26L
18   66   F      (-dc/a + f)/(bd/a - e) = y
     NO   28L    y to 29L
19   75   22L    by
     L4   23L
20   66   21L    (by +c)/a = -x
     22   11L    Control to 11L
21                    a
22                    b
23                    c
24                    d
25                    e
26                    f
27                    x
28                    y
```

Table 6.6 (continued)

Solution of Two Simultaneous Equations

CHAPTER 7

MACHINE METHODS AND CODING TRICKS

There are usually a number of special techniques which can be used on any particular digital computer and which will simplify programming. Some of these techniques are applicable on many different computers, but usually, as is the case in those which follow, they result from particular orders or combinations of orders which are peculiar to an individual machine. This chapter is concerned with a number of unrelated sections having to do with operations which frequently arise in programming.

In general, unless space is at a premium, it is better to resist the temptation to use unusual combinations of orders. Anyone who has to alter a program not his own will realise the importance of this remark: discovering the purpose of someone else's program, especially when only an inadequate write-up is available, can be a time-consuming business. However, the order combinations given in this chapter are sufficiently well known to be regarded as standard coding conventions.

7.1 THE SUMMATION OF PRODUCTS. We often need to form sums of products, and on the SILLIAC this cannot be directly done in the accumulator. The accuracy can often be enhanced by performing a summation either exactly or with only one round-off. This is comparatively easy to do using the 74 order. All that we need to do is to place the least significant half of the partially summed products into the accumulator before performing the 74 order. In fact, this can usually be done by an -5 order because the quotient register will usually hold the last single half of a summed product. Then, since 74 gives $(n)(Q) + 2^{-39}(A)$,

we obtain the double-length product in AQ. Of course, the most significant part of the previously summed products needs to be added by means of an L4 order.

Using similar schemes we can also arrange to add or subtract products with double-length accuracy in a program. As a first example we shall sum 50 double-length products, assuming we do not exceed capacity.

Example 1: Place the rounded sum

$$\sum_{i=0}^{49} (100 + i) \times (150 + i)$$

in location 0. The program is given in Table 7.1.

```
m      41   F      Clear location 0 for sum.
       39   F      Waste order.
m+1    50   8L     Put 1/2 in Q.
       L5   11L    Set i = 0.
m+2    40   3L
       -5   F      Round off (see Section 7.2).
m+3    50   ()F
       74   ()F    (100 + i) x (150 + i).
m+4    L4   F
       40   F
m+5    L5   9L     Increase i by 1.
       L4   3L
m+6    40   3L
       L0   10L    Test for i = 150.
m+7    32   2L     Re-enter loop.
       0F   F      Stop.
m+8    40   F      Roundoff constant = 1/2.
       00   F
m+9    00   1F     Increment
```

```
      00   1F
m+10  J0   150F   End of constant
      74   200F
m+11  50   100F   Starting constant
      74   150F
```

Table 7.1

Program for Example 1

7.2 REVERSING THE CONTROL TRANSFER. FURTHER DISCUSSION OF EXAMPLE 1.

There is a second coding trick in Example 1. The end constant, instead of being 50 150F 74 200F has had -1 added to it, making the first order J0 150F. The effect is to reverse the sense of the following 32 order (in location m + 7) so that we transfer control to re-enter the repetitive loop. If this had not been done a 22 order following the 32 order would have been required, and the 32 would have transferred to OFF. Thus, a half word was saved. This technique is equivalent to having a conditional transfer order act when the accumulator is negative.

Another coding trick might have been used to save a full word. The left-hand order at m + 1 puts 1/2 in Q so that it may be used to round off on the first step of the sum, this being the sole roundoff. Instead of storing 1/2 in m + 8 we could have used the order pair in m + 2 as the roundoff constant. This order pair is 1/2 plus at most 2^{-9} and would serve quite well.

As a second example we consider a summation of two products with a single roundoff.

Example 2: Given

cos θ in 10,

sin θ in 11,

x in 12,

y in 13,

place the rounded quantity (x cos θ + y sin θ) in location 20 and the rounded quantity (-x sin θ + y cos θ) in location 21. The program is given in Table 7.2.

m	50	10F	x cos θ + 2^{-40}
	7J	12F	
m+1	40	F	Most significant half to location 0.
	-5	F	Least significant half to A.
m+2	50	11F	y sin θ + 2^{-39} (l.s. half of x cos θ + 2^{-40}).
	74	13F	
m+3	L4	F	Add most significant half of x cos θ + 2^{-40}.
	40	20F	Store x cos θ + y sin θ + 2^{-40}.
m+4	50	11F	-x sin θ + 2^{-40}.
	79	12F	
m+5	40	F	
	-5	F	
m+6	50	10F	y cos θ + 2^{-39} (l.s half of -x cos θ + 2^{-40})
	74	13F	
m+7	L4	F	
	40	21F	Store -x sin θ + y cos θ + 2^{-40}.

Table 7.2

Program for Example 2.

7.3 BINARY SWITCHES. It is sometimes necessary to do two different operations alternately. This can be done by changing the sign of a number each time we pass it so that it will be alternately positive and negative. Usually it is not necessary to use a special number for this because some number or order pair in the rest of the program may be used. To accomplish the switch an L1 order

followed by a 40 order is used to change the sign of the number and put it back in its location with sign changed. A conditional transfer order may then be used to decide which of two sequences will be performed.

A variation is the requirement that an order (or order pair) take on two different values alternately. This can be accomplished by using the identities

$$b = (b + a) - a,$$

and $$a = (b + a) - b.$$

Thus, if the current value of an order (or order pair) is subtracted from the sum the other value is obtained.

Example 3: Arrange a program to alter the address of the left-hand order at (m+2) so that it takes on the values 0 and 5 alternately. A program for this is given in Table 7.3.

```
m     any order
      L5  pF        Put sum of orders in A
m+1   L0  2L        Form alternate address
      40  2L        Store alternate address at (m+2)
m+2   L5 (0)F
      L4  12F       Normal program order
 p    F+  6F        Sum of L5  0F  L4  12F
      F8  24F       and    L5  5F  L4  12F
```

Table 7.3
Binary Switch

In Table 7.3, only a single address is taking on alternate values and it is also possible to carry out the switch using the program of Table 7.4. In Table 7.4 the 50 5F order at location m is provided for the switch. If some order which needed address 5F could be used here, we should be very well-off indeed.

```
m      50   5F
       L5   mF
m+1    L0   (m+2)F
       46   (m+2)F
m+2    L5   (0)F
       L4   (12)F
```

Table 7.4
Binary Switch

7.4 TESTS FOR 0 AND -1. In order to test for a particular number value held inside the machine it is generally necessary to use two tests. However, the numbers 0 and -1 can be tested for using absolute value orders and a single test. In machine language 0 is the only number whose negative absolute value is positive, and -1 is the only number whose positive absolute value is negative. Thus we can test for 0 using an L3 order followed by a conditional transfer order, and we can test for -1 by using an L7 order. Similarly, we can test for -2^{-39} or $1-2^{-39}$ using F3 and F7 orders, respectively.

Example 4: Transfer control to location 200 if A is zero but transfer control to location 300 if A is non-zero. Two ways to do this are given in Tables 7.5 and 7.6. The program of Table 7.5 has only two words. The program of Table 7.6 has four words but will be faster than the other if A is usually negative. Moreover, the program of Table 7.6 can be used to transfer control to any of three locations depending upon whether A is positive, negative, or zero.

m	40	F	(A) to 0
	L3	F	- |(0)| to A
m+1	36	200F	To 200 if -|(A)| $\geq$ 0, i.e., if (A)= 0
	26	300F	To 300 if (A) $\neq$ 0

Table 7.5

Testing for Zero

m	36	1L	
	26	300F	To 300 if (A) < 0
m+1	L0	pF	(A) - 2^{-39}
	36	300F	To 300 if (A) - 2^{-39} $\geq$ 0, i.e., if (A) > 0
m+2	26	200F	Tc 200 otherwise, i.e., if (A)= 0
p	00	F	Constant 2^{-39}
	00	1F	

Table 7.6

Testing for Sign

7.5 USE OF ORDERS AND ADDRESSES AS CONSTANTS.-V,+V,39 and 3J orders do not use their addresses, so these addresses can often be used for other purposes. For instance, they may be used to store a starting address taken by a cycling order or an increment which is used to change an address. In such cases we naturally use 42 or 46 orders to make certain that the function digits do not become altered.

In the following example the address of a +5 order is used as a counter.

Example 5: Given the positive number a in A, write a closed subroutine with old style entry which will furnish the positive integer m such that $1/2 \leq 2^m a < 1$. The program

is given in Table 7.7.

```
          m     40  F     Store a at location 0
                +5  F     Form link
          m+1   42  4L    Plant link
                43  L     Clear counter
          m+2   L5  F  ↰
                00  1F    Shift a
          m+3   40  F
                36  5L    Test to see if 2^p a < 1
          m+4   L5  L     Counter to A
                22  ()F    Link
          m+5   F5  L     Count
                40  L
          m+6   26  2L ↱  Re-enter loop
                00  F     Waste order
```

Table 7.7

Address Use in +5 Order

7.6 RESETTING AND STARTING OF CYCLES OF ORDERS. In many cases we have cycles of orders, some of which are being modified by the same increment. In such cases, if the indexing order is not used, the variable addresses can all be modified by modifying one order and then deriving the other orders (or addresses) by adding the constant difference between the variable orders. When this is done it is economical to use the same orders to set these addresses when the cycle is begun. The following example with the program given in Table 7.8 illustrates this:

Example 6: For i = 0, 1,....., 99 place in location 200 + i the sum (10) + (100 + i) + (200 + i).

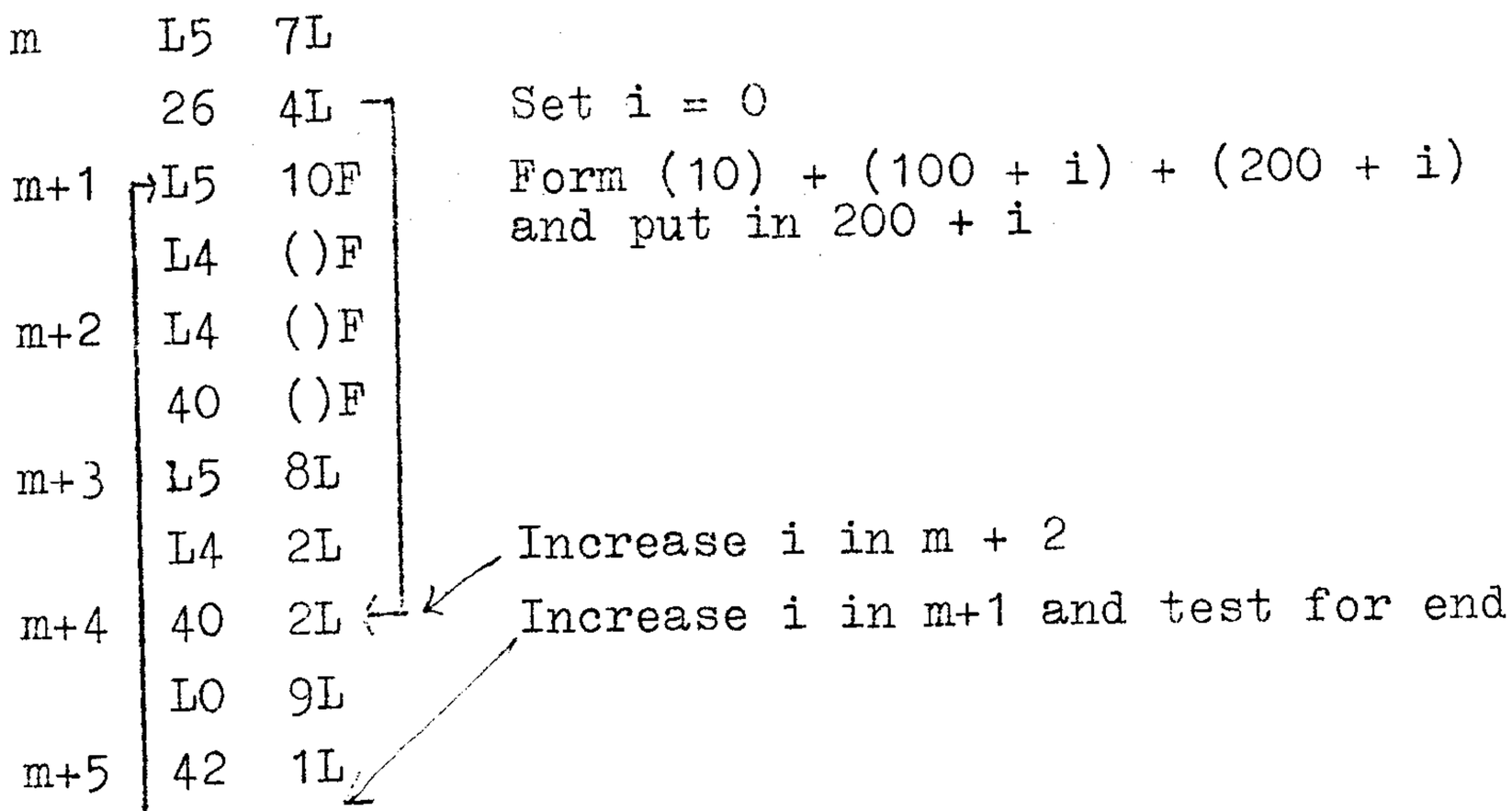


Table 7.8

Resetting of Addresses

In Example 6 the addresses in location m+2 have been changed and then that in m+1 has been obtained from one of them. Notice that the end test has been combined with the second address change and that the end test constant is not L4 but 74 so that the 36 order at location m+5 will cause re-entry to the repetitive loop.

When two orders have to be varied in a single cycle, it

is advantageous to let these form a single order pair as in Example 6 so that they can both be modified simultaneously by the same orders. This arrangement is not always possible and the second best arrangement is to place the variable orders on the same side of their respective order pairs so that the orders required to modify them will be as simple as possible. Of course, the use of the indexing order makes such considerations unnecessary.

For very simple operations it is sometimes advantageous to do three or four operations in a single cycle. This saves time although the advantage is bought at the expense of more orders. This is illustrated in Example 7.

Example 7: Add the numbers in memory locations 10 to 14, putting the sum in location 15. It is simpler and faster, both in coding and in machine operation, to use the program given below than to write a repetitive code which counts.

```
L5  10F
L4  11F
L4  12F
L4  13F
L4  14F
40  15F
```

7.7 USE OF THE QUOTIENT REGISTER FOR INTERCHANGES.

In many programs we wish to replace the number in a certain storage location and yet use the value that is there to continue with the calculation. In such cases the old value can be placed in the quotient register before the new value replaces it in the memory. Thus, the old value is available in the quotient register for further computation.

Example 8: Store A in location 10, but use the old

(10) as a dividend to form (10)/(11). The program is then

```
50    10F
40    10F
-5    F
66    11F
```

7.8 TESTING IF NUMBERS ARE GREATER THAN ONE-HALF

When scaling numbers it is very often necessary to test when numbers are larger in magnitude than one-half. This can easily be done with the appropriate L or - order. For example, the order LL n will cause the accumulator to be negative if the magnitude of (n) is greater than or equal to 1/2.

Example 9: If $|(10)| \geq 1/2$ replace it by half its value. The program is given in Table 7.9.

m	LL	10F	1/2 + $\|(10)\|$ to A
	32	(m+2)F	
m+1	L5	10F	
	10	1F	
m+2	40	10F	

Table 7.9
Scaling by Testing for One-Half

7.9 CONVERGENCE CRITERIA

When iterations or repetitive calculations are carried out we frequently want to stop when we have achieved the maximum accuracy. In some cases it is difficult to specify in advance the tolerances which can be used as end criteria because we have to compromise between achieving the greatest accuracy and yet assuring that we terminate the processes (i.e., don't loop).

In such cases it is worthwhile to use more complicated criteria which will give us maximum accuracy but which will not loop. One such criterion is to terminate the process if $e_n \leq e_{n+1}$ where e_n is some positive number which tends to zero as the process converges with increasing n. This criterion will terminate the process only when either e_n is the same for two successive iterations or when the roundoff error has actually caused it to increase.

7.10 MARKING It is often possible to use marking techniques instead of the more usual counting processes. The simplest illustration of such a technique is Library Routine N3 where a sequence of numbers read from the tape is automatically terminated by the mark N. In this code instead of counting up to some predetermined number we test each character as it is read from the tape until it is N.

A binary digit is sometimes shifted as a marker; this is illustrated in the next example.

Example 10: Using a print routine stored at locations beginning with 50 print the 7 numbers in locations 10-16 The program is given in Table 7.10:

0 19 7F 2^{-8} to 1F. This is the marker

```
        40    1F
1       L5    1F
        00    1F      Advance marker by shifting
2       40    1F
        32    3L      Test for end
3       0F    F       Stop
        L5    (10)F
4       39    F       Waste Order
        50    4L
```

```
5    26    50F    Enter print routine
     F5    3L
6    40    3L     Increase address of number
                      to be printed
     26    1L
```

Table 7.10
Use of a Marker

In the program of Table 7.10 the marker is shifted into the sign digit to indicate the end of the repetitive process. This technique can be used in a similar way for completely internal programs. For example, when we are dealing with a group of numbers in the memory we may arrange that the storage location following the group contains some unique number such as 0 or -1. Then the code merely has to test for the presence of one of these numbers rather than for a predetermined count.

7.11 REMAINDER IN INTEGER DIVISION In the general case it is difficult to compute the remainder from the residue that is left in the accumulator after a division. However, if we are dealing with positive integers less than 2^{38} in magnitude we can do this quite readily. We place twice the dividend integer in the quotient register, clear the accumulator and divide by the divisor integer. The accumulator then contains twice the integer remainder and the quotient contains twice the integer quotient. We store an integer y as $y \times 2^{-39}$.

Example 11: Divide the positive integer in location 20 by 10. Place the quotient in location 11 and the remainder in location 12. The program is given in Table 7.11.

m	51	20F	$y \times 2^{-39}$ to Q
	00	1F	$y \times 2^{-38}$ to AQ
m+1	66	4L	$(y \times 2^{-38})/(10 \times 2^{-39}) \times 2^{-1} = y/10$
	10	1F	
m+2	40	12F	Store remainder
	NO	11F	Store quotient
m+3	26	pF	
	00	F	Waste order
m+4	00	F	
	00	10F	

Table 7.11
Remainder in Integer Division

7.12 BINARY CHOPPING This is the method of repeated subdivision of an interval. It is easy to code although it may be wasteful of memory space and it is slow because it will usually take the full 39 steps to go from an interval of length unity down to one of length 2^{-39}.

If we use binary chopping to find the zero of a function, we proceed as follows. We choose bounds for the zero, perhaps -1 as a lower bound and +1 as an upper bound. Then we bisect the interval and compute the function at the midpoint. Depending upon whether the sign of the function at the midpoint agrees with the sign of the lower or upper bound, we substitute the midpoint for the appropriate bound. After 39 steps the difference between the two bounds will be 2^{-39} and the zero will be determined.

The code will be simpler if the signs of the upper and lower bounds are known so that a comparison with the sign of the midpoint is not needed at each step. This is

the case in Example 12.

Example 12. Find the square root of a = 0.26943 by using a binary chopping technique on the function $a_n^2 - a$.

Here we choose initial upper and lower bounds (written $\bar{a}$ and $\underline{a}$) of $1 - 2^{-39}$ and 0. Instead of counting 39 steps, a test has been included so that if $a_n^2 - a < 2^{-37}$ the code stops. The program is given in Table 7.12. on page 19.

7.13 EVALUATION OF POLYNOMIALS Polynomials are best evaluated by use of a recurrence relation. Given the polynomial

$$f(x) = \sum_{i=0}^{n} a_i x^{n-i} = a_0 x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n,$$

we can express it in the form

$$S_{i+1} = x S_i + a_i,$$
$$S_0 = 0,$$
$$S_{n+1} = f(x).$$

See Table 7.13 on Page 19.

If, as is often the case, the coefficients a_i are the quantities (m+1) the recurrence relation becomes

$$S_{i+1} = x S_i + (m+i),$$
$$S_0 = 0,$$
$$S_{n+1} = f(x).$$

Example 13 shows how a polynomial may be evaluated. In practice the summation of products should be done with a 74 order as in Example 1, but we do not wish to obscure the general idea here.

Example 13: Given x in location 49 and coefficients in locations 50 through 67 evaluate the polynomial

$$\sum_{i=0}^{17} x^{17-i} (50 + i).$$

The program is given in Table 7.13 on Page 20.

7.14 PRINTING OF SIGNS IN PRINT ROUTINES When the sign of a number is to be printed, this can be done in one less order than by the straightforward method by the use of logical shift orders. Thus, if we wish to output the sign of the number in location a, then depending on whether we wish to begin with a left-or right-hand order, we may use one of the methods:

```
        m
                50 a

        m+1     L5 m+1
                04 37F

        m+2     82 1F
```

```
or      m       50 a
                F1 m+1

        m+1     04 37F
                82 2F
```

Table 7.14
Output of a Sign with Logical Shift Orders

Alternatively, if we have a constant available where right-hand address differs from 5 by a multiple of 8 and b is the address of this constant, a slightly quicker method is as follows:

```
m     L5 a
      50 b
m+1   16 3F
      82 1F
```

Here we may start with either a right-or left-hand order.

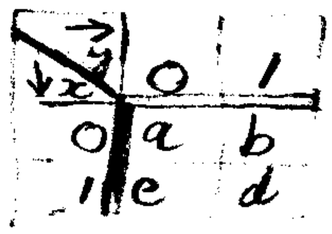
7.15 LOGICAL OPERATIONS ON SILLIAC The operations we consider replace any digit by either 0 or 1 depending only on the values of the corresponding digits of two numbers, say X and Y. Suppose we have a scheme as follows:

Digit of X	Digit of Y	Digit in result of operation
0	0	a
0	1	b
1	0	c
1	1	d

where a, b, c, d are given digits (0 or 1).

The four numbers, a, b, c, d, characterize the operation. We shall write them in an array as follows:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$



E.g.

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

is collation as performed in SILLIAC using a J0 order.

To form the complement of a number we use an F1 order.

All other cases to be considered will depend on both X and Y. It will be assumed that X and Y are stored in

locations x and y respectively and that the result is to appear in the A register. The sets of instructions are of course not unique.

Operation	Definition	Orders	Operation	Definition	Orders
X.Y (and)	0 0 0 1	50 x J0 y -5	X↓Y (Pierce)	1 0 0 0	50 x L1 y +0 J0 y -4
X∨Y (or)	0 1 1 1	50 x L5 y -4 J0 y -0	X=Y (equals)	1 0 0 1	50 x L1 y +0 J0 -4 -4
X•Y, X≠Y (not equi- valent)	0 1 1 0	50 x L5 y -4 J0 y -0 -0	X.$\overline{Y}$	0 0 1 0	50 x -5 J0 y -0
X|Y (Sheffer stroke)	1 1 1 0	50 x J0 y +1	X<Y (implies)	1 1 0 1	50 x +1 J0 y -4

Table 7.15

Various Logical Operations

0	L5	12L	
	10	1F	
1	40	F	$\bar{a}/2$ to location 0
	L5	13L	
2	10	1F	$(\bar{a} + \underline{a}) / 2 = a_n$ to 14L
	L4	F	
3	40	14L	
	50	14L	
4	7J	14L	
	L0	11L	
5	40	15L	$a_n^2 - a$ to 15L
	32	7L	
6	L5	14L	
	40	13L	Change $\underline{a}$
7	22	8L	
	L5	14L	
8	40	12L	Change $\bar{a}$
	L7	15L	
9	L0	16L	Test for $a_n^2 - a < 2^{-37}$
	36	L	
10	0F	F	Stop
	00	F	
11	00	F	$a = 0.26943$
	00	26943000000J	
12	7L	4095F	$\bar{a} = 1 - 2^{-39}$
	LL	4095F	
13	00	F	$\underline{a} = 0$
	00	F	
14	00	F	a_n
	00	F	
15	00	F	$a_n^2 - a$
	00	F	
16	00	F	2^{-37}
	00	4F	

Table 7.12
Square Root by Binary Chopping

p	41	F	$S_0 = 0$
	50	49F	x
p+1	7J	F	x S_i + N(50 + 1)
	L4	(50)F	
p+2	40	F	
	F5	1L	Increase i
p+3	40	1L	
	L0	5 L	Test for end
p+4	32	L	
	OF	F	
p+5	LJ	F	End test constant
	L4	68F	

Table 7.13

Evaluation of a Polynomial

CHAPTER 8
CODE CHECKING METHODS

Code checking is an art, and can be learned only by practice. Nevertheless, there are certain rules, most of them commonsense, which greatly help both yourself and other people. The purpose of this chapter is to present some of this advice, in the hope that it will aid users of the SILLIAC in getting their programs working rapidly and with a minimum of headaches.

8.1 WHAT TO DO BEFORE STARTING CODING. Observation of a number of points in the laying out of a program will considerably increase the ease with which it can be put into service.

The first and most important of these rules is that one should not start coding until a complete flow digram (i.e. a diagram setting out the course of the whole calculation) has been made, setting out the whole of the problem. There is a temptation to leave some thorny sections of the work until later, and to get on with coding the easy bits. This temptation should be stoutly resisted because, inevitably, it will be found that the sections of the flow diagram which were left unfinished will considerably affect the approach to the problem, and will probably invalidate much of the coding based on the incomplete flow diagram. Moreover, the flow diagram should be sufficiently complete to allow annotations of the order list to be identified with corresponding sections of the diagram.

In preparing the flow diagram, thought should be given to the way in which the program is to be checked. For example, wherever possible, printout routines should be

chosen, or constructed, in such a way that they can be used with sections of the program during testing.

The ideal way to assemble a program is to test individual sections as fully as possible and then to combine these sections into a complete program for final testing. Certainly all closed sub-routines written by the programmer should be tested individually before their inclusion into the full program. This ideal can be approached with very little effort if arrangements are made which will call for greater amounts of printout during the testing of a program than will finally be required. This can be done either by modifying the program to use its own printout routines or by use of suitable diagnostic routines, as is described below.

8.2 WHAT TO DO WHILE CODING. Individual coding lists should be accompanied by sufficient explanation to make their interpretation as easy as possible. Measures which help here are:

(a) drawing of "transfer of control" lines and "switch" symbols, preferably in red pencil;

(b) adding descriptions of groups of orders, which correspond to the descriptions on the flow diagram;

(c) writing beside "switch" orders, constants and pseudo-orders, locations of any other orders, constants and pseudo-orders with which they are associated. (This last is particularly important when alterations are being made to a routine in the process of error eradication.)

When a routine has been written, it should be read through and checked immediately. This is not a very good

check, because of inevitable blind spots which the coder develops towards his own work. A better check, if this is possible, is to put the routine away for a few days or weeks, and then to go over it again. A substitute for this is for somebody (not a member of the SILLIAC Laboratory staff!) who can be depended upon not to say "Yes" out of mere politeness, to go over the routine with the originater. When punching programs up, sufficient delay characters should always be left for splicing (see next chapter) at the end of each group of 10 or 20 words, between subroutines and after directives and lists of parameters; these also help in locating positions on the tape.

It is important to avoid having to think while at the key-board, and for this reason, it is best that all information, including directives, N orders etc. should be written out in detail beforehand.

A program of any length should always be checked by the programmer himself, by reading out the manuscript to somebody who will check it against a printed version produced from the tape. This minimises the possibility of errors introduced by "wishful thinking" in reading someone else's handwriting.

In arranging the lay-out in the store, it is desirable not to use storage locations from 900 onwards (or even from 800 onwards if this is possible). This will allow Diagnostic Routines to be used to fullest effect. Also, if sufficient storage space is available, the working space used by individual routines should be kept grouped: the usual convention is to use locations which are associated with the positions of routines themselves, or are at the beginning of the store (do not use storage locations 0, 1, or 2, the contents of which are not available to Post Mortem routines input with a bootstrap start).

If a D.O.I. other than X13 is being used, at the end of each routine, a group of 10 or so storage locations should be left for the insertion of corrections associated with the routine. If the saving of storage space is important, it may be necessary to rewrite sections of routines once a complete program is working; however, it is better strategically to get the program as a whole working first and then to concentrate on "polishing it up" as a subsequent operation, than to spend too much time on "polishing" before the program as a whole is working.

The tests to be carried out on individual sections of a program and on a completely assembled program should be devised so that they are as searching as possible but are at the same time so simple that they can be checked easily by hand. For example, if the program evaluates a polynomial with specified coefficients, for code checking purposes, all but two or three coefficients can be made equal to zero, and the independent variable x chosen to be some simple number, such as 1/2. If the code solves a differential equation coefficients can be picked so that the equation has a tabulated solution. If the program is to analyse experimental data, pick some particularly simple fake "data". (This is preferable, at least in the early stages of code checking, to picking data which have already been analyzed by hand. The hand computation is often found to be wrong, or of insufficient numerical accuracy for code checking, or both.)

All test runs should be prepared with provision for adequate printout at key points of the program. If adequate printout arrangements are not available as part of the program, they can be provided by use of D1 (or D7) or D15-

-see next section. Always insert a 20 1019N order on the tape

immediately before the order which starts the program to enable read-in of modifications and of D1(or D7) or D15.

Standard forms are available for coding: these are pre-printed with numbers 0 - 9. Although it is not essential to use them, failure to do so often results in simple mistakes in numbering orders.

A number of coding blunders occur frequently The more typical blunders are listed in Table 8.1. It is unfortunately true that the more trivial ones seem to occur most frequently.

At this point, we shall examine the way in which information about the progress of a program can be obtained when it is presented to the machine.

8.3 SOURCES OF INFORMATION. The Punch. The punch is the most effective way by which data can be extracted from the SILLIAC. However, there are other ways to obtain information, particularly when a program does not run far enough to punch any data at all.

The Reader. If the input tape stops before the entire program has been read into the memory, then an examination of the characters punched on the tape just ahead of the place where it stopped will often provide an explanation.

The Order Counter and Control Counter. If the program is read in correctly but comes to a sudden unexpected stop, then the order register will exhibit the order on which the program stopped while

(1) L5 orders used instead of L4 orders.

(2) Orders terminated by L instead of F and vice-versa.

(3) The renumbering of a code not complete after a modification has been made.

(4) Simple mistakes in numbering when standard coding pads are not used.

(5) Rounded multiplication used when dealing with integers.

(6) Control transfers to the wrong address or wrong order of an order pair.

(7) Accumulating storage registers not cleared before a cycle of orders is entered.

(8) The end condition for a cycle of orders not correct.

(9) Allowing the temporary storage of a subroutine to erase useful data.

(10) Using a 46 order instead of a 42 order and vice-versa.

(11) Omitting directives and starting orders on the program tape.

(12) Incorrectly remembering the specifications of a subroutine.

(13) Forgetting to reset addresses when coming back to a cycle of orders.

(14) Making corrections incorrectly.

(15) Using the same relative addresses on correction words although the preceding directive is different from that of the program.

(16) Overlooking the digits shifted from the quotient register to the accumulator on a left shift.

(17) Attempting to convert fractions greater than one-half by using the J terminating symbol.

Table 8.1

Typical Blunders

the order counter will contain a number one greater than the storage location from which the order pair came. With this information the programmer can often discover the cause of failure of his program.

If the SILLIAC loops, observation of the slave or monitor tube will give an indication of how extensive the loop is. Then, if the machine is stopped and the contents of the order register and the control counter noted, we usually have enough data to identify the loop in our program. The slave tube can also be used to ensure that orders are being input into approximately the right locations, thus enabling inadvertent overwriting to be detected.

Printout of specific items of information can be obtained by the use of one of the techniques described in the following sections--either by specific modification to the program, or by the use of post-mortem or diagnostic routines.

8.4 BLOCKING ORDERS. This is the name given to control transfer orders which are used to replace normal program orders, so that some printing or checking can be done at the point of replacement. Before control is restored to the program the replaced order of the program is executed and the contents of the arithmetic registers are restored. Thus the original program is unaffected, but the extra orders that are obeyed can be utilized to do printing of desired data.

An example may illustrate this: Let us suppose that

(a) we have a program in which we wish to
print (19) after the left-hand order
of the order pair 40 9F I5 29F at storage

location 100, has been obeyed,

(b) we wish to preserve the quotient register but not the temporary storage of the print subroutine,

(c) locations 800-804 are unused by the program,

(d) the program print routine starts at 200,

(e) the original program ends with 24 999N 26nN.

We then prepare the following modification tape:

MODIFICATION TAPE

```
00  100+        Directive

40  9F          Plants blocking order in program
26  800F

00  800+        Directive

NO  4L          Stores Q at 804F
L5  19F

22  1L          Enters print subroutine
50  1L

26  200F
L5  29F         Does ommitted order

50  4L          Restores Q
26  101F        Control back to program

26  nN          Starts program
```

It might perhaps be noted here that control can at any stage of a calculation be transferred to any point required by bootstrapping in the order pair 2DXYZ00000 where XYZ is the required destination of control in sexads. and D is 2 or 6 according as control is to be transferred to a right- or left-hand order. This procedure will overwrite the contents of location 0.

8.6 TYPES OF CHECKING ROUTINES. There are several types of checking routines. One type prints out the contents of certain memory locations after a program has stopped. We call this a post mortem routine.

A second type takes a given program and allows it to be obeyed order-by-order while printing out information about the course of the program. It is often called a sequence checking code.

A third type arranges for information to be printed out at specified points of a program. It is usually known as a check point or blocking order routine. The second and third types are known as diagnostic routines.

8.7 POST MORTEM ROUTINES Library Routines C3, C4, C5, C9. These are printing routines and have been arranged so that they can be used with little or no preparation on the part of the programmer. They are read into the memory with bootstrap input routines and are located in storage locations at the end of the memory, the longest occupying locations from 986 to 1023 and using locations 0, 1, and 2, as temporary storage. The end of each of the post mortem tapes contains 100 two-decimal-digit numbers. These numbers are used to specify the locations from which printing will occur. Suppose, for example, that we wish to know the order pairs in memory locations 540 to 549. Then we read in Routine C5. When it stops we find the number 54 on the end of the tape and place it in the reader. When the START switch is moved, the order pairs in locations 540 to 549 will be printed and then the SILLIAC will stop. If we start again we will get the order pairs in locations 550 to 559. Codes C3 and C4 perform similar functions for decimal fractions and decimal integers.

Note: There is no section numbered 8.5.

The programmer should keep in mind the storage locations used by the post mortem routines so that he will be able to take full advantage of these important checking routines.

8.8 POST MORTEM VERSION OF DECIMAL ORDER INPUT ROUTINES. Library Routines C1 and C12. These are very important checking routines and are usually the first ones used after a program failure. C1 is used when X1 has been used as a D.O.I., C12 when X12 has been used as a D.O.I. They compare the contents of the memory with the contents of the input tape. Only discrepancies are printed out, enabling programmers to discover which orders of a program have become altered while the program was in the memory. This is an aid in making sure that orders which should have been modified have been modified correctly and that no order has become modified accidentally.

C1 and C12 are used in the following manner. The Post Mortem Version of the Decimal Order Input is read into the memory in the usual way with a bootstrap input routine. (C1 occupies storage locations 962 to 1023, and C12 occupies locations 986 to 1023.) The program tape is then placed in the reader after its copy of the Decimal Order Input, so that the Decimal Order Input is not read into the SILLIAC. Then when the machine is started the program tape will be read and the words created from it compared with the corresponding ones in the memory. When a discrepancy is found it will be printed on a line giving, first, the location at which the discrepancy was found, second, the word read from the tape printed as an order pair and, third, the word found in the memory, again printed as an order pair. Thus a

typical line of printing might be

345 L5 000 40 354 L5 546 40 354

indicating that the left-hand address of the order pair in location 345 had become modified, taking the value 546.

It should be pointed out that closed subroutines which have been used will usually have their links printed. This gives an indication of the part of the program from which they were last called in. Interludes cause a large amount of printing because the contents of the interlude locations are changed twice in the course of input and are printed out both times.

Post Mortem Versions of the Decimal Order Input routines cause the memory to be changed to the original states as they are being executed, so that if the program is started the original program will be performed again.

If desired, only selected parts of the tape need be compared with the contents of the memory. However, when doing this it is necessary that all the preset parameters pertaining to that part of the program be input and that the selected part of the program begin with a directive.

All post mortem codes occupy locations at the end of the memory and all of them use storage locations 0, 1, 2, as temporary storage, so that it is desirable when coding not to use these storages for constants or numbers which may need to be printed out. It is worthwhile to note that if addresses which are to be changed are initially read in with their final values then no printing will take place on the post mortem if they have been modified correctly.

8.9. THE ADDRESS SEARCH ROUTINE. Library Routine C_6.
Programs sometimes fail because of a transfer of control to an order which causes the machine to stop. The usual order causing the stop is a zero left shift order because the memory is normally cleared to zeros before a new program is read in.

Under such circumstances the offending transfer of control order may be searched for with the aid of Library Routine C2. The search routine is read into the memory, occupying storage locations normally occupied by the Decimal Order Input. Next the address to be searched for is read into the machine as a three character sexadecimal address. The routine then searches the memory (exclusive of itself) for order pairs containing this address. When found they are printed out (in sexadecimal form) together with their location (in sexadecimal form).

The routine naturally has other similar uses. For example, if it is known that some number becomes modified but it is not known why, then the store order which does the damage can be sought for in the above manner.

8.10 INTERPRETIVE ROUTINE POST MORTEMS.C10 and C11.
These routines, when bootstrapped into the machine, together with a parameter giving the location of the first order of the interpreting routines with which they are working will give the last interpretive order obeyed, its address, and the contents of the floating point accumulator.
C10 is used with A1 or A8 (the single length floating point scheme) and C11 is used with A4 (the 1.7 precision floating point scheme).

8.11 CONTROL TRANSFER CHECK. Library Routine D15.

This routine takes charge of a program and allows it to be obeyed order by order. Each transfer of control that is obeyed is placed in a list kept in a specified place. The list is cyclic, that is to say, the later entries overwrite the earlier ones in a cyclic fashion. At the end of a program, the list can be printed, so that it can be discovered how the program reached its final end. There is no printing during the execution of the program, so that this routine allows the program to be obeyed at a speed much greater than of similar routines which print. It is possible to suppress storage of successful transfers of control over a section of the store. This facility makes it possible to place all tested routines in one section of the store, so that transfers of control in the routine under test only are stored.

8.12 THE CHECK POINT ROUTINE. Library Routines D1, D7.

These routines are designed to print out intermediate information about some other program. They use the blocking order principle, and the programmer prepares a specification tape to describe the kinds of information he wishes to obtain. These two routines differ in that D7 has been written to use only as much storage as is used by the D.O.I. It is possible with D1 to go through iterative loops and print results on various passages through the loops; with D7, however, each use of a blocking point requires separate specification. Data can be obtained with both routines as an order pair, a right-hand address, a left-hand address, a 10 character sexadecimal word, a signed integer, a signed 12 decimal place fraction or a signed 5 decimal place fraction.

Library Routines D1 and D7 are very powerful checking routines because of the great latitude given the programmer in choosing where and how he will obtain information and because they utilize the programmer's own knowledge of his program.

Other checking facilities not listed here are introduced into the library from time to time, and for this reason if the routines described do not appear to be effective in any case, a member of the laboratory staff should be consulted.

8.13 WHAT TO DO BEFORE A CODE CHECK RUN.

Now that we have been introduced to the main "tools" available to a programmer as an aid to debugging his program, let us return to some more practical aspects of the debugging process. When a program is written, it should not be expected that it will be free from mistakes. Finding mistakes is as much part of getting a problem running as coding itself, and must be attacked with equal care and deliberation.

A list of the memory locations of the routines and lists of constants etc. contained in the code, both in decimal and sexadecimal notation, should be made. The latter is very useful while on the machine, since the order counter is read sexadecimally. This list of locations should be on a separate page, so that the information is clearly set out and readily available when code checking at the computer.

A general rule concerning all papers associated with a program is that they should be placed on a file as soon as possible after they are originated, and from which they are detached in case of emergency only. This applies to flow diagrams, routines, lists of correction, tape

printouts and machine printouts. Each sheet should be accompanied by its date of origination, and, if relevant, the time of origination. (This is sometimes important if a number of code checking runs are carried out in one day.) The liberal use of scissors and paste in condensing column printouts will reduce the amount of paper involved, and the use of transparent Sellotape to make small side tabs will facilitate reference to important sections of the file.

In addition to

D1 and D7 (Check Point) and
D15 (Control Transfer Check Routine)

one of which will probably be used as an integral part of the code checking run, familiarity with the following post mortem routines is necessary.

C1 and C12 (Post Mortem Versions of D.O.I's X1 and X12),
C3, C4, C5, and C9 (Post Mortem Printouts),
C6 (Address Search Routine) and
C10 and C11 (Special Post Mortems for Floating Point Routines).

The programmer must be prepared to call for any one of these post mortem routines at short notice if difficulty arises with his program during the code checking period. Although the operator can give advice, only the programmer knows the intricacies of the problem and hence the best Diagnostic Routine to use.

It is important to appreciate the relative roles of D4 and D1 (or D7). Problems may be divided broadly into two categories--those which are largely mathematical in nature, and those which are largely logical. These two categories frequently overlap.

A problem which requires considerable thought in

planning its logical arrangement is likely to incorporate blunders which result in control pursuing the wrong course through the program. Blunders of this type should be eliminated before those involving incorrect mathematical manipulation, and it is here that D15 is particularly useful. A copy of D15 is available on the console, but should it be desired to put D15 in specified locations or preserve the D.O.I., a decimal version is available which is attached to the end of the main tape and input by the D.O.I. If, say, it was required to put D15 in location 700 onwards, the end of the main tape should read as follows:

201019N
24 pN
Fifth hole characters
00700+
Code D15
End of tape.

Here p is the location of the first order in the main program to be obeyed.

When the tape is read in, it stops on encountering 201019N. For code checking, the tape is moved on manually to the fifth hole characters, and reading in is continued by lifting the black switch. (In a production run, the 201019N is bypassed via the black switch.)

It is unusual for D1 (or D7) to be used the first time a run is presented to the machine for test, because this would require the preparation of a specification tape which would almost certainly call for much more printout than is really necessary. The use of D15 with the first run is more usual because it is at this stage that the program is most likely to run "off the rails" and the course

taken by the program up to the time of stopping on an illegal order can be traced.

After the first test run, D1 (or D7), will usually be required and for these routines a specification tape must be prepared. It is important that the instructions for using these routines should be properly understood; although they are the most useful Diagnostic Routines in our Library, they must be handled with care. Certain precautions are necessary, and if these are ignored, the code checking method will itself introduce errors into the program.

As mentioned in section 8.12, the basic difference between D1 and D7 is that D7 occupies only the space normally occupied by the D.O.I., and is designed for use when space is at a premium. D1 is more versatile and occupies more space. When using D1 (or D7) to printout intermediate results in a test case that has been done manually, write out on a sheet of paper the locations of blocking orders and the printout expected when each blocking order is encountered. File this with your other papers, for, although one should not try to use this information while the machine is punching out information during the code check, it will be required later, after leaving the machine. Because of the frequency with which D1 (or D7) specification tapes are usually produced in the course of assembling a program, it is best always to produce these as separate lengths of tape. Always remember to place on the specification tape terminating symbols which result in control being transferred to the program--with D1, pN4 is the usual combination to use initially, as this gives the stop control to location p of the program, enabling the operator to insert the data tape in the reader if this is required.

Parameters and data associated with the various tests the programmer has devised should all be inserted on the main tape after the point where D4 and/or D1 (or D7) is incorporated. These tests should each be preceded by a clearly written label, and separated by an adequate series of delays.

There are two ways in which corrections can be inserted in the program. One is to have a separate correction tape which is inserted just before control is transferred from the D.O.I. to the program proper. As there are usually a number of corrections required, and these corrections will be made at different times, this method necessitates reproducing, at each stage, the correction tape containing information recorded so far and then adding it to the new corrections. With longer programs, this approach is both wasteful of editing equipment time and likely to introduce errors if the editing equipment happens to be operating unsatisfactorily at the time: however, it is quite adequate for shorter programs.

A second and more satisfactory technique is to splice corrections into the tape as they are made. If this is done it is best to splice each correction into the tape at the end of the routine with which it is associated, rather than at the end of the complete tape. This allows individual routines to be taken from a corrected tape and re-arranged at a later time without the risk of missing corrections. In splicing tape the precautions outlined in the next chapter should be observed.

One last point concerns the technique of making corrections on the code sheets. These should never be made by rubbing out, because subsequent diagnoses may

involve spotting the fact that a given instruction has been altered. The best plan is to make the first alterations in ink, the second in red pencil, the third in blue etc. Also, considerable care should be exercised to ensure that the alterations are actually made on the tape: the safest plan is to follow a rule that tape alterations and alterations in code lists are always carried out at the same time.

When the machine is to be used for a code check the following should be taken:

(1) The tapes, preferably not more than two. Any tape more than a yard long should be rolled up so that it will not become tangled while reading in.

(2) The operating instructions for the check run written down on a sheet of paper.

(3) A list of the store locations(in sexadecimal as well as decimal) at which the routines and tables of data should be stored. When the machine stops on an illegal order the programmer must be able to decide immediately whether its location (read from the order counter in sexadecimal) places it within his program or in an unused part of the store.

(4) At least a mental picture of what should appear on the monitor screen as the program is read in.

8.14 <u>WHAT TO DO (AND NOT TO DO) DURING CODE CHECK RUNS.</u>

If the above recommendations are followed, the number of separate tapes brought to the machine will have been minimised.

The programmer should put his name and problem number on the blackboard queue and wait his turn, standing well away from the machine while waiting so as not to block the operator's view of the lights in the order register. A

hopper should be obtained on reaching the head of the queue to take tape as it passes through the reader and to take the output tape.

When the operator calls the programmer's name, the programmer should hand her the tapes and read out the operating instructions. Far more will be learned about the operation of the program if the handling of the reader is left to the operator and the programmer focuses his attention upon the monitor screen and/or the order counter and register.

As the program is read in, the programmer should watch the monitor screen and check that it is being stored into the correct part of the store.

The operating instructions for an initial run might be as follows:

(1) Read in main tape until it stops. If it does not stop on a legitimate stop, mark the position at which it stops before the tape is removed from the reader. (This precaution is helpful in diagnosing the reason for incorrect stopping.)

(2) Bootstrap in a copy of D15, or if the decimal version is being used, move tape on to beginning of D15, re-starting tape with black switch. D15 will be read in and the specification tape and the remainder of D15 should be read in as described in the write-up.

(3) Move main tape onto the parameter combinations under test. Re-start with black switch.

If the output produced is clearly not as expected (e.g. if there is no output at all), transfers of control stored by D15 can be output (see D15 write-up). Output

will normally be taken on the punch at this stage, and so incorrect output will not always be detected immediately. Only if very little output is called for should it be taken on the printer. If the output available gives no clue as to the best locations for blocking orders for D1 (or D7) and the coder has erroneously assumed that output is correct, it is best to repeat the above operations, and obtain the D15 output before proceeding to use D1 (or D7).

If a test is not being tried for the first time, the procedure will normally be varied after the first stop, perhaps as follows:

(1) Read in operator's copy of D1 until a stop is encountered.

(2) Read in specification tape.

(3) Read in remainder of operator's copy of D1.

(4) Insert main tape in reader in front of appropriate test parameters and start again with black switch.

Whenever the machine stops, the address at which it stops should be read from the order counter and noted in sexadecimal. The operator will normally read this information out.

If the program stops on the division hang-up, write down the memory location of the division order, look at the number register (R^3 register) to see whether the divisor is zero or very small, and then by-pass the division order with the white switch. If the program stops on an illegal order, insert the address search routine C6, followed by a short tape on which the address of that order is punched in sexadecimal. Search for instructions containing these and the ten preceding locations (this is done by raising the black switch when the address

search routine stops on each of nine separate occasions). If the illegal order is a zero left shift order in an unused part of the store there is no point in searching for transfers to previous locations. Familiarity with the technique of using C6 is desirable before going to the machine, as this will make it easier for the operator to assist.

This address search routine is useful if the illegal order is in part of the store outside the program: if this is the case the usual cause is a transfer of control in which an F termination has been used in lieu of an L termination.

This technique is also useful if the illegal order is in part of the store occupied by the program, for the usual cause of such an occurrence is a transfer of data to the storage location concerned. The store order with the offending address can be found quickly in this way: the blunder is usually due to the use of an L termination in lieu of an F termination.

The operating instructions given above, or some equivalent instructions which apply to the particular program and the method of checking employed, should be written down on a sheet of paper which is kept separate from other papers so as to be readily available during the code check.

The code check output should be taken on the punch in the interests of speed. An important point here is that the characteristic rhythm of the punch will quickly indicate when an output of storage locations containing zeros (with the post mortem routines), or an unsuccessful search (with C6) for orders with certain addresses, occurs.

This information often helps to curtail output considerably. The operator will be able to assist in this respect.

Tapes should be taken away from the machine in tape hoppers as soon as the code check has finished, and wound up either in the tape preparation room or with the help of the winder on the far side of the SILLIAC room. Hoppers must be emptied as quickly as possible so that they can be used by some-one else. Similarly, tape printout should be carried out using printing facilities in the tape preparation room: this relieves the congestion around the console.

8.15 ADDITIONAL HINTS The result of the code checking run should be taken away to a quiet place for consideration--for preference, a place with a desk computing machine. The SILLIAC and tape preparation rooms are not well suited to this purpose.

In the initial stages of code checking, the calculation is likely to stop in some odd place in the memory, and it is necessary to determine the reason for this kind of failure. This is usually rather easy with the help of D15, and also if liberal use has been made of D1 (or D7) blocking orders. It is often useful to place blocking orders in various places, without any printout being specified, just so it can be determined whether a certain section of the program was in fact reached during the abortive computation.

In later stages of code checking, the machine will run through the calculation and stop on the programmed OFF order--but the answer will still be quite wrong. By using D1 (or D7) to print out intermediate results at various stages of the calculation, and checking these on a hand computer one by one, it is possible to localise

the errors in the code, and eliminate them one by one.

It is not always wise to stop as soon as one error has been discovered in a program. It is desirable if possible to make sure that this particular error does in fact produce the particular printout obtained. Otherwise there must be at least one other error in the program, and already enough information may have been obtained to locate this other error as well. However, such a procedure is not always feasible, as it may often be quicker to try out a program again having rectified the blunder you have found, than to spend time working out its possible effects in detail--a balance must be struck.

Do <u>not</u> be satisfied with qualitative 2-or 3-figure accuracy checks on answers. Use a desk calculator and work to 8-figure accuracy, or else use parameters which give very simple answers which can be checked to 12-figure accuracy by mere inspection. Remember that errors in the program may result in wrong answers in the fifth significant place for the particular set of parameters which have been chosen for code checking, but in the first significant place for some other set of parameters.

When the code is suspected of working correctly, don't give in to such unsound suspicions--make sure! Try before extreme values, pronouncing a program O.K. It is unfortunately true that programs which work perfectly with the usual run of parameters, sometimes fail badly on somewhat unusual parameters, which may nevertheless occur in final production runs. Division hang-up failures are particular culprits in this respect.

A final run with D1 (or D7) and extensive print-out is a wise precaution. The record of this run should be filed along with other information concerning the

development of the program, and the file should be retained.

When a program is finally working, it is desirable to insert check sums with the use of X12 and, in the case of long programs which have been input with X13 to obtain a sexadecimal dump of the program together with the check sum using routine X16. This increases the speed of input considerably.

With long runs, provision should be made for program interruption (e.g. with X10), and no run of more than about 15 minutes should be made unless the calculation is checked in some way. Although error-free runs of many hours are usual with SILLIAC, occasionally faults which cannot be tracked down quickly because of their intermittent nature are present. When this happens, errors induced by a machine fault will cause too much loss of time if this precaution is not observed.

A point about machine errors is that occasionally obscure behaviour of a program during development may be due to a machine fault. If, after a reasonable time, you cannot see why the machine behaved in a certain way, repeat the run.

If the same thing happens twice in succession it is almost certain that the fault is the coder's. If this is not the case, do not omit to have the operator note in the log book that the time of the previous run was lost due to machine malfunctioning.

Remember that, in addition to the periods formally set apart for code checking, it is possible to have a "two-minute break-in" between any two problems as they are taken from the queue. The purpose of this facility is to enable points arising during code development which can be settled quickly to be handled without delay. To

get the most out of this short period, considerable streamlining of code-checking techniques is necessary.

As well as these "two-minute break-ins" it is possible to carry out development runs from the queue of work awaiting processing. Once a routine is in a state for D1 (or D7) to be used on it with profit, then there is no reason why its development should not be handled in this way: and in fact, particularly with larger programs in an advanced state of development, this technique should be adopted as normal procedure.

8.16 CODE CHECKING DO'S AND DON'T'S The following list of suggestions for efficient code checking is provided for quick reference:

(1) Always stand well clear of SILLIAC when waiting a turn for a code check.

(2) Always have any tapes more than about three feet long wound up so that they can be read into SILLIAC without tangling. Do not carry tapes to the machine loose in a hopper

(3) When running a program more than once in the one code check period (for example, it may be desired to try several different values of the parameters) the programmer's name should be written the appropriate number of times on the queue list on the blackboard provided. The programmer should avoid running the same tape twice in succession through the reader, for time is needed to wind it up again.

(4) The programmer should have the complete instructions for running a test written out in full. Attention may then be fully concentrated on what SILLIAC does, and not on checking

procedure.

(5) Read the instructions out to the operator and let her put the tapes into the reader. This speeds things up and leaves the programmer more opportunity to watch the monitor screen and order register.

(6) Have at hand a list of the locations of the various parts of the program in sexademical as well as decimal. When the machine stops on an illegal order it can be immediately determined whether its location (read from the order counter in sexadecimal) places it within the program under test or out in an unused part of the store.

(7) Whenever SILLIAC stops on an illegal order note immediately the function digits and the location of the order.

(8) Immediately after leaving the machine after a code check, wind up the tapes (using the table on the far side of the room--not the operatore's table), and leave the tape hopper in the SILLIAC room for use by someone else during his code check.

(9) If D1 (or D7), or one of the type C checking routines has been called for, the operator's copy of the routine will go into the tape hopper. Do not walk off with the hopper before giving the operator a chance to take it out and hang it back on the rack. If an operator's copy of a tape is taken by accident, do not delay in returning it--someone may be waiting for it.

(Note: an operator's copy is

coloured blue.)

(10) Do not place anything on the operator's desk and do not use the tape winder on the operator's desk to wind tape--use the table on the other side of the room or the one in the tape preparation room.

(11) If a program development calls for a run which should fairly certainly go through without stopping and which will take more than three minutes total elapsed time, do not do it during code checking periods. Write up the instructions on the official form and hang it up on the production queue. (If the form is marked Development, the programmer, or his Department will be charged at the same rate as in code checking periods.)

PURPOSE	ROUTINE	HOW USED
Investigate contents of memory.	C3: signed 12-figure fractions. C13: signed 5-figure fractions. C4: signed integers. C5: order pairs with decimal addresses. C9: floating decimal numbers.	Input with bootstrap.2-digit numbers from 00 to 99 are on end of tape. If for example we specify "73" routine will print locations 730 to 739.
Investigate changes from original.	C1: use with X1 C12: use with X12	Used in lieu of D.O.I. Prints out address of discrepancy, the word on the tape and the word in the memory. Program left ready for running.
Address search.	C6: all orders with given address C7: only orders of types 2,3 C8: only store orders	Input with bootstrap. Also specify 3 character sexad. address. Will print all order pairs which use this address. Restarting with black switch repeats with address one less with C6,C7, or one more in case of C8. C7 and C8 may be restarted with white switch to read in new address.
Investigate last integer interpretive order and contents of floating accumulator.	C10: use with A1 or A4 C11: use with A4 only	Input with bootstrap. Also specify 3 character sexad. address of first word of interpretive routine, A1 or A4.
Investigate contents of store or registers at specified points during running of a program.	D1 D7	Specification tape to indicate the points and the printout required. D7 is more economical in storage space but not as simple to use as D1.
Control transfer check routine.	D15	A specification tape is required. D15 prints out details of final stop and control transfer orders last obeyed. Operation of program under test is slowed down twenty-fold.

CHAPTER 9

TAPE PREPARATION

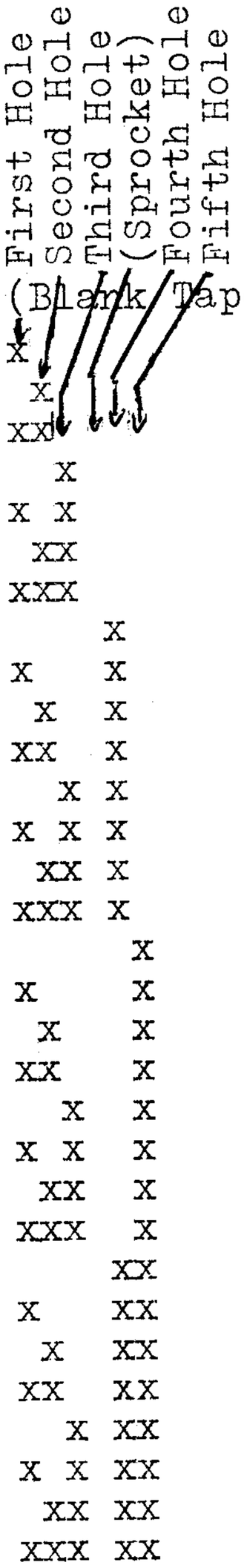
9.1 THE SILLIAC INPUT The input unit of the SILLIAC is a photoelectric tape reader that transfers binary digits from a punched paper tape to the A register. The tape preparation equipment is used to translate instructions and data from the programmer's manuscript into a binary-coded punched tape acceptable to the SILLIAC.

9.2 THE SILLIAC OUTPUT. Output from the SILLIAC is usually in the form of punched tape. This tape may be printed on the Teletype page printer in the SILLIAC room, or the Teletype page printer in the tape preparation room. The page printer performs a conversion from the five-hole code on the tape to the characters on the printed page.

9.3 THE PERFORATED TAPE, THE SILLIAC TAPE CODE The tape preparation equipment described in this chapter consists of Creed and Teletype equipment which has been modified to work with the SILLIAC tape code. The paper tape can be punched in any one of the six positions across its width, one of these positions (the feed or sprocket hole, which is smaller than the others) always being punched.

The code is set out in Fig. 9.1, the order in which the symbols are displayed corresponding to the view of the tape obtained by an operator sitting at the keyboard of a keyboard perforator and looking to the left.

Teletype and Creed page printers are capable of interpreting the five information holes in a total of 62 different ways by virtue of the fact that two of the 2^5 possible interpretations of the five hole positions are reserved to mean: "Change the code". According to which of these two codes

Character	Figure Shift	Letter Shift
 (First Hole, Second Hole, Third Hole, (Sprocket), Fourth Hole, Fifth Hole) (Blank Tape)		
	0(Zero)	P
	1	Q
	2	W
	3	E
	4	R
	5	T
	6	Y
	7	U
	8	I
	9	O
	+	K
	-	S
	N	N
	J	J
	F	F
	L	L
	Delay	
	#	D
	Carriage return & line feed	
	(	B
	Letter Shift	
	)	V
	/	A
		X
	Space	
	=	G
	.	M
	Figure Shift	
	'(prime)	H
	:	C
	×(multiplication)	Z
	Erase	

Notes: (a) x indicates the presence of a hole.
(b) The table shows the configuration of holes on the tape produced by a keyboard perforator, as viewed from the perforator.
(c) Delay, Letter Shift, Figure Shift and Erase cause no action of the Teleprinter or Teletype.

Figure 9.1

has been encountered last, the page printer will interpret symbols as being in one of two codes, conventionally reserved for "figures" and "letters". The two codes have accordingly become known as "figure shift" and "letter shift", and, because of the mechanical construction of the page printers, the five-bit combination used for "figure shift" (which would normally only be required in letter shift) cannot be used for anything else in figure shift, and mutatis mutandis with letter shift. In fact, if the "figure shift" character is used when a page printer is already in figure shift, there is no effect; similarly, there is no effect if "letter shift" is used when a page printer is already in letter shift.

In the SILLIAC code, for convenience, eight other characters are common to both shifts, and so the total number of different characters is, in fact, 54.

The first sixteen characters in figure shift are the normal sexadecimal code used in laboratory literature with the exception that in older literature, + and - appear as K and S, their letter-shift equivalents.

It is usual to refer to the 10 sexadecimal characters 0-9 as punched on the tape as being in <u>binary coded decimal</u>. The reason for this is that, if we regard the hole positions as corresponding to 2^0, 2^1, 2^3 respectively, the coded form of the digits is also their binary form. Note that in Fig. 9.1, the codes are displayed so that the least significant digit is written first, a practice which is of course the reverse of the normal one.

There are three printer control characters. The first causes the printer to space; the second causes the carriage to return and the paper to be advanced one line, and the third is a dummy character which merely serves as a delay. If the carriage of a Teletype page printer is to be returned from

more than 20 characters from the left hand margin, it should be followed by a delay code to allow enough time for the complete return of the carriage to the left hand margin. Otherwise, the next sexadecimal character would operate the printer mechanism too soon, and print the character somewhere out in the middle of the line. If the carriage is within about 20 characters of the left margin, a delay code need not be used after a carriage return and line feed code.

The format control codes have a character in the fifth hole position. When a punched tape is placed in the SILLIAC tape reader, the circuits cause any character with a hole in the fifth position to be skipped (i.e. not read into the computer) with a normal input order. (There is, however, a special input order to read such a character - see the order code.) This means that in the preparation of tape, these format codes can be interspersed with sexadecimal order codes in any desired way. Then, when an instruction tape is printed these codes will control the printer. The usual method in preparing instruction tapes is to follow each order pair with a carriage return and line feed code. This produces a single column print of words.

The ability to print letters and some other special characters is useful for headings, and sometimes these codes are used as special codes in input programs.

9.4 <u>TAPE EDITING</u> The several operations which one wishes to perform, in handling the punched tape which forms the input-output medium for SILLIAC, are as follows:

(a) preparation of a tape from a manuscript of program or data in written form;

(b) reperforation to produce a copy of an existing tape with or without occasional corrections;

(c) printing out the contents of a tape;

(d) comparison of two tapes to detect errors;

(e) comparison of two tapes which may contain errors, together with simultaneous production of a third, correct, tape.

For performing these operations the following equipment will be initially provided;

(i) Teletype keyboard perforators and modified Murray keyboard perforators;

(ii) Creed 7TR/3 non-printing reperforators;

(iii) Editing sets consisting of Creed 54/N4 page-printers, with reperforating attachments, coupled to Creed 6S/5M transmitters;

(iv) Creed tape comparators;

(v) Teletype Model 28 receiving-only page-printers, coupled to Teletype transmitters;

(vi) High-speed comparer-reperforator sets consisting of two Ferranti photoelectric tape readers, one Teletype BRPE high-speed punch, and associated electronic control circuits.

9.5 DETAILS OF EQUIPMENT Of item (i) keyboard perforators, there are two Teletype and two Murray keyboard perforators. One of these keyboard perforators will be close to the computer itself, for use in preparing short lengths of tape for immediate reading into the computer, particularly during maintenance and program checking procedures. The other keyboard perforators are installed in the hand computing machine room in the basement (G.16) and will be available for use on operation (a), preparation of tape from manuscript. This simple form of machine, which does no more than produce a punched tape in response to manual operation of the keyboard, is satisfactory for use by an experienced operator, but lacks facilities which are a help to the general user. For him it is a great convenience to use the more elaborate

equipment of item (iii) which, simultaneously with producing a punched tape, produces also a printed record of what has been punched.

Of item (ii), non-printing reperforator, one only will be provided and this one will be used for the production of library tapes; it will be available for general use when not required for this purpose. The Creed machine used is capable of punching two tapes simultaneously, and this facility will be used when library tapes are being produced. In this case, one of the tapes will be a special long-life tape of heavy grade which will be the library tape for general use. The other tape will be of ordinary grade, but coloured green to distinguish it from ordinary tape; this tape will form the master copy, which will be filed away for use in producing a new library tape when required. Each time a master tape is used to produce a new library tape, a new master will also be produced, and the old master may be discarded. In this way the file of master tapes should remain always in first class condition.

The equipment of item (iii) constitutes the normal means for production and reproduction of tapes by the general user. There will be two sets of this equipment initially provided. Additional sets may become necessary as the use of SILLIAC increases. The set consists of a Creed Model 54/N4 page printer with reperforating attachment, and a Creed 6S/5M transmitter, interconnected via a junction box on which necessary control switches are mounted. By means of this set any of the operations (a), (b) and (c) as listed above may be conveniently performed.

When a copy of a tape is produced by a reperforating operation one does not trust the machines to be completely free of error. Immediately after reperforation, the copy

and the original tape must be compared to check that they do in fact agree. This may be done on a Creed comparator (item (iv)), one of which will be associated with each set of item (iii). The Creed comparator has two reading heads; when two tapes are placed one in each of these reading heads, and the machine is started, the tapes are read and compared at 400 characters (i.e. rows of holes) per minute, until a discrepancy between the two tapes appears. Upon detecting such a discrepancy the machine stops. Tapes may also be compared by means of the equipment of item (vi) as described below.

When results are obtained from SILLIAC on punched tape it is generally necessary to print out the contents of the tape to obtain the results in readable form. This could be done on the Creed equipment of item (iii), but since these machines are likely to be in considerable demand for the production of program and data tapes, separate machines will be provided for the sole purpose of printing out results. These machines will be Teletype Model 28, receive-only, page printers which operate at 600 characters per minute; two of these will be provided, of which one will be placed close to the computer and connected so that it may, when required, receive results direct from the computer instead of from a tape.

The equipment listed above is all telegraph equipment which has been slightly modified to suit the special needs of a computing installation. In order to make use, in tape-editing operations, of the higher speeds obtainable from photoelectric tape readers and the Teletype BRPE punch as used for input and output on the computer itself, a unit (item vi) has been designed using two Ferranti photoelectric readers and one BRPE punch, together with necessary electronic control circuits. With this unit any of the operations (b), (d) and (e) listed above, may be performed,

and it will be the only equipment available for operation (e). The speed of this unit will be 3600 characters per minute when reperforating tape (nine times faster than the Creed equipment), or 12,000 characters per minute when the readers only are in use for tape comparison (thirty times faster than the Creed equipment).

Operation (e) is useful particularly in the preparation of long data tapes. One way to check such tapes is to print them out and proof-read against the original manuscript; but this process is both tedious and unreliable. A preferred way is to punch the data twice and then to compare the resulting two tapes, simultaneously punching a third tape so long as the two agree, but stopping as soon as a discrepancy appears. The correct character can then be manually punched on the third tape, and the process of comparison and reperforation resumed. At the end of the process a correct tape should be available; this tape however should again be checked by comparison with one of the first two tapes.

9.6 HOLLERITH CARD-TO-TAPE CONVERTER AND NATIONAL ADD PUNCH The laboratory has recently (November, 1958) acquired a Hollerith punched card to paper tape converter, and expects to have an adding machine with punched paper tape attachment early in 1959.

Also, it is expected that a verifier will be made available from C.S.I.R.O. early in 1959. This equipment will compare the character corresponding to a depressed key with a corresponding one on a punched paper tape. If they agree, the character will be punched on a second punched paper tape. If they disagree, the equipment will indicate the disagreement and the operator must then choose the correct one.

Literature will be issued to describe the use of these items of equipment as soon as possible after they are available for use. (See also section 9.18.)

9.7 KEY BOARD OF TELEPRINTER MODEL 54. The rest of this chapter is concerned with detailed operating instructions for the main items of equipment which have been described briefly in the preceding sections (viz. (iii) and (vi)).

The most frequently used keys of the teleprinter Model 54 (except zero key which is the centre front pad) are in red.

A combination of holes on the punched paper tape (called a character) will cause a printer to punch one of two characters according to whether the printer is in Letter Shift or Figure Shift. One character (marked LTRS) has the effect of changing a printer to Letter Shift, and one character (marked FIGS) has the effect of changing the printer to Figure Shift. In general, where two symbols are marked on a key, the upper symbol will be printed if the printer is in Figure Shift, the lower symbol if it is in Letter Shift. Characters which result in the same symbol in both shifts have only that symbol on the corresponding key.

In most input routines it does not matter in which shift the character concerned is referred to. In particular K and + are interchangeable, as are S and -.

The two space keys cause the same character (which will cause a printer to space without printing) to be punched on tape. A similar remark applies to the two Carriage-Return/Line Feed keys (marked $\frac{CR}{LF}$). Note that the character corresponding to 1 (one) is obtained by pressing the key marked $\frac{1}{Q}$, and not $\frac{8}{I}$. Similarly, the character corresponding to O (zero) is obtained by pressing the centre pad marked $\frac{O}{P}$, and not the key marked $\frac{9}{O}$.

Most programs ignore ancillary characters on input (in fact they ignore any character with a hole in the fifth

position -- i.e., the hole in the rearmost position as punched). This means that the operator can choose any form of layout according to his own preference.

The RUN-OUT key to the right and above the keyboard, when depressed, causes continuous punching of the last character punched. About eight or nine inches of tape should be run out on Delay characters (see next paragraph) at the beginning of every tape, followed by a Carriage-Return/Line-Feed and about two inches of Figure Shifts and then a final CR/LF followed by a delay. The tape should be terminated by about two inches of CR/LF's and run out on delays, the CR/LF's ensuring that the printed version is clear of the shield on the Teleprinter. Tape should always be cut with the scissors available so that the leader is pointed in the forward direction. This and the easily recognizable sections of tape punched with Figure Shifts and CR/LF's serve to label the beginning and end of the tape respectively, and so help to prevent the otherwise easily committed error of placing the tape in a reader wrong end first.

After every CR/LF throughout the text, it is advisable to punch a delay character. This character has the action of wasting one cycle of the printer, thereby ensuring that the carriage, which takes a finite time to return, is back in the initial position before the next character is printed.

Characters entered from the key-board will appear on the printed sheet in red, characters printed from tape run through the tape transmitter (see below) will appear in black. Note that the punch is operated by bowden cables from the printer mechanism and so a punching error may occur in some cases even though the printed text appears correct. To avoid this possibility going unnoticed, visual checks should be made from a separately printed copy of the perforated tape.

The erase key has the effect of punching five holes across the tape. It is used in conjunction with the BACK SPACE key (see next section) to negate the effect of incorrect punching. Having a hole in the fifth character position, it is ignored by most routines on input.

9.8 PUNCH CONTROL OF TELEPRINTER TYPE 54. If the cover to the left of the key-board is lifted, the punch becomes visible. The last character punched can just be seen beyond the punch block and this can be recognized from the attached list of characters. To the left is the BACK SPACE KEY. This must be firmly depressed in order to ensure correct action, and moreover it is usually wise to check by inspection that the action has occurred correctly. It is usually used in conjunction with the ERASE key mentioned above. The metal lever to the right is the PUNCH CONTROL LEVER, and has two positions: if pressed fully to the left (the INHIBIT position), it results in the tape punch being inhibited; if pressed fully to the right (the PUNCH position) it results in the tape punch being rendered operative. Printing occurs regardless of the setting of this lever.

9.10 TAPE TRANSMITTER. To the left of the teleprinter is Tape Transmitter Model 6S/5M with the Control Panel mounted above it. On the base of the Tape Transmitter is the ON/OFF switch (OFF-up, ON-down). This switch should be turned on only when the transmitter is actually in use. To insert the tape in the transmitter GATE the RETAINING CATCH should be pushed to the right and the GATE lifted. The tape should then be inserted and register with the SPROCKET DRIVE ensured. The GATE should then be closed and the action of the CATCH checked. The easiest way of doing this is to ensure that it is not possible to raise the GATE.

The tape should be fed under the TAUT TAPE LEVER to the right of the GATE and over the GUIDE immediately to the

Note: There is no section numbered 9.9.

right of the TAUT TAPE LEVER. Note that raising the TAUT TAPE LEVER will stop the transmitter. Also, when the end of the tape has run through the reader, an END-OF-TAPE-SENSING-DEVICE will cause the transmitter to stop. This is located under the GATE.

The character about to be read can be seen by looking at the GATE from above. The READING PECKERS will show through any holes on the character about to be read, since they are darker than the transmitter plate which shows through the holes of neighbouring characters. The character about to be read will be immediately to the right of the raised section of the grid of the GATE.

9.11 EDITING STATION CONTROL UNIT. This is mounted on top of the tape transmitter and has a SWITCH and a KEY. The SWITCH is an ON/OFF SWITCH. The KEY has three positions: in its central position (MANUAL) it allows tape to be punched from the key board; in its upper position (AUTO) it will cause tape to be copied from the transmitter; in its lower position (SINGLE SHOT) it will cause re-perforation to take place one character at a time.

9.12 NOTE ON CORRECTION OF TAPE. If the correction of a character requires merely the insertion of another hole, this can be carried out by back spacing the tape until the character concerned is directly under the punch and then punching the character. Alternatively, the ONE-HOLE PUNCH can be used.

9.13 SPECIAL REMINDERS. WHEN THE TAPE BEING PERFORATED IS RED, CALL THE PROGRAM LIBRARIAN OR ANY OTHER MEMBER OF THE LABORATORY STAFF. This is an indication that the tape supply is running out.

TURN OFF CONTROL UNIT AND TRANSMITTER ON/OFF SWITCHES WHEN FINISHED.

TURN ON TRANSMITTER ONLY WHEN REPERFORATION REQUIRED.

REPORT ALL MALFUNCTIONINGS TO THE PROGRAM LIBRARIAN OR ANY OTHER MEMBER OF THE LABORATORY STAFF.

9.14 SAMPLE OPERATIONS OF EDITING SETS.

Punching

CONTROL UNIT ON/OFF switch to ON.
CONTROL UNIT KEY to MANUAL.
Ensure PUNCH CONTROL LEVER in PUNCH position.
Use keyboard as explained above.

To Correct an Error with ERASE KEY

BACKSPACE the tape until the character concerned is under punch (i.e. has just disappeared). Press ERASE key. If subsequent characters are satisfactory and it is desired to advance tape without causing any other effect, press $\frac{O}{P}$ pad until last character punched is just visible.

To Copy Tape Correcting a Single Character

CONTROL UNIT ON/OFF SWITCH to ON.
TRANSMITTER TO ON.
CONTROL UNIT KEY to MANUAL.
Insert tape in GATE.
Check that PUNCH CONTROL LEVER is in OPERATE position.
Run out leader as instructed above.
CONTROL UNIT KEY to AUTO.

Check from printed copy when proposed correction is drawing near. At this point, place CONTROL UNIT KEY in MANUAL position, and then use SINGLE SHOT facility until incorrect character is about to be read.

PUNCH CONTROL LEVER to INHIBIT position.
CONTROL UNIT KEY to SINGLE SHOT.
PUNCH CONTROL SWITCH to OPERATE position.

With CONTROL UNIT KEY in MANUAL, insert correct character from keyboard.

Set CONTROL UNIT KEY to AUTO to continue reperforation.

A tape reperforated in this way should be checked against the original in one of the comparators.

9.15 HIGH-SPEED COMPARER-REPERFORATOR DESCRIPTION AND OPERATING INSTRUCTIONS: INTRODUCTION. Many people who have used the telegraph-type equipment for the preparation of tapes for SILLIAC will have been exasperated by its slowness. The equipment here described has been constructed to speed up some, at least, of the operations, by taking advantage of the higher speeds obtainable from the photoelectric tape reader, and high-speed punch, of the kind used for SILLIAC input and output.

The equipment consists of two tape readers and one tape punch, together with the necessary circuits to link these units together so as to perform several possible operations. These operations include: reperforation, using either reader together with the punch; comparison, using both readers; and, using all three units, the comparison of two tapes with simultaneous perforation of a third tape so long as the two initial tapes agree. Controls are provided to facilitate editing operations such as changing, adding, or deleting characters during a reperforation process.

9.16 REGISTERS AND CONTROLS. Two registers, one associated with each reader, and a third register between them, have their contents displayed on neon lights on the front of the machine. The two reader registers are always set to delay characters when the machine is first turned on, and they may be set manually to any other characters by means of the push buttons provided; otherwise they always display the last characters read by the corresponding readers.

The third register is also set to a delay character on turn-on. It is called the punch register since it is

primarily concerned with the punch. When the punch is in use, the punch register displays the last character punched. When the punch is not in use, i.e. during comparison of two tapes, the punch register is still used. In this case it displays the last character which has been successfully compared in the two reader registers.

On the main control panel a FUNCTION SWITCH is provided, by means of which any one of four modes of operation may be selected. These are as follows:

1. COMPARE
2. COMPARE & REPERFORATE
3. MANUAL PUNCH from selected register
4. REPERFORATE from selected reader.

Operation of the equipment in each of these four modes is described in some detail in the following sections.

As well as the FUNCTION SWITCH there are a number of other controls, which are all necessary in order to give the machine the desired flexibility, but which make it necessary to proceed with care if the correct results are to be obtained.

There is a switch with labels S/BY, ON. The equipment is normally left connected to the power all day, with the switch turned to standby (S/BY) when the equipment is not in use. It is then ready for use immediately the switch is turned to ON. Tapes should not be placed in the readers while the switch is on S/BY, since the reader brakes are not operating, and the tapes will tend to drift through. It is important that the switch be returned to S/BY after use. There are also power switches on each reader, and on the punch. These switches MUST NOT be turned off. Such action may cause inconvenience to other users, and moreover, as the switches control blowers in the readers, may lead to overheating. This point is stressed particularly, because it is

different from the procedure requested for the other tape-preparation machines, where it is important that the user should turn off the power switches after he has finished.

When reperforating, either reader may be selected by means of the switch labelled SELECT and the reader may be caused to read all characters, to skip over erase characters, or to skip over all 5th-hole characters, according to the setting of the SKIP SWITCH. The normal setting for the skip switch is position 2, SKIP ERASE, since erase characters are not normally wanted on a finished tape. The skip switch performs the same function when comparing, so that, for example, a tape which has been reperforated with deletion of erase characters may be compared with the original without stoppages, except on actual errors.

For the purpose of deleting unwanted characters during reperforation, or of bringing two tapes into step during comparison, each reader is provided with a JUMP BUTTON. These buttons are located just under the shelf in front of each reader. When one of these buttons is pressed the reader concerned advances to the next wanted character, as determined by the setting of the skip switch, and reads this character into the corresponding register, but no other action takes place.

The punch is, of course, provided with a run-out button for the purpose of running out a section of tape punched with delay characters. This button is located on the punch housing.

The remaining controls are the three push buttons labelled 1-SHOT, RUN and STOP respectively, on the main panel. Since these labels are almost self explanatory the user should be able from what has already been said, to carry out straightforward reperforation and comparison operations. However, for the correct performance of editing operations, including the

changing, deleting, or adding of characters while reperforating, it is necessary to know exactly the unit operation for each of the four functions selected by the FUNCTION SWITCH. By the unit operation one means that operation which occurs just once each time the 1-SHOT button is pressed, or occurs continually after pressing and releasing the RUN button. It is set out, for each of the four cases, in the next section.

9.17 THE UNIT OPERATION. For the first case, COMPARE, the unit operation is as follows:

The two characters standing in the two reader registers are compared. If they differ, no action takes place. If they are the same character, this character is placed in the punch register, and then the next wanted character on the tape in each reader (as determined by the setting of the skip switch) is read into the corresponding register.

From this it follows that:

(a) a comparison operation cannot be started until the two reader registers have been set to the same character;

(b) if, at any time after a comparison has been started, differing characters have been read from the two tapes, the machine will immediately stop. The two differing characters will be displayed in the two reader registers, and the previous character, which was the same on both tapes, will be displayed in the punch register.

In the COMPARE & REPERFORATE mode the above description of the unit operation for COMPARE still applies as far as it goes. In addition, a character, having been placed in the punch register, is then punched. At the end of the operation the punch register thus displays the last character punched.

The mode labelled MANUAL PUNCH from selected register is normally used only for the addition of an extra character or two during an editing process. The unit operation consists simply of copying the character in the selected register into the punch register, and then punching it. If the RUN button should be pressed while the machine is switched to this mode, the punch will run out tape continuously punched with the character standing in the selected register.

In the REPERFORATE mode, only one reader is used, as determined by the SELECT switch. The unit operation is the same as just described for MANUAL PUNCH, except that while the character initially in the selected reader register is being punched, the next wanted character on the tape in the selected reader is read into the reader register. Thus, at the end of the operation the reader register displays the last character read, and the punch register displays the last character punched. It will be observed that the punch lags the reader by one character, and when a reperforate operation is started, the first character punched will not be the first character read, but the character initially standing in the selected reader register.

9.18 GENERAL OPERATING HINTS. With this equipment, as with the slower mechanical equipment, a reperforation process should always be followed by a comparison of the copy with the original tape. Furthermore, it will be wise to observe the rule that if the original tape has been read by reader 1 during reperforation, it should be read by reader 2 during comparison, and vice versa. This will reduce the risk of an error being missed because of a tape being read wrongly twice in succession in the same manner, as may happen if it is slightly malformed, and it is read by the same reader each time.

When comparing, the tape readers operate at their full speed of 200 characters per second, and this is quite fast enough for the operator to have difficulty in controlling the tapes if his technique is poor. It is suggested that when a comparison has been started, it should be stopped again after three or four feet of tape has run through, and the bins for reception of the tape arranged so that the tape runs fairly into them. Any attempt to fight tapes into the bins while the equipment is running will probably end in disaster. After re-starting the comparison, watch the tapes feeding into the readers, and hold a finger close to the stop-button so as to stop the process immediately any snarling of tape is observed.

Under no circumstances should one attempt to feed a tape of any length from the bottom of a bin. This may generally be avoided by carrying out the comparison backwards, starting at the end of the tape. Failing that, the tape should be handled into another bin, or rolled up and placed on one of the un-winding spools provided.

In order to bring the two tapes into step at the start of a comparison the simplest method is to make use of the 5th hole sections that normally appear at the ends of all tapes. If the skip switch is set to SKIP 5TH HOLE CHARACTERS, and the 1-SHOT button pressed, the tapes will jump forward to the first sexadecimal character, and thus be ready to start comparison.

The inclusion of the COMPARE & REPERFORATE mode in this equipment is to some extent experimental. It was fairly easily included, and has been suggested as a possibly useful operation in ensuring accuracy of data tapes. The process of printing out such tapes, and proof-reading, is not a very reliable method of detecting errors. The preferred method is undoubtedly to use an add-punch if check sums are avail-able for the data. This equipment consists of an adding machine with punching facilities. As data is punched, a total

is accumulated. This total may be used for checking purposes and also transferred to the tape, where it will serve to ensure correct input. An add punch will be available in the laboratory in the near future.

Failing this, a verifier should be used if possible. With a verifier, which consists essentially of a tape reader and a keyboard, a punched tape may be compared with the original manuscript by means of a second keying operation.

It is expected that a verifier will be made available to the SILLIAC laboratory on loan from C.S.I.R.O. in the near future, but at present there is not one available. An alternative, satisfactory in some cases at least, may be to use the equipment at present under discussion, in the COMPARE & REPERFORATE mode. The data would be punched twice, preferably by different operators, and the two tapes run through, producing a third tape so long as they agree. Discrepancies would be investigated as they occur, and the equipment operated so as to place the correct information on the third tape in each case (see below for suggestions). Other uses of this COMPARE & REPERFORATE mode may occasionally occur to the ingenious tape-editor.

When a tape being reperforated requires correction, the position of the error should be clearly marked on the tape so that the reperforation may be stopped as the portion in error approaches the reading position. The tape can then be advanced, using the 1-SHOT button, until in position for making the correction.

If the correction requires changing a character, the tape should be advanced until the character-in-error stands in the reader register. It can then simply be changed manually to the character required, and the reperforation continued. Do not switch to MANUAL PUNCH for this operation.

The MANUAL PUNCH function is provided for use when a character not present on the original tape is to be added. The character to be added should be set up in the register which is not being used for reperforation. The tape should then be advanced by 1-SHOT operation until the character preceding the one to be added has been punched, i.e. shows in the punch register. One may then reverse the setting of the SELECT switch, switch over to MANUAL PUNCH, press the 1-SHOT button to punch the extra character and then restore the original switch settings to continue reperforation.

Another method of adding a character, involving less switching, but a better grasp of how the equipment operates, is as follows: advance the tape until the character preceding the addition stands in the reader register, and so has not yet been punched. Turn to MANUAL PUNCH and press the 1-SHOT button to punch this character. Then manually set up in the reader register the extra character required, switch back to REPERFORATE and continue.

In order to delete a character, the tape is advanced until the unwanted character stands in the reader register. The reader JUMP button is then pressed to bring up the next character and reperforation continued.

When operating in the COMPARE & REPERFORATE mode, a wrong character can be corrected in the reader register, and the RUN button pressed to continue, without moving the FUNCTION SWITCH. An extra character on one tape can be deleted with the JUMP button, and the RUN button pressed to continue, again without moving the FUNCTION SWITCH. If however, a character is missing from one tape, the simplest procedure is to select the correct tape by the SELECT switch, turn the FUNCTION SWITCH to REPERFORATE, and press the 1-SHOT button. The tapes will then be again in step, so that one can turn back to COMPARE & REPERFORATE, and continue.

The above examples of editing operations are by no means exhaustive, but it is hoped that they will serve as sufficient illustration to enable the user to work out the procedure required in any particular case.

9.19 RECOVERY FROM MISHANDLING . If, when one wishes to stop the tape, one accidentally presses the wrong button, i.e. the 1-SHOT or RUN button, the machine will stop, but will go on again as soon as the button is released. If the error is noticed in time, one can recover from the situation without letting the tapes run on. If it is the 1-SHOT button that has been pressed, the STOP button should be pressed and released before releasing the 1-SHOT button. If it is the RUN button that has been pressed, the STOP button should be pressed and held down until the RUN button has been released. If the machine has been stopped in one of these abnormal ways, the punch register contents may be incorrect.

If a tape has been allowed to run right through a reader, the reader will be left running, looking for more tape. The STOP button should then be pressed to restore a normal condition before trying to commence another operation.

Any suspected malfunctioning should be reported immediately to the Program Librarian, or, if she is not available, to some other member of the Laboratory staff.

9.20 SAMPLE OPERATIONS OF COMPARER-REPERFORATOR.

Before use.

SWITCH FROM S/BY TO ON.

Reperforation of Tape.

PLACE TAPE IN READER AND SWITCH READER SELECTOR TO THAT READER.

SELECT REPERFORATION FUNCTION AND SKIP SWITCH POSITION.

SET DELAY IN READER REGISTER.

PRESS RUN BUTTON.

STOP TAPE AT END BY PRESSING STOP BUTTON.

Comparison of Tapes:

PLACE TAPE LEADERS IN READERS.
SELECT COMPARE FUNCTION.
SET DELAYS IN READER REGISTERS.
SET SKIP SWITCH TO "SKIP 5TH HOLE CHARACTERS",
AND PRESS 1-SHOT BUTTON.
SET SKIP SWITCH TO SETTING REQUIRED.
START BY PRESSING RUN BUTTON.

Continuing When an Error Stop Occurs:

LINE UP TAPES USING SKIP BUTTONS UNDERNEATH READERS.
PRESS RUN BUTTON.

Correction of Marked Error on Tape:

RUN AS IN REPERFORATE UNTIL ERROR APPROACHES READER.
PRESS STOP BUTTON TO STOP TAPE.

Proceed then according to the nature of the correction, e.g.

(i) To change a character:

ADVANCE TAPE USING 1-SHOT BUTTON UNTIL CHARACTER-
IN-ERROR STANDS IN READER REGISTER.
CHANGE CHARACTER MANUALLY IN REGISTER.
PRESS RUN BUTTON TO CONTINUE.

(ii) To add a character not on the tape:

ADVANCE TAPE USING 1-SHOT BUTTON UNTIL CHARACTER
BEFORE POINT OF ADDITION STANDS IN READER REGISTER.
SELECT MANUAL PUNCH FUNCTION.
PRESS 1-SHOT BUTTON.
MANUALLY SET UP CHARACTER TO BE ADDED IN REGISTER.
SELECT REPERFORATE FUNCTION AND PRESS RUN BUTTON.
TO CONTINUE.

(iii) To delete a character or characters:

ADVANCE TAPE USING 1-SHOT BUTTON UNTIL UNWANTED
CHARACTER STANDS IN READER REGISTER.
PRESS READER SKIP BUTTON UNTIL A DESIRED CHARACTER
STANDS IN READER REGISTER.
PRESS RUN BUTTON TO CONTINUE.

After Use.

SWITCH FROM ON TO S/BY.

9.21 A FEW REMINDERS. These notes in this section are intended to assist in the rapid acquisition of good technique in handling tape. Furthermore, the amount of tape editing equipment that can be made available is strictly limited, and has to be used by a large number of people. In these circumstances it is of the greatest importance that all users, to their mutual advantage, use the tape editing room in such a manner as to cause the minimum of inconvenience to other users. In particular the following injunctions should be carefully observed:[1]

1. Do not leave waste tape, paper or other rubbish lying about on tables or in tape hoppers.
2. Always use the hoppers provided for handling loose tape; tape which is dusty from trailing on the floor has a bad effect on the photoelectric reader.
3. All tapes should commence with a leader about eight or nine inches long consisting entirely of 5th-hole characters. The standard form for this leader is: about 7 inches of delay characters, followed by a carriage return and line feed, followed by about 2 inches of figure-shifts, followed by another carriage return and line feed, followed immediately by a delay and the first character of the program or data on the tape.
4. At the end of a tape run out with a few carriage-return/line-feed characters to get the printed copy

---

[1] Some of these points have already been made elsewhere. They are included here again for convenience of reference.

---

clear of the platen, and then with a number of delay characters to give a tail of several inches on your tape. The difference between the block of figure shift characters at the beginning and the block of carriage-return/line-feed characters at the end is sufficient to distinguish one end from the other, but in addition it is desirable to trim the ends thus:

as a further precaution against the common error of inserting a tape in a reader the wrong end first.

5. All tapes should be labelled on the leader with the owner's name, a short title or description of the material on the tape, and the date.

6. Every effort should be made to keep the number of separate tapes involved in any one computer run to a minimum. It should not be necessary to use more than two tapes in most cases, one having all program material (including diagnostic routines for use during development) and the other having all parameters and data. In many cases these can be combined in the one tape.

7. The several sections of a complete tape should be separated by blocks of several inches of delay characters where labels should be written indicating the content of the following section. Moreover, in punching up a routine, it is desirable to leave a gap of about a dozen delay characters at the end of each group of ten orders: this enables a section of the tape to be located quickly.

8. If you have several sections of tape prepared at different times, they may be joined up, to comply with 6. above, by splicing. Splices are made with $\frac{1}{2}$" black Sellotape and should be preceded and followed by four inches of delay characters. Not more than two thicknesses

of black tape should be used (one each side of the paper tape), and the black tape should cover completely an integral number of rows of holes. Any black tape extending beyond the edge of the paper tape should be trimmed off with scissors. Such splices will run through the Ferranti photoelectric reader, but are liable to damage by the sprocket if they are run through the mechanical readers.

9. When using a tape-editing station, turn on power and motor switches as required, and turn off again when you have finished, but during operation control the equipment by means of the black key-switch alone. Switching motors off and on while reperforating is in progress will certainly cause errors.

10. When punching a directive 00n+, punch it as 0000000000n+. The extra zeros do no harm and provide space for manually punching a new directive if a change has to be made. By this means the reperforation of a tape can often be avoided.

11. After preparing a tape from manuscript throw away the printed copy and then print out the tape to obtain a fresh printed copy for proof-reading.

12. Never assume that the equipment has operated without error. In particular whenever a tape has been reperforated, the new tape must be compared by machine with the original, to check that they are in fact identical.

13. Library tapes must be treated with great care, and must be replaced in their boxes and returned to their drawers immediately after use. If a library tape accidentally becomes damaged, do not simply put away the damaged tape, but hand it to the Librarian or other responsible person.

14. Record all malfunctioning of the tape editing equipment in the book provided in the tape editing room.

15. No responsibility whatever can be accepted for tapes or other material left lying about in the tape room. In particular, unlabelled tapes will be thrown out without question. Labelled tapes will be preserved for a day and then thrown out if still not claimed.

16. Avoid using blocks of erases except in emergencies. They weaken the tape.

17. When inserting corrections by splicing in a length of tape containing a directive and the corrected orders, ensure that it is inserted in the correct position in the tape, that it is not inserted back to front, and that addresses of orders with D and L parameters are converted to fixed form (unless the correct preset parameters are available in the required addresses).

18. Only in cases of extreme emergency should individual holes be covered with Sellotape. The Sellotape inevitably becomes dislodged (unexpectedly restoring the tape to its original form) and often adheres somewhere else - or gets caught in the tape reader.

19. Beware of "chads" - the circular discs of paper which are punched from the tape. These can cause errors if they find their way into a tape hopper. If this should happen - e.g. by the inadvertent spilling of the contents of a chad box - put the hopper on one side and inform the Librarian or some other responsible person.

9.22 PRINTING LAYOUT. When printing layouts are being designed, they should be based on the assumption that the Teletype machines are set up to give a margin of one inch on the left hand margin, and 70 characters across the page. The line spacing is six to the inch.

Rolls of paper are available which will provide

an original and one carbon, and rolls providing an original and three carbons can be obtained. Wax stencils can be cut by the Teletypes if the hammer tension is adjusted, but require to be fed in page by page. Continuous stencil suitable for a spirit duplicator is also available, and is quite satisfactory for runs of several hundred copies.

For a quarto layout, overall dimensions of 10" x 8" should be used as a basis for planning. For a foolscap layout, the figure is 13" x 8".

9.23 THE N.C.R. RECORDING ADDER. This unit consists of a ten column adding machine coupled to a paper tape punch and is used for the punching of sequences of numbers together with check sums, the sums being formed by the adding machine as the punching takes place. The paper tape punched by the recording adder can be read by the SILLIAC input routine N31, but cannot be read by any input routine which does not accept check sums.

If data is being prepared for a SILLIAC library program, the recording adder should be used only if the instructions for using the program state explicitly that a tape prepared by the recording adder is acceptable.

9.231 CONTROLS

Before starting to use the recording adder locate the following controls. Full details of their use will be described later.

THE ADDER

Main Switch

This is located on the right hand side of the front panel of the adding machine. Push it to the left to turn the machine on. One of the two pilot lights will then glow. (If neither light goes on, the switch at the wall socket may be off). Do not neglect to turn the machine off with this switch before leaving it.

Punch Control Key

This key is located between the main switch and the pilot lights. If the recording adder is to be used to prepare a punched tape this key should be turned to the right until the green light goes on. If the machine is to be used purely as an adding machine turn the key to the left until the red light glows.

Digit Keys

The keyboard consists of ten columns of keys indicating the digits 1 to 9. Only one key in any one column should be depressed at one time. A zero is indicated by having no keys depressed.

The adding machine always accepts, prints, and adds numbers of up to ten digits, but the punch is wired to record on tape only a certain number of digits. It can be wired to record all ten digits, the nine, six or three <u>right hand</u> digits -- giving four different choices of number length. Eventually a switch will be attached enabling manual selection of number length, but at the moment the machine is normally set to punch the right hand six columns and can be readjusted only by the maintenance engineer who will do so on request. (This adjustment may take up to an hour to carry out).

With the machine set to record six digits, fractions should be entered on the keyboard with the decimal point between the fourth and fifth columns (counting from the left) -- note that there is a change of key colour here -- and integers with their decimal points on the far right hand side of the keyboard.

Keyboard Clear Key

This button, marked "C", is located at the bottom left hand corner of the keyboard and is used to re-set the keyboard to all zeros. It is used to correct errors in setting numbers in the keyboard.

Void Key

This button, marked "Void", is located at the top left hand corner of the keyboard. It is used to correct numbers which have been wrongly entered and recorded. The function of the Void key is to punch a special void symbol (the sexadecimal digit J). The void symbol is not punched directly by the Void key. This key is "set" by depressing it, and when in the set condition it causes a void symbol to be punched as part of the information recorded on actuation of any one of the operating keys. The Void key then is automatically reset.

Operating Keys

The keys which initiate a cycle of operation of the adder and punch are grouped in a column on the right hand side of the keyboard. They are, from top to bottom, as follows.

Total Key

This button is marked with an asterisk. It should be actuated

only when the keyboard is clear. It causes the total accumulated by the adding machine to be printed and punched, and the total in the machine to be cleared to zero. The symbols F and N are punched on the tape immediately before and after the total, respectively.

The Total key is used to terminate a sequence of numbers. The total that it punches is used as a check sum by SILLIAC.

Sub-Total Key

This button is marked "S". It causes the total accumulated by the adding machine to be printed only -- nothing but a CR/LF is punched and the total is not cleared. The Sub-total key is used to:

1) Punch CR/LF's within a sequence of numbers so that the numbers will print in separate groups.[x]

Footnote: [x]It should be pointed out here that, owing to the check provided by the check sum, it is not necessary to print and proof read the tape punched by the recording adder, but this may be done if desired. If the tape is printed out the numbers will print in a single solid column. Sequences are not separated by blank lines, but the sequences are easily identified by the N's printed after each check sum.

---

2) Print a sub-total at any point in a sequence where such a sub-total has been provided by the originator. This provides a printed proof on the adding machine roll only.

3) Punch a CR/LF after each sequence and provide a printed proof that the total within the adding machine is cleared between sequences.

4) Punch a void symbol without punching a number from the keyboard. This is done by using the Void key in conjunction with the Sub-total key. By this means one can punch a sequence of successive void symbols.

Minus key

This button is used to enter a negative number in the sequence. The number is entered on the keyboard, and the Minus key then depressed. The number in the keyboard is punched on the tape, preceded by a minus sign, and the number is subtracted from the total.

Plus Key

This bar is used to enter positive numbers. The number is entered on the keyboard and the Plus key then depressed. The number is punched, preceded by a plus sign and is added to the total.

Code Key

This key is identified by the symbol #. If a number is entered in the keyboard and this key depressed, the number will be punched, preceded by the symbol L, and will not be added to the total. The Code key is therefore used for runs of zeros, code numbers and stop symbols.

Repeat Key

This key is located immediately to the left of the Plus key. It is used to record a string of equal positive or negative numbers. It has three positions, up, down and central, of which the central position is the normal or "off" position. When set down it inhibits the clearing of the keyboard which normally occurs immediately after operation of the Plus, Minus or Code keys. A number entered in the keyboard will remain there and may be recorded by any of the three above-mentioned operating keys any number of times. The operating key may be held down while the machine cycles through the repeated operation.

When pushed upwards the repeat key becomes an operating key, acting in the same way as the Plus key, but recording the number in the keyboard repeatedly until released.

Space Control Lever

This lever is located towards the left hand side on the top of the adding machine. It controls the printing and spacing of numbers and totals on the adding machine roll. It has no effect on the punching operation. The lever has four positions:

a) Answer only. Only totals and sub-totals are printed, with single spacing.

b) 1 Space. Numbers are printed with single spacing, totals and sub-totals with double spacing.

c) 2 Space. Numbers are printed with double spacing, totals and sub-totals with triple spacing.

d) Total space. Numbers are printed with single spacing, totals and sub-totals with multiple spacing, so providing a large spacing between sequences.

Operators will generally find it best to use the "Total Space" position if they do not make very much use of the Sub-total key, and to use the "1 Space" position if they do make a practice of using the Sub-total key.

Total Dials

Just above the keyboard, through a glass window, can be seen the Total Dials. The number visible here is the number that will be recorded on actuation of the Total key.

THE PUNCH

Raise the cover of the punch. On the right hand side is a take-up spool. The tape may be threaded on to this spool, and will be automatically taken up as it is punched -- but note that this results in a spool of tape wound in the reverse direction. It is generally more convenient to leave the cover of the punch open (or partly open) and allow the tape to fall into a plastic bin.

On the front of the punch near the bottom left hand corner, is the run-out button. This causes the tape to run out on delays. (There is no way of running out figure shifts or carriage returns).

The tape feed spool is located an the left hand side of the punch. See that there is sufficient tape here for your purposes before starting. If a new spool of tape is required inform the SILLIAC operator, and she will arrange for a new one to be mounted. The same applies if the tape should break. (The punch is equipped with safety devices which stop it if the tape should break or run out).

9.232 PREPARING A SILLIAC DATA TAPE

Before starting:

1) Check that there is sufficient tape in the punch feed spool.
2) Check that there is sufficient paper in the adding machine roll.
3) Check that the total held by the adding machine is zero. If not, clear it by depressing the Total key.
4) Make sure that the punch is turned on. (Turn the Punch Control key so that the green light glows).

5) Make sure you know which columns on the keyboard are "active", i.e. are being punched onto the paper tape. Normally the six right hand columns are active.

6) Run out the tape to give a clear leader of about eight inches of delays. Tear off any section of tape containing spurious punchings, including the total that may have been punched in step 3.

It is now just a matter of entering the individual numbers in the active columns of the keyboard, and actuating the appropriate operating keys. The following notes describe how different types of entries are made.

Normal Members of the Sequence.

Enter the number in the keyboard. If a fraction, it is to be entered so that its decimal point is located just to the left of the left-most active column. If an integer, so that the decimal point is just to the right of the far right hand column.

Record the number by pressing either the Plus or Minus key, depending upon the sign of the number.

If several successive members of the sequence have the same magnitude (their signs may be different), the Value may be locked in the keyboard by using the Repeat key.

Runs of Zeros

To record a run of n zeros, enter the integer n in the keyboard and press the Code key. Note that n must lie within the range $1 \leq n \leq 100$.

Terminating a Sequence

A sequence is terminated, and the check sum punched, by pressing the Total key while the keyboard is clear. The punched tape should then be run out a couple of inches so that the SILLIAC operator can see where each sequence begins. This enables her to repeat the reading of any one sequence when the check sum test fails.

It is also good practice then to press the Sub-total key. This provides a printed proof that the total has been cleared to zero, and punches an additional CR/LF to separate the sequences.

If a pre-calculated check sum has been provided the total formed by the adding machine should be checked against it before the Total key is pressed and the sequence terminated. If the sums do not agree the sequence should be voided by pressing the Total key after setting the Void key. The sequence should then be repeated. If the total again disagrees with the check sum provided, it should be compared with the total printed at the end of the previous (voided) sequence. Agreement of these two totals is an indication of an error in the data or its check sum as provided.

In some application it may be desired that the sum of the elements of each sequence should be appended as an additional element at the end of the sequence. If such a "sum element" is required it must be entered as an additional member of the sequence, for the check sum punched automatically at the end of each sequence cannot be used for this purpose. If the members of the sequence are fractions it will generally be necessary to scale the sums down (say by a power of ten) to reduce them also to fractions.

When such sums are provided with the data they can be used by the adding machine operator as checks by comparing the number in the Total Dials with the check number as the latter is entered in the keyboard.

Code Numbers

Any integer greater than 100 can be recorded as a code number by entering it in the keyboard and pressing the Code key.

Stop Symbol

A stop symbol is punched by pressing the Code key twice in succession with the keyboard clear.

Void Symbol

If it is desired to punch the void symbol J without it being followed immediately by a number, set the Void key and press the Sub-total key.

9.233 CORRECTING ERRORS

The method of correcting an incorrectly entered number depends upon when the error is disoovered and also upon which operating key was used to enter the number.

1) If the error is discovered before the number has been recorded, i.e. before an operating key has been pressed, clear the keyboard with the Keyboard Clear key and enter the number again.
2) If the error is discovered after the number has been recorded, but before the next number has been recorded, the method of correction depends upon which operating key was used:
   a) If an incorrect number has just been recorded by the Plus or Minus key (this also includes recording the correct

number with the wrong operating Key) it can be corrected by depressing the Void key, setting the correct number in the keyboard, then pressing the correct operating key (which may be Plus, Minus or Code).

The correct number will then replace the incorrect one when the tape is read by N31, but the effect of the incorrect number on the check sum is not voided. Hence, if a pre-calculated total has been provided, this should be preserved by first entering the wrong number again, using the Void key and the operating key of opposite sign, then entering the correct number, again with the Void key and the correct operating key.

b) If an incorrect number of zeros has been entered by use of the Code key some ingenuity may be required to correct the error. This is because the function of the void symbol is to void the previous member of the sequence of numbers. Hence a void symbol punched after a run of zeros voids only the last zero of the run, thereby reducing by unity the number of zeros in the run.

If the number of zeros is too small, then one simply records the remaining number of zeros by using the Code key again.

If the number of zeros is too large by exactly unity, then the Void key should be used with the next number in the sequence which will then replace the last of the zeros.

If there are just a few too many zeros they can be voided by repeating the operation of setting the Void key and pressing the Sub-total key. Each time this is done one zero is voided -- also of course the sub-total is printed. (It is a good plan to set the Space Control Lever to Answer Only while doing this so that all the sub-totals will be printed with single spacing). This procedure should not be used if the excess number of zeros is greater than the number of elements remaining in the sequence. This is because the incorrect zeros are actually voided by being overwritten by the following numbers, and hence are not voided if there are not enough numbers to overwrite them all. In this case the procedure outlined in the following paragraph should be followed.

If there is a large number of excess zeros, but still no more than 100 zeros altogether, the only cure available which is always "safe" is to mark the punched tape at the place where it has **just** been punched and later erase the number, together with its preceding L, by use of the one-hole punch. The erasing of a number entered with the Code key is allowed because such numbers are not added to the check sum.

If the number incorrectly punched with the Code key is greater than 100, then the accident should be ignored completely, for such a number will be interpreted by N31 as a Code number and will therefore be effectively ignored. An

attempt to void this number will result in the voiding of the preceding member of the sequence.

If a number which is supposed to be recorded by the Plus or Minus key is accidentally recorded by the Code key, it is usually true that this number (treated as an integer) is greater than 100, and hence the accident can be ignored. If such a number should not be greater than 100 then the appropriate corrective procedure outlined above must be carried out.

If the Code key is depressed accidentally when the keyboard is clear, the punching that results will be interpreted by SILLIAC as part of a stop symbol. This error cannot be voided by use of the Void key. It can be voided only by erasing it from the tape by use of the one-hole punch. Otherwise, complete the stop symbol by pressing the Code key again and inform the SILLIAC operator that the stop which will occur at this point is to be by-passed. If the stop symbol is not completed in this way (and the error not erased from the tape) it will still act as a stop symbol, but will also cause the following number to be ignored.

3) If an error is discovered after only a few more numbers have been recorded, then the incorrect number and the following numbers can all be voided by use of the Void and Sub-total keys. Each time the Sub-total key is pressed while the Void key is set one number is voided from the sequence. This procedure is not satisfactory if a pre-calculated check sum has been provided; in this case the whole sequence should be voided.

4) If an error is discovered some time after it is made, or if, at the end of a sequence, it is discovered that the total is incorrect, then the whole sequence should be voided. This is done by setting the Void key and pressing the Total key. The sequence can then be started again.

Further Information

Persons desiring more information about the mode of operation and capabilities of the Recording Adder should consult document number I1 30 in the Laboratory pamphlet library.

CHAPTER 10

CALCULATION OF RUNNING TIME

Each programmer should estimate as well as he can the length of time his program will run. This information is needed for efficient scheduling of machine time and to enable the computer operator to decide whether a program may have failed to stop when it should have.

The estimation of running time consists of summing up the times for executing all of the orders of the program, taking into account that many orders are executed more than once. Thus, the time is calculated for one passage through an iterated loop and then multiplied by the number of trips through the loop. The number of passages through a loop is not always known (as in the square root code where the number of passages depends upon the quantity whose square root is being calculated) and the programmer may have to make rough estimates for such cases. He can make use of the code checking periods to help him here.

10.1 ORDER TIMES The programmer must know how much time the SILLIAC requires for each order. The order times are given in Table 10.1, the time required for getting the order from the memory being included.

The values given in Table 10.1 are in some cases not exact. The 700 microsecond value for multiplication is a maximum, the time depending upon the multiplier. It could be as low as 625 microseconds.

ORDER TYPE	
On, 1n	16 n microseconds
2,3,4,5,J	55 microseconds
3 not executed	18 microseconds
6	800 microseconds
7	700 microseconds
80, 81	4 milliseconds/character
82, 92	17 milliseconds/character in punch
+,-,F,L	75 microseconds

Table 10.1
Order Times

The time required to execute an 80 or 81 order so as to read one character from the tape is 4 milliseconds. Of this time just over 2 milliseconds is available for computing: if the computing time is longer than this, input will be slowed down. Similarly, with output, 4 milliseconds will be available for calculations. If a greater time than this is required between characters, a computer cycle of the punch will be missed: i.e. the output time per character will double.

10.2 EXAMPLE OF RUNNING TIME CALCULATION Let us consider a simple example. The program given in Table 10.2 will transfer the contents of memory locations 100 to 199 into memory locations 200 to 299. How long will it take?

The program in Table 10.2 consists of 7 words. There is a loop consisting of the 9 orders beginning with the left-hand order at location 0 and ending with the left-hand

order at location 4. This loop is executed 100 times. There are no other orders executed except the stop order.

Thus we have the following orders:

```
      00  50+
0     L5  (100)F
      40  (200)F
1     L5  L
      L0  5L
2     32  4L
      L5  L
3     L4  6L
      40  L
4     26  L
      0F  F
5     L5  199F
      40  299F
6     00  1F
      00  1F
      24  50N
```

Table 10.2

Repetitive Program

Five L orders at 75 microseconds,
Two 4 orders at 55 microseconds,
One 2 order at 55 microseconds,
One 3 order at 18 microseconds.

The time T is then given as

$$T = (5 \times 75 + 2 \times 55 + 18)\ 100 \text{ microseconds},$$
$$= 53{,}800 \text{ microseconds},$$
$$= .054 \text{ seconds}.$$

10.3 A SIMPLE RUNNING TIME FORMULA A rule which is accurate enough for most calculations is the following one:

Let N_O = number of orders obeyed,
N_m = number of multiplication orders obeyed,
N_d = number of division orders obeyed,
N_s = number of shifts.

Then the running time T, exclusive of input and output orders, is

$$T = \frac{N_O}{16} + \frac{N_s}{60} + \frac{N_m + N_d}{2} \text{ milliseconds.}$$

In the example given above N_O is 900 while $N_s = N_m = N_d = 0$. We thus have T = 900/16 milliseconds or about .056 seconds.

22nd Jan., 1959.

MODIFICATIONS TO CHAPTER 11 OF SILLIAC HANDBOOK (PROGRAM A9)

Page

11-16 Program A9 has now been modified so that it does not stop after printing out ROOT NEG, LOG NEG or TOO BIG, but continues to compute. In the case of the square root or log, the faulty argument remains in the accumulator. In the case of exceeding capacity, the number which is too large remains in the accumulator, and any attempt to store it will result in an incorrect value being stored.

11-14, 11-17 Since π is a value frequently required in computations, A9 has been modified to store π in the first location of the block of storage reserved for constants. Thus the number of other constants which may be stored is now 60. The address of π for use in programming is 18F. Thus the instruction 85 18F places π in the accumulator.

11-14, 11-17 The diagnostic routine no longer uses the last 12 locations of the -J block. Instead, it overwrites the constant listing routine and parts of A1 and the -F block. These overwritten portions are reset by the resetting routine. It is now quite impossible to use the diagnostic routine twice in succession without resetting in between.

11-13 The diagnostic routine has now been modified so that the user may choose how many digits of the accumulator he would like printed, and also at what point in his program he would like printing to commence. The maximum number of accumulator digits which may be printed is 9. If zero digits are specified, the diagnostic routine will print out the location of instructions obeyed only, and not the contents of the accumulator.

As before, until (iii) is reached. After bootstrapping in the diagnostic routine, SILLIAC stops on a 20 order. When started with the black switch it expects to read a specification tape of the form

```
OOpF  OOF
OOF   OOmF
OOF   OOnF
24 16N
```

where p is the number of accumulator digits to be printed out.

m is the number of instruction at which printing is to start (instructions are numbered from 0).

n is the number of times instruction m is performed before printing starts.

E.g.
```
009F  00F
00F   003F
00F   0010F
24 16N
```
specifies that 9 digits of the accumulator are to be printed out, the printing to start when instruction number 3 is performed for the 11th time.

If printing is to take place as soon as the program starts, m and n should both be specified as zero. After this specification tape is read in, SILLIAC stops on a 24 order, ready to perform item (iv) of the operating instructions.

For convenience, immediately after the diagnostic routine on the A9 tape (and before the resetting routine) is the specification

```
009F  00F
00F   00F
00F   00F
24 16N,
```
which will cause printing to start at the beginning of the program, 9 digits of the accumulator being printed.

CHAPTER 11

A SIMPLIFIED CODING SCHEME

The coding scheme A9 described in this chapter has been prepared to enable programs to be written by coders with little or no knowledge of SILLIAC. It is also a convenient scheme for even experienced programmers when programming time is the major consideration, and computing time of secondary importance.

The unstarred sections of this chapter have been written with the neophyte in mind. It should be possible to make use of the scheme with no more knowledge of computers than is contained in these unstarred sections.

The scheme is an interpretive one. However, for convenience the words "accumulator", "instruction", etc., are used throughout the unstarred sections of the chapter without any special indication that they refer, not to the machine accumulator or machine instructions, but to the "accumulator" of the interpretive scheme and interpretive "instructions".

11.1 PREPARATION OF INSTRUCTIONS. To instruct SILLIAC to carry out a calculation under the control of the program A9, we must write a list of "instructions", each such instruction specifying one step in the computation. A single step may, for example, be to add a number into the accumulator, or to multiply the contents of the accumulator by a number, or to replace the number in the accumulator by its logarithm, etc. Each instruction must be prepared in a coded form. The symbols used in this code are the numerical digits 0,1,2,3,4,5,6,7,8,9, the two signs + and -, and four alphabetical characters N,J,F and L. The fact that the number of symbols used, namely 16, is a simple power of two is connected with the binary code used both on the punched paper tape and inside SILLIAC.

The number of instructions in any program written according to this scheme must be less than 201. For an exact definition of this number and means of extending it, the reader is referred to section 11.13. For data, 163 registers are available: this too can be increased (see section 11.13).

Each instruction consists of three parts, viz:

(1) The first part is a single character and is normally an 8. In some instructions we can use a different character as will be explained later.

(2) The second part is also a single character and can be any one of the 16 symbols. It specifies the type of operation to be performed. Thus, for example, 84 means "add a number to the accumulator" while 8- means "copy the contents of the accumulator into a memory register". The combination 8J means "compute one of the special mathematical functions (such as square root or logarithm)".

(3) The third part of the instruction we call the "address" part because in the simpler arithmetic instructions it specifies which register in the memory is to supply the number concerned. In other types of instructions the address part has different meanings as can be seen from the following table.

11.2 TABLE OF INSTRUCTIONS UNDERSTOOD BY PROGRAM A9

Group I. ARITHMETIC ($0 \leq n \leq 162$)[1]

80 n-N	Subtract the number in memory register n from the accumulator.
81 n-N	Put minus the number in register n into the accumulator.
84 n-N	Add the number in register n into the accumulator.

---

[1] See however section 11.13.

---

85 n-N Put the number in register n into the accumulator.

86 n-N Divide the number in the accumulator by the number in register n.

87 n-N Multiply the number in the accumulator by the number in register n.

8N n-N Subtract the arithmetic value of the number in register n from the arithmetic value of the number in the accumulator.

8- n-N Copy the number in the accumulator into register n.

8+ nF If $0 \leq n < 200$ put the integer n into the accumulator.

If $200 \leq n < 400$ put the integer n-400 into the accumulator.

Group II. FUNCTIONS

Each of these instructions replaces the number in the accumulator by a certain function of it. The functions are as follows:

8J -6 Square Root.

8J -7 Sine.

8J -8 Exponential. (I.e. replace x by e^x.)

8J -9 Natural Logarithm.

8J -+ Arc Tangent. (I.e. angle whose tangent is x, in radians.)

8J -- Cube Root.

Group III. INPUT AND OUTPUT

88 F Read a number from the punched paper tape into SILLIAC and place it in the accumulator.

89 nF Print out the number in the accumulator to n significant digits. $1 \leq n \leq 9$.

8F nF Print the next number on a new line and all future numbers in a group of n columns.

Group IV. BREAK SEQUENCE

Those instructions to which a transfer of control is to be made must be preceded by a tag, or label, of the form //nL. The tag number, n, is an arbitrarily

assigned positive integer which must be less than 2048 and greater than zero. Although the tag numbers may be chosen at will it is generally advantageous to allot them serially, starting at 1, in order to avoid duplication.

82≠nL If the number in the accumulator is positive (or zero) carry out the instruction preceded by the tag ≠nL next. If the number is negative go straight on.

92≠nL Carry out the instruction preceded by the tag ≠nL next.

90≠nL Stop. When re-started with the black switch the instruction preceded by the tag ≠nL will be carried out next.

Group V. SPECIAL

8J -F Stop. This instruction must be the last one to be carried out in every program.

26 16N Start. This instruction must appear at the end of the program tape to instruct SILLIAC to stop reading instructions from the tape and to start carrying them out.

24 16N This instruction performs the same function as 26 16N, except that the program will stop after it is read into the machine, and must be restarted with the black switch in order to carry out the instructions. This stop enables a diagnostic routine (see below) to be read into the machine if it is found necessary to trace the course of the program in detail because of some error in the list of instructions,or to place a separate data tape in the tape reader.

11.3 REPEATED OPERATIONS. We sometimes want to repeat a sequence of steps a certain number of times. An example of

this would arise if, for example, we had to compute log(sin x) for 10 different values of x. To facilitate counting the number of repetitions we are provided with a set of eight separate counting registers and the following instructions.

b+ pF Prepare to execute a sequence exactly p times, using counting register b to count the number of times the sequence is repeated.

b2 ≠ nL Carry out the instruction preceded by the tag ≠ nL next the first p-1 times, then go straight on. The value of p is as specified by the preceding b+ pF instruction.

The first of these orders is placed just before the beginning of a sequence, the second at the end. The digit b may have any one of the eight values 0 to 7.

For example, to read in 12 numbers x and print out the 12 values of log(sin x), we would write:

```
          8F 3F          Print in 3 columns.
          0+ 12F         Prepare to repeat 12 times.
≠  1L     88 F           Read x.
          8J -7          Compute sin x.
          8J -9          Compute log(sin x).
          89 5F          Print to 5 figures.
          02≠1L          Repeat the sequence from
                          instruction tagged with
                         ≠ 1L.
          8J -F          Stop.
          26 16N         Start program.
```

A repeated sequence of this form is generally known as a "loop".

We can write a loop inside a loop too. Thus to compute $x^{1/8}$ for 16 values of x we would repeat the square root operation 3 times on each of the 16 numbers.

```
        8F  4F      Print in 4 columns.
        0+  16F     Prepare to repeat loop number
                    0 16 times.
#1L     88  F       Read x.
        1+  3F      Prepare to repeat loop No. 1
                     three times.
#2L     8J  -6      Compute square root.
        12  #2L     Repeat loop 1 from instruction
                    tagged with #2L.
        89  7F      Print to 7 figures.
        02  #1L     Repeat loop 0 from instruction
                    tagged with #1L.
        8J  -F      Stop.
        26  16N     Start.
```

When a loop is to be repeated, not a specific number of times, but as often as is required to obtain a convergence of some iterative process we use the instruction 82#nL which repeats the loop as long as the number in the accumulator is still positive.

11.4 MODIFICATION OF ADDRESSES Suppose we wish to read a list of 50 numbers into SILLIAC and place them in consecutive memory registers, say into registers 0 to 49. To do this we need a loop which is repeated 50 times and which contains an 8- instruction (copy to memory) whose address part is increased by unity each time the loop is repeated.

Special provision is made for us to do this. We simply replace the first digit 8 of the 8- instruction by the digit b where the value of b must be the same as the loop number.

Thus, to read in 50 numbers and place them in registers 0 to 49 we would write

```
          0+ 50F     Prepare loop 0 for 50 repeats.
#1L       88 F       Read a number.
          0- 0-N     Store in memory registers 0
                     onwards.
          02#1L      Repeat loop 0 from instruction
                     tagged with #1L.
          Next instruction.
```

11.5 DATA . Each number that is to be used in the calculation must be punched on the paper tape in a special form. The number must first be written as a fractional part multiplied by a power of 10. The fractional part and the exponent (an integer) are then punched, each preceded by a sign (either + or -). For example, to punch the number 1.47, we write

$$1.47 = 0.147 \times 10^{1}$$

and punch it as

+147+01.

The fractional part may have any number of digits up to a maximum of 9, but the integral exponent must always have two digits. The exponent must not be larger than 63 in magnitude, otherwise the number will be outside the allowable range of magnitudes.

The number zero is punched as ++00.

The numbers on which we operate are normally taken into SILLIAC by a read instruction (88 F). The numbers are accordingly punched onto the program tape immediately after the 26 16N instruction which starts the program operating. However, if we need a constant which is used in only one or two places in the program it is easier to specify the value of that constant within the program itself. An example should make clear how this is done. If we wish to multiply the number in the accumulator by 0.3 we can do this by writing the instruction

This trick of using the address part of the instruction to specify the actual value of the number used, instead of the register in which it may be found, can be used only on the arithmetic instructions in Group I. The instruction 8+ nF can be used for putting small integers into the accumulator.

A list of useful constants is included as Appendix III of this handbook.

11.6* ADDITIONAL LOOP INSTRUCTIONS. Each loop counting register, or index register, consists of a pair of ten binary digit integers (g_b, c_b). When one of the arithmetic type instructions is modified by an index register, the number g_b is added to the address before the instruction is performed, the instruction in the store being unaltered.

The instruction b+ pF replaces g_b, c_b by 0, -p.

The instruction b2//nL replaces g_b, c_b by g_b+1, c_b+1, then transfers control to the instruction tagged with //nL if (c_b+1) is negative.

There are in addition two more instructions which modify index registers, viz.:

bL nF Replace g_b, c_b by g_b+n, c_b.

8L nF Replace g_b, c_b by n, c_b where b is the index register referred to by the previous order using index registers.

TABLE 11.1 INSTRUCTION CODE FOR PROGRAM A9

Note: An asterisk precedes instructions mentioned only in sections marked with an asterisk.

I. ARITHMETIC

x = the number in register n
$0 \leq n \leq 162$.
Addresses of form $\overline{n}$-N may be replaced by N followed by the actual number.

85 n-N	Put x in A.
84 n-N	Add x to A.
81 n-N	Put -x in A.
80 n-N	Subtract x from A.
87 n-N	Multiply by x.
86 n-N	Divide by x.
8N n-N	Subtract |x| from |A|.
8- n-N	Store A in register n.
8+ nF	Put n in A.
8+ mF	Where m=400-r, put -r in A., where $r \leq 200$.

V. SPECIAL

8J -F	Stop the program.
26 16N	Stop reading the program and start obeying it.
24 16N	Stop reading the program. Stop the machine. Start with the black switch to obey the program.
* bL nF	Replace g_b, c_b by g_b+n, c_b.
* 8L nF	Replace g_b, c_b by n, c_b (where b is last index register used).

II. FUNCTIONS

8J -6	Square Root
8J --	Cube Root
8J -8	Exponential
8J -9	Natural Logarithm
8J -7	Sine
8J -+	Arc Tangent

III. INPUT AND OUTPUT

88 F	Read a number into A.
89 nF	Print A to n digits $1 \leq n \leq 9$.
8F nF	Print future numbers in n columns.

IV. BREAK SEQUENCE $0 \leq b \leq 7$

82 #nL	Go to instruction tagged with #nL if A positive.
92 #nL	Go to instruction tagged with #nL.
90 #nL	Stop. Go to instruction tagged with #nL when started with black switch.
8+F	Set A to zero (therefore positive)
b+ nF	Loop b to be repeated n times
b2#nL	Repeat loop b by going to instruction tagged with #nL

11.7 EXAMPLE OF USE OF PROGRAM A9. As an example of a program written for interpretation by the program A9, let us compute values of

$$\log_{10}(\sin x)$$

for values of x from 5^{o} to 90^{o} in steps of 5^{o}. We will go through the process of preparing the program step by step.

First we note that the SILLIAC program measures all angles in radians and computes only natural logarithms. Consequently we will have to convert x to radians by multiplying it by $\pi/180$, and we must also convert $\log_e(\sin x)$ to $\log_{10}(\sin x)$ by dividing it by $\log_e 10$.

Secondly we note that we will need to use 18 separate values of x, so we will set up a loop to be repeated 18 times. Then we decide that we will get a useful layout if we print our answers in four columns, thus

5^{o}	10^{o}	15^{o}	20^{o}
25^{o}	30^{o}	--etc.	

the last row containing only two entries. We must remember that this is the layout so that we can interpret the answers that SILLIAC produces.

We next plan the "logic" of the program, that is, we write down the various steps that must be carried out. At this stage we do not bother about using the coded form of the instructions.

Step 1. Set to print in 4 columns.
2. Set x = 0.
3. Set loop to repeat 18 times.
4. Add 5^{o} to x.
5. Multiply x by $\pi/180$.
6. Compute sin x.
7. Compute $\log_e(\sin x)$.

Step 8. Divide by $\log_e 10$.
9. Print to 5 figures.
10. Repeat loop from step 4.
11. Stop.

Before writing the program in its final coded form we check the above "flow diagram" carefully for logical errors. For example we make sure that the first value of x used is in fact 5° and that the last value is 90°. We check that the "set loop to repeat 18 times" instruction is placed just outside the loop and not inside it, and so on.

And now, making a note of the constants:

π = +314159+01 (to 6 figures)
180 = +18+03
$\log_e 10$ = +230258+01 (to 6 figures)

we can prepare the coded form of the program.

```
        8F 4F              Print in 4 columns.
        8+ F               Set accumulator to zero.
        8- 0-N             Store x = 0 in register 0.
        0+ 18F             Set loop 0 for 18 repeats.
#1L     85 0-N             Put x in accumulator.
        84 N+5+01          Add 5 to x.
        8- 0-N             Copy new value of x into
                            register 0.
        87 N+314159+01     Multiply by π.
        86 N+18+03         Divide by 180.
        8J -7              Compute sin x.
        8J -9              Compute log_e(sin x).
        86 N+230258+01     Divide by log_e 10.
        89 5F              Print to 5 figures.
        02 #1L             Repeat loop 0 from instruction
                            tagged with #1L.
        8J -F              Stop.
```

26 16N Start obeying instructions.

11.8* OPERATING A9. The following are the operating instructions for A9:

(i) Load the A9 program. If the sum-check of the read-in fails SILLIAC will stop on an FF order. The tape must then be read in again. If the program is read in correctly SILLIAC will stop on a 24 order, ready to read in the user's program.

(ii) When the black switch is raised the program is read into the machine and performed. (If the start order 24 16N is used at the end of the list of instructions, the black switch must be raised when the machine stops on the 24 order, telling the machine to perform the instructions.)

(iii) The program stops on a 20 order. Provided the D.O.I. and constant listing routine have not been overwritten by data, SILLIAC may be made ready to read in another interpretive program by raising the black switch. SILLIAC then stops on a 24 order, ready to read in the program.

Data stored in the -L locations will be lost if a new program is read in, but that in the -N and -J locations is still intact; therefore a program may be run in successive stages, provided that each stage finishes with the 8J-F order. Each stage may contain as many as 61 constants to be stored by the constant listing routine.

When SILLIAC stops on a 90~~z~~nL instruction, the address from which the instruction was taken appears in the left-hand address of the SILLIAC accumulator.

11.9 USING THE DIAGNOSTIC ROUTINE. The diagnostic routine is used to assist in locating ("diagnosing") programming

blunders, and will work through the list of instructions, printing out the location of the instruction and the contents of the accumulator before the instruction is performed. The printing instructions supplied by the user are suppressed by the diagnostic routine, since the contents of the accumulator is printed for every instruction. The list printed by the diagnostic routine should be sufficient to show the user where his program has left the intended course.

11.10* TO OPERATE THE DIAGNOSTIC ROUTINE The following are the operating instructions for the diagnostic routine:

(i) Load the A9 program. SILLIAC stops on a 24 order.

(ii) Read in the user's program by raising the black switch - the program must be terminated by the order 24 16N so that SILLIAC stops before performing the instructions.

(iii) Bootstrap in the diagnostic routine. (The diagnostic routine is on the A9 tape, immediately after A9 itself.) SILLIAC stops on a 24 order.

(iv) Place the data tape for the program in the reader and raise the black switch. The instructions are then performed under the control of the diagnostic routine, and SILLIAC stops on a 20 order.

Provided that X13 and the constant listing routine have not been overwritten by data, the A9 program may now be restored to its original form by bootstrapping in an amendment tape (this is placed at the end of the tape containing A9 and the diagnostic routine). The SILLIAC will then stop on a 24 order, ready to read in another interpretive program.

Note that, if two successive programs are run under the control of the diagnostic routine, SILLIAC will print out a large number of orders which are performed when it clears the directory, resets the constant list, and places

N.B. See Modification sheet.

new constants in the list before entering the second program.

11.11* LAYOUT OF STORE FOR A9. A9 is a combination of various library routines. The following is a list showing the distribution of storage space and allocation of S parameters.

Locations	S number	Description
0-2		Used as working space by various routines.
3-15		S parameters.
16-17		Two word routine to start the program.
18-78		Storage space for 61 constants produced by the constant listing routine.
79		Special order performed before the user's program.
80-179		Storage space for 200 single instructions of the user's program.
180-279	- N	Storage space for 100 items of data.
280-354	- J	Storage space for another 75 items of data. N.B. If the diagnostic routine is to be used, locations 343-354 must not be used as data storage. I.e. the number of - J locations is reduced to 63.
(344-354)		The sum check routine X7 occupies these locations but is no longer required once the check has been made.
355-364	- F	Program to clear the X13 directory, set the constant storage to its initial condition, set the directive 80 and enter the D.O.I.

Locations	S number	Description
365-382		Constant listing routine (modified form of XA1).
383-384	- 3	Floating decimal accumulator.
385-560	- 4	Floating decimal interpretive A1 (modified).
561-587	- 5	A3 Number conversion.
588-603	- 6	RA1 Square root.
604-629	- 7	TA1 Sine.
630-655	- 8	SA2 Exponential.
656-685	- 9	SA3 Natural logarithm.
686-733	- +	TA2 Arc tangent.
734-774	- -	RA2 Cube root.
775-795		Modifications to RA1, RA2 and SA3.
796-799		Modifications to X13.
800-877	- L	Directory for X13. Available for data storage during operation of program (78 locations).
878-1023		X13 D.O.I. (slightly modified).

At locations 16-17 a two word routine prints a figure shift and enters A1 in such a way that the first interpretive order is taken from the right hand side of location 79. In location 79 is a right hand order which instructs A1 to print all numbers in a single column (i.e. an 8F 1F order); this order only has effect if the user has neglected to provide an 8F nF order in his program. This is followed by the user's program which has been automatically stored at locations 80 onwards.

Each program tape should therefore end with the "start" order 26 16N (or the variant 24 16N).

11.12* EXCEEDING CAPACITY AND OTHER EFFECTS . The routines RA1, RA2 and SA3 have been modified to enable them to handle negative and/or zero numbers. A1 has been modified so as to yield a special indication when a number exceeds the allowable range. Additional orders for this purpose are stored in locations 775-795. If an attempt is made to find the square root of a negative number SILLIAC will print out ROOT NEG and then stop on order 24 3F7. If an attempt is made to find the logarithm of a negative or zero number SILLIAC will print out LOG NEG and stop on 24 3F7. If a number exceeds the allowable range for floating point numbers (10^{64}) SILLIAC will print out TOO BIG and stop on 24 3F7. N.B. See Modification sheet.

If, after stopping on the 24 3F7 order, SILLIAC is started with the black switch, it will read in a new interpretive program and carry it out.

If, due to a programming blunder, control of A9 is lost, the Post Mortem routine C10 may be used to print out details of the last interpretive instruction attempted and the contents of the accumulator. Control of A9 may then be regained by bootstrapping in a tape containing the word 2216300000.[1] SILLIAC will then stop on a 24 order, ready to read in another interpretive program (unless, of course, the programming blunder has corrupted any section of A9).

11.13* STORAGE SPACE AVAILABLE FOR THE USER'S PROGRAM AND DATA. The user's program may occupy locations 80-179 (2 instructions per storage). There are 100 -N locations for data (locations 180-279).

There are also 75 -J locations (locations 280-354) which may be used in exactly the same way as the -N locations.

It will be noted that these three blocks are consecutive in the store, therefore the user may write a longer program at the expense of the data storage if he so desires.

1) This word is punched at the extreme end of the A9 tape.

In addition, there is a block of 78 -L locations which may be used for data storage. This block is at locations 800-877 and is used by X13 to store its directory of tagged instructions while the user's program is being read into the machine.

If the user does not require to read in another interpretive program immediately after the current one, he need not retain the D.O.I. and constant listing routine. In this case, there are 224 -L locations, and 27 -F locations (0-F must be excluded if it is used as the final stop for the program) available for data storage.

Note that if the diagnostic routine is to be used, the last 12 locations of the -J block may not be used for data storage.

There are 61 locations available for the storage of constants by the constant listing routine. If a larger number of constants is required, some must be read in by interpretive input orders and stored in the data storage.

The D.O.I. X13 permits the use of many more S parameters than X1 - they may be numbered from 3 to 255 (decimal). Therefore if the user wishes to preset any of the S parameters from 051 to 0LL (sexadecimal) he may do so by setting these S parameters after the required directive, and then resetting the directive to 80 before reading in his program proper. He should take care that these parameters do not interfere with the course of his program, which must start from location 80, and that the S parameters are always referred to as 3 digit sexadecimal numbers (see description of X13 in Chapter 5).

Additional S parameters may be useful for setting the counting numbers for loops, if the user requires to use his program with several sets of values.