WEIZAC

THE WEIZMANN INSTITUTE OF SCIENCE, DEPARTMENT OF APPLIED MATHEMATICS

NOTES ON CODING FOR WEIZAC

CONTENTS

Section	T I T L E	Page
I	Basic	1
II	Addition and Subtraction Operations	4
III	Transfer Orders	6
IV	Complex Examples	7
V	Subroutines	12
VI	Multiplication and Division	15
VII	Scaling	18
VIII	Input-Output	21
IX	Control Panel	36
X	Off Line Printer	40

September 1960

THE WEIZMANN INSTITUTE OF SCIENCE, DEPARTMENT OF APPLIED MATHEMATICS

NOTES ON CODING FOR WEIZAC

Section One - BASIC

Information is stored in the computer in electronic or magnetic spaces called "memory locations", or "cells" for short. The information stored in a cell is known as a word. No distinction is made whether a word contains arithmetic or alphabetic information, or orders.

There is no single order which will copy a word from one memory location into another. To copy the word out of a cell never destroys the original, it simply produces a duplicate.

The RI Register

The RI register is a special cell. It is the principal arithmetic register where addition, subtraction, multiplication and division is carried out.

Example 1: Suppose that cells 001, 002 and 003 in the memory each hold a number, and we wish to find the sum of these three numbers and store the result in cell 001. The following sequence of instructions will do this.

Symbolic Code		Read	Statement
001	+c	plus clear	Copy into RI the number in cell 001
002	+h	plus hold	Add to RI the number in cell 002
003	+h	plus hold	Add to RI the number in cell 003
001	S	store	Store the number in RI into cell 001

After these orders have been executed, the contents of cells 002 and 003 remain unchanged, and the number in RI is the same as the number in cell 001.

Each order consists of two parts:-

a) The "address" specifies the cell which is involved in the operation. It occupies the first twelve binary digits from the left (This gives the possibility of counting up to 4096 - the capacity of the memory).

b) The "operation" specifies what sort of action is to be taken. It occupies eight binary digits to the right of the address.

Each cell can hold two orders. The order on the left side (Phase I) is carried out first, then the order on the right side (Phase II), and then the order in the next consecutive memory location (provided that no transfer order has been given, for which see later).

Thus the programme given in the example will need to occupy 2 cells, as well as the three cells for the numbers in 001, 002 and 003. Let us suppose that the programme is in cells 004 and 005, then the contents of the memory for this programme will be as follows:

Memory location number	Contents
001	1st number, later sum
002	2nd number
003	3rd number
004	001 +c 002 +h
005	003 +h 001 S

Operations are given in coded form

The machine receives its information not in a symbolic coded form, but in a numerical form.

The codes used for the operations in the previous example are as follows:-

Coded Hexadecimal Form	Symbolic Form
79	+c
71	+h
57	S

Below is given a portion of the Weizac memory containing the coded programme given above.

Memory location number	Contents
001	1st number
002	2nd number
003	3rd number
004	00179 00271
005	00371 00157

Example 2: Numbers p, q and r are stored in cells 101, 102, 103 respectively. Code a routine which will store (p+q) in cell 112 and (p+q+r) in 113.

Solution:

Location	Symbolic Order	Notes	Coded Form
101		p	
102		q	
103		r	
104	101 +c 102 +h	p+q → RI	10179 10271
105	112 S 103 +h	p+q → 112, p+q+r → RI	11257 10371
106	113 S	p+q+r → 113	11357

After p and q have been added and (p+q) stored, (p+q) is still present in RI, making it possible to add r immediately. This technique of using an earlier result which is still in RI will be found useful in problems 4 and 5 below.

Problems

3. Numbers called s, t and u are stored in cells 110, 111, 112 respectively. Code a routine which will store t+u in cell 113, u+s in cell 114, and s+t in cell 115.

4. Numbers called v, w, x and y are stored in cells 120, 121, 122, 123 respectively. Code a routine which will store (v+w) in cell 124, (x+y) in cell 125, and (v+w+x+y) in cell 126.

5. Numbers called g and h are stored in cells 003 and 004 respectively. Code a routine which will store (g+h) in cell 005, (g+2h) in cell 006, (2g+3h) in cell 007, (3g+5h) in cell 008, (5g+8h) in 009 and (8g+13h) in cell 00A.

Section Two - ADDITION AND SUBTRACTION OPERATIONS

C(n) means the contents of memory location n. Thus operation n +c means C(n) → RI, i.e. copy the contents of memory location n into RI.

There are 8 of the above operations. Here are the coded forms, symbolic orders, and the quantities they leave in RI

71	n +h	C(RI) + C(n)
73	n +Mh	C(RI) + |C(n)|
75	n -h	C(RI) - C(n)
77	n -Mh	C(RI) - |C(n)|
79	n +c	C(n)
7B	n +Mc	|C(n)|
7D	n -c	-C(n)
7F	n -Mc	-|C(n)|

It will seem that there exist pairs of orders whose only difference is a clear or a hold operation, e.g. 71 is plus hold and 79 plus clear. The presence of the 8-bit in the second digit of the order signifies that RI is to be cleared before the operation is executed. This is true in general for other operations, e.g. 57 is a store order and 5F is a store clear, i.e. RI is first cleared and then the contents of RI are stored to the location given in the address.

Example 6: Assets are summarised in cell 117. Liabilities are summarised in cell 118. Code a routine which will store the net worth in cell 119.

Solution: The value of the assets is copied into RI and the value of the liabilities is subtracted from it. Then the net worth is stored in cell 119.

100	117 +c	118 -h	11779 11875
101	119 S		11957

Example 7: Numbers called r, s, t and u are stored in cells 100, 101, 102 and 103 respectively. Code a routine which will store s-r in cell 104, s-r-t in cell 105, and r+t-s-u in cell 106.

Solution:

107	101 +c	100 -h	10179 10075
108	104 S	102 -h	10457 10275
109	105 S	105 -c	10557 1057D
10A	103 -h	106 S	10375 10657

The first five steps are easy. Then by using some algebra it can be seen that r+t-s-u = -(s-r-t)-u. A little analysis of this sort will greatly simplify problems 9 and 10.

Problems

8. Assets consist of three different accounts, the amounts of which are stored in cells 101, 102 and 103. Liabilities consist of four accounts, the amounts of which are stored in cells 111, 112, 113 and 114. Code a routine which will store the net worth in cell 101.

9. Numbers called v, w, x, y and z are stored in cells 201, 102, 203, 204 and 205 respectively. Code a routine which will store (w-v) in cell 206, (x-w+v) in cell 207, (y-x+w-v) in cell 208, and (z-y+x-w+v) in cell 209.

10. Numbers called p, q and r are stored in cells 301, 302, and 303 respectively. Code a routine which will store (q-p) in cell 304, (2p+r) in cell 305 and (q-2p-r) in cell 306.

Section Three - TRANSFER ORDERS

Unconditional Transfer Orders (U.0 and U.1)

These orders break the sequence of instructions.

The sequence of carrying out instructions one after another down the column is broken off by an "unconditional transfer", and a new sequence is started, beginning with the cell specified by the address part of the transfer instruction.

It is possible to start this new sequence at any instruction, regardless of whether it is in the left hand or right hand part of its cell. If the transfer order is to the same part as the order to which the control is to be transferred, i.e. right-hand to right-hand, this is called the "parity" transfer order and its hexadecimal representation is 40. If the control is to be transferred from right-hand to left-hand instruction, i.e. "non-parity", the hexadecimal representation of this order is 41.

Conditional Transfer Orders (C.0 and C.1)

If the number in RI is negative, Weizac will continue to follow the old sequence of instructions. If the number in RI is zero or positive, the transfer order will be obeyed and the sequence of instructions will be changed starting with the order in the address referred to by the transfer order.

The matter of parity or non-parity is identical with that for an unconditional transfer order. The hexadecimal representation of parity conditional transfer is 42, and that for non-parity 43.

Example 11: There are numbers in cells 101 and 102. Code a routine which will rearrange them, if necessary, so as to leave the larger in 101.

Solution:

103	101 +c 102 -h	10179 10275
104	107 C.1 101 +c	10743 10179
105	108 S 102 +c	10857 10279
106	101 S 108 +c	10157 10879
107	102 S	10257
108	Temporary store	

The portion of this routine from cell 104 Ph.II to cell 107 Ph.I simply exchanges the words in cells 101 and 102. This portion is left out if the word in cell 101 is already the larger, i.e. if word 101 minus word 102 is greater or equal to zero. Note also the use of the temporary store 108 for holding intermediate results. Later in these notes a better method will be given for interchanging the contents of two cells.

Example 12: Code a routine which will replace the number in cell 100 with zero if it is negative.

Solution:

101	100 +c	102 C.0	10079 10242
102	100 -h	100 S	10075 10057

If the number in 100 is less than zero we subtract it again from RI making RI zero, and then we store. (Later a better way will be given using different orders but for the moment it is better not to complicate the matter).

Problems

13. Code a routine which will replace all negative numbers in cells 101, 102, 103, 104 with their absolute values (Note: Use only the orders discussed up till now).

14. Code a routine which will place in cell 101 the largest of the numbers in cells 102, 103 and 104.

15. There are numbers in cells 101 and 102. Code a routine which will copy one of them to cell 103 if they are the same, but will store the absolute value of their difference in cell 103 if they are different.

Section Four - COMPLEX EXAMPLES

The following thirteen operations have been discussed:

n +c	79	C(n) → RI
n -c	7D	- C(n) → RI
n +h	71	C(n) + C(RI) → RI
n -h	75	C(RI) - C(n) → RI

n +Mc	7B	$\|C(n)\| \rightarrow RI$
n -Mc	7F	$-\|C(n)\| \rightarrow RI$
n +Mh	73	$C(RI) + \|C(n)\| \rightarrow RI$
n -Mh	77	$C(RI) - \|C(n)\| \rightarrow RI$
n S	57	$C(RI) \rightarrow n$
n U.0	40	Transfer control to n (parity)
n U.1	41	Transfer control to n (non-parity)
n C.0	42	Conditionally transfer control to n (parity)
n C.1	43	Conditionally transfer control to n (non-parity)

With just these thirteen operations it is possible to code some complex routines, which will be done next to illustrate some important coding techniques.

Each coder will develop his own method of attacking a new problem. The first step should always be scaling, which is discussed in Section Seven. Then the typical coder will produce two or three flow diagrams before he has one that is sufficiently detailed to work from. The next step is often a simple listing of the instructions that will be needed, with no attempt to assign them to cells. The names of operands are written in place of addresses, since operands are usually not assigned to cells either.

At this stage it is fortunate that addresses have not been assigned, since many deletions, modifications and additions have to be made. Often it is found that changes must be made in the flow diagram at this stage.

When all the instructions are arranged in the right sequence, they are copied to a coding sheet and explicit addresses filled in. There is also space available on the coding sheet for making notes. The last step is to translate the symbolic code into the machine code reading for making up a tape.

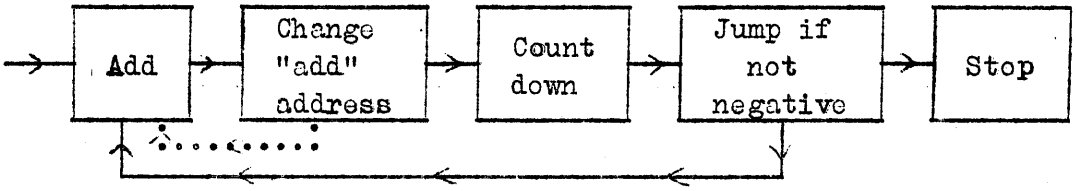
Example 16: Nine numbers are stored in consecutive cells from 110 to 118. A number called n, which is smaller than nine, is stored in cell 119. Code a routine which will store in cell 11A the sum of the first n+1 of the nine numbers in 110 through 118.

Solution: This routine will involve two new features:

a) The routine will cycle on itself. We will go around the cycle n times, adding one of the nine numbers each time. Each time we will subtract 1 from n so that after n+1 times this number will become negative, and we can use a conditional transfer order to leave the cycle.

b) We will perform an arithmetic operation on the second instruction, so as to increase by one each time the address of the number to be added to the accumulating sum.

The final flow diagram looks like this:



The final list of instructions looks like this:-

120	T	Sc	clear location of sum
	T	+c	
121	(110)	+h	Add
	T	S	
122	121	+c	
	*1	+h	Change address
123	121	S_L	
	119	+c	
124	*1*	-h	Count down
	119	S	
125	120	C.1	Jump if not negative
		Stop	

In location 123 Ph.I a "store left" is used instead of a "store" order. The symbol *1 means $n.2^{-11}$, 1* means $n.2^{-31}$ and *1* stands for $n.(2^{-11}+2^{-31})$. It is assumed that the number n is given as *n* (in 119). The left-hand address in location 121 is taken to be *110. (In a real problem this address would have to be set by the programme). The use of a symbolic temporary location (T) is also shown, whose final address in the store need not be known at this stage.

Example 17: Employers are required to deduct 2% for K.K.L. from each employee's pay until his total pay for the year reaches IL.4200, after which no more should be deducted.

A certain employee's pay for this week is stored in cell 101. His previous total pay for the year is stored in cell 102. 4200 is stored in cell 103 (suitably scaled).

Code a routine which will store in cell 104 that portion of this week's pay which should be taxed at the 2% rate.

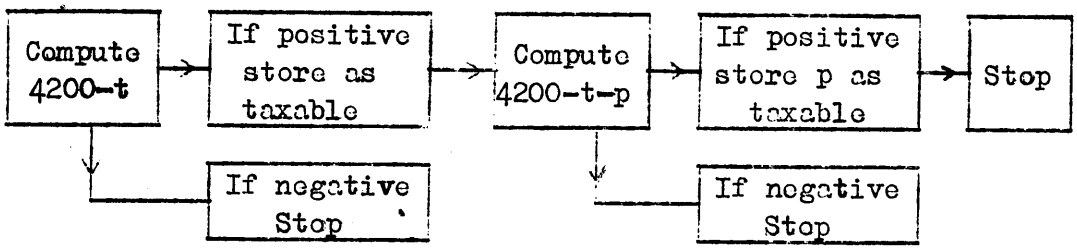
Solution: The only new feature in this problem is the three-branched split that is necessary in the flow diagram, depending on whether the employee has already passed IL.4200, reached IL.4200 this week, or still has not reached it. In the first case none of his pay should be taxed. In the second case IL.4200 minus his previous total should be taxed. In the last case all his pay should be taxed.

Let t represent the previous total and p represent this week's pay.

Since 4200-t is the answer in one of these cases, it would seem wise to compute this first. If it is negative, he has already passed IL.4200 and none of his pay should be taxed.

If 4200-t is positive, let us store it temporarily as the answer, and then test to see if he still has not reached IL.4200. If 4200-t-p is positive, he still has not reached IL.4200 and the whole amount p should be taxed.

Thus we get the flow diagram:



and this list of instructions:

120	103 +c 102 -h	Compute 4200-t	10379 10275
121	122 C.0 104 Sc'	If positive, store (4200-t) as taxable	12242 1045E
→122	104 S 101 -h	Compute (4200-t-p)	10457 10175
123	124 C.0 Stop	If positive, store	12442 00000
→124	101 +c 104 S'	p as taxable	10179 10456

The use of "stop" orders is introduced. An order whose second instruction digit is even, i.e. whose "stop" digit is zero, will be executed, after which the machine will stop (except for transfer orders). The order "00" will stop the machine without doing anything.

Problems

18. Code a routine which will store in cell 100 the larger of these two numbers:
    a) The smaller of the two numbers in cells 101 and 102
    b) The smaller of the two numbers in cells 103 and 104
19. Without looking at the sample coding, recode Example 16.
20. Numbers called m and n are stored in cells 101 and 102. Using multiplication by repeated additions, code a routine which will store mn in cell 103. (Weizac can multiply, but the operation has not yet been discussed).

Section Five - SUBROUTINES

There are many pieces of coding, such as the calculation of square roots and logarithms, which occur so frequently that much effort can be eliminated by preparing a subroutine which can easily be made a part of any routine that one is coding. Indeed, it often happens that a calculation which is used in only a single problem is used so often that it is profitable to prepare it as a subroutine.

Entry and Exit

Subroutines, as they are written for Weizac, are all based on the same principle of operation. A subroutine is entered through a special group of orders incorporated in the "master routine" or "main programme". It is designed so that its last order returns control to the master routine at a point immediately following that from which it entered. There are various methods of arranging the operation of entering a subroutine and returning to the master routine after the operation of the subroutine has been completed. The method chosen for use with Weizac is given below, it is known as the "left hand entry, right hand exit"; n is the address of the first order of the subroutine.

The code for the entry and exit is as follows:

m	- - -	m +c	adds address m* into RI
m+1	n U.0		transfer control to n

The orders in the subroutine are as follows:

n	1* +h	(n+p) S_R	Stores (m+1)* into the right hand address of location (n+p)
.			
.			
n+p	- - -	( ) U.0	Return control to master routine

The address of m is placed in RI. It should be noted that if there is no left-hand instruction in m, then a "dummy" order must be used. The first order of the subroutine will be to increase m by one, then to store (m+1) at the last memory location of the subroutine in the right-hand address. Thus, with the completion of the subroutine, control will be transferred back to m+1, right-hand side.

Subroutine Organisation

Detailed information on the subroutines is obtained from the subroutine file. The tapes themselves are kept in separate compartments in the teletype-room. The subroutine file lists all the subroutines available with an information sheet for each one.

Each sheet will have the file number, title, use, length, number of words to be modified (about which refer to the paragraph on incorporation), temporary storage, non-standard constants, entry parameters and exit indicated. Entry information will include the location of parameters and arguments. Exit information will give the location of the resultant function. Often both argument and result will be in the same location, i.e. 001. This is done where several subroutines are used in succession. The result of one is used as the argument of the following subroutine, e.g. $e^{\sin x}$.

Standard Constants

There are a number of parameters which are used very frequently with many programmes. Hence it was decided to allocate them a fixed place in the memory, so that anybody writing a programme will be able to make use of them. There are eight such "preset" parameters and they are stored as follows:

Memory location	Constant	Hexadecimal Representation
020	0	00000 00000
021	-1	80000 00000
022	$\frac{1}{2}$	40000 00000
023	*1	00100 00000
024	1*	00000 00100
025	*1*	00100 00100
026	1.2^{-39}	00000 00001
027	*2*	00200 00200

Temporary Storage

For temporary storage, memory locations 001-00A are used. When there is another subroutine within a subroutine, then there must be another pool of

temporary storage locations. Usually locations 00B-014 are assigned. Programme parameters which vary for each programme are usually stored at 015-01E. These are required for the mechanics of the programme itself, and are not themselves data. Programme parameters may be either prepared by the machine or they may be preset.

Incorporating Routine

Since a subroutine may go into any part of the memory, we do not want to have to rewrite the subroutine for each memory location. We want the machine to make the necessary changes. This is the task of the incorporating routine, which changes all the internal addresses of the subroutines. There may at times be words in a subroutine, e.g. constants which are placed at the end of the subroutine, whose address digits one does not want changed. The number of words to be modified by the incorporating routine will then be all those excluding these constants.

The incorporating routine is itself an interpretive routine, i.e. it processes information according to a certain scheme. The routine is stored at 028' - 038', and its programme parameters at 039 - 039+h. These will be the key words which indicate where the subroutines to be incorporated begin and end. This list will be followed by a word of zeros which will be the sign that all the subroutines have been incorporated and control is sent to 01F where there will be an order to start the computation. After incorporation, the memory locations used by the incorporating routine may be used for temporary storages, except for 028 which is used as a constant by the "print" subroutines.

Here, then, is the memory allocation list:-

001-014 temporary storage

015-01E programme parameters

01F-01F' transfer order to main routine

020-027 standard constants

028-038 incorporating routine

039-039+h programme parameters for incorporating routine

Section Six - MULTIPLICATION & DIVISION

The RII Register

Up till now only orders using RI have been discussed. There is another register called RII which is connected to RI in a way which will now be explained.

Multiplication

The RII register holds multiplier and quotients. Before multiplication it is necessary to load RII with the multiplier by means of the order R_2L (47). There are three multiply orders:-

1. X_{RO} (6B) is used primarily to multiply two fractions. No overflow can occur since we multiply two numbers each of which is less than one.
2. X_{NRO} (63) is used in several contexts, e.g.:
   (i) in double precision arithmetic
   (ii) to retain greater accuracy when a multiplication is followed by a division, in which case $\div_L$ is used
   (iii) for multiplying two numbers with binary scale factor in which case it is followed by L_L
   (iv) in interpolation problems as illustrated by the following example:

Example 21: Assume that a table of values of a function $f(x)$ is given for $x = x_o, x_o+h, x_o+2h, \ldots, x_o+nh$, where $\frac{1}{h}$ is an integer. Then to find $f(\bar{x})$ for a given $x_o \leqslant \bar{x} < x_o+nh$, we need to know m such that $x_o+mh \leqslant \bar{x} < x_o+(m+1)h$ and $\Delta\bar{x} = \bar{x}-(x_o+mh)$. This is achieved very simply by the sequence

$L\ (\bar{x}_o-x)$	R_2L
$L\ (\frac{1}{h} \cdot 2^{-39})$	X_{NRO}
$L\ (m.2^{-39})$	S
	A + c
$L\ (\Delta\bar{x})$	S

3. X_{NROH} (61) is a difficult order. The best advice is to use it only with positive numbers. Then it can be used for cumulative multiplication of integers.

Example 22: If m, n, p, q are positive integers, then we can compute r = mn + pq with the following orders:-

L $(m.2^{-39})$	R_2L
L $(n.2^{-39})$	X_{NRO}
	A + c
L $(p.2^{-39})$	R_2L
L $(q.2^{-39})$	X_{NROH}
	A + c
L $(r.2^{-39})$	S

Exam.23: The other major use of X_{NROH} is in double-length arithmetic. This application will be illustrated by means of an example of the multiplication of two numbers given in Integer-Fraction form. A number x, in integer fractions consists of two parts, the integral part I(x) and the fractional part F(x) where F(x) is always positive. Each part occupies a separate cell. To multiply two such positive numbers x and y located in 001/2 and 003/4 respectively, the following sequence of orders is necessary (T_1 and T_2 are temporary storage locations):-

001	R_2L
003	X_{NRO}
	A + c
T_1	S
002	R_2L
004	X_{RO}
001	R_2L
004	X_{NROH}

T_1	+ h
T_2	S
	A + c
002	R_2L
003	X_{NROH}
T_2	+ h
$L[I(xy)]$	S
	A + c
$L[F(xy)]$	S

Problems

24. w, x, y, z, are numbers stored in 101–104 respectively. Code a routine to compute $(w+x)(y-1)z$

25. g, h, j, k, are positive integers stored in 800–803 respectively, stored as $g.2^{-39}$ etc. Code a routine to compute $2gh + 4jk$.

Division

There are two division orders of which the first one (67) is the usual case and the other one (65), the long division, is for special purposes.

Division in which only the quotient is sought does not cause any difficulty. Care must be taken that overflow does not occur, by ensuring that the absolute value of the numerator is less than the absolute value of the denominator. The quotient appears in RII and hence must be brought to RI before storing. One must also remember that the 40th bit of the quotient is always 1 so that for example zero divided by a non-zero number gives 2^{-39} and not zero. The remainder in RI is twice the remainder after 38 division steps and presents some tricky problems. Dealings with it should be left to the expert.

Long division (65) with a double-precision dividend is performed in the computer in a similar way to division (67) and all remarks which pertain to the order (67) apply to (65).

Example 26: It is required to compute $d = \frac{ab}{c}$. The most efficient way is as follows:

L (a)	R_2L
L (b)	X_{NRO}
L (c)	$\div_L$
	A + c
L (d)	S

Example 27: It is required to divide a fraction by an integer, $\frac{x}{m}$. Since m is given as $m.2^{-39}$, we must form $\frac{x.2^{-39}}{m.2^{-39}}$ and proceed as above. If x is positive, L(x) R_2Lc gives $x.2^{-39}$ in RI and RII. If x is negative, more care must be taken. Hence the final procedure to compute $\frac{x}{m}$ is

α	L (x)	+ c	β	C.1
$\alpha + 1$	L (2^{-39})	- c	L (x)	R_2L
$\alpha + 2$	L ($m.2^{-39}$)	$\div_L$		A + c
$\alpha + 3$	L ($\frac{x}{m}$)	S		
β	(α+ 1)	U.1c		

Problems

28. Code a routine to compute $\frac{pq + r - s}{t}$ which will stop on overflow.

29. Code a routine to compute $\frac{w + x + y}{z}$ where w, x, and y are fractions, and z is an integer.

Section Seven - SCALING

Although numbers in Weizac have to lie in the range $-1 \leq x < 1$, it often happens that it is required to make use of numbers outside this range. This generally involves the introduction of scale factors.

Example 30: It is required to evaluate a function which is given by

$$f = \frac{x^2}{1+x^3} \qquad \text{where x lies in the range } 0 < x < 3.$$

First of all, x cannot be used or stored inside the machine since it can

exceed the range, hence we can use

$y = \frac{1}{4} x$, which will always be within range. In terms of y, f becomes,

$$f = \frac{16y^2}{1+64y^3} \qquad \text{where } 0 < y < \frac{3}{4} .$$

The numerator has a maximum of 9 and the denominator has a maximum of 28. Hence it is convenient to scale them both by a factor of 32. This yields

$$f = \frac{\frac{1}{2}y^2}{\frac{1}{32}+2y^3}$$

The final programme is

L (y)	R_2L
L (y)	X_{RO}
T_1	S
T_1	R_2L
L (y)	X_{NRO}
1	L_L
L ($\frac{1}{32}$)	+ h
T_2	S
L (y)	R_2L
L (y)	X_{NRO}
1	R_{NRO}
T_2	$+_L$
	A + c
L (f)	S

Note: y^2 is computed a second time by multiplication rather than by adding $C(T_1)$. If y is small, a more precise value of y^2 is thus obtained and long division is used to obtain the quotient.

31. $x = \frac{yz + w}{u}$ where y is less than 20, z is less than 40, w is less than 200, u is less than 90 but more than 15. Scale and code the computation of x (which will be scaled down).

32. Eight numbers are stored in 100 to 107. Code a routine which will compute their average and store the result in 108.

Section Eight - INPUT-OUTPUT

The following notes are an addendum to the Order Code, in order to clarify the situation with respect to the above equipment. A knowledge of the controls and their terminology is assumed throughout.

Hexadecimal Character

Although the machine works in the Binary System, it is convenient for the programmer to use as his basic digit not the binary digit (bit for short), but a combination of four bits which is called a hexadecimal character. The list below explains the latter system:-

Binary Digits	Hexadecimal Digit	Decimal Equivalent
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

In the text following the hexadecimal characters will be used without further explanation.

Equipment

The input-output equipment is of two kinds:-

a) Perforated paper-tape devices

b) Magnetic tape devices

a) Perforated Paper Tape Equipment

(i) Two tape readers are used alternatively with Weizac, a Ferranti Mk I and a Ferranti Mk II. The Mk I instrument can read only 5-hole tape whereas the Mk II can also read 6-hole tape. Both are high-speed photo-electric tape readers, reading 200 characters a second, and stopping on a character.

(ii) High speed Teletype paper punch (output only) made by Teletype Corporation model BPRE2 capable of punching 50 hexadecimal characters a second.

(iii) Flexowriter (output only) made by Commercial Controls capable of punching and printing simultaneously 12 characters a second.

(iv) Teletype paper punch (output only) made by Commercial Controls, capable of punching 15 characters a second. This instrument is connected to the buffer equipment for the off-printing from magnetic tape and instructions for use are given in Section Ten.

b) Magnetic Tape Equipment

There are four magnetic tape handlers now in use: two made by the Potter Instrument Company, Type 905 with tape speed of 75 inches per second; and two made by the Ampex Corporation, Model FR-308, with tape speeds of 75 inches per second. The computer can both read and write on this magnetic tape. At present the magnetic tape is not strictly speaking an input/output device, since primary data still has to be fed in via punched paper tape, but information from magnetic tapes can now be punched out onto paper tape on the Off-Line-Printer. Since the orders for reading from and writing on the magnetic tapes are similar to the orders for paper tape, they will be included here.

Input-Output Orders

Three input-output orders are used with all the input-output media:

a) Block Input - Code Number D7

A block consisting of a variable number of computer words is read into the high-speed memory from either paper or magnetic tape. Each word may con-

tain up to 10 tetrads or 8 pentads. The tape (paper or magnetic) must include marks signifying the end of a word and marks signifying the beginning and end of a block. The choice of input unit is put into R_2.

b) Single Word Input - Code Number A7

A single word consisting of up to 10 tetrads or 8 pentads is read into R_1 without storing. The order must specify the input unit desired.

c) Single Word Output - Code Number A5

A single word consisting of up to 10 tetrads or 8 pentads is taken from R_1 and either typed out by the Flexowriter, or punched out, or written on one of the magnetic tapes. The order must specify the output unit desired.

d) A number of additional orders which are essential for the magnetic tape handlers such as "advance tape one block" etc., are also available.

Code Numbers Allocated for Choice of Input-Output Unit

	Input	Output
Ferranti	00	
Output Punch		00
Flexowriter		20
Magnetic Tape I	18	18
Magnetic Tape II	30	30
Magnetic Tape III	28	28
Magnetic Tape IV	08	08

Information Given by the Tape (Paper or Magnetic)

This consists of one of three forms:

a) Hexadecimal Characters

(i) On 5-hole paper tape this is characterized by the presence of the I-hole.

(ii) On 6-hole paper tape this is characterized by the presence of the I-hole and the absence of the VI-hole.

(iii) On magnetic tape this is characterized by a "1" on the I-channel and a "0" on the VI-channel.

b) Alphabetic Characters

(i) On 6-hole paper tape, recognized by the presence of the VI-hole

(ii) On magnetic tape, recognized by a "1" on the VI-channel

c) Operational Characters

(i) On 5-hole paper tape, recognized by the absence of the I-hole

(ii) On 6-hole paper tape, recognized by the absence of the I-hole and the VI-hole

(iii) On magnetic tape, recognized by "0" on both the I- and the VI-channels

The position of each character is defined by an additional small "sprocket" hole on paper tape, or an additional "1" on the "synch-track" of the magnetic tape.

A list showing the various characters is given below. X means a hole on paper tape, a "1" on magnetic tape. The numbering of the channels follows the conventions for Flexowriter paper tape. The "sprocket" hole lies between the II-hole and the III-hole.

a) Hexadecimal Characters

	I	II	III	IV	V
0	X				
1	X				X
2	X			X	
3	X			X	X
4	X		X		
5	X		X		X
6	X		X	X	
7	X		X	X	X
8	X	X			
9	X	X			X
A(10)	X	X		X	
B(11)	X	X		X	X
C(12)	X	X	X		
D(13)	X	X	X		X
E(14)	X	X	X	X	
F(15)	X	X	X	X	X

b) Alphabetical Characters

	VI	I	II	III	IV	V
A	X	X				
B	X		X			
C	X	X	X			
D	X			X		
E	X	X		X		
F	X		X	X		
G	X	X	X	X		
H	X				X	
I	X	X			X	
J	X		X		X	
K	X	X	X		X	
L	X			X	X	
M	X	X		X	X	
N	X		X	X	X	
O	X	X	X	X	X	
P	X					X
Q	X	X				X
R	X		X			X
S	X	X	X			X
T	X			X		X
U	X	X		X		X
V	X		X	X		X
W	X	X	X	X		X
X	X				X	X
Y	X	X			X	X
Z	X		X		X	X
Full stop	X	X	X		X	X
Code delete on P.T.) Block marker on M.T.	X	X	X	X	X	X

c) Operational Characters

	VI	I	II	III	IV	V
Stop code (P.T. only)						
Space (word mark)				X		
Carriage return					X	
Tab						X

Some additional remarks have to be made:

1) "Code delete" (6 holes) on alphabetical (6-hole) paper tape means "delete this character", i.e. do not read it in. On magnetic tape, this character (6 "ones") signifies the beginning or end of a block of words.

2) "Stop Code": a) Sprocket hole only on paper tape means the beginning or end of a block when paper tape is fed into the reader. When this tape is fed into the Flexowriter is causes the typewriter to stop.

b) Synch track present on magnetic tape, has no significance, i.e. the tape goes on to the next character without causing any additional operations.

3) The "Space" character signifies the end of a word on paper or magnetic tape. When this character is present on paper tape fed into the Flexowriter it causes a "space" to be typed.

4) "Carriage return" and "Tab" are interpreted as such only when paper tape is fed into the Flexowriter. On input tapes (paper or magnetic) these characters are ignored.

Operation of the Reader

There are four items of information necessary in order to execute a read-in order:-

a) Where to store the information

b) Start of block

c) End of word

d) End of block

A single order can read-in a single word into R_1 or an entire block of words on perforated paper tape into the high-speed memory. The block input order is of the form n D7 where n specifies the address in the high-speed memory where the first word is to be stored. The order must occupy the left-hand part of a word and the right-hand part should be an unconditional transfer to the next part of the programme.

N.B. If an unconditional transfer is not given in the R.H. part of the word with the D7 order, the R.H. order will be obeyed after which control will be transferred to the address in the dispatch counter which will be one location greater than the last word of the block just read in.

The address associated with the order for block input specifies the memory location in which the first word on the tape is to be stored. The dispatch counter will then increase the address each time a word is stored, so that the input block will sit in successive locations in the memory. Items 2), 3) and 4) are given as perforations on the tape.

There are two modes of operation: 5-hole tape, with 4 information bits per character, and 6-hole tape, with 5 information bits per character. The fifth bit on the 5-hole tape, and the 6th bit on the 6-hole tape, do not get read in as information but are operation characters either for items 2), 3) and 4) above or flexowriter operations. The latter will be dealt with later.

Perforations on the tape supply the following information:

(i) Stop code, (blank tape with only sprocket-hole) which is used as the start and end of a block;

(ii) The actual, 4 or 5, information bits;

(iii) End of word, signifying that the word now in R_1 is to be stored.

Manual Control of the Reader for Block Input

Assuming that the read-in order is sitting in R_3, and 00 in R_2, the following sequence of events will take place:-

a) With a tape threaded into the reader on stop code, the main switch S_2 is turned to "fast". The reader will then commence to read and store until "Stop Code" appears again on the tape, and then the control will go to Phase II of R_3.

b) With the tape as before, if after turning the switch S_2 to "fast" it is turned to "interrupt fast", the reader will read and store until "stop code" appears again and then the computer will stop and await further instructions. If it is then turned back to "fast" control will be sent to Phase II of R_3.

c) With the tape as in a), the switch S_2 is turned to "slow", and microswitch S_1 is depressed continuously, the reader will read and store as long as the switch S_1 remains depressed or until "stop code" appears again on the tape.

A switch has been provided to allow ignoring the input-output selector called the "External Button". When this is in the "In" position, the contents of R_2 do not effect the input orders. However, in this case the only possible input order is "Read from paper tape".

Manual Control of the Magnetic Tape Handlers for Block Input

Assuming that the read-in order is sitting in R_3, and the code number corresponding to the unit desired is sitting in R_2, the following sequence of events will take place:

The main switch S_2 is turned to "fast". The tape handler will start running forwards. For technical reasons the code which signifies the start

and end of a block, is not blank tape with synch-track only, but a single character of "code delete", i.e. a "1" on all six channels. This is known as the "block marker", and must be present to signify the start and end of a block. After receiving the presence of the "block marker" the information (excluding the block marker) will be read into R_1 and stored in a similar way to paper tape. After a second block marker appears on the tape, the reading is discontinued and the magnetic tape is stopped. The alternative operation is as in b) and c) for paper tape.

Execution of D7 Order

As mentioned previously, there are two modes of operation. In the first mode, reading from 5 or 6-hole tape, first of all R_1 is shifted left 4 places, making room for the next character and then 4 information bits are gated into the 4 least significant digits of R_1. This operation is repeated until a "space" operation appears on the tape (which is not read into R_1) and the word assembled in R_1 is then stored at the appropriate memory location. In the other mode, reading alphabetic information from 6-hole tape, 5 left shifts are executed and 5 information bits (including the I-hole) are gated into the 5 least significant digits of R_1. The storing is as for 4 bit operation.

Both modes of operation may be summed up by the following statement:- When there is a "1" on the VI channel (or the VI-hole in the case of paper tape) 5 shifts will be executed and 5 bits will be gated. When there is a "0" on the VI channel (or VI-hole in the case of paper tape) 4 shifts will be executed and 4 bits will be gated.

This makes it possible to read-in less than 10 tetrads (or 8 pentads) per word, provided one remembers that the whole of the word in R_1 is stored. It also provides a way to erase, since providing no "space" has been punched, the word will not be stored and the incorrect part will be shifted out of R_1.

When reading from paper or magnetic tape, where there is a mixture of numerical and alphabetic information, since the computer will automatically do 5 shifts if the information is alphabetic and 4 shifts if the information is numerical, care must be taken that the sum total of a word is not more than 40 bits otherwise information will be lost.

When reading from magnetic tape, the process is similar except that the "External Button" must be in the "out" position, and R_2 has to be loaded with the appropriate order specifying the input unit desired.

Example 33: To read a block from paper tape starting from location 100. Order 100D7. (External Button "in" or clear R_2)

To read a block from magnetic tape I starting from location 100. Orders: 20047 , 100D7, where C(200) = 18000 00000

Execution of Single-Word Input - A7 Order

The order A7 is used for a number of purposes, but under this heading only its use for reading in single-words will be explained.

a) Single-Word Input from Paper Tape - 005 A7

Reading from paper tape, (5-hole and 6-hole), 5 information bits (excluding the VI-hole) are gated into the 5 least significant digits of R_1 and then R_1 and R_2 are shifted 5 places left, thus making room for the next character. At the same time as the shifting, 5 "1's" appear in the 5 least significant digits of R_2. This operation is repeated until a "1" appears in the sign digit of R_2. This order is different from D7, since no

storing is done and the length of the word read in depends on the contents of R_2; also the shifting of R_1 and R_2 is not connected, i.e. the most significant digits of R_1 do not shift into the least significant digits of R_2.

It is also possible to read less than 8 pentads. By putting 2^{-5n} into R_2, only n pentads ($n < 8$) will be read into R_1. This order may be used for copying a tape including the same number of operations, i.e. stop-codes, carriage returns, etc.

b) Single-Word Input from Magnetic Tape

N.B. All information given here is given for the sake of completing the explanations. It is highly desirable that every one using magnetic tapes will only use the standard tested subroutines. Every one using programmes other than these does so at his own risk.

The order contains the choice of input unit and R_2 controls the end of the order as for paper tape. However, since the tape remains on the move, there is a maximum safe interval (600 μsec.) between read-in orders when it is possible to use the arithmetic organ for the purpose of constructing a loop for checking purposes, etc.

The standard block written by the subroutine is arranged in the following manner:

Block marker

Block number + number of words (n+3)

Information (n words)

Quick Sum

Block number (in reverse)

Block marker

Thus it is seen that for an information block of n words the block actually contains (n+3) words.

With the magnetic tape it is possible to use single-word input both forward and reverse. For this reason the block number is written in the normal direction at the beginning of the block and backwards at the end of the block.

This order may be used for searching for the desired block. If, when reading forward, the right block has been found, it is possible to read the rest of the block. If the right block has not been found, it is possible to disconnect reading to the arithmetic organ and to allow the tape to run to the next block marker, while meanwhile using the arithmetic organ for further computing.

Control of the Output Order - A5

There are three items of information necessary to be given to the computer in order to extract the output:-

a) Choice of output unit

b) Choice of hexadecimal, alphabetic or operational characters

c) Number of digits to be printed

R_1 contains the information to be printed. The address in the output order has three hexa digits. The first two signify the output unit as listed on p. 23. To this must be added also information about the kind of digit required, i.e. hexadecimal or operation. The third address digit controls the number of channels (i.e. corresponding to 5 or 6-hole paper tape) to be printed, and also the corresponding number of shifts to be executed, about which further anon, i.e. whether numerical or alphabetic.

Execution of the Output Order - A5

R_2 controls the number of digits to be printed in the following way (similar to the A7 order). The print cycle takes the 4 (or 5) most significant bits of R_1 and gates them to the output unit and then executes 4 (or 5) shifts left. With each shift a "1" appears in R_2 at the least significant end. R_1 and R_2 do not shift together connectedly, i.e. the most significant digits of R_1 do not go into the least significant digits of R_2. This cycle is repeated and the print order is terminated when a "1" appears in the sign digit of R_2. This means that it is possible to print a maximum of 10 tetrads (or 8 pentads) with one output order. In order to print only n ($n < 10$) tetrads, it is therefore necessary to put 2^{-4n} into R_2. In order to print only n pentads ($n < 8$) it is necessary to put 2^{-5n} into R_2.

Code numbers to be added to the address are as follows:-

000	Hexadecimal Character (or Alphabetic)
400	Operational Character

If the third digit of the address is even, the output will gate tetrads, and if it is odd it will gate pentads, but the number of shifts is controlled by the whole of the third address digit.

Example 34: To punch 10 hexadecimal characters on the punch

Synthesis of output order:

Punch + hexa char. ==>000 + 004 ==> 004A5

Also needed is 02047 where C(020) = 0

Example 35: To punch sp. + carr. ret. on magnetic tape I

Mag. Tape I + operations ==> 180 + 404 ==> 584A5

Hence orders are 20047 where C(200) = 42800 00000

00239

584A5

Example 36: To punch 40 times stop code on the punch

Punch + op + 1 shift only ==> 000 + 400 + 001 ==> 401A5. Hence orders are:

0204F

401A5

Example 37: To punch the word "key" on the flexowriter

Flexowriter + alph. char. + 5 shifts ==> 200 + 000 + 005 ==> 205A5

Hence orders are: 20079 where C(200) = KEY00000

20147 where C(201) = 0008000000

205A5

N.B. All programmers are hereby warned that for technical purposes it is not permitted to punch two carriage returns on the Flexowriter following each other. For output to magnetic tape it is recommended that the W7 routine be used.

Miscellaneous Information on the Use of Magnetic Tapes

a) The use of the Internal/External Button has already been mentioned in connection with input. For output if it is the "in" position, it is only possible to output to the punch and not to the Flexowriter or the magnetic tapes.

b) There is another button on the extreme left of the operating console. This is to stop the magnetic tape from running wild. If the block marker denoting the end of a block has been erased, the tape will continue running, in the worst case, to the end of the tape. The additional switch prevents this possibility and one is able to stop the tape.

c) There exists a programme W3 which enables one to transfer the contents of the whole memory except for 000 and 001 to the magnetic tape called the Memory Dump routine. This is to prevent machine time going to waste by en-

abling the computer user to throw the contents of the memory on to tape and to restart next time without having to redo all the work up to this point.

Section Nine -- THE CONTROL DESK

It is the task of the switches on the control desk to make it possible for the programmer to read his programme into the machine and to operate it with the minimum of manual attention and yet be flexible enough to help him locate and correct faults even manually, if the need arises. For this purpose the following facilities are provided:-

a) Display of Registers and Counters

Looking through the window in front of the control desk, there are a number of registers fitted with neons for display. The object of the neons is twofold:

(i) In order to read information off the machine in case it has stopped either because of a technical or programming failure or because a breakpoint had been inserted in the programme.

(ii) In order to check changes which have been made manually to the registers.

The first two rows from the top form the double shifting register of R_1. The lower of the two shows the actual content of R_1.

The next two rows make up the double shifting register of R_2. The lower of the two shows the actual content of R_2.

The next row consists of 40 display neons giving the contents of R^3. This is the register which holds the multiplicand during multiplication, the divisor during division and the addend during addition. The programmer has no control over this register.

The last two rows show the content of R_3. The 20 digits in the upper row in the middle show the content of the left-hand order and those in the lower row, the right-hand order.

In the same row as the left-hand half of R_3 there are two more sets of display neons. On the far left there are 12 neons of the Address Counter which show the address of the next order-pair to be obeyed except after TT_{in} and Q Σ. On the far right there are the 3 neons of the Phase Counter. If this shows 001, the next order to be obeyed will be a left-hand order. If this shows 111, the next order to be obeyed will be a right-hand order.

b) Main switch S_2.

This switch has four positions:-

(i) Off

(ii) Slow - for use in conjunction with one-at-a-time execution

(iii) Interrupt fast

(iv) Fast. In this position the computer is controlled by the stored programme, as interpreted by the main control. When going to position "interrupt fast", the order that is at the moment being executed will be completed and the machine will then stop. However, if the switch is turned back to "fast", the machine will continue.

c) Microswitch S_3

This is used in conjunction with the main switch S_2 when on position (ii). When this is then depressed, a single order will be obeyed according to the contents of R_3.

d) 40 switches are provided to insert a digit (up for zero and down for one) into any of the forty positions of R_1 or R_2. These switches are arranged in groups of four to facilitate insertion of hexadecimal information. Whether

these switches insert information into R_1 or R_2 is determined by the position of switch S_4. When switch S_4 is in the "up" position, digits will be inserted into R_1; when this is in the "down" position, digits will be inserted into R_2.

e) 20 switches are provided to insert a digit (up for zero and down for one) into any position of R_3. The upper part of R_3 belongs to the L.H. order, the lower part of R_3 belongs to the R.H. order. These switches are arranged in two groups, the left group consisting of 3 hexadecimal groups (the address) and the right group of 2 hexadecimal groups (the operation). Whether these switches insert a L.H. order or a R.H. order is determined by the position of switch S_5. When switch S_5 is in the "up" position, digits will be inserted into the L.H. order. When switch S_5 is in the "down" position, digits will be inserted into the R.H. order.

f) At the right-hand side of the control desk are three white switches. The function of the left-hand switch is to clear R_1 to zero (up) or to fill with 1's (down). The function of the middle switch does the same for R_2, and the right-hand switch clears R_3 to zero (up). When the main switch is in the "off" position, it is possible to clear R_1 and R_2 to "1" or to zero, but not if it is in any other position. The switch to clear R_3 to zero is <u>always</u> connected and an accidental touch can stop the computer. This clear switch is the only way to stop a dynamic stop-order of, e.g., too many shifts, which never reaches termination in the normal way, and which cannot be stopped by turning the main switch to "off".

g) At the right-hand side of the control desk is a switch whose function is to change the phase counter, i.e. decide whether the next order to be obeyed, after the machine has stopped, is to be a left-hand order or a right-hand order. If the switch is in the "up" position, the next order to be obeyed

is the left-hand order in R_3; if the switch is in the "down" position, the next order to be obeyed is the right-hand order in R_3. In order to make the decision, it is sufficient to give the switch a flip in the required direction and then to let it return to the neutral position; it is not necessary to hold the switch either up or down.

h) There is a button on the right-hand side of the control desk, the "External Button", whose function is to simplify the input-output when only paper tape is read from the Ferranti reader and the only output is to the punch. With the button depressed, it is not necessary to specify separately the unit desired; instead of this the D7 order reads paper tape and the A5 order punches tape on the high-speed punch. With the button in the "out" position, it is necessary to specify in the usual way the input-output unit desired.

i) On the far right-hand side on the side of the control desk is a switch to suppress the VI-hole when 5-hole tapes are read in. For 5-hole tapes the switch is in the "down" position; for 6-hole tapes, in the "up" position. Since most of the work now is with 5-hole tapes, the switch is usually down.

j) Excessive temperature rise will operate either a warning light or an alarm bell on the control table. On the far left-hand side there is a switch to make the choice.

k) To the right of the switch (j) there is a switch to stop the magnetic tapes from running away, which may happen due to a programme or machine error. In order to stop the tape, it is necessary to move the main switch to the "off" position and then to press on the microswitch.

Section Ten - OFF-LINE-PRINTER

Introduction

The Off-Line-Printer is an auxiliary device for the Weizac computer, which has punched paper tape input-output. The purpose of the Off-Line-Printer is to save computer time by printing information on magnetic tape rather than mechanically by punching. Since writing on magnetic tape goes a hundred times faster than with the present punch, considerable time is saved for more productive computer operations. The tape delivers the information much faster than any printer can print, so a buffer is needed for intermediate storage of the tape information (the loading of the buffer with this information will be called the "Load"-mode). After the buffer has thus been loaded we print out: we unload the buffer (the "Unload"-mode). A magnetic drum with eight data tracks was chosen as the buffer.

The output device is a punch (50 msec/character). The tape-handler is Potter model 902 (36"/sec).

Operator's Instructions

A "Print Request" character (alphanumeric), 110111, has to be printed on the tape after the first block mark, in order to print out the information appearing directly after this "Print Request" and up to the next block mark.

By block length we understand the number of words (10 characters + space = 1 word) between "Print Request" and next block mark.

Maximum allowed block length : 80 words on flawless tape.

By "full cycle" we understand the reading from the tape onto an intermediate storage medium (a magnetic drum) (the "Load"-mode), and the reading from this drum and printing out via a punch (the "Unload"-mode).

Note: As the packing density of the tape handlers connected with the computer has been increased by a factor of two, we need to skip every second tape sync-pulse when writing for the Off-Line-Printer.

Operator's Panel

The panel holds 3 push-buttons (I, II, IV), one switch with 5 positions (III) and two neon lights (a and b).

a) Functions

I: Start

II: Tape-Feed

III: Operator switch : Position 1 : full cycle
" 2 : interrupt
" 3 : unload only
" 4 : reverse one block
" 5 : forward one block

IV: Pre-clear

(i) Neon-indicator for "Overload"

(ii) Neon-indicator for : "We are not in position 1 of the operator switch"

b) Short description of the functions

I: The "start" punch button starts any operation of the Off-Line-Printer as specified by the position of switch III.

II: Tape-feed is a push button which causes the punch to punch out paper tape, with only sprocket holes. (It automatically interrupts print when pressed during the "Unload"-mode).

III: Operator switch. This is a 5-position switch, specifying the desired operation of the Off-Line-Printer.

Position 1 : "Full-cycle", i.e. reading of a tape-block and printing it out.

Position 2 : "Interrupt". In this position we interrupt the printing during the "Unload"-mode. In order to resume printing we have to switch back to either position 1 or 3.

Position 3 : "Unload only". Pressing the "start" button will reprint the drum contents from the beginning, i.e. reprint the last block read.

Position 4 : "Reverse one block". Pressing the "start" button will reverse the tape one block, i.e. it stops at the beginning of the last block, marked with a "Print Request". (Without printing anything out).

Position 5 : "Forward one block". Identical to 4, only the tape will now move forward and stop after the next block, marked with a "Print Request".

N.B. In the positions 1, 4 and 5 the original contents of the drum will be erased.

IV: Pre-clear. This push button is used only directly after turning on the Off-Line-Printer, and in case of "Overload".

(i) Neon indicator for "Overload". By "overload" we understand a block length, longer than the available space on the magnetic drum. So the last portion of the block cannot be printed out.

To warn the operator, this neon will light up, and the system stops without printing out, i.e. before entering "unload".

In order to print out this block so far as it is recorded, we have to switch the operator switch III to position 3, ("unload only"), press the "Pre-clear" button to remove the "Overload" inhibition, and then press the "start" button.

(ii) Neon indicator that we are not in position 1.