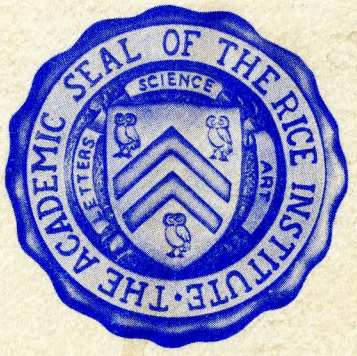
D. A. Chesnut
~~1003~~ Chem. Bldg.
310

# WILLIAM M. RICE INSTITUTE

*For the*
*Advancement of Literature, Science, and Art*




## COMPUTER MANUAL

# A MANUAL

# FOR

# THE RICE INSTITUTE COMPUTER

September 1, 1958

October 1958

To the reader of this manual:

In the text which follows references have been made to sections which are not included at this time because they are currently being prepared. Also, we expect to compile from time to time addendum and errata which would render your copy of the manual more useful.

If you would like to receive this material, please fill out the form below and mail to:

Computer Project
The Rice Institute
Houston 1, Texas

---

Please mail material to be included in A Manual for the Rice Institute Computer to:

Name _____

Position _____

Address _____

_____

_____

TABLE OF CONTENTS

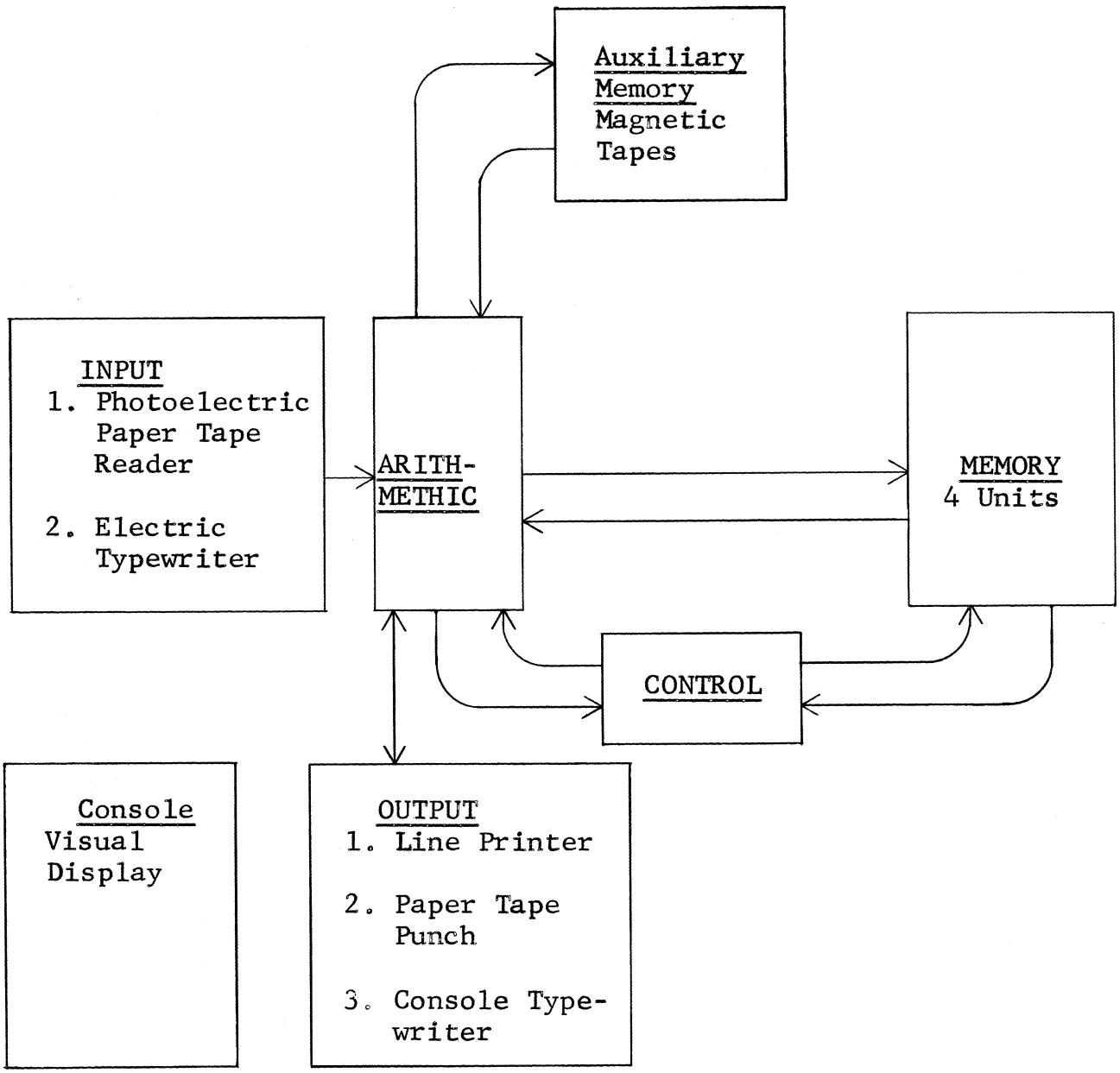



Figure I. Functional parts of the digital computer and their relationships. The arrows represent information flow.

# I.

## COMPUTER ORGANIZATION

The modern digital computer consists of five distinct groups of equipment which perform the following functions:

(1) input
(2) memory or storage
(3) arithmetic
(4) control
(5) output

Figure I is a block diagram of these units showing the relationships among them.

The input section consists of a photoelectric reader which takes information from punched paper tape and places it in memory and an electric typewriter which can be used to type information into the arithmetic and control sections. The arithmetic unit is always an intermediate in the flow of input information to memory. The information in question may be anything which can be stored in memory: numbers, instructions, or alphabetical and numerical comments.

The memory is an information-holding device composed of electrostatic storage tubes. One memory contains ~~56~~ 64 storage tubes and is subdivided into distinct units called words. The memory is needed to record numbers and hold instructions. Thus, each word may be a number, an instruction, or a coded comment. Each memory unit is capable of recording up to 8,192 words, and the computer in its final form will have 4 memory units.

The memory may be thought of as N little boxes or locations where numbers or instructions can be located. Each of the locations is given an identification number from 8 to N (the numbers 0 to 7 are reserved for a purpose to be explained later). The label of a location is called its address (synonyms: cell, location, box). Note that the address 1371 does not mean that we can find the number 1371 stored there - except by accident; the address is purely a label or identifying number.

A memory location can hold only one word at a time, and placing a word in a location automatically destroys whatever was there previously. It is possible to read a number out of memory without destroying or removing it. A detailed description of the word and the memory unit is given in the section on electrostatic memory.

The arithmetic section does what its name implies. In addition to the basic arithmetic operations, this unit can shift numbers right and left and assist in certain operations which make it possible for the computer to make decisions. If we use the analogy of a desk calculator, this section corresponds to the upper, lower and middle dials plus the wheels and gears that actually do the calculation. A detailed description is given in the section under the heading of arithmetic unit.

Register is a term commonly used in connection with these various units. It denotes a device for temporarily storing a

piece of information while or until it is used. A register corresponds quite closely to the dials on a desk calculator. Not only numbers but also instructions may be stored in a register.

The whole computer is controlled by a certain set of specified permissible operations, and no two such operations can occur simultaneously. The permissible operations may be executed in any desired sequence. It is up to the user to specify the sequence of operations or, as it is commonly called, the program. Each permissible operation can be specified in a concise coded form called an order (synonym: instruction). For a problem to be solved on a computer, it must be broken down into a series of precise steps and this sequence is coded and usually stored in memory as ordinary numbers. The correspondence between the set of permissible operations and the set of numbers which specify them is called the order code and is described in the section on instructions.

The control section of the machine has the function of accepting orders one by one and of interpreting or decoding these instructions and then sending signals to the other units telling them what to do. The control unit is equivalent to the operation buttons which are pushed on a desk calculator. The control section is described in detail in another section.

The output units are an automatic punch for paper tape

and a fast line printer. The printer can print up to 600 lines per minute - each line containing up to 108 characters. Information may also be permanently recorded (or written) on magnetic tape.

SUMMARY of MACHINE CHARACTERISTICS:

The Rice Institute Computer is a megacycle computer (i.e., a basic pulse time of about 1 microsecond) with a speed that is appropriate to:

(1) memory access time for reading of 10 microseconds

(2) memory access time for writing of 20 microseconds

(3) an addition time of 4 microseconds

(4) an average multiplication time of 120 microseconds.

The machine is asynchronous, binary and parallel in operation and will have a random access memory of 32,000 words.

## II.

## OCTAL NOTATION

Binary numbers are very well adapted to representation by electronic circuits. Since each digit can have only two different values, zero or one, the digits of a binary number can be put into one-to-one correspondence with the electrical conditions of off-on, open-closed, non-conducting-conducting, etc. We pay for this simplicity (i.e., small amount of information per digit) by needing more digits to represent a given amount of total information than if we had used a larger number base. For example, a decimal number with N significant figures is equivalent to a binary number with N ln 10/ln 2 = N/0.30103 = 3.321 N digits. For example, the standard numerical word in the Rice Computer will have between 40 and 47 significant binary places. This is equivalent to about 12 to 14 decimal places.

The problem of conversion between base two and base ten is actually simple but need not concern the reader at the moment. The process will be carried out essentially automatically by the computer by means of subroutines, so that the average machine user will supply decimal input data and the computer will deliver decimal final results.

In order to discuss the instruction word and numerical word structure of the computer, we must use the full 54 bit binary machine words. It is very inconvenient to write out

such words in full and it is equally inconvenient to type them into a typewriter-tape punch. As a shorthand, we shall introduce "octal" notation. The binary number is divided into triads (groups of three bits). Instead of writing each triad in full, we shall write instead an integer between zero and seven inclusive:

| binary | octal |
|---|---|
| 000 | 0 |
| 001 | 1 |
| 010 | 2 |
| 011 | 3 |
| 100 | 4 |
| 101 | 5 |
| 110 | 6 |
| 111 | 7 |

Each triad is thought of as an octal integer, and the digit written is the usual symbol for this integer. The reader is advised to memorize as soon as possible this conversion table. This conversion is of course very easy in either direction. The resulting shorthand number is actually the equivalent of the binary number written to base eight, i.e., an octal number. A 54 bit machine word becomes an 18 octal digit number, much more manageable in length. We shall use expressions such as "the second octal figure" and "the second triad" essentially synonymously. In the computer we shall have triads; on paper or at the typewriter punch we shall use octal figures.

As an example, 000101011001010100111 is equivalent to 000, 101, 011, 001, 010, 100, 111 is equivalent to 0531247.

The octal form is obviously much easier to write and to absorb at a glance.

In referring to an octal or binary number we read it from left to right. For example, "the first octal figure" refers to the figure furthest to the left (0 in the above example); "the second octal figure" or "the second triad" in the number above is 5.

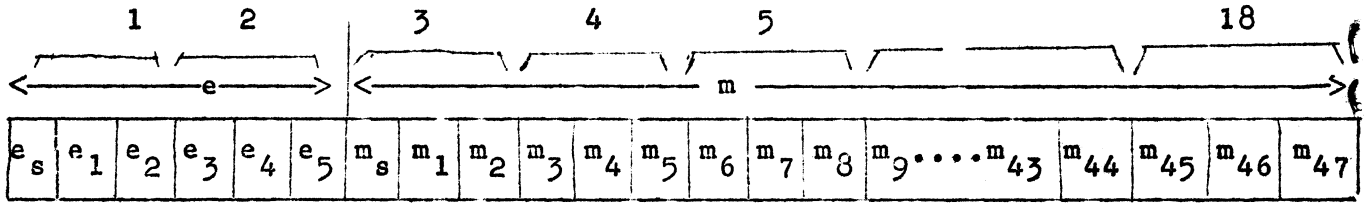
# III

## NUMBER REPRESENTATION

A number $x = m \cdot (256)^e$ is represented by the number pair (m,e) where m and e are referred to as the mantissa and exponent, respectively. The actual computer representation of m and e depends upon the location of x. There are two cases one must consider, (a) the representation in storage; and (b) the representation in the arithmetic registers U and S.

(a) Representation in storage.

In all storage locations (electrostatic memory, R and T registers) the number is a sequence of 54 binary bits, 6 exponent bits and 48 mantissa bits, numbered as follows:




Each bit is a 0 or a 1. $m_s$ is termed the sign bit of m and $e_s$ the sign bit of e (0 means +, 1 means - ). The number representation is what is usually called the 1's complement system and can be explained as follows. For simplicity and convenience in explanation we will regard the binary point as being located between $m_s$ and $m_1$. In floating point operations this is the only interpretation possible, while in fixed point work the location of the binary point can be chosen by the coder.

<u>mantissa</u>

$$m_s = 0;\ m = m_1 \cdot 2^{-1} + m_2 \cdot 2^{-2} + \ldots + m_{47} \cdot 2^{-47}$$

$$m_s = 1;\ m = -\left(\bar{m}_1 \cdot 2^{-1} + \bar{m}_2 \cdot 2^{-2} + \ldots + \bar{m}_{47} \cdot 2^{-47}\right)$$

where $\bar{m}_i$ is the bit-complement of $m_i$: that is, if $m_i = 0$, $\bar{m}_i = 1$ and if $m_i = 1$, $\bar{m}_i = 0$. To clarify this system consider as an example a 4-bit mantissa:

| $m_s$ | $m_1$ | $m_2$ | $m_3$ | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | $= 1 \cdot 2^{-1} = 1/2$ |
| 0 | 0 | 1 | 1 | $= 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 3/8$ |
| 1 | 0 | 1 | 1 | $= -(1 \cdot 2^{-1}) = -1/2$ |
| 1 | 1 | 0 | 0 | $= -(0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}) = -3/8$ |

exponent

$e_s = 0$; $e = e_1 \cdot 2^5 + e_2 \cdot 2^4 + \ldots + e_6 \cdot 2^0$

$e_s = 1$; $e = -(\bar{e}_1 \cdot 2^5 + \bar{e}_2 \cdot 2^4 + \ldots + \bar{e}_6 \cdot 2^0)$

As an example consider the following exponents

| $e_s$ | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | $= +3$ |
| 1 | 1 | 1 | 1 | 0 | 1 | $= -(1 \cdot 2^1 + 0 \cdot 2^0) = -2.$ |

The zero is a special case that must be given careful consideration. A sequence of all zeros in m is called a logical zero (sometimes a + zero) while a sequence of all ones in m is called an arithmetic zero (sometimes a -0). The word "zero" by itself will refer to either +0 or -0.

In the exponent e, all ones represent the arithmetic zero and all zeroes, by convention, represent an exponent $\leq -32$. The reason for this will become clear when floating point operations are discussed.

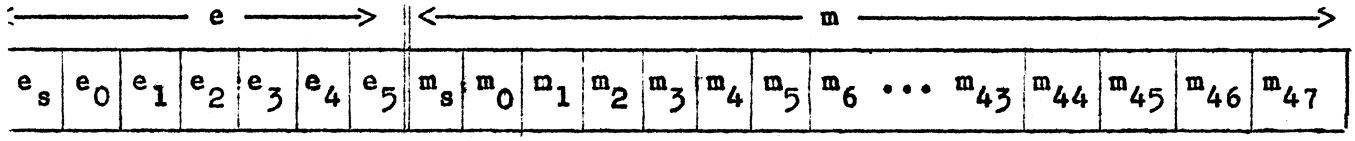
Since the octal notation is often useful, the reader is advised to study the following examples.

| Octal Machine Number | | Value |
|---|---|---|
| 00000 ... | Fixed point numbers | zero (or "plus zero") |
| 00200 ... | | 1/2 |
| 00777 ... | | zero (or "minus zero") |
| 00577 ... | | - 1/2 |
| 77000 ... | Floating point numbers | $(256)^{0} \times 0.0$ |
| 75200 ... | | $(256)^{-2} \times 0.5$ |
| 02577 ... | | $(256)^{2} \times (-0.5)$ |
| 40020 ... | | $(256)^{-31} \times (0.03125)$ |

In fixed point numbers the choice of the exponent is left to the coder, who will, however, find it advantageous to use a "plus zero" exponent. A nonzero floating point number will never have 00 for an exponent.

(b) Representation in U and S.

In the arithmetic registers the number is a sequence of 56 binary bits, 7 exponent bits and 49 mantissa bits, numbered as follows:




The two bits $e_0$ and $m_0$, called the exponent overflow and the mantissa overflow respectively, have no representation in storage. In S, $e_0$ is always equal to $e_s$ and $m_0$ is always equal to $m_s$. Thus the $e_0$ and $m_0$ bits in S as described below are only virtual concepts. However in U the $e_0$ and $m_0$ bits are actual stages in the register. Whenever a number is transferred to U from some other register, the computer automatically sets $e_0 = e_s$ and

$m_0 = m_s$. We note, however, that these two bits ($e_0$ and $m_0$) in U are unaffected when (U) -> U as described later in the interpretation of Field 1.

The bits ($e_0$, $m_0$) are only altered upon arithmetic operations and arithmetic shifts as explained in the section describing overflow and underflow. Whenever a number is stored, all bits except $e_0$ and $m_0$ are transferred to their corresponding positions in storage. It is convenient to consider the binary point in the mantissa to be located to the right of the $m_0$ bit. In this manner we can give the following interpretation to the numbers in U:

mantissa

U is capable of working with mantissae in the range -2 to +2, using the following convention:

| $m_s$ | $m_0$ | range | | |
|---|---|---|---|---|
| 1 | 0 | $-2 < m \leq -1$ | $\lvert m\rvert - 1$ | $= \bar{m}_1 \cdot 2^{-1} + \bar{m}_2 \cdot 2^{-2} + \ldots + \bar{m}_{47} \cdot 2^{-47}$ |
| 1 | 1 | $-1 < m \leq 0$ | $\lvert m\rvert$ | $= \bar{m}_1 \cdot 2^{-1} + \bar{m}_2 \cdot 2^{-2} + \ldots + \bar{m}_{47} \cdot 2^{-47}$ |
| 0 | 0 | $0 \leq m < 1$ | $m$ | $= m_1 \cdot 2^{-1} + m_2 \cdot 2^{-2} + \ldots + m_{47} \cdot 2^{-47}$ |
| 0 | 1 | $1 \leq m < 2$ | $m - 1$ | $= m_1 \cdot 2^{-1} + m_2 \cdot 2^{-2} + \ldots + m_{47} \cdot 2^{-47}$ |

Whenever, as a result of an operation, m falls in the range -2 to -1 or 1 to 2, it has overflowed the range of the storage locations and an overflow indicator will be turned on when appropriate (see section entitled "overflow").

exponent

In U the exponent has an extended range in which $-63 \leq e \leq +63$.

| $e_s$ | $e_0$ | |
|---|---|---|
| 1 | 0 | $-63 \leq e \leq -32$ |
| 1 | 1 | $-31 \leq e \leq 0$ |
| 0 | 0 | $0 \leq e \leq +31$ |
| 0 | 1 | $+32 \leq e \leq 63$ |

Exponents in range +32 to +63 are said to have overflowed and an appropriate overflow indicator will be turned on if this occurs. Exponents in the range -63 to -32 are said to have underflowed the storage range. By convention a zero exponent is represented by all 1's, and all zeros will indicate an exponent smaller than -31.

The question probably occurs to the reader: Why do we use the large base 256 rather than the more obvious value of 2? The choice of base depends upon consideration of a number of factors (the number range desired, the minimum and maximum mantissa accuracy desired, the details of the shifting technique used in standardization of numbers, the proportion of arithmetical combinations that may be expected to require a final standardization, etc.). Our choice appears to be about optimum. We lose very little as compared with base two; we gain a great deal, particularly in that floating point operations will be carried out (on the average) very nearly as fast as fixed point operations.

IV.

## ADDRESSING SYSTEM

The Rice Computer will have eight full length (54 bit or 18 triad) registers (the A series), eight address or indexing (~~16~~ 15 bit or 5 triad plus ~~one sign bit~~) registers (the B series), eight special purpose (15 or 16 bit) registers, and an electrostatic memory of $2^{15}$ -16 or 32,752 full length words. The A and B series are jointly known as F registers. The special purpose and F reigsters are fast registers, having an access time of the order of one microsecond. The electrostatic memory addresses have an access time of about ten microseconds. The A series and the special purpose registers, together with the electrostatic series are known as M addresses.

The execution of every instruction by the computer involves (1) procuring two operands, one from an F address and one from an M address, (2) some arithmetical or logical work on these two operands, and (3) the storage of some result at an F address or a modification of the contents of some B register.

A SERIES

| address | abbreviation | full name |
|---|---|---|
| 0 | 0 | zero or null register |
| 1 | U* | universal* |
| 2 | R | remainder |
| 3 | S | storage |
| 4 | $T_4$ | temporary store 4 |
| 5 | $T_5$ | temporary store 5 |
| 6 | $T_6$ | temporary store 6 |
| 7 | $T_7$ | temporary store 7 |

*Note: The U register plays a special role in arithmetic operations and has 55 bits - the extra bit is called an overflow bit and its use will be explained in the detailed description of arithmetic operations. This overflow bit is always set ~~to 0~~ equal to the sign bit when a number is sent to U.

B SERIES

| address | abbreviation | full name |
|---|---|---|
| 0 | CC | control counter |
| 1 | $B_1$ | B register 1 |
| 2 | $B_2$ | " 2 |
| 3 | $B_3$ | " 3 |
| 4 | $B_4$ | " 4 |
| 5 | $B_5$ | " 5 |
| 6 | $B_6$ | " 6 |
| 7 | PF1 | pathfinder |

M addresses range from 0 to 77777 (octal) (i.e., 0 to 32767 decimally), with 0 to 7 being from the A series and 10 to 77767 being true electrostatic memory addresses and 77770 to 77777 being the addresses of a class of special registers.

Register 0 does not actually exist. By definition it always contains zeros. The uses of address 0 will appear later. U and S (and sometimes R) are used to hold the operands of arithmetical or logical operations. After an operation, U and R hold the result. $T_4$ through $T_7$ are used to temporarily store words.

CC holds the address of the next instruction word to be fetched to the instruction register (see discussion of instruction register in section on control unit). $B_1$ through $B_6$ contain address increments to be used in certain logical manipulations of instructions. PF1 is set to the current reading of CC immediately before an unconditional class 0 transfer or skip is executed.

## SPECIAL PURPOSE REGISTERS

| address | abbreviation | name | description |
|---|---|---|---|
| 77770 | PF2 | pathfinder 2 | CC ~~PF~~1 → PF2 ~~before transfer or skip executed~~ (on all modifications of CC other than the normal advance by 1.) |
| 77771 | SL | sense light register | holds sense light information |
| 77772 | X | increment or index register | adds special purpose increments to B series registers |
| 77773 | ML | mode light register | holds mode light information |
| 77774 | TR | trapping register | holds trapping information |
| 77775 | IR | indicator register | holds indicator information |
| 77776 | reserved for future use | | |
| 77777 | | | |

PF2 and X are 16 bit registers (5 triads plus sign); the use of the X register is described in the discussion of field 3 operations. SL and ML are 15 bit registers and are described fully in the section on indicators.

V.

# INSTRUCTION WORD STRUCTURE

All instruction words are divided into four major fields. These fields are further subdivided as follows:

| field | triad | name | use |
|---|---|---|---|
| 1 | 1 | IF | inflection on F |
| | 2 | F | F address |
| 2 | 3 | C | class |
| | 4 | Op1 | 4 operation triads |
| | 5 | Op2 | |
| | 6 | Op3 | |
| | 7 | Op4 | |
| 3 | 8 | ISt | inflection on store |
| | 9 | St | store address |
| 4 | 10 | IM | inflection on M |
| | 11 | IA: 1 bit | indirect address bit |
| | 12 | BM: 8 bits | B modification of M |
| | 13 | | |
| | 14 | M | memory address |
| | 15 | | |
| | 16 | | |
| | 17 | | |
| | 18 | | |

An instruction word is decoded and interpreted by the computer in the I register. We shall first take an overall view of the results of this decoding, then return later to a detailed view of each section.

(1) The computer consults CC and fetches the contents of the indicated address to the I register. CC is advanced by 1.

(2) Field 1 (2 triads, IF and F) is decoded. Consequence: a word in an F address (generally a numerical word) is sent to U.

(3) Field 4 (9 triads, IM, IA, BM, M) is decoded. Consequence: a word is sent to S and an address (possibly new) is left in position M of I.

(4) Field 2 (5 triads, C, Op1, Op2, Op3, Op4) is decoded. Consequence: in general, arithmetical or logical work is done using the contents of U and S and/or the final address M (which have been set up by the two preceeding steps). If the operation is arithmetical, the primary answer will be found in U and the secondary answer (if there is one) in R.

(5) Field 3 (2 triads, ISt, St) is decoded. Consequence: the contents of either U or R is sent to an F address or certain changes may be made in the contents of one of the B registers.

(6) Return to step (1).

We shall use from now on the convention that an address symbol in parentheses means the contents of that address location; the address symbol alone means the numerical value of that address. For example:

(M) represents the 54 bit word located at M

M represents a 15 bit numerical address

($B_4$) represents a 15 bit number and sign stored in $B_4$

SUMMARY:

The common features of every instruction, without exception, are:

(1) a number $\rightarrow$ U

(2) an address $\rightarrow$ M in I

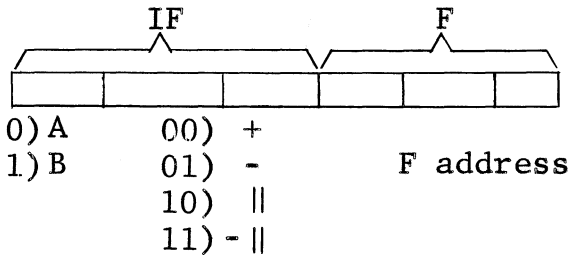
(3) a number $\rightarrow$ S

The instruction may then operate with any or all of these three results.

1. Field 1

The two triads of field 1 (IF,F) determine the F address of a word which is brought to U and the modification of its sign. F is a fast address. Bit 1 of IF determines whether we mean an A or a B address, a zero for A and a 1 for B. The contents of this address are fetched to U. Bits 2 and 3 of IF determine a sign modification according to the scheme:

| | | | |
|---|---|---|---|
| 00 | means | + | (no sign change) |
| 01 | " | - | (change sign) |
| 10 | " | \|\| | (absolute value) |
| 11 | " | -\|\| | (negative of absolute value) |




For convenient reference we quote again a table of F addresses:

| | | A | B |
|---|---|---|---|
| 0 | 000 | 0 | CC |
| 1 | 001 | U | $B_1$ |
| 2 | 010 | R | $B_2$ |
| 3 | 011 | S | $B_3$ |
| 4 | 100 | $T_4$ | $B_4$ |
| 5 | 101 | $T_5$ | $B_5$ |
| 6 | 110 | $T_6$ | $B_6$ |
| 7 | 111 | $T_7$ | PF1 |

Examples of field 1:

| | | | |
|---|---|---|---|
| 00 | means | zero $\longrightarrow$ U | clear |
| 01 | " | (U) $\longrightarrow$ U | no change |
| 40 | " | (CC) $\longrightarrow$ U | |
| 05 | " | $(T_5) \longrightarrow$ U | |
| 15 | " | $-(T_5) \longrightarrow$ U | |

Examples of field 1 (continued):

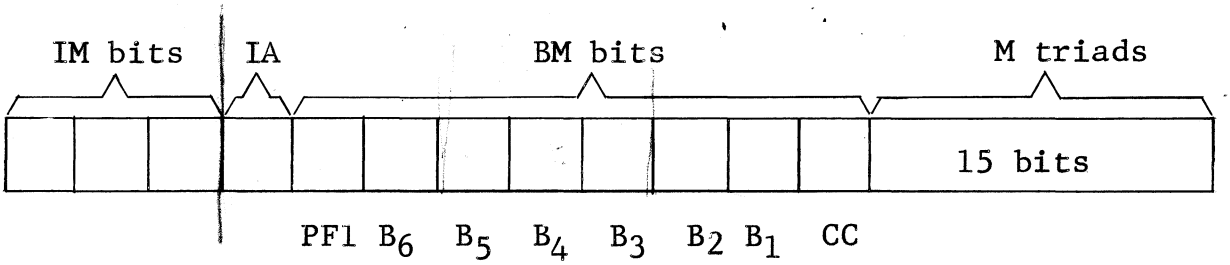
25 means $|(T_5)| \longrightarrow U$

35 " $-|(T_5)| \longrightarrow U$

45 " $(B_5) \longrightarrow U$

Whenever an integer from any B register is sent to a 54 or 56 bit register (for example, U), the 15 ~~bit magnitude is~~ bits of the B register are written in the right hand end of the register and the sign bit of the B register is sent to the sign bit of the long register. All remaining bits of the ~~register are cleared to zero~~ [mantissa, including the overflow bit, are set equal to the value of the first bit (that on the left end) of the B register, and all of the bits of the exponent are set to zero. When using operations of this sort it is convenient to regard this number as an integer (either positive or negative).

2. Field 4




The nine triads of field 4 (IM, IA, BM, M) determine what word is brought to the S register and the final address residing in the I register. M is a 5 triad octal address. 00000 through 00007 refer to A addresses; 00010 through 77767 refer to electrostatic memory addresses; and 77770 through 77777 refer to the special purpose registers. BM consists of

8 bits. Counting from right to left, they refer to the 8 B addresses: CC, $B_1$ through $B_6$, and PF1. The M address as written will be modified by the sum of the contents of the B registers referred to. A zero means ignore, a 1 means use. For example:

| BM | meaning |
|---|---|
| 01 001 010 | add $(B_1)$ + $(B_3)$ + $(B_6)$ to M |
| 10 000 000 | add (PF1) to M |
| 00 000 001 | add (CC) to M |
| 00 000 101 | add (CC) + $(B_2)$ to M |

Field 4 is decoded according to the following sequence.

(1) BM is decoded and a new M is formed in I

$M + (B_i) \longrightarrow M$

(M is incremented by the contents of all of the B registers referred to in BM.)

(2) Test IA bit

if 0, go to step 3

if 1, form new IA, BM, M in I

[last 24 bits of (M)$\longrightarrow$I]

then return to step 1

(3) Test bit 1 of IM

if 0, (M)$\longrightarrow$S

if 1, M$\longrightarrow$S (last 15 bits of S, all others cleared to zero)

(4) Test bits 2, 3 of IM and modify the sign of (S) as

Field 4 is decoded according to the following sequence:

(1) If bit 4, 5, or 6 of trapping register is 1 and control tag register = 1, 2, or 3 respectively, transfer to 41, 49, or 57, respectively.

(2) BM is decoded and a new M is formed in I

$$M + \Sigma(B_i) \rightarrow M$$

(M is incremented by the contents of all of the B registers referred to in BM.)

(3) Test IA bit

if 0, go to step 4

if 1, form new IA, BM, M in I (last 24 bits of (M) -> I), then return to step 1.

(4) Test bit 1 if IM

if 0, (M) -> S

if 1, M -> S (last 15 bits of S, all others cleared to zero)

(5) Test bits 2, 3 of IM and modify the sign of (S) as follows:

00 means + (no sign change)

01 means - (sign change)

10 means || (absolute value)

11 means -|| (negative absolute value)

(6) If bit 1, 2, or 3 of the trapping register is 1 and the arithmetic tag register = 1, 2, or 3 transfer to 9, 17, or 25 respectively.

follows:

| | | | |
|---|---|---|---|
| 00 | means | + | (no sign change) |
| 01 | " | - | (change sign) |
| 10 | " | \|\| | (absolute value) |
| 11 | " | -\|\| | (negative absolute value) |

Examples:

| | | | |
|---|---|---|---|
| 0 | 000 | 00000 | zero $\longrightarrow$ S |
| 0 | 000 | 00005 | $(T_5) \longrightarrow$ S |
| 1 | 000 | 00077 | $-(77) \longrightarrow$ S i.e., the contents of location 77 goes to S |
| 4 | 002 | 00000 | $(B_1) \longrightarrow$ S |
| 0 | 001 | 00005 | $((CC) + 5) \longrightarrow$S i.e., the word 5 later in the code past the current reading of CC |
| 0 | 002 | 00005 | $((B_1) + 5) \longrightarrow$S |
| 0 | 026 | 00005 | $((B_4) + (B_2) + (B_1) + 5) \longrightarrow$S |
| 4 | 000 | 00077 | ...00077 $\longrightarrow$ S |
| 3 | 002 | 00077 | $-\,\lvert((B_1) + 77)\rvert \longrightarrow$ S |
| 0 | 402 | 00077 | $((B_1) + 77) \rightarrow$I (last 8 octal digits) followed by a reinterpretation of I. |

At the end of this sequence some word will have been sent to S, with or without sign modification. This word may have come from the original M address (no 1's in BM). It may have come from M incremented by any or all of the B registers

(note the possibilities and flexibility in the fact that the B series includes CC, PF1, and the regular B indexing registers, singly or in combination). It may have come from an address (with B modification) looked up in memory (IA bit 1). This procedure may be repeated indefinitely. Finally, we have a choice of obtaining either the contents of the final M address or the address itself, with or without sign modification in either case.

Most operations will work with the contents of U (set up by field 1) and the contents of S. Some operations, however, ignore (S) and use the final M. Examples: shifts of U and/or R, set or increment B from address, and transfers. In these cases, a 10 microsecond memory fetch time can be saved by writing a 1 in bit 1 of IM. These operations can be controlled by M. They can also be controlled by the M portion of (M) by use of the IA bit.

### 3. Field 2

The five triads of field 2 determine what arithmetical or logical operation takes place. C is decoded first and determines the class of the operation:

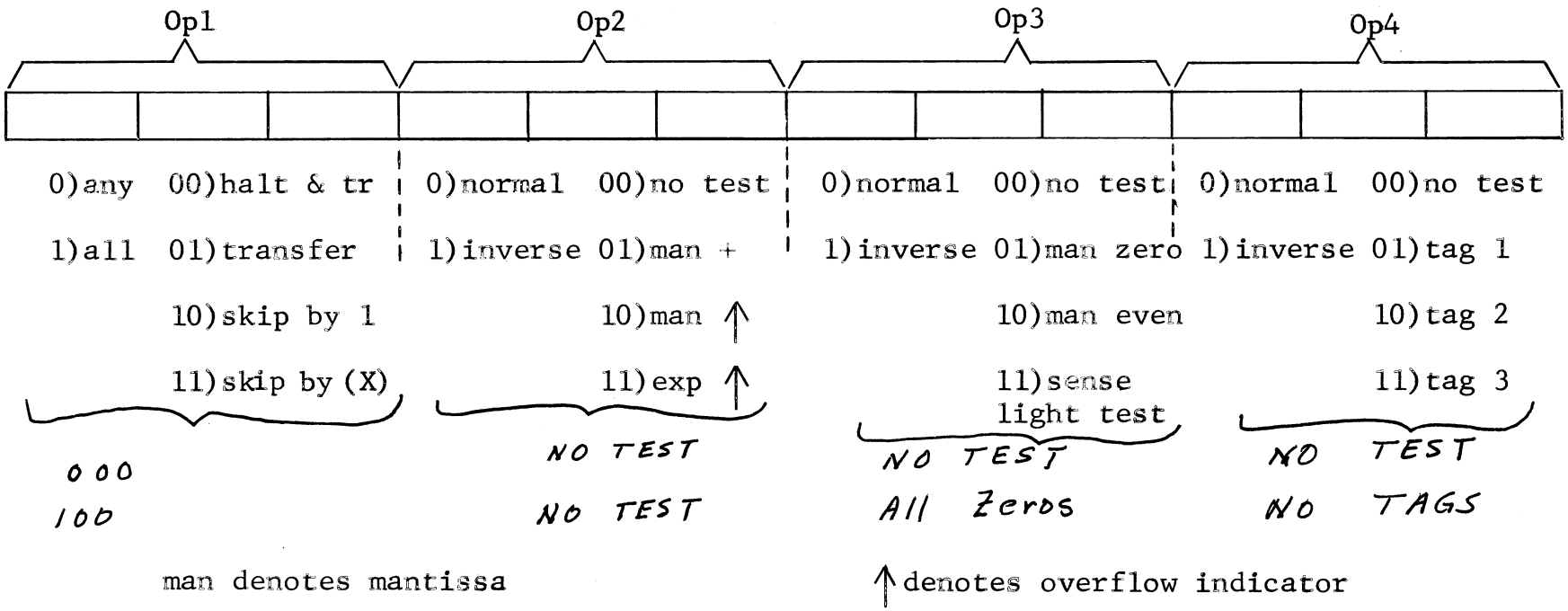
| C | class |
|---|---|
| 0 | control: compare, skip, or transfer |
| 1 | arithmetic |
| 2 | store, substitute, set tag |

| C | class |
|---|---|
| 3 | not used - reserved for future use |
| 4 | B register modify, set sense, shift |
| 5 | logical arithmetic |
| 6 | input-output |
| 7 | special functions |

Class 0 - Control: Compare, Skip, or Transfer

This is a very flexible family of conditional transfers. Op2, Op3, and Op4 are used to specify a set of zero, one, two, or three tests. The inverse of every test is possible. Bit 1 of Op1 specifies whether we mean a favorable outcome for the whole test to be (0) a favorable outcome from any one test, (1) a favorable outcome from all tests. Bits 2 and 3 of Op1 give the action to be taken on favorable outcome, i.e., some special adjustment of CC.

class 0 operations




man denotes mantissa

↑denotes overflow indicator

The first bit of Op2, Op3, and Op4 indicates whether we mean the normal form or the inverse form of the indicated test.

The various tests are applied to:

(1) special indicators (e.g., overflow, tag, etc.)

(2) (U) - (S) (floating point subtraction)

(3) sense register

In case (2), the exponents of (U) and (S) are tested. If they are both zero, effectively a fixed point subtraction is carried out; the result of the subtraction goes to U, and S is cleared to zero. ~~and both (U) and (S) are left unchanged~~. If both exponents are not zero, a floating point subtraction is carried out, and (U) and (S) are possibly changed by shifting and normalizing procedures. In either case, the quantity (U) - (S) is ~~not~~ available in U after the test.

In a transfer order M denotes the address to which one is transferring; thus the number in S is not used. If Op1 = 0, 1, 4, or 5, zero→S before the test and we test (U) - 0 = (U). In view of this, one can also say that the computer has two classes of control orders:

(1) test (U) and/or indicators and then transfer to M

(2) compare (U) with (S), test indicators and then skip by 1 or (X)

If no tests are specified (i.e., an ignore test in Op2, 3, and 4), the function specified in Op1 will be executed unconditionally.

Op1:

0) on any test successful, halt; transfer to M when start button is pressed

1) on any test successful, M→(CC) (transfer)

2) " " " " , (CC) + 1→(CC) (skip)

3) on any test successful, (CC) + (X) $\rightarrow$ (CC) (relative transfer)

4) on all tests successful, halt; transfer to M when start button is pressed

5) " " " " , M $\rightarrow$ (CC)

6) " " " " , (CC) + 1 $\rightarrow$ (CC)

7) " " " " , (CC) + (X) $\rightarrow$ (CC)

Op2:

0) ignore Op2

1) mantissa positive?

2) mantissa overflow?

3) exponent overflow?

4) ignore Op2

5) mantissa negative?

6) no mantissa overflow?

7) no exponent overflow?

Op3:

0) ignore Op3

1) mantissa [redacted] + 0 or - 0?

2) mantissa lower bit zero? (equivalent to "mantissa even"?)

3) sense lights designated by 1's in M on?

4) ~~ignore Op3~~ is every bit in U zero?

5) mantissa nonzero?

6) manitssa lower bit one? (equivalent to "mantissa odd"?)

7) sense lights designated by 1's in M off?

Op4:

0) ignore Op4

1) tag indicator 1 on?

2) tag indicator 2 on?

3) tag indicator 3 on?

4) tag indicators all off?

5) tag indicator 1 off?

6) tag indicator 2 off?

7) tag indicator 3 off?

(4 is not the true inverse of 0 but appears to be too useful to leave out.)

Class 1 - Arithmetic

~~At the present, only Op2 is used. Op1, Op3, and Op4 will be ignored and may be written as zeros.~~ As denoted by OP2 (U) and (S) are combined as follows:

| Op2 | operation |
|---|---|
| 0) | fixed point addition |
| 1) | fixed point subtraction |
| 2) | fixed point multiplication |
| 3) | fixed point division |
| 4) | floating addition |
| 5) | floating subtraction |
| 6) | floating multiplication |
| 7) | floating division |

With OP1 = 3 or 7 (division order), OP3 may be used as follows:

OP3:

0) Leave (U) and (R) unchanged (double length dividend)

1) Clear (Rm) to sign of (Um) before division (single length dividend)

2) Clear (Um) to sign of (Rm) before division (integer division)

3) ⋮ 7) } not used

That is, bit 1 determines fixed or floating arithmetic while 2 and 3 determine +, -, x, ÷. In addition and subtraction, (U) ± (S)→U. In multiplication, the most significant 47 bits of the mantissa of (U) x (S) go to U, with the remaining 47 bits going to R. The exponent and sign of mantissa of R are set to agree with U. The manitssa of R is merely the continuation of the mantissa of U. Division is exactly the reverse of multiplication. (U), with the mantissa of R being understood as the continuation of the mantissa of U, is divided by (S). The exponent and sign of (R) is ignored. The quotient appears in U and the remainder in R.

In fixed point addition, the exponent part of S is simply transferred to U, replacing the previous exponent. Thus, a fixed point addition with field 1 = 0, 0 is a load (or fetch)command. However, a logical "or" with field 1=0,0 is the usual load order.

← INSERT

Class 2 - Store or Substitute, Set Tag

Op1:

0) store (U)→M

1) substitute part of (U)→M ; in detail, part of (U)→S, then (S)→M

2) add to memory: (U) + (S)→M

3) ~~substitute to memory: part of [(U) + (S)]→M (unless a 1 has been used in bit 1 of IM, (S) will have come from M)~~ Not used

4), 5), 6), 7) tag location M (no. in M is not affected)

Op2: not used

Op3: this triad is interpreted only for substitute orders (1 or

3 in Op1)

0) substitute the M triads of (U), i.e., last 5 triads into S and store (S)→M; note that the former contents of M are in S if correct IM is used.

1) substitute right half (27 bits) of (U)→M

2) substitute left half (27 bits) of (U)→M

Op4: this triad may adjust the tag at address M

0) set tag to no tag

1) set tag to tag 1

2) set tag to tag 2

3) set tag to tag 3

4) Send contents of arithmetic tag register to tag bits at M

~~4~~, 5), 6), 7) ignore tag (do not change tag) on M)

We can store (U), substitute from (U), or add (U) to memory and at the same time clear, adjust, or ignore the tag condition at M. One note of caution about the substitute order is needed. The order actually substitutes part of U into S and then stores (S) at M. In order to be a true substitute in memory order, one must be sure that bit 1 of IM is zero (i.e., (M) must first come to S).

We also note that the overflow bit in U cannot be transferred to memory since it exists only in U.

Class 3 - Not Used

Class 4 - B Register Arithmetic, Shift, Set Sense and Mode Lights

The operations of this class are all controlled by the final 15 bit integer M in I. The original M is used as a number or control symbol unless IA = 1. The particular operation is specified by the triad in Op1 as follows:

Op1:

0) increment ($B_i$) by M, i.e., ($B_i$) + M$\longrightarrow B_i$

1) arithmetic shift of U/R by M places

2) ~~set sense~~ turn on lights designated by 1's in M

3) logical bit count of (R) for M places

4) set ($B_i$) to M, i.e., M$\longrightarrow B_i$

5) logical shift of U/R by M places

6) ~~set mode~~ turn off lights designated by 1's in M

7) not used

With Op1 = 0 or 4, Op2 designates which B register is to be set to M or incremented by M, as follows:

Op2:

0) $B_0 \equiv$ CC

1) $B_1$

2) $B_2$

3) $B_3$

4) $B_4$

5) $B_5$

6) $B_6$

7) $B_7 \equiv$ PF1

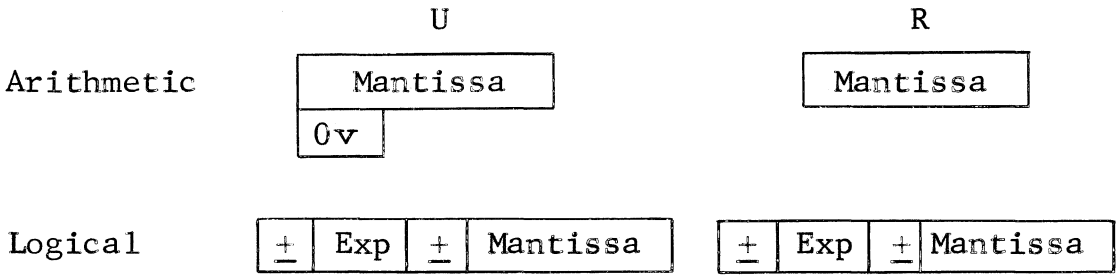
INSERT:

In an arithmetic shift of U or R to the right the sign is propogated to the right as many places as the contents of the register are shifted. In an arithmetic shift of U or R to the left, the sign fills in those places on the right into which nothing is being shifted.

Shifting: a short discussion of shifting in the Rice Computer will help clarify this set of instructions.

Consider a shifting register (e.g., U or R). This register may be considered to have a "donor" and an "acceptor" stage for bits. For example, when U shifts right, $\longrightarrow$, the right hand stage is the "donor" and the left hand stage is the "acceptor".

The shifts fall into two major classifications: (1) arithmetic and (2) logical. In the arithmetic class, the left hand bit of U is the overflow bit just to the left of the mantissa and the right hand bit is bit number 54; the left hand bit of R is bit number 8 (the highest order bit in the mantissa) and the right hand bit of R is again number 54. INSERT In the logical shifts, the entire word for both U and R is used, that is, the exponent sign, exponent, sign, and mantissa (the overflow bit in U being ignored). Thus for the logical shifts, the left hand bit is the exponent sign and the right hand bit is bit number 54.




Each register (U or R) may be shifted right or left M (mod $2^7$) times, taking into its "acceptor" stage either zeros or the spill from the "donor" of the other register. The

number of bits entering the acceptor stage is equal to M(mod $2^7$), that is, the number of positions shifted.

The pattern of shifts is specified by the triads Op3 and Op4, with Op3 determining the behavior of the U register and Op4 that of the R register in the following manner:

Op3

| Bit 1 | Bit 2 | Bit 3 |
|---|---|---|
| into U "acceptor" | U left | U right |
| 0) zeros | 0 times | 0 times |
| 1) spill from R | M times | M times |

Op4

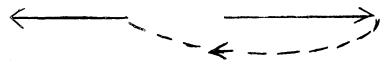
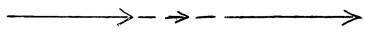
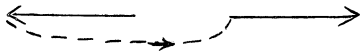
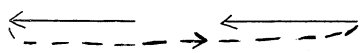
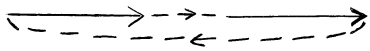
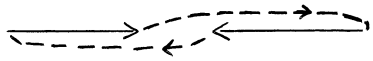
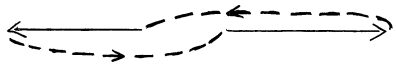
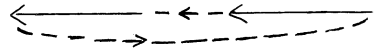
| Bit 1 | Bit 2 | Bit 3 |
|---|---|---|
| into R "acceptor" | R left | R right |
| 0) zeros | 0 times | 0 times |
| 1) spill from U | M times | M times |

The pattern of shifts can also be pictured in the following way:

| Op3 | Op4 | U | R |
|---|---|---|---|
| 1 | 0 | → | |
| 2 | 0 | ← | |
| 0 | 1 | | → |
| 0 | 2 | | ← |
| 1 | 1 | → | → |
| 2 | 1 | ← | → |
| 1 | 2 | → | ← |
| 2 | 2 | ← | ← |
| 5 | 1 | → | → |

The only connections that are meaningful for the arithmetic shifts are $U_{54}$ to the left end of R and the left end of R to $U_{54}$, i.e.

In other words, any spill from the extreme ends of U or R are lost, and only ones or zeros will be accepted at these places. Hence, the full range of shifting possibilities is available in logical shifts only.

| Op3 | Op4 | U | R |
|---|---|---|---|
| 5 | 2 |  | |
| 6 | 1 |  | |
| 6 | 2 |  | |
| 1 | 5 |  | |
| 2 | 5 |  | |
| 1 | 6 |  | |
| 2 | 6 |  | |
| 5 | 5 |  | |
| 5 | 6 |  | |
| 6 | 5 |  | |
| 6 | 6 |  | |

← *Insert*

All shifts are controlled by the 15 bit integer M in the instruction register, modulo 128.

The procedure for logical bit count is as follows:

(1) U and S are cleared to zero.

(2) The shift pattern specified by Op3 and Op4 is executed with the spill out of the low end of R going to the lowest order stage of the adder. Thus the bits are added one at a time (with each shift) to U. By convention we use 0, 1 for the Op3, Op4 code in the bit count.

~~The set sense lights and set mode lights instructions~~

With Op 1 = 2 or 6 the lights turned on or off respectively are as follows:

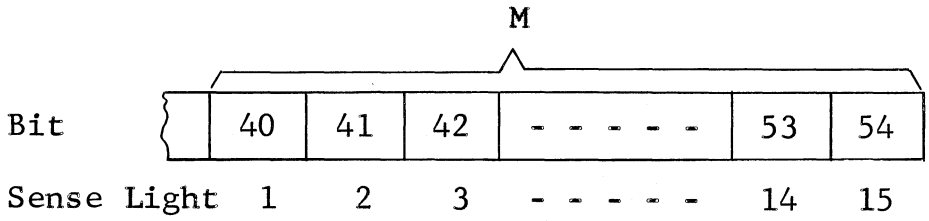
0) sense lights

1) mode lights

2) trapping lights

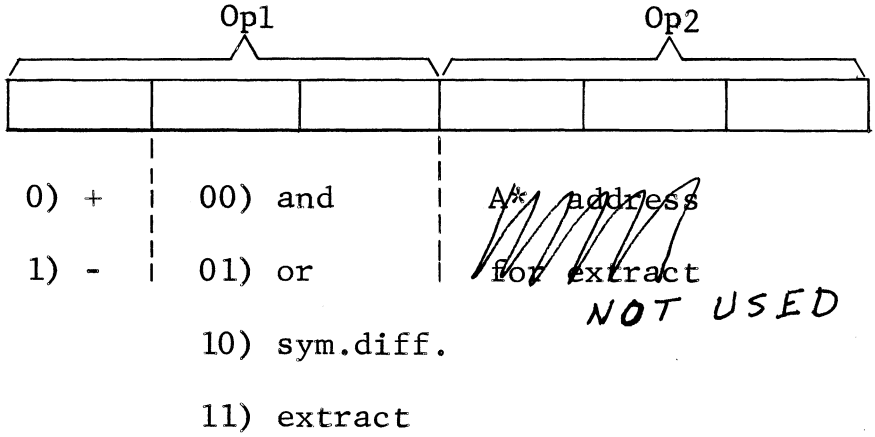
3), 4), 5), 6), 7) not used

Lights corresponding to zeros in M are not affected.




Class 5 - Logical

Only Op1 ~~and Op2 are~~ is used. (U) and (S) are combined by a logical operation.



These operations will work on all 54 bits. The first bit enables us to negate (one's complement) the final result. In "and", "or", and "symmetric difference" corresponding bits in U and S are compared and the usual logical result placed in U. Extract is more complicated and involves three words. Below we give the combination tables for "and", "or", and sym. diff. and their negations:

Op1:

0) and

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

4) neg and

| | 0 | 1 |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 0 |

1) or

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

5) neg or

| | 0 | 1 |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 0 |

2) sym diff

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

6) sym sum

| | 0 | 1 |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 1 |

3) extract

7) neg extract

The extract order is outlined by:

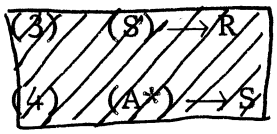
(M) thru (R) into (F) ⟶ U

The contents of ~~an address A* (set up in Op2)~~ S is extracted through the contents of R as a mask ~~(this word is set up by~~

~~memory address M and inflections~~) into (F), set up as usual by field 1. The sequence of operations is as follows:

(1) (F) $\longrightarrow$ U (from field 1)

(2) (M) or M $\longrightarrow$ S (from field 4)




(3) The bits of (S) corresponding to 1's in (R) are substituted into U; the remaining bits of U are unchanged.

The operation neg extract merely forms the 1's complement of this final result in U.

Class 6 - Input-Output

The Rice Computer will have the following auxiliary input-output equipment:

(1) one optical paper tape reader

(2) one fast line printer

(3) one console typewriter

(4) one paper tape punch

(5) several magnetic tape units (Provision will be made in the vocabulary to be able to add an arbitrary number of units in the future. Initially there will be two units.)

Of these five pieces of equipment, the magnetic tape units will have the most complicated and versatile order code.

For a complete understanding, the reader should read the sections in this manual devoted to the detailed description of each unit. However, for the purpose of following the description of this class of orders the following brief outline will suffice.

The paper tape used is seven hole tape (i.e., the optical reader may read seven bits at a time). The information in six positions (a hexad) is transferred to and from the machine. The seventh position is used for the purpose of control and is not read into the memory. A punch in the seventh position means that the corresponding hexad is not to be read into the machine but is to be interpreted as a control on reading (e.g., delete or ignore, end of word, end of tape).

Paper tape will be prepared on an electric typewriter punch which will punch one hexad at a time.

The optical reader may be used to read (1) a whole tape at a time (terminated by an "end-of-tape" control punch) or (2) one to nine hexads at a time (as specified by the coder). The first choice may also be initiated by a load switch on the control console.

The line printer will have 64 characters. These will include the numbers from 0 through 9, mathematical symbols, lower case letters a through f, upper case letters A through Z, and special symbols. In addition, there will be provisions for format control by means of a control tape on the printer itself.

The console typewriter can be used to type octal (i.e., binary) information into the instruction register and into the S register. In addition, the console typewriter can be used to obtain the octal contents of any of the F registers (both A series and B series). This can be accomplished by means of a stored instruction or keys on the typewriter. There will be one key for each register. When a given key is struck, the contents of the corresponding register will be typed in octal.

The paper tape punch can be ordered to punch one to nine hexads at a time or special control punches.

The contents of the memory can be recorded (i.e., written) on magnetic tapes in blocks of arbitrary length. These blocks can be grouped in files.

The triad, Op1, will be used to designate the unit or units selected and their functions. The remaining triads, Op2-Op4, then are used to designate various inflections and details.

Op1:

0) paper tape control (~~either~~ read and/~~or~~ punch)

1) type [(M) $\longrightarrow$ console typewriter in octal]

2) printer control

3) magnetic tape control

4) dump memory control

5) cathode ray display control

6), 7) not used

The detailed description of each function is as follows.

Op2:

This triad specifies the various options on the "read paper tape" and "punch paper tape" order.

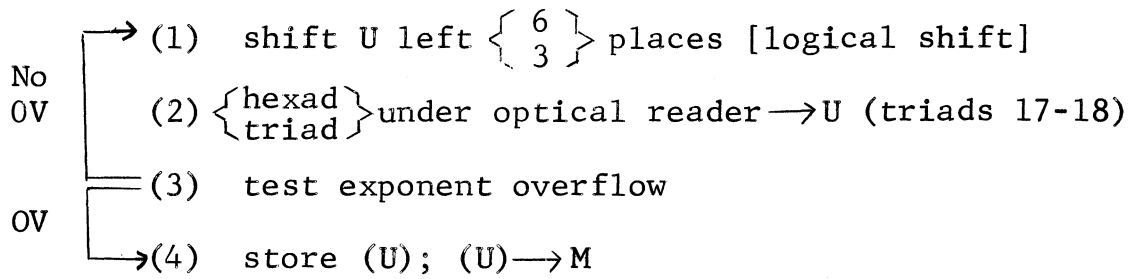
Op2

| | Bit 1 | Bit 2 | Bit 3 |
|---|---|---|---|
| 0) | read | punch hexads | hexad mode |
| 1) | punch | punch hexads + 7th hole | octal mode |

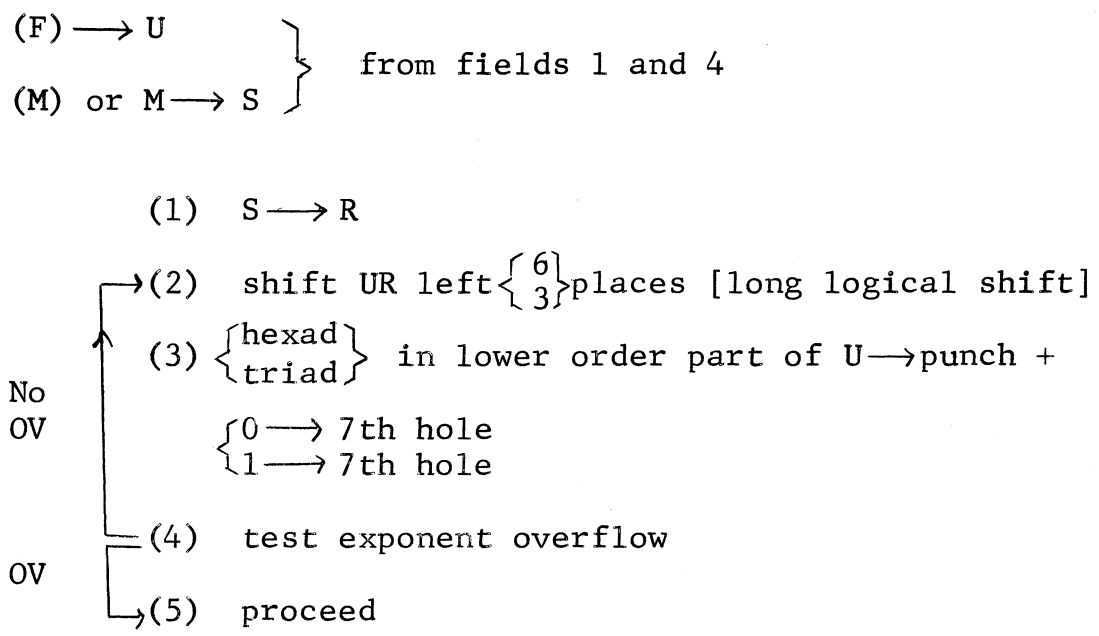
Description of read and punch paper tape orders:

In the read $\left\{\begin{matrix}\text{hexad}\\\text{octal}\end{matrix}\right\}$ mode the following sequence is executed:

$$\left.\begin{matrix}(F) \longrightarrow U\\(M) \text{ or } M \longrightarrow S\end{matrix}\right\} \text{ usual decoding of fields 1 and 4}$$




In the punch {hexad / octal} mode, the following sequence is executed:




Bit 2 of Op2 refers only to the punch orders.

Op3:

This triad controls the various output formats of the line printer on the print order.

Loading of paper tape

A. Automatic starting of the machine is accomplished by loading the paper tape reader and pressing the load button on the control console. Pressing the load button does the following:

1) Sets CC = 1
2) Turns on the repeat mode light
3) Sets I to the load instruction (see below)
4) Starts paper tape feed

The load instruction consists of the following, in the order of decoding:

Field 1

| IF | F | |
|---|---|---|
| 1 0 0 | 0 0 0 | (CC)→U, placing a 1 in the lowest order bit of U. |

Field 4

| IM | IA | BM | M |
|---|---|---|---|
| 0 0 0 | 0 | 0 0 0 0 0 0 0 1 | 000 000 000 000 111 |

When Field 4 is decoded the first time ($I_M$) is the first true memory address 000 000 000 001 000. Each time the order is repeated, ($I_M$) is increased by (CC) = 1.

Note: If it is desired to start loading at an address other than 10, the desired read instruction must be manually typed into the I register and the start button must be used.

Field 2

| C | OP 1 | OP 2 | OP 3 | OP 4 |
|---|---|---|---|---|
| 1 1 0 | 0 0 0 | 0 0 0 | 0 0 0 | 0 0 0 |

This field indicates that the order is an input-output order and specifically denotes a read hexad paper tape function.

Field 3

Not used.

The overall operation of the load procedure is the following:

The paper tape reader is loaded with the desired tape and the load button is depressed. Field 1 is decoded and places a 1 in the lowest order bit of U. This is accomplished by sending (CC)→U. The machine begins reading hexads and shifting them into the U register. This is continued until the 1 which was previously placed in the lowest order bit of U reaches exponent overflow. Then (U)→M which initially is 10...

Since the repeat mode light is on the order is repeated until the end of the tape punch turns off the repeat mode light. The last word is then read from tape into U and (U) -> M, leaving the last word in U. The instruction register then consults CC for the address of the next instruction. (CC) = 1. Therefore the next instruction is (U) which is the last word from paper tape. Thus the memory is loaded and the first instruction is in the I register.

B. For manual starting or alteration of the normal loading procedure, the control console also has a selector switch and a start button. The selector switch (15 push buttons) selects registers for manually typing in instructions and the start button serves to remove halt conditions (Field 2, Class 0) as well as manually starting the machine from a typed instruction.

Op 3:

0) no space.

1) space ½ line

2) space 1 iine

3) space 2 lines

4) format #1 on printer (usually restore half page)

5) format #2 on printer (usually restore full page)

6) format #3 on printer

7) format #4 on printer

The coder has the option of preparing up to four of his own formats for spacing control on the printer.

The orders pertaining to magnetic tape should be ignored at this time. The principal reason for this is that a simple buffer and checking system is now being prepared for the computer. A new description of magnetic tape input and output will be written as soon as possible.

Class 7 - Special Functions.

We plan initially to have the square root ( $\sqrt{(U)} \rightarrow U$) and eventually a number of other special functions, depending upon the need for such functions and the feasibility of building the necessary circuits.

The ~~seven~~ eight manipulations which may be performed are as follows:

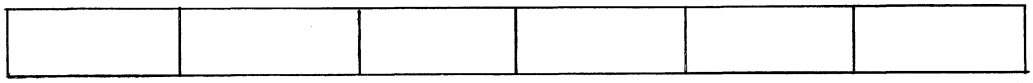
| | | |
|---|---|---|
| 000 | 0 | contents of U → an A address |
| 001 | 1 | " " R → an A address |
| 010 | 2 | $(B_i) + 1 \rightarrow B_i$ |
| 011 | 3 | $(B_i) + (X)$ [increment register] $\rightarrow B_i$ |
| 100 | 4 | address portion of U → $B_i$ |
| 101 | 5 | " " " R → $B_i$ |
| 110 | 6 | $(B_i) - 1 \rightarrow B_i$ |
| 111 | 7 | address portion of instruction register → $B_i$ |

Note that if I St = 0 or 1, St is an A address

" " = 2 - 7 " " a B "

## 4. Field 3

The two triads of field 3 (ISt, St) offer an additional flexibility to each instruction by allowing the coder to obtain "free" (i.e., without an additional order or access time) one of seven convenient manipulations. The decoding of this field is independent of the operation code and all seven choices are available with all possible orders. St is a fast address (see discussion of addresses). Bit 1 of ISt determines whether we mean an A or a B address. Bits 2 and 3 of ISt specify one of the four permissible manipulations, namely, store (U) in a fast register, store (R) in a fast register, advance $B_i$ by 1, or advance $B_i$ by the contents of X, the increment register.




**ISt:**

| | | |
|---|---|---|
| 0) | $(U) \rightarrow A_i$ | 4) $(U) \rightarrow B_i$ |
| 1) | $(R) \rightarrow A_i$ | 5) $(R) \rightarrow B_i$ |
| 2) | $(B_i) + 1 \rightarrow B_i$ | 6) $(B_i) - 1 \rightarrow B_i$ |
| 3) | $(B_i) + (X) \rightarrow B_i$ | 7) $(I)_{40-54} \rightarrow B_i$ |

**where $A_i$ is an A-series register and $B_i$ is a B-series register.**

**St: i = 0, 1, ..., 7, the address of the desired register in the A- or B-series.**

field 3 is left blank (all zeros), this section will be ignored and the computer will then fetch the next instruction.

Examples of field 3:

| octal code | meaning |
|---|---|
| 00 | ignore |
| 62 | $B_2 - 1 \rightarrow B_2$ |
| ~~37~~ | ~~meaningless - do not use~~ |
| 03 | ~~(U) → S~~ $(U) \rightarrow S$ |
| 14 | $(R) \longrightarrow T_4$ |
| 24 | $(B_4) + 1 \longrightarrow B_4$ |
| 33 | $(B_3) + (X) \longrightarrow B_3$ |
| 40 | $(U)_{14-18} \longrightarrow CC$ (an effective transfer) |
| 75 | $(I)_{40-54} \rightarrow B_5$ |

Whenever the last 15 bits of a long register (e.g., U) are sent to a B register, ~~the mantissa sign of the 54 bit register is also examined. If the sign is plus, the 15 bits are transferred without any change. If the sign is negative, the 2's complement of the 15 bits is placed in the B register (i.e., the number in $U_{14-18}$ plus the sign is transferred to B).~~ they are transferred without any change

Examples should be disregarded as printed in view of changes in machine language.

VI.

## EXAMPLES OF SINGLE INSTRUCTIONS

In order to illustrate the procedure of microprogramming instructions (that is, composing a single instruction) in machine language, the following arbitrary list of instructions is presented.

The instructions are grouped according to class. The numerical code in octal is first given for each order. The octal digits are arranged according to the following pattern:

| field 1 | field 2 | field 3 | field 4 | |
|---|---|---|---|---|
| XX | XXXXX | XX | XXXX | XXXXX |
| IF, F | class & Op | | IM, BM | M |

Below each numerical pattern is a symbolic description of what the order will accomplish. The reader is strongly advised to check the numbers against the previous outline of the order code and to try to compose a few instructions. After a few attempts, one will realize how easy it is to memorize the various field codes to compose orders.

Class 0

00 00000 00 0000 00000

Stop. A class 0 (control) order, halt and transfer variety which is unconditional; S and U are cleared, CC is set to zero.

01 01200 06 4000 06122

Transfer to location 6122 if mantissa overflow indicator is on; U is unchanged and $(U) \longrightarrow T_6$; $6122 \longrightarrow S$.

01 01000 00 4001 00015

Jump forward 15 instructions. U is unchanged and the address (CC) + 15 $\longrightarrow$ S, thus saving a memory access time.

51 06150 00 5000 00007

Skip CC by 1 when $-(B_1) + 7 > 0$ (positive, non-zero)

42 03120 61 4002 00007

Skip CC by (X) when $+(B_2) - 7 - (B_1)$ is positive or even. In any case, advance $B_1$ by 1.

04 07105 61 0002 00555

This order can be used to search consecutive memory locations starting at 555 until a number larger algebraically than the one in $T_4$ is found or tag 1 is reached (the end of the table). This is accomplished by assuming that X contains -1 (i.e., the two's complement of 1). As a note of caution we recommend that such an order be in the T registers or that the repeat mode be used. Otherwise the chance of a read-around error may become appreciable.

Class 1

00 10000 00 0000 12345

$0 \longrightarrow U$ and $(12345) \longrightarrow S$; fixed point (U) + (S) $\longrightarrow$ (U) with exponent of (S) $\longrightarrow$ exponent of (U). Hence, $(12345) \longrightarrow U$.

04 10600 04 0002 12345

$(T_4) \times (12345 + (B_1)) \longrightarrow U$ (floating point) then $(U) \rightarrow T_4$

41 10200 53 4004 00000

$(B_1) \times [(B_2) + 0] \longrightarrow U, R$ (fixed point)

$(R)_{14\text{-}18} \rightarrow B_3$ i.e., $[(B_1) \times (B_3)] \pmod{2^{15}} \longrightarrow B_3$

Note that since we consider the numbers in B registers as integers, they are sent to the lower bits of U and S and the product is formed in the lower order 30 bits of R.

Class 2

01 20000 00 0000 12345

$(U) \longrightarrow 12345$ (clearing tag to zero)

07 21023 00 0002 12345

Bits 28-54 of $(T_7) \longrightarrow$ bits 28-54 in location $[12345 + (B_1)]$ with a tag 3.

03 20000 61 0002 12345

$(S) \longrightarrow U$

$(12345 + (B_1)) \rightarrow S$

then $(U) \longrightarrow 12345 + (B_1)$ and $(B_1) + 1 \longrightarrow B_1$

Together with the repeat mode, this can be used to shift a block of numbers in storage.

Class 4

44 44400 07 4002 12345

$(B_4) \longrightarrow U \longrightarrow T_7$ (i.e., $(B_4)$ is saved in $T_7$)

$12345 + (B_1) \longrightarrow B_4$

In the same instruction $12345 + (B_1) \longrightarrow S$, thus the new number in $B_4$ is also available in S.

01 40400 00 4400 12345

$(B_4) + (12345)_{14\text{-}18} \longrightarrow B_4$

We assume that location 12345 only contains a non-zero number in the M section (triads 14-18), that is, with no modifications. This is a means of incrementing B with a number in memory.

01 45056 00 4000 00066

This instruction inverts the order of the bits in U and sends the results to R and at the same time inverts the order of the bits in R, sending the result to U.

22 46010 73 4000 50000

$-(R) \longrightarrow U$

Turn on sense lights 1 and 3

then increment $B_3$ by (X), i.e., $(B_3) + (X) \longrightarrow B_3$

Class 5

01 55000 00 0002 12345

(U) "or" (12345 + $(B_1)$), then complement result.

06 53700 00 0000 12345

$(T_7)$ through (12345) into $(T_6)$, result $\longrightarrow$ U

For example, consider the initial bits:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $T_7$ | 1 | 1 | 0 | 0 | 1 | - | - | - | - |
| 12345 | 0 | 1 | 1 | 0 | 0 | - | - | - | - |
| $T_6$ | 0 | 0 | 1 | 1 | 1 | - | - | - | - |
| result | 1 | 1 | 0 | 1 | 1 | - | - | - | - |

Class 6

10 60000 61 4002 01000

Read one hexad from paper tape to location 1000 + $(B_1)$. The hexad is located in the lower order 6 bits; all other bits are zero except the exponent sign which is negative. In addition, $(B_1) + 1 \longrightarrow B_1$. If the repeat mode is used in conjunction with this order, a series of hexads can be read into a block of storage. The repeat mode light is turned off by a special control punch.

02 60100 61 4002 01000

We assume that R contains a one in bit 54 and zeros elsewhere. This order then reads 18 triads or a word from tape into memory location 1000 + $(B_1)$. Then $(B_1) + 1 \longrightarrow B_1$.

```
10  60600  00  0002  12345
```

Punch the exponent plus sign of location 12345 + ($B_1$) onto paper tape and punch control hole.

Special purpose register $(77775)_8$ is known as the indicator register. The bits of this register are themselves indicators which describe certain conditions of the computer. If the bit is 1, the indicator is said to be "on" or the corresponding condition has been recorded.

| bit | condition |
|---|---|
| 1 | arithmetic tag indicator #1 |
| 2 | arithmetic tag indicator #2 |
| 3 | arithmetic tag indicator #3 |
| 4 | U mantissa overflow indicator |
| 5 | U exponent overflow indicator |
| 6-15 | not used |

# VII

# INDICATORS

The Rice Computer will have available a number of indicators useful in determining the logical control of the problems.

← INSERT

## Overflow and Tag Indicators

These indicators are either on or off and can be tested by means of a class 0 test instruction. Transfer of control can be effected by such instructions on either the "on" or the "off" status of the indicator. Whenever an indicator is turned on, it remains on until it is tested. When it is tested, it is turned off regardless of its prior condition. The reader is advised to study the numerical word structure before reading this section.

| abbreviation | full name |
|---|---|
| MAN ↑ or MANOV | U register mantissa overflow |
| EXP ↑ or EXPOV | U register exponent overflow |
| tag 1 | tag indicator # 1 |
| tag 2 | " " # 2 |
| tag 3 | " " # 3 |

The status of these indicators is displayed on the console in the form of small neon lights.

Mantissa Overflow Indicator

This indicator is turned on when the word in U has a positive sign on its mantissa and a 1 is carried past the binary point into the overflow position or when the word in U has a negative sign on its mantissa and a 0 is carried past the binary point into the overflow position. This can be the result of the execution of a fixed point arithmetic or an arithmetic shift command. An example is a carry resulting from an algebraic addition. The indicator can be turned off by testing it. [See also the section on floating point arithmetic.]

Exponent Overflow Indicator

This indicator is turned on when the exponent of U has a positive sign and a 1 is carried past the first bit of the exponent of the U register (i.e., bit 2 in U) or when the exponent of U is negative and a 0 is carried past the first bit. This can be the result of the execution of a floating point arithmetic order or a logical shift left. [See also the section on floating point arithmetic.]

Tag Indicators

When a word from memory enters the arithmetic unit through the central distributor, the two tag bits are noted according to the following code:

| | |
|---|---|
| 00 | no tag |
| 01 | tag 1 |
| 10 | tag 2 |
| 11 | tag 3 |

If the number in memory is tagged, the corresponding tag indicator is turned on. Note that the two tag bits exist

~~only~~ in memory and are ~~not~~ transferred to the arithmetic ~~unit~~ tag register in the arithmetic unit.
For a discussion of the uses of tags the reader should refer to the section entitled, "Tagging and Trapping."

## Sense Light Register

The sense lights are numbered 1 through 15 and are available to the coder for general use, e.g., control of printing, sequencing of orders, transfers of control, visual indication of certain phases of a calculation, and so forth.

The lights are located on the console with a switch below each light. The lights themselves correspond to the bits in the sense light register (location 77771). When a given bit position has a one in it, the corresponding light is on. The switch below each light may be (1) in a neutral position (the sense indicator is then under internal control); (2) depressed momentarily to turn on the given sense light; (3) locked in the down position which sets the sense indicator to "one" as long as the switch is down; or (4) locked in an "off" position. When a switch is in the neutral position, the sense light may be turned on or off by a set sense instruction. The status of any sense light or group of sense lights may be tested at any time. The test does not affect the status of these indicators.

## Mode Light Register

The mode lights are numbered one through 15 and correspond with machine operation modes one through fifteen. Modes one through six have been assigned as noted below; the remainder are reserved for assignment to be made at a later time as need dictates.

Mode lights are essentially sense lights which control and indicate the status (in use or not in use) of their corresponding machine modes of operation. The lights are located on the console with a switch below each light. The lights themselves correspond to the bits in the mode light register (location 77773). When a given bit position has a 1 in it, the corresponding light is on, indicating that the mode is in use. The switches associated with the lights are used in exactly the same manner as those associated with the sense lights (see the previous section).

| mode light | control specification |
|---|---|
| 1 | ignore error stop mode |
| 2 | repeat mode |
| 3 | trapping mode |
| 4 | significance mode |
| 5 | round mode |
| 6-15 | reserved for future assignment |

When mode light one is on, the machine ignores all automatic error stops (e.g., improper division). When mode light 2 is on, the "fetch and advance CC" operation is omitted and the computer will repeat the current order. When mode light 3 is on, the machine will operate in the trapping mode (see the section on tagging and trapping). When mode light 4 is on, the machine will perform floating point arithmetic in the significance mode (see section on floating point arithmetic). When mode light 5 is on, the high order bit of R is added into the low order end of U after floating point additions and subtractions and after all multiplications.

# VIII.

## TAGGING, TRAPPING, AND REPEAT

The numerical word and instruction word in the Rice Computer is 54 bits in length. However, in memory each word has two additional bits called tag bits which allow the coder to tag a word with one of three possible labels. This concept offers many novel features and possibilities in coding.

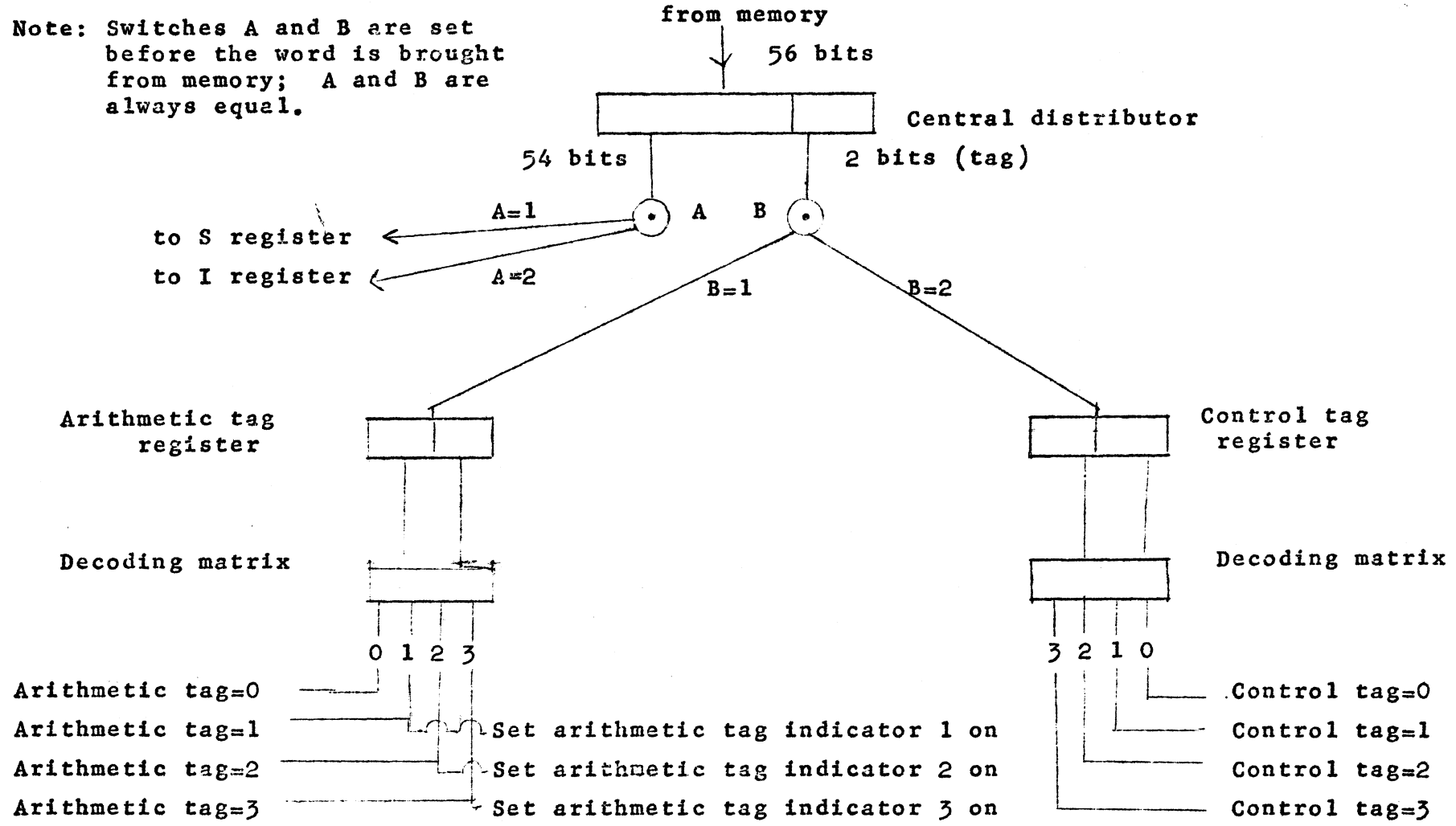
### Tag Registers

The arithmetic tag register is set every time a word goes through the central distributor to the S register. The word in memory consists of 56 bits, the last two of which are the tag bits. All 56 bits come into the central distributor: then the first 54 go to the S register and the last two go into the two-bit arithmetic tag register. The bits of this tag register then go into a decoding matrix which has an output 0, 1, 2, or 3, and this output sets the corresponding tag indicator on and is available for the satisfaction of trapping conditions. It should be remembered that tag indicators remain on until tested by a Class 0 instruction, but the contents of the arithmetic tag register always corresponds to the tag of the last word taken through the central distributor into the S register.

The control tag register is set every time a word goes through the central distributor to the instruction register. The last two bits of the 56 in the central distributor go to the two-bit control tag register. The bits of this tag register go into a decoding matrix with output 0, 1, 2, or 3 which is then available for the satisfaction of trapping conditions. The contents of the control tag register always corresponds to the tag of the last word (or part of the word if indirect addressing is being used) taken through the central distributor into the instruction register.

In cycles or loops which use a series of numbers stored in memory, it is possible to tag the last number and end the cycle or loop by testing for this tag. In two or three dimensional network problems where moving boundaries occur, it is possible to tag certain functions at the boundary points and follow their progress more conveniently. Examples of the use of tagging may be found in the section on coding examples.

Note: Switches A and B are set before the word is brought from memory; A and B are always equal.



## Trapping mode

When mode light 3 (ML3) is on, the computer will operate in the trapping mode. This mode allows the programmer to specify certain tests without using the class 0 or control commands. The computer will automatically execute these special tests on every instruction. If a test is successful the regular sequence of events is interrupted and control is transferred to one of seven fixed trap addresses. This process is called trapping. If mode light 3 is off, all of these special tests are ignored.

In order to use this mode of operation one must know (a) the method of specifying the tests, (b) the point at which the regular sequence of events is interrupted (called the trap time) and (c) the trap addresses. This information is summarized in the table below.

One can specify up to fifteen special test conditions by means of the trapping registers, $(77774)_8$. Corresponding to each bit in the trapping register there is one specific test (see table below). If this bit is a one the test is executed at the proper time but if this bit is a zero the test is ignored. The coder specifies his tests by storing a bit pattern in register $(77774)_8$ before entering the trapping mode.

| Trapping register bit | Test Condition | Trap Time | Trap Address |
|---|---|---|---|
| 1<br>2<br>3 | Arithmetic = 1<br>tag = 2<br>register = 3 | Immediately before decoding field 2 | Transfer to {9, 17, 25} & turn off ML3 |
| 4<br>5<br>6 | Control = 1<br>tag = 2<br>register = 3 | After decoding of field 1 or when indirect address enters I | Transfer to {41, 49, 57} & turn off ML3 |
| 7<br>8<br>9 | Control = 1<br>tag = 2<br>register = 3 | After execution of Field 3 | Transfer to {41, 49, 57} & turn off ML |
| 10<br>11<br>12<br>13<br>14<br>15 | (U) positive<br>(U) negative<br>Mantissa OV indicator on<br>Exponent OV indicator on<br>------<br>------ | After execution of Field 3 and test of Trap bits 7, 8, 9. | If transfer to 41, 49, or 57 is not to be made, then transfer to 33 and turn off ML3. |

The transfer to 41, 49, or 57 after execution of the instruction takes precedence over the transfer to 33; e.g., if TR8 = 1 and control tag register = 2, also TR13 = 1 and the exponent overflow indicator is on after execution of the instruction, then the transfer to 49 takes place and the transfer to 33 is not effective at this time.

When the trapping mode light is off, all trap transfers are ignored and this light is automatically turned off in the process of every trap transfer.

Trapping has extensive application in check-out routines and interpreter-type systems as well as in the conventional control of overflow conditions in the arithmetic unit. The coder should naturally insure, if he uses this feature, that unexpected conditions of the machine do not lead to unnecessary trap transfers.

We also note that the trapping address is given by

Arithmetic tag trap address = 8(contents of tag reg.)+1

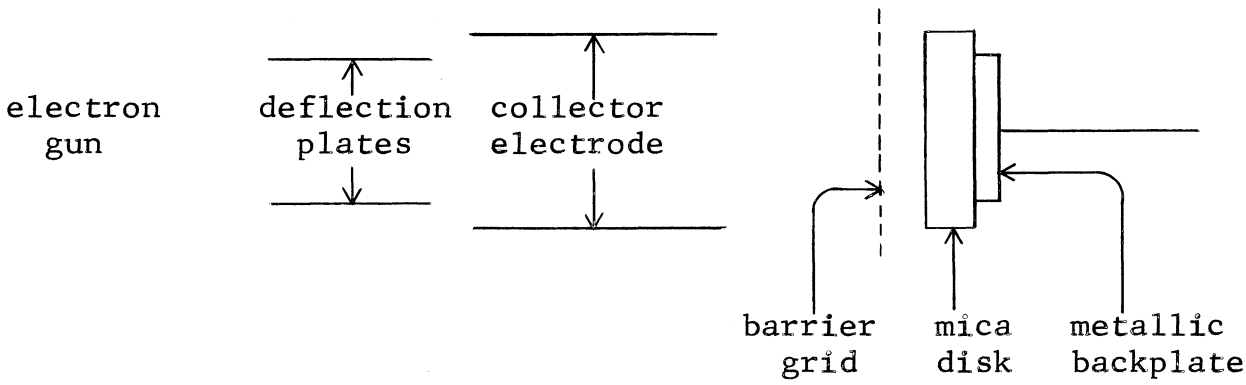
Control tag trap address = 8(contents of tag reg.)+33

# IX.

## ELECTROSTATIC STORAGE OR MEMORY

The Rice Institute Computer will have a random access electrostatic storage. This storage section will consist of 4 independent memory units, each capable of storing up to 8,192 words. [Note: this may be changed to 4,096 words for greater reliability if necessary.]

The basic unit in this memory is a barrier grid storage tube which is pictured schematically in the following drawing:




Information may be stored in the form of a charged spot on the front (left in the above diagram) surface of the mica disk. This is used as a binary element. For example, a plus charge at +25 volts potential relative to the metallic backplate may be interpreted as the digit "0" and a minus charge at -25 volts potential relative to the backplate may be interpreted as the digit "1".

The spot may be "written" or "read" by focusing an electron beam (produced by the electron gun) at the given position in conjunction with appropriate manipulations of the backplate voltage. The detailed electronic explanation of both reading and writing may be found in the technical sections on this subject. For the purposes of this discussion it is sufficient to think of the charged spot together with the backplate as a small condenser which may be charged (writing mode) or discharged (reading mode) by a beam of electrons striking the given spot. The position at which the electron beam hits the mica surface is governed by the voltage on the deflection plates. Thus there is a one to one correspondence between the voltage on the deflection plates and the position of the spot. The storage tube being used has a mica surface 2 inches in diameter and can store up to 8,192 distinct spots without any appreciable overlap.

The 8,192 spots in a given storage tube correspond to one binary position of all the words in memory. For example, each of the spots in Tube No. 5 is the fifth bit of each of the 8,192 words in this memory unit. Thus we see that for a 54 bit word we must use 54 storage tubes in each memory. In addition, each memory contains two extra tubes for tagging purposes (see section on tagging), one tube for a parity ~~check~~ bit (~~see section on error detection and checking~~ and six tubes for check bits), and a monitor tube on which the pattern of "ones" and "zeros" in any tube can be visually displayed.

In reading or writing a word in memory the computer must accept a numerical location number, convert it by means of a digital to analogue device into a set of deflection voltages and apply this to the deflection plates of all 58 tubes. The electron beam is then at the same position in all tubes and the information is transferred in parallel (i.e., all the bits in a word are read or written simultaneously).

Due to the actual physical spread of the electron beam the continuous process of reading and writing tends to destroy the overall information on the mica surface (see discussion of read-around error in the technical sections). Thus each memory unit is engaged in a regeneration process in which each spot is read and clearly rewritten. Regeneration is always interrupted when it is necessary to read or write a number. The entire pattern can be regenerated in about 50 milliseconds and as a general safety rule one should not try to read one particular word more than 1000 times within one continuous 50 millisecond period.

## Error detection and correction

An outstanding feature of the memory of the Rice Computer is that it is self-correcting for errors that occur only one at a time per word. The parity bit indicates the number of ones in a word, and the check bits indicate the arrangement of these ones. Thus if a bit of a word reads differently when it comes from memory than when it went to memory, this condition is sensed and the correction made.

Assuming that the sixty-four bits of the word in memory are numbered 0 through 63, the function of error detection is performed by the parity bit and the six check bits as follows:

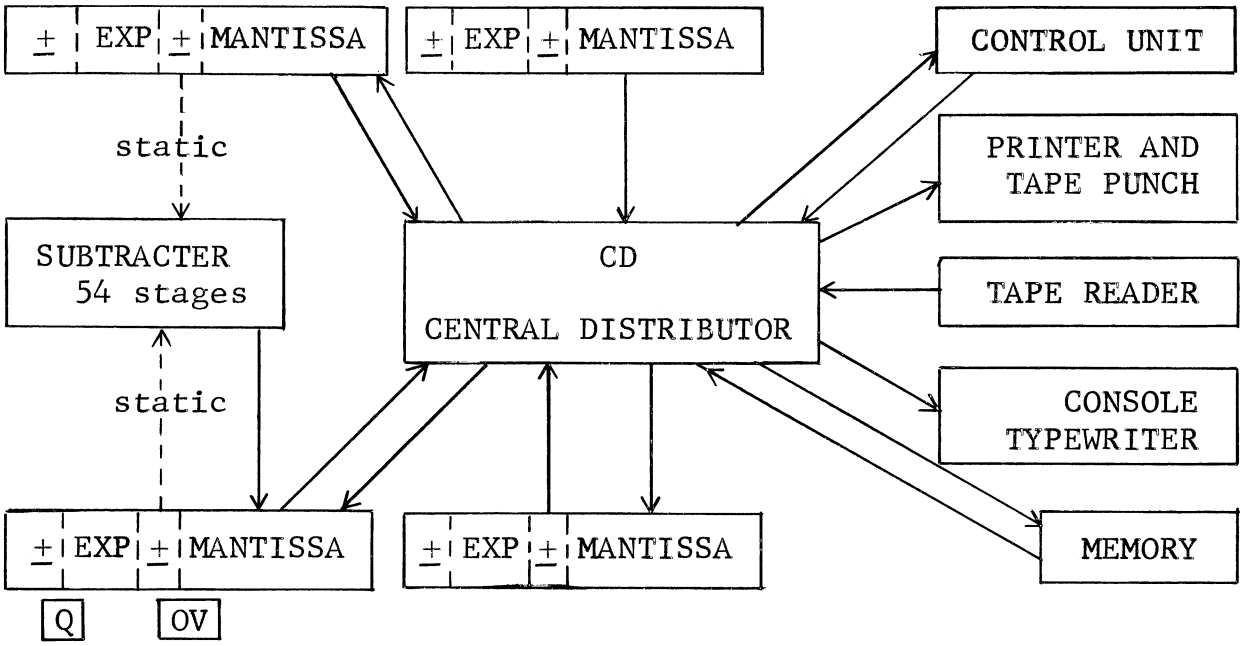
| Parity bit | | whole word | |
|---|---|---|---|
| Check bit #6 | | last half of word | |
| Check bit #5 | ...is 1 if | 2nd and 4th quarters of word | ...is odd |
| Check bit #4 | number | 2nd, 4th, 6th, 8th eighths of word | |
| Check bit #3 | of 1's | 2nd, 4th,...,16th sixteenths of word | |
| Check bit #2 | in ... | 2nd, 4th,...,32nd thirty-seconds of word | |
| Check bit #1 | | bits 1,3,5,....,61, 63 of word | |

These bits are computed when the word is written into memory and again when the word is read out of memory. If the second check bit #6 agrees with the first, there is no check #6 output; likewise for the other check bits and the parity bit. If there is no parity output and no check outputs, then the word is free of errors. If there is parity output, one error is present and the check outputs read the number of the bit that must be complemented in order for the word to again be free of errors. For example, by

| parity output | check outputs | | | | | |
|---|---|---|---|---|---|---|
| | #6 | #5 | #4 | #3 | #2 | #1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |

it is indicated that bit 13 of the word must be complemented. In this case, any larger odd number of errors might be present and then the configuration of check outputs would bear no significance; the chances of this case arising are so small that the possibility is profitably neglected. If there is no parity output but check output is present, then the word contains two (or possibly a larger even number of) errors. Again the check output is meaningless, so the error cannot be corrected; but this condition will result in an error stop.

S register 54 bits  T registers 54 bits (4)




U register 54 bits  R register 54 bits

plus exp overflow bit, Q
and mantissa overflow bit, OV

Figure 2: Block diagram of the arithmetic unit. Arrows represent, schematically, permissible paths for the flow of information excluding shifting. A detailed explanation of the parts of each register and the terms exp, mantissa, etc. can be found in the description of the numerical word structure. "Static" indicates a permanent (or static) connection. Thus the subtracter always contains the difference of the current (U) and (S)

X.

# ARITHMETIC UNIT

The arithmetic unit accomplishes all arithmetic functions and has, in addition, facilities for temporary (or erasable) storage. The execution time for each function will vary from function to function and from operand to operand depending upon the number of zero bits in the operand.

The arithmetic unit is built around an information distributing device called "the central distributor," or CD. CD is not a register since it cannot store information but is only used to transfer information between registers and to and from the other units in the machine. Figure 2 is a block diagram of the arithmetic unit and is an expansion of part of the general diagram in Figure 1. The various registers are listed below and a paragraph description of each register and its function is given:

registers

| abbreviation | name |
|---|---|
| U | universal register |
| R | remainder register |
| S | subtrahend or storage register |
| $T_4$ | temporary store No. 4 |
| $T_5$ | temporary store No. 5 |
| $T_6$ | temporary store No. 6 |
| $T_7$ | temporary store No. 7 |

Universal Register

The U register is involved in every arithmetic and logical arithmetic operation. Before subtraction it contains the minuend (note that the Rice Computer uses a subtractor rather than an adder), before the multiplication the multiplicand, and before division it contains the higher order part of the dividend. At the end of each operation the principal result is found in U. In subtraction U contains the difference, in multiplication it holds the higher order part of the product, and after division the quotient also appears in U.

Remainder Register

The R register is used partly as an intermediate in some operations and as a storage for secondary results of a calculation. The multiplier is placed in R before the execution of a multiplication instruction. This is done automatically by the computer. The lower order part of the dividend before a division is found in R. After multiplication R contains the less significant half of the product and after a division the absolute value of the remainder appears in the mantissa of R, with the sign of the dividend. In fixed point addition or subtraction R plays no role, but in floating point addition it is used (see section on floating point arithmetic). R is also used in certain logical orders (see extract order in the order code).

Subtrahend or Storage Register

Numbers coming from electrostatic memory or the control unit first appear in S before each operation. Before subtraction S contains the subtrahend, before multiplication S holds the multiplier (which is subsequently transferred to R), and before division the divisor is found in S. The contents of S after a general operation will usually be some complicated intermediate and as such will seldom be used.

Temporary Storage Registers

The computer will also have four fast registers used primarily for storing intermediate or temporary results. These are non-shifting registers and will be designated by $T_4$ to $T_7$. Instructions may be stored in these registers and will be fetched at the proper time by the control unit.

Subtractor

U and S are connected statically to the subtracter and the quantity (U) - (S) is available at the output of the subtracter. This output may then be gated into U. An addition is performed by first complementing the sign of S. In actual practice (U) and (S) may be complemented during the process of subtraction for electronic reasons. This will only concern the coder when he is interested in the contents of S after an operation. Most computers in the past have used an adder instead of subtractor,

and common terminology or jargon has been the word "adder" since the result available to the coder in this case was the sum of (S) and (U).

Complementing

Both U and S have the facilities to form the 1's complement of their contents (i.e., each "0" → "1" and each "1" → "0"). The 1's complement of U may be obtained using the logical arithmethic orders. The 1's complement of S is often formed as an intermediate step in an arithmetical calculation. This is one of the reasons why (S) is generally not useful after an operation.

Shifting

Registers U, R, and S are all shifting registers and may be shifted both right and left. On any arithmetic shift order the machine automatically uses the optimum number of shifts of 8 plus shifts of 1. The logical shifts are carried out one bit at a time. U and R may be treated as a double length register and shifted together both in arithmetic operations and in logical operations.

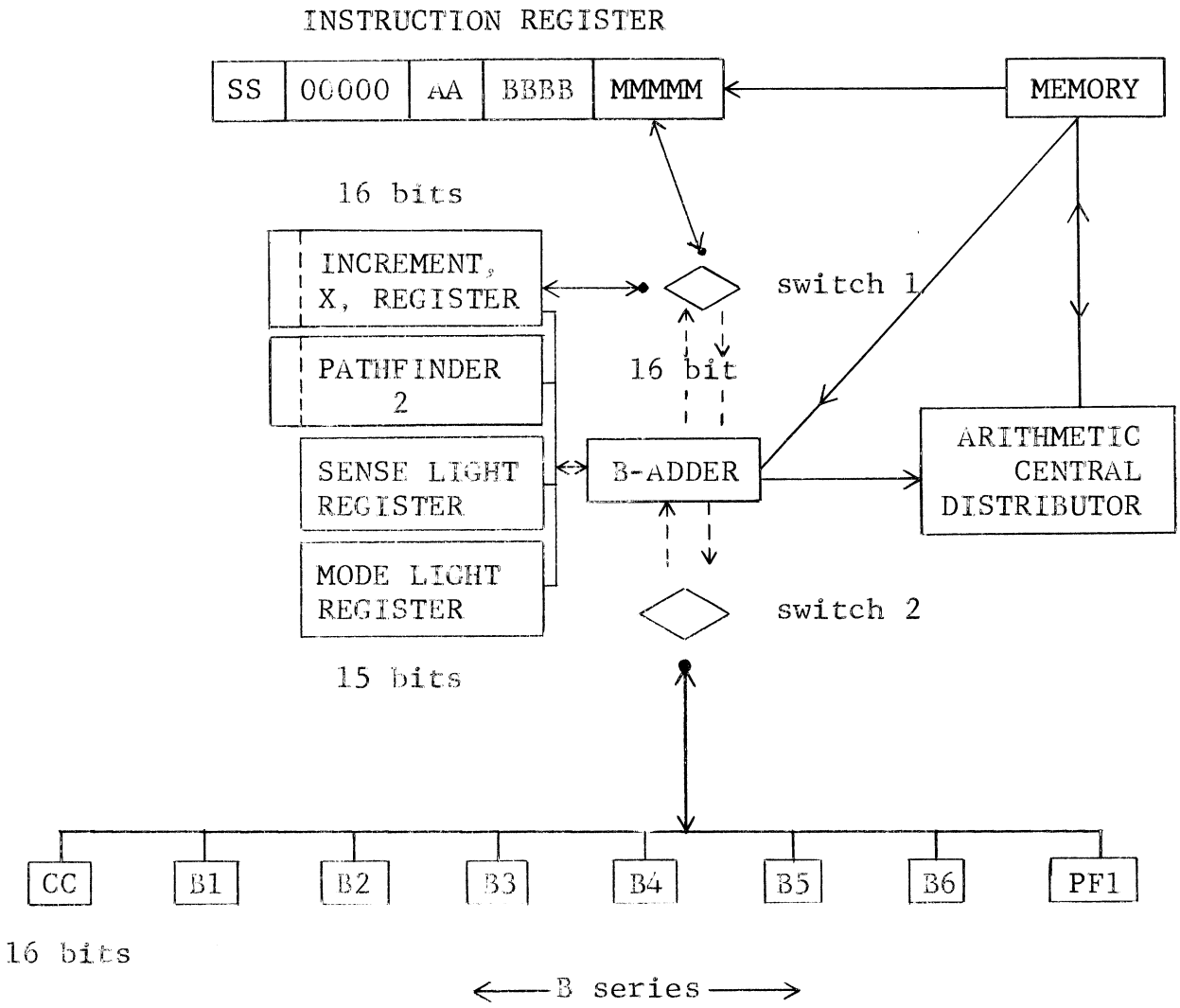



Figure 3: Block diagram of the control registers and the relationships among them. The solid arrows represent permissible transfers of information. The dashed arrows represent static connections to the adder. Switch 1 is a 2-fold position electronic switch and switch 2 is an 8-fold position electronic switch. In this manner one of two registers may be connected to the upper side of the adder while one of 8 registers may be connected to the lower side. The B-adder also acts as a central distributor for the control section.

# XI.

## CONTROL UNIT

The machine's control unit has the task of accepting orders one by one into the I register and of causing the machine to carry out the operations specified according to the order in the I register. All address modification also takes place in this unit. Normally orders are obeyed by the control unit in the sequence in which they are stored in the memory. Sometimes, however, this sequence is broken and the control unit starts over at some new position in the memory. This is called a transfer of control. If control is transferred a few locations back in the memory, the machine will repeat the operations specified by the intervening orders. It is possible to cause this repetition to occur any number of times. The machine also has special facilities for the repetition of a single instruction (see section on indicators).

The occurrence of cycles and conditional transfers of control are some of the things that complicate the programming of a calculation and at the same time make the computer of practical use. If each order in the memory were to be carried out only once, the Rice Computer would get through them all in about 3 to 4 seconds (even if the memory contained nothing but orders). In actual practice, calculations vary in duration from a few minutes to many hours.

Because of the importance and the difficulty of programming this type of cyclic control, emphasis has been placed on special facilities in the control section to help the coder. This tends to make the description of the control section of this machine fairly involved. However, the prospective coder is advised to master these additional features since they make possible most of the interesting calculations and will greatly shorten his time spent in coding.

The control unit is centered around a B-adder. This plays the part of a central distributor for the control section. The last 15 bits of the instruction register or the X register can be connected to one side of this adder and one of 8 other registers described below can be connected to the other side of the adder. The adder output may then be gated (i.e., transferred) to any of these control registers, the arithmetic central distributor, or the memory. Figure 3 is a block diagram of the control registers and their interrelations. The various registers are listed below, and a brief description of each one is given:

registers

| abbreviation | name |
|---|---|
| I | instruction register |
| CC | control counter (or location register) |
| $B_1$ | B register 1 |
| $B_2$ | B register 2 |

| abbreviation | name |
|---|---|
| $B_3$ | B register 3 |
| $B_4$ | B register 4 |
| $B_5$ | B register 5 |
| $B_6$ | B register 6 |
| PF1 | pathfinder 1 |
| PF2 | pathfinder 2 |
| SL | sense light register |
| ML | mode light register |
| X | increment register |

Instruction Register

When an instruction is brought from memory into the control unit, it is placed in the instruction register where it is decoded. The instruction register is a full length 54 bit word. The last 15 binary digits (bits 40-54) specify an address or location number. It is this number that is subject to modification.

Control Counter or Location Register

This register with a capacity of 15 bits ~~(fifteen binary digits and one overflow bit)~~ determines the location in memory from which the next instruction is taken. Whenever a new instruction is brought into the instruction register from memory, the CC (control counter) is advanced by 1. However, during the execution of a transfer or skip, the contents of CC may be changed to any number in the address range. ~~When acting in the~~ The control counter may also act as a B-register.

~~capacity just described, the remaining bit plays no role. However, the control counter may also act as a B register (described below) and then the overflow bit indicates whether or not it represents a negative integer (see section on complement arithmetic).~~

B Registers

The Rice Computer has six B registers each containing ~~16~~ 15 bits. One of the primary uses of B registers arises from their ability to modify the instruction address. When a given B register is appropriately specified (this is explained in the detailed discussion of the order code), the instruction is executed as if its address had contained the stated address plus the contents of the specified B register. The actual addition is carried out in the B-adder, and ~~[illegible]~~ The result of the addition is also placed in M, the address section of I.

~~The remaining bit (bit position 1) has the following interpretation:~~

~~"0" means the number in B register is a true integer~~

~~"1" means the number in B register is the two's complement of an integer~~

~~In the latter case, the addition of the two's complement is equivalent to a subtraction of the integer (see section entitled "Complement Arithmetic").~~

When a number is transferred from a full length register in the arithmetic unit to a B register, only the last 15 bits of the full length register are transferred.

When a number is read out of a B register into the arithmetic unit, the contents of the B register are placed in the last 15 bits of the specified arithmetic register. All other bits (including sign and overflow) are set to the value of the first bit of the B register.

Pathfinder Register 1

Whenever an unconditional class 0 transfer or skip is about to be executed, the contents of CC are placed into the pathfinder register. The PF1 register may also be used as a B register.

Pathfinder Register 2

On all modifications of (CC) except the normal advance by 1, the contents of CC (before the above modification) are placed into PF2. PF2 may not be used as a B register. Its address is $(77770)_8$.

~~PF2. PF2 may not be used as a B register. Its address is $(77770)_8$.~~

Sense Light Register

The sense light register is a 15 bit register with address $(77771)_8$. The uses of the sense light register are outlined in the section entitled "Indicators."

Mode Light Register

The mode light register is a 15 bit register with the address $(77773)_8$. Its uses are outlined in the section entitled "Indicators."

Increment Register

The X or increment register is a 16 bit register with address $(77772)_8$. The contents of this register may be added to the contents of any of the B series registers, with the sum being placed back into the B register.

B-Adder

As mentioned earlier, the B-adder controls the transfer of information and carries out the address modification specified. It is purely an adder and has not complementing facilities, and thus, the complement system of numbers must be used. The reader is referred to the discussion on complement arithmetic.

Pages 76 - 78

DELETED.

# XIII.

# PRINTER OUTPUT

When paper tape is read into the memory, the only requirement is that the information gets into memory in convenient form. The format on the paper tape is not of great importance. However, the printer output must not only satisfy the peculiarities of the machine but must also produce an easily readable report.

This requirement is partially met by having 64 different symbols available on the printer and a format control tape on the printer.

## Available Symbols

The printer will have 64 symbols which may be used in any way the coder desires. The 64 symbols may be put into a one to one correspondence with the 64 binary numbers which a hexad (six binary bits) is capable of representing. For convenience and uniformity of notation the correspondence given in the following table has been adopted. The order of symbols is also the order in which they occur on the printer type wheel. The binary code only refers to the position of the symbol on the print wheel, and it is not in general the binary code that is used on punched paper tape.

| symbol | octal code | binary code | comments |
|---|---|---|---|
| 0 | 00 | 00 0000 | ordinary |
| 1 | 01 | 00 0001 | decimal |
| 2 | 02 | 00 0010 | numbers |
| 3 | 03 | 00 0011 | |
| 4 | 04 | 00 0100 | |
| 5 | 05 | 00 0101 | |
| 6 | 06 | 00 0110 | |
| 7 | 07 | 00 0111 | |
| 8 | 10 | 00 1000 | |
| 9 | 11 | 00 1001 | |
| a | 12 | 00 1010 | |
| b | 13 | 00 1011 | |
| c | 14 | 00 1100 | |
| d | 15 | 00 1101 | |
| e | 16 | 00 1110 | |
| f | 17 | 00 1111 | |
| + | 20 | 01 0000 | |
| - | 21 | 01 0001 | |
| / | 22 | 01 0010 | |
| . | 23 | 01 0011 | plotting symbols: |
| . | 24 | 01 0100 | each symbol is |
| x | 25 | 01 0101 | duplicated so that |
| x | 26 | 01 0110 | graphs may be |

| symbol | octal code | binary code | comments |
|---|---|---|---|
| Δ | 27 | 01 0111 | plotted to 1/2 |
| Δ | 30 | 01 1000 | a character width |
| * | 31 | 01 1001 | |
| \| | 32 | 01 1010 | absolute value sign |
| ( | 33 | 01 1011 | |
| ) | 34 | 01 1100 | |
| × | 35 | 01 1101 | multiplication symbol |
| = | 36 | 01 1110 | |
| , | 37 | 01 1111 | |
| A | 40 | 10 0000 | full alphabet in |
| B | 41 | 10 0001 | upper case type |
| C | 42 | 10 0010 | |
| D | 43 | 10 0011 | |
| E | 44 | 10 0100 | |
| F | 45 | 10 0101 | |
| G | 46 | 10 0110 | |
| H | 47 | 10 0111 | |
| I | 50 | 10 1000 | |
| J | 51 | 10 1001 | |
| K | 52 | 10 1010 | |
| L | 53 | 10 1011 | |
| M | 54 | 10 1100 | |
| N | 55 | 10 1101 | |
| O | 56 | 10 1110 | |

| symbol | octal code | binary code | comments |
|---|---|---|---|
| P | 57 | 10 1111 | |
| Q | 60 | 11 0000 | |
| R | 61 | 11 0001 | |
| S | 62 | 11 0010 | |
| T | 63 | 11 0011 | |
| U | 64 | 11 0100 | |
| V | 65 | 11 0101 | |
| W | 66 | 11 0110 | |
| X | 67 | 11 0111 | |
| Y | 70 | 11 1000 | |
| Z | 71 | 11 1001 | |
| < | 72 | 11 1010 | less than sign |
| ≤ | 73 | 11 1011 | less than or equal |
| ↑ | 74 | 11 1100 | exponent symbol |
| ← | 75 | 11 1101 | |
| → | 76 | 11 1110 | |
| ↓ | 77 | 11 1111 | subscript symbol |

~~The printer can print up to 108 characters on each line but only 54 at a time. The left hand 54 positions are handled in parallel by a single "PRINT LEFT" command, while the right hand 54 positions are controlled by the "PRINT RIGHT" instruction. Thus to print one full line we must use the sequence:~~

~~(1) "print left-do not space" followed by "print right"~~

The printer can print up to 108 characters on each line. Each print order produces one line of printer output in 0.1 seconds. Thus one can print up to 600 lines per minute.

In discussing the print format it is helpful to consider a layout chart as illustrated in figure 4. This chart has a square for each position on the paper where a symbol can be printed. Charts of this sort are useful in planning the programming of the output. This layout may be done for each line individually or once for the entire output if a standard format is used for all the lines.

To print <u>one</u> line the coder must reserve 128 consecutive words in memory, this block being called a "print-matrix." Each bit in this block of words is an element $P_{c,t}$ in this matrix and may have the value 0 or 1.

$$P_{c,t} = 0,1 \qquad \begin{matrix} 0 \leq c \leq 63 \\ 1 \leq t \leq 108 \end{matrix}$$

The index c denotes the character to be printed and the index t denotes the column. If $P_{c,t} = 1$, the character c will be printed in column t in the line under consideration.

The location of these print-matrix elements is given as follows

$P_{c,t}$ = bit t in word c if $t \leq 54$

$P_{c,t}$ = bit t-54 in word c+64 if $t \geq 55$

Thus the first 64 words control the format of the left hand side of the page and the second 64 words control the format of the right hand side of the page.

The print matrix which will produce the left half of line 1 shown on the layout chart of Figure 4 is given schematically in Figure 5. Note that a new print-matrix must be constructed for every line of print.

If all 64 characters are used the maximum speed of 600 lines per minute can not be attained because the actual printing will consume an entire drum revolution or 0.1 seconds and another drum revolution will be required for the paper advance. However, if only the first n characters are used one can tag the last significant word in the print matrix with a tag 3. The print order will proceed to print characters until a tag 3 or character number 63 is reached; the paper will then be advanced as specified in the order. If $n \leq 54$ (i.e., if only the first 54 characters are used) the paper advance will not consume an extra 0.1 second but will be accomplished within the 0.1 second period required to print one line.

The words of the print matrix are placed in two buffer registers when they are needed. During the time interval between the printing of successive characters the computer will proceed with the calculation unless another print order or magnetic tape order is encountered (both types of orders require the buffer register). In fact, about 75 per cent of the time during the printing process will be available for calculation.

In a print instruction the coder specifies the address of the first word of the print-matrix and whether or not the paper should be advanced after printing. Eight choices for the paper advance

will be available. Some of the choices might be, for example, space ½ line, 1 line, 2 lines, 1/3 page, 1/2 page, 1 page.

Only one symbol should be specified for a given column in a single print-matrix. For example, in the sample given in Figure 5 a one in bit 32 of word 5 and a one in bit 32 of word 35 should not be given. In general, only the first symbol specified will be printed, but it is possible that the second symbol will be printed over the first; the actual output in such an instance is difficult to predict.

In actual practice the formation of the print-matrix will be accomplished by a set of subroutines. Subroutines will be available for the common types of tables in fixed and floating point numbers. The coder will be able to select a subroutine for his purposes (e.g., a routine that will print five columns of ten digit numbers in standard floating point form) including a limited number of alphabetical routines that will print specified column headings, etc.

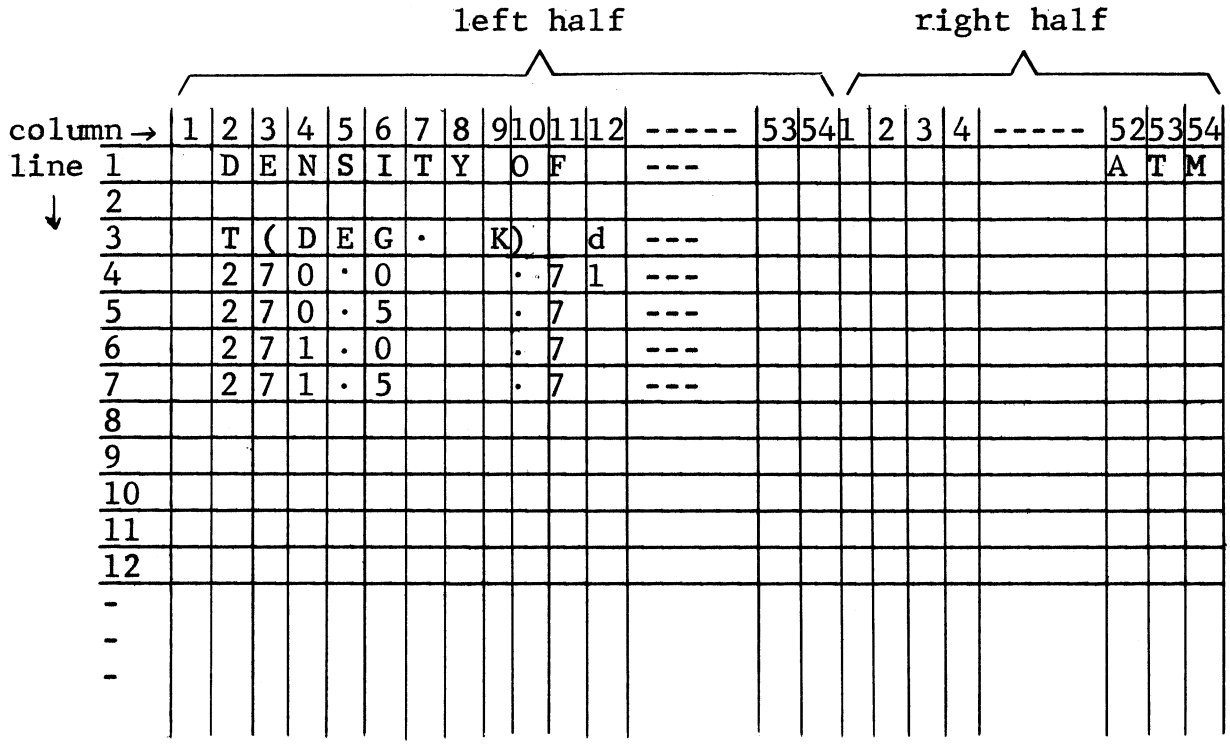



Figure 4: Example of a print layout chart.

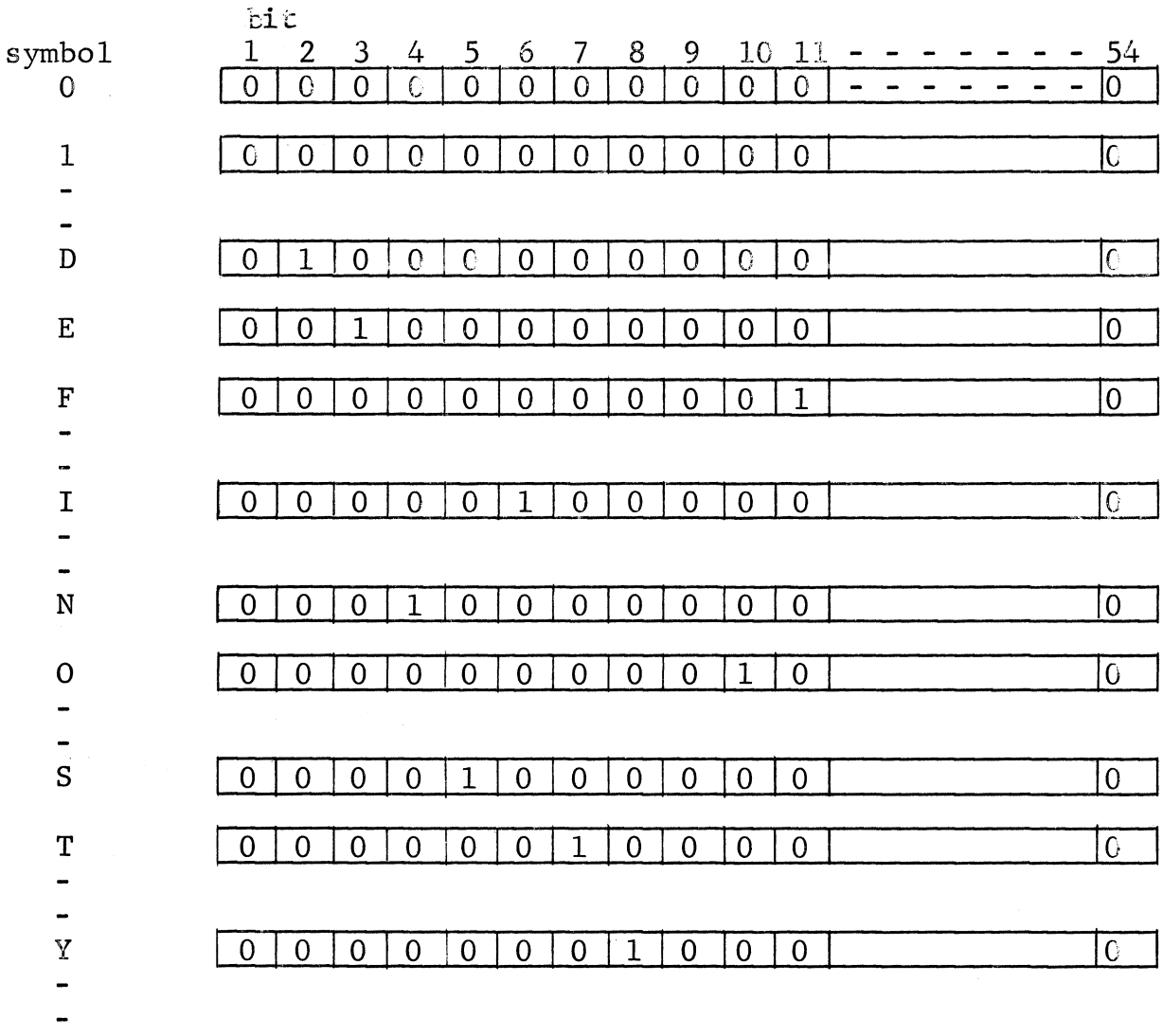
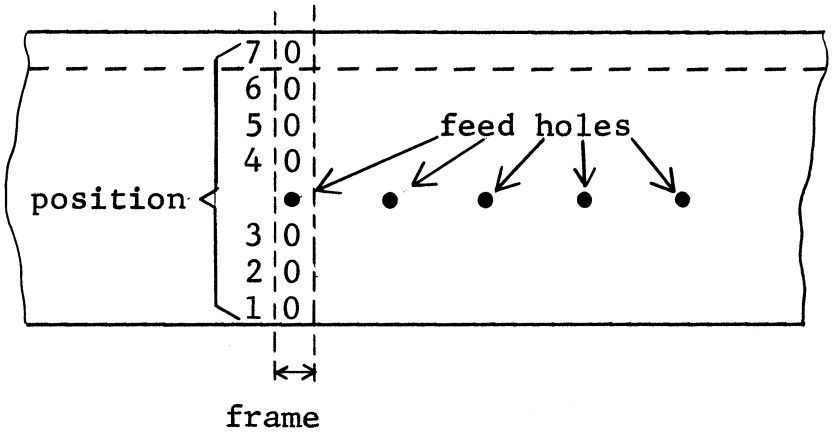



Figure 5: Schematic drawing of print-matrix for left half of line 1 in Figure 4.

XIV.

PUNCHED PAPER TAPE

The initial input to the computer will be 7-hole punched paper tape which will be read by a photoelectric reader. The diagram below illustrates the terms used in discussing punched paper tape:




The positions are located across the width of the tape and are places where data holes can be punched. A frame is a column whose location along the length of the tape is defined by the feed hole.

The computer reads positions 1-6 in a frame sensing a hole as a 1 and a blank as a zero. If position 7 is blank, the information in 1-6 enters the U register. If position 7 is punched, the information in 1-6 is <u>not</u> entered into the computer proper. Thus, a hole in position 7 is effectively a delete mark. However, there are a few combinations of punches (all of

which have the 7th hole) which act as control punches. They are not read into the U register but may affect the control of the reading process. A table of control punches may be found at the end of this section.

The tape is prepared by a "Flexowriter" combination typewriter and punch, whose keyboard layout is illustrated in Figure 6. Every key on the central keyboard will type a character on the paper in the typewriter and at the same time if the "punch on" control key is down, a six bit code will be punched into positions 1-6 on the paper tape frame under the typewriter punch. A table giving the binary code for each key can be found below. Many of the control keys (upper case, lower case, carriage return, tab, etc.) will not only carry out their designated function, but will also punch a six bit code into the paper tape. Depressing the ~~"7th hole also"~~ "punch 7th hole" switch ~~key~~ by itself will not have any visible effect, but if this key is down and another key is struck, the corresponding six bit code plus a hole in position 7 will be punched. ~~Several~~ The special control ~~keys~~ switches (tape feed, code delete, ~~[illegible]~~ and ~~space~~) automatically punch the 7th position since this information need never be read by the computer proper. Note that the upper case and lower case symbols on a given key have the same binary code. The computer can only distinguish the two symbols by noting whether or not the symbol is preceded by an "upper case" or "lower case" punch.

A punched paper tape may be read by the Flexowriter which will type the characters specified and if desired punch a duplicate tape. All punches and operations may be duplicated in this manner except "code delete" and "stop code." "Code delete" punches will be ignored and "stop code" punches will stop the automatic action of the typewriter. A 7th position hole will be duplicated but will not affect the typing of the machine. The back space key will simply back space the carriage and paper tape one position but will not punch the tape.

| key | | code | | key | | code | |
|---|---|---|---|---|---|---|---|
| L-case | U-case | 7654 | 321 | L-case | U-case | 7654 | 321 |
| a | A | 0100 | 000 | w | W | 0110 | 110 |
| b | B | 0100 | 001 | x | X | 0110 | 111 |
| c | C | 0100 | 010 | y | Y | 0111 | 000 |
| d | D | 0100 | 011 | z | Z | 0111 | 001 |
| e | E | 0100 | 100 | 0 | ( | 0000 | 000 |
| f | F | 0100 | 101 | 1 | ) | 0000 | 001 |
| g | G | 0100 | 110 | 2 | * | 0000 | 010 |
| h | H | 0100 | 111 | 3 | # | 0000 | 011 |
| i | I | 0101 | 000 | 4 | $\lambda$ | 0000 | 100 |
| j | J | 0101 | 001 | 5 | $\sigma$ | 0000 | 101 |
| k | K | 0101 | 010 | 6 | $\alpha$ | 0000 | 110 |
| l | L | 0101 | 011 | 7 | $\beta$ | 0000 | 111 |

| key | | code | | key | | code | |
|---|---|---|---|---|---|---|---|
| L-case | U-case | 7654 | 321 | L-case | U-case | 7654 | 321 |
| m | M | 0101 | 100 | 8 | ↑ | 0001 | ~~001~~ 000 |
| n | N | 0101 | 101 | 9 | △ | 0001 | ~~010~~ 001 |
| o | O | 0101 | 110 | π | Π | 0001 | ~~011~~ 010 |
| p | P | 0101 | 111 | × | Σ | 0011 | 101 |
| q | Q | 0110 | 000 | + | / | 0010 | 000 |
| r | R | 0110 | 001 | - | → | 0010 | 001 |
| s | S | 0110 | 010 | < | ≤ | 0111 | 010 |
| t | T | 0110 | 011 | . | , | 0011 | 111 |
| u | U | 0110 | 100 | = | \| | 0011 | 110 |
| v | V | 0110 | 101 | carriage return | | ~~0011~~ ~~001~~ 0010 | 100 |
| upper case | | ~~0001~~ ~~011~~ 0111 | 100 | 1/2 space down | | ~~0001~~ ~~110~~ 0010 | 010 |
| lower case | | ~~0001~~ ~~100~~ 0111 | 110 | 1/2 space up | | ~~0001~~ ~~101~~ 0010 | 011 |
| tab | | 0010 | 010 | space (bar) | | ~~1111~~ ~~110~~ 0001 | 100 |
| tape feed (key) | | ~~1011~~ ~~000~~ 0011 | 000 | code stop (switch) | | ~~1111~~ ~~101~~ 0010 | 111 |
| back space | | 0010 | 110 | code delete | | 1111 | 111 |
| tape feed (switch) | | 1011 | 000 | | | | |

control punches

"stop code" - turns off repeat mode light

7th hole - frame is <u>not</u> read into U

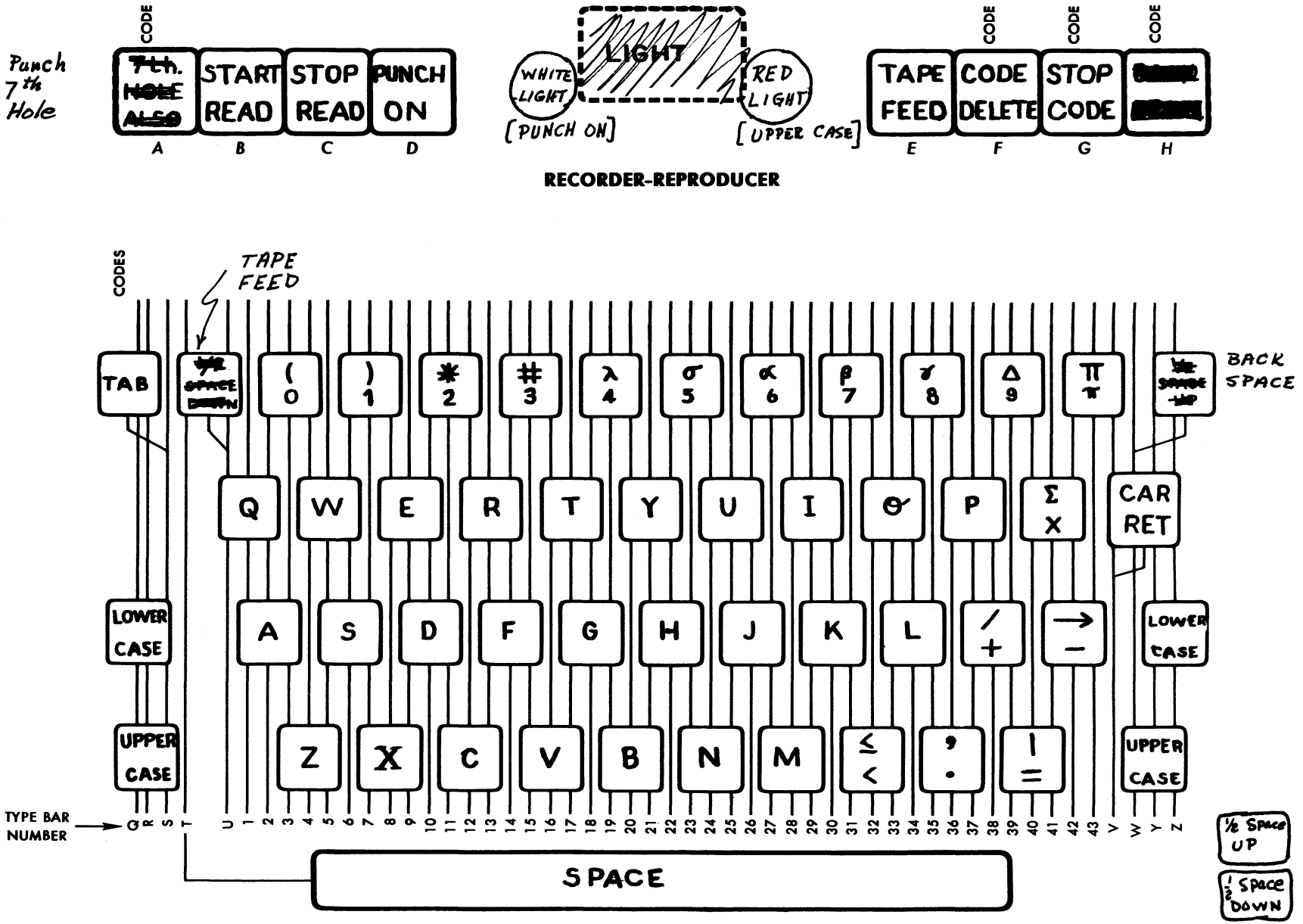



Figure 6: Keyboard Layout - Flexowriter Model - FL

# FIXED POINT ARITHMETIC

## XV.

For a complete understanding of coding, it is necessary to have a fairly thorough grasp of how the computer handles arithmetic. For the purpose of illustration we will often consider here a 5 bit mantissa. The reader may easily extend this to a larger number of bits.

(a) Addition and Subtraction.

The fixed point addition is carried out by simply adding two 49 bit numbers (bits $m_s$, $m_0$,...,$m_{47}$) with one modification: the carry (either 0 or 1) from bit $m_s$ is considered to be a carry into bit $m_{47}$. In other words, addition is performed with an "end-around" carry. In this operation (Exp S)—> Exp U . Thus at the end of the operation both U and S have the exponent part of the number from memory. Aside from this transfer, the exponent plays no role in the fixed point addition.

To illustrate this we consider a five bit mantissa and the following examples:

| | | $m_s$ | $m_0$ | . | $m_1$ | $m_2$ | $m_3$ | |
|---|---|---|---|---|---|---|---|---|
| 1 | U | 0 | 0 | . | 1 | 0 | 0 | 1/2 |
| | S | 0 | 0 | . | 0 | 1 | 0 | + 1/4 |
| | | 0 | 0 | . | 1 | 1 | 0 | + 3/4 |
| 2 | U | 0 | 0 | . | 1 | 1 | 0 | 3/4 |
| | S | 1 | 1 | . | 0 | 1 | 1 | (-1/2) |
| | 1 | 0 | 0 | . | 0 | 0 | 1 | |
| | | end around carry | | | | | 1 | |
| | | 0 | 0 | . | 0 | 1 | 0 | +1/4 |

| 3 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | U | 0 | 0 | . | 0 | 1 | 0 | 1/4 |
| | S | 1 | 1 | . | 1 | 0 | 0 | + (-3/8) |
| | | 1 | 1 | . | 1 | 1 | 0 | -1/8 |

B-register arithmetic takes place in the B-adder and is an addition of two 15-bit integers exactly analogous to that discussed above for 49-bit numbers.

(b) Overflow and Underflow:

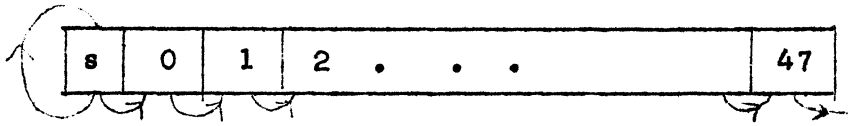
The following conditions describe when the arithmetic operations result in an overflow of the storage capacity.

| $m_s$ | $m_0$ | condition |
|---|---|---|
| 0 | 1 | overflow |
| 1 | 0 | underflow |

See section on overflow for more details.

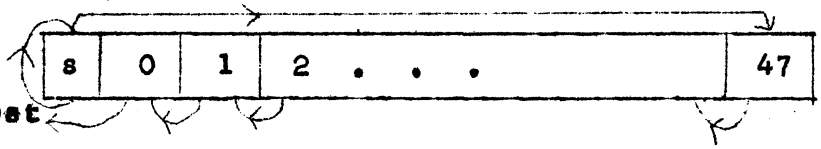
(c) Arithmetic Shifting.

U and R are shifting registers and in the arithmetic shifts behave in the manner described by the following diagrams.



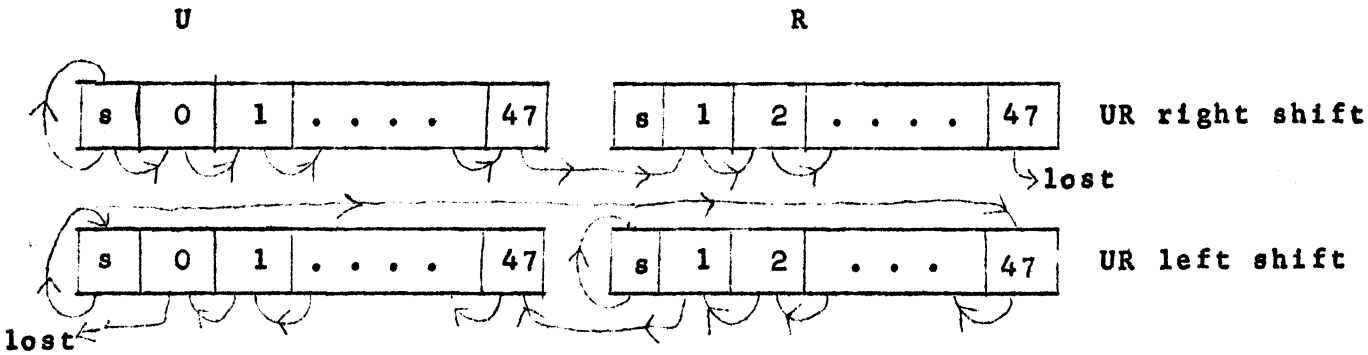

U right shift

On a right shift the sign of U is propagated and preserved. On shifting R alone the same connections (with the overflow bit omitted) are made.




U left shift

On a left shift the sign of U is carried into the lowest order bit. On shifting R alone the same connections (with the overflow bit omitted) are made.




For example,

| U | | | | | | U | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $m_s$ | $m_0$ | $m_1$ | $m_2$ | $m_3$ | shifted Right 2 | $m_s$ | $m_0$ | $m_1$ | $m_2$ | $m_3$ |
| 1 | 1 | 0 | 0 | 1 | | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | shifted left 2 | 1 | 0 | 1 | 1 | 1 |

(d) Fixed point multiplication.

Multiplication is carried out by a process of repeated addition using a positive multiplier. When a multiply order is initiated the two factors are in S and U respectively, and the following sequence of steps is executed.

(1) Test (U) for +0 or -0

(a) (U) = 0, clear mantissae of U and R, proceed to next instruction.

(b) (U) ≠ 0, go to step (2).

(2) Test sign of (S)

(a) (S) +, go to step (3).

(b) (S) -, complement (U) and (S), then go to step (3).

(3) (S) -> R, (Exp S)-> Exp R

(U) -> S, Exp U remains in U

Note that R always contains a + number.

-0 (i.e., all ones) -> U if sign U is - .

+0 (i.e., all zeros) -> U if sign U is + .

(4) Multiply by repeated addition

(a) If $m_{47}$ in R = 0, shift (UR) right one place using an arithmetic shift.

(b) If $m_{47}$ in R = 1, (U) + (S) -> U, then shift (UR) right one place.

(c) Repeat process 46 times, testing for groups of 8 zeros in R. If a group of 8 zeros is encountered, (UR) is shifted right 8 places.

(5) $m_s$ of U -> $m_s$ of R.

Each of the above additions in step (4) is carried out using the "end-around" carry in U. As an example we consider a couple of three bit numbers: + 1/2 in U and - 3/4 in S. At the end of step (3) we proceed as follows:

| S | U | R | Procedure |
|---|---|---|---|
| $m_s$ $m_1$ $m_2$ $m_3$ | $m_s$ $m_1$ $m_2$ $m_3$ | $m_1$ $m_2$ $m_3$ | |
| 1 0 1 1 | 1 1 1 1 | 1 1 0 | (after step 3) |
| | 1 1 1 1 | 1 1 1 | Shift UR |
| | 1 0 1 1 | 1 1 1 | Add U + S |
| | 1 1 0 1 | 1 1 1 | Shift UR |
| | 1 0 0 1 | 1 1 1 | Add U + S |
| | 1 1 0 0 | 1 1 1 | Shift UR |

Answer is -3/8 in the one's complement system and is a double length result in UR.

The process of multiplication can be explained in analytical terms. Let the algebraic value of the two factors be x and y (x in R and y which may be negative in S). The machine representation of negative y may be considered as

$$(2^{50} - 2^{-47} + y)$$

with the leading ones being virtual in the sense that they appear only on right shifts of U.

We then form

(a) $(2^{50} - 2^{-47} + y)x$ from repeated addition scheme.

(b) $2^{-47}x$ from the "end-around carry" into $m_{47}$ of U instead of $m_{47}$ of R.

(c) $+(2^{50} - 2^{-47})2^{-47}$ from the initial setting of U to all ones.

The sum of all these contributions is

$$2^{50}\, x + 2^{3} + xy - 2^{-94}$$

And since any bits to the left of $m_s$ in U are suppressed this is equivalent to

$$4 - 2^{-94} + xy \quad ,$$

which is the double length representation of xy in the 1's complement system.

(e) Rounding after Multiplication.

In the normal mode of operation, fixed point multiplication will be performed without rounding. However, when operating in the "rounding mode" the computer will carry out the following sequence of steps after the normal multiplication is complete.

Step (1): Compare $m_s$ and $m_1$ in R.
- (a) $m_s = m_1$, proceed to next operation.
- (b) $m_s \neq m_1$, proceed to step (2).

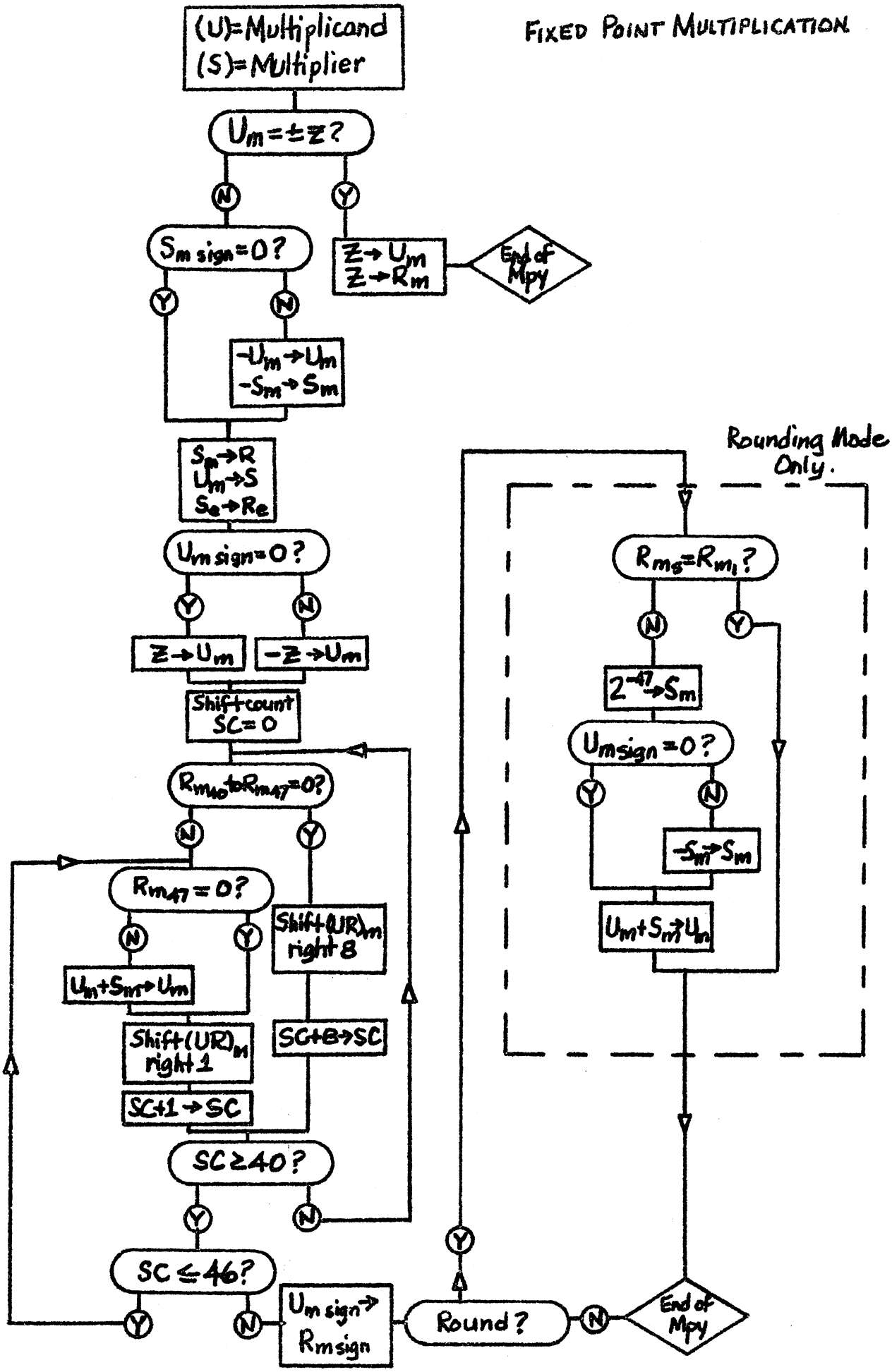
Step (2): Clear S and set $m_{47}$ in S to a 1.

Step (3): Examine $m_s$ in U.
- (a) $m_s = 0$, proceed to step (4)
- (b) $m_s = 1$, complement S and then proceed to (4).

Step (4): (U) + (S) $\rightarrow$ U, proceed to next operation.

The effect of this sequence of steps is to add 1 to the magnitude of (U) only if the magnitude of (R) is greater than or equal to

# FIXED POINT MULTIPLICATION



(f) Fixed Point Division.

Division is carried out by a process of repeated subtractions using a divisor and dividend of the same sign. If the sign of (S) which is the divisor differs from the sign of (U) which is the dividend, (S) is complemented before division and the quotient is complemented after division. The mantissa of (U) must be less than the mantissa of (S) and the exponents are ignored. With the divisor in S and the dividend in UR, the following sequence of steps is executed.

(1) Test Op 3 of divide instruction.

(a) Op 3 = 0, 3, 4, 5, 6, or 7, proceed to step (2).

(b) Op 3 = 1, clear $R_m$ to sign of ($U_m$) and proceed to step (2).

(c) Op 3 = 2, clear $U_m$ to sign of ($R_m$) and proceed to step (2).

(2) Compare sign of (U) and sign of (S).

(a) $U_s = S_s$, then will not complement quotient after division in step (8).

(b) $U_s \neq S_s$, - (S) → S and will complement quotient after division in step (8).

(3) Subtracter output $\neq 0$ and sign of subtracter output $\neq U_{ms}$?

(a) Yes, shift mantissa of (UR) left 1, set shift count SC = 0; proceed to step (4).

(b) No, test ML #1.

(i) On, go to end of divide.

(ii) Off, stop.

(4) Subtracter output $\neq 0$ and sign of subtracter output $\neq U_{ms}$?

(a) Yes, $0 \rightarrow R_{m47}$; proceed to step (5).

(b) No, subtracter output $\rightarrow$ U, $1 \rightarrow R_{m47}$; proceed to step (5).

(5) Advance SC by 1.

(6) SC = 47?

(a) Yes, proceed to step (7).

(b) No; (UR) arithmetic shift left 1; return to step (4).

(7) (R) $\rightarrow$ S, (U) $\rightarrow$ R.

(8) (S) $\rightarrow$ U or -(S) $\rightarrow$ U as indicated in step (1).

(9) End of divide.

After division the remainder is in R and the quotient is in U. The exponent of the word that originally came to S from memory ends up in R, and the exponent of the word that was originally in U remains in U.

Division is actually accomplished with a double length dividend, (U) and its extension in (R). If the sign of (R) is not the same as the sign of (U), the results of the division process are not the correct quotient and remainder. Any necessary adjustment of the sign of (R) is left up to the coder in the use of a double length dividend. In order to use a single length dividend and obtain the correct results, R must be cleared to the sign of U, and this may be done by proper use of OP3 of the divide code. Division of two integers is accomplished by placing the dividend in R and clearing U to the sign

of R, which may also be accomplished by use of OP3 in the divide code.

As an example of division we consider two three bit numbers: +1/4, a single length fraction in U, and +1/2 in S. After step (1) we proceed as follows:

| S | | | | U | | | | R | | | | Procedure |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $m_s$ | $m_o.m_1$ | $m_2$ | $m_3$ | $m_s$ | $m_o.m_1$ | $m_2$ | $m_3$ | $m_s$ | $m_o.m_1$ | $m_2$ | $m_3$ | |
| 0 | 0 .1 | 0 | 0 | 0 | 0 .0 | 1 | 0 | 0 | 0 .0 | 0 | 0 | after step (1) |
| | | | | 0 | 0 .1 | 0 | 0 | 0 | 0 .0 | 0 | 0 | $\|S_m\| > \|U_m\|$ so shift (UR) left 1 and $0 \rightarrow R_{m47}$ by (2) |
| | | | | 0 | 0 .0 | 0 | 0 | 0 | 0 .0 | 1 | 0 | $\|S_m\| \leq \|U_m\|$ so subtracter to U, $1 \rightarrow R_{m47}$, shift by steps (4)-(6) |
| | | | | 0 | 0 .0 | 0 | 0 | 0 | 0 .1 | 0 | 0 | $\|S_m\| > \|U_m\|$ so shift (UR) left 1 and $0 \rightarrow R_{m47}$ by steps (4)-(6) |
| | | | | 0 | 0 .1 | 0 | 0 | 0 | 0 .0 | 0 | 0 | interchange (U) and (S) by (7) and (8) |

The quotient +1/2 is in U and the remainder zero is in R.

If we consider the signel length numerator N and denominator D with the same sign and (U) = N, (S) = D, (R) = Z initially, then in fixed point division the bits of the quotient are obtained by application of the algorithm

$p_o$ = N. Then for j = 0, 1, ..., 46

$$\left.\begin{aligned} &q_{j+1} = 1 \text{ and } p_{j+1} = 2p_j - D \text{ if sign } (2p_j - D) = \text{sign } (2p_j) \\ &q_{j+1} = 0 \text{ and } p_{j+1} = 2p_j \quad \text{if sign } (2p_j - D) \neq \text{sign } (2p_j) \\ &R = p_{47} \\ &Q = q_1 2^{-1} + q_2 2^{-2} + \ldots + q_{47} 2^{-47} \end{aligned}\right\} \text{ I}$$

If $|N| > |D|$ division overflow occurs.

If sign (N) $\neq$ sign (D), complement D before division and complement Q afterwards.

Example: Assume N and D negative 4-bit fractions.

$-D = -(-7) = (S) = 00.\ 0111$

$p_o = N = -5 = (U) = 11.\ 1010$

$q_1 = 1$ and $p_1 = 11.\ 1100$

$q_2 = 0$ and $p_2 = 11.\ 1001$

$q_3 = 1$ and $p_3 = 11.\ 1010$

$q_4 = 1$ and $p_4 = 11.1100$

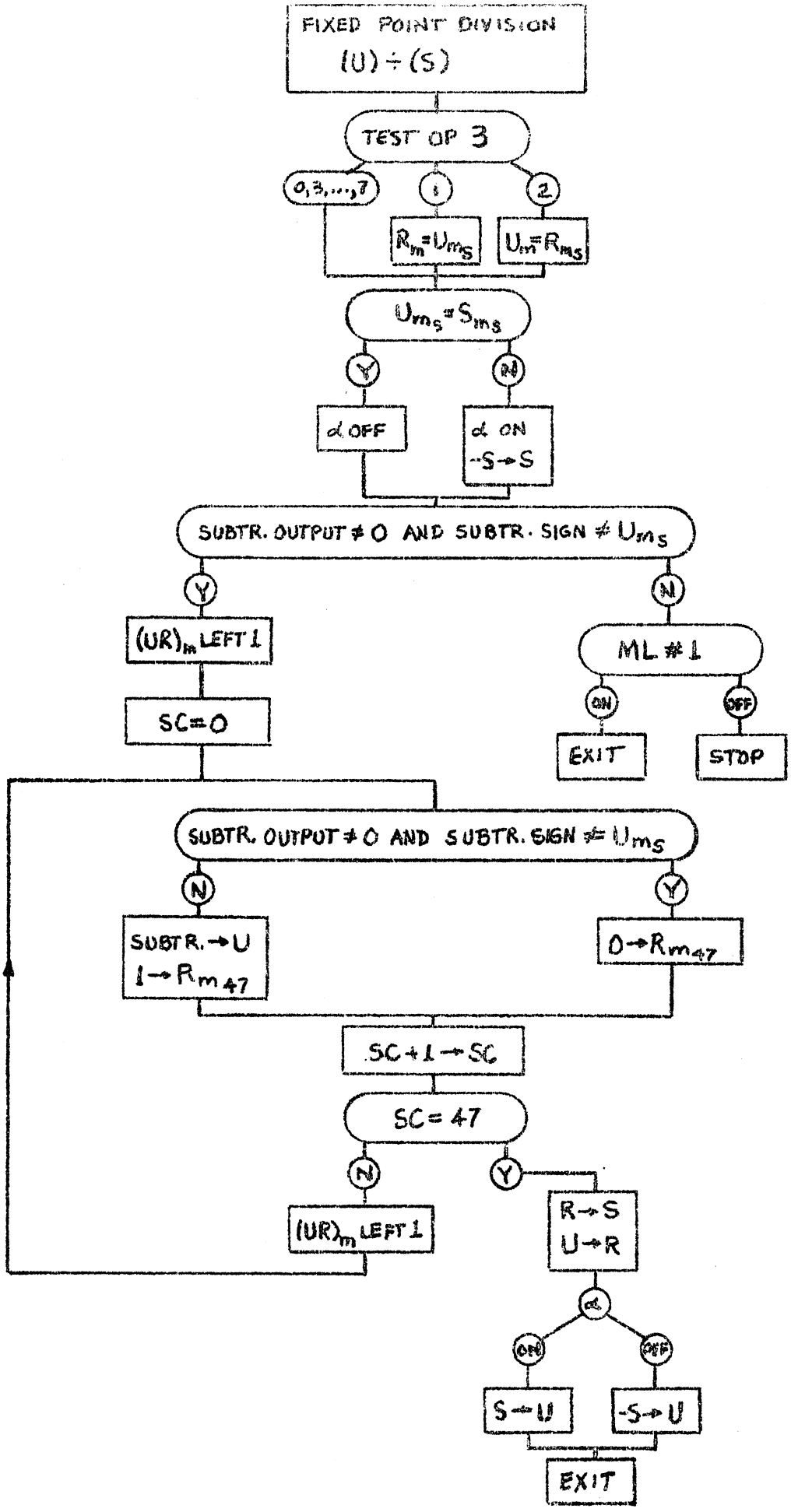
so answer is $q = .1011$ and $R = 11.1100 \times (2^{-4})$ .

This can be compared with

$$\frac{-5 \cdot 2^{-4}}{-7 \cdot 2^{-4}} = \left(\frac{-80}{-7}\right) 2^{-4} = \left(11 - \frac{3}{7}\right) 2^{-4} = .1011 \text{ and } -\left(\frac{.0011}{7}\right)$$

It can be shown that the following equation connects the partial quotient (in R) and the remainder (in U) after j applications of I with N and D:

$$N = 2^{47-j} q_j D + 2^{-j} p_j .$$


FIXED POINT DIVISION
(U) ÷ (S)
TEST OP 3
0,3,...,7
1
2
$R_m = U_{ms}$
$U_m = R_{ms}$
$U_{ms} = S_{ms}$
Y
N
α OFF
α ON
-S → S
SUBTR. OUTPUT ≠ 0 AND SUBTR. SIGN ≠ $U_{ms}$
Y
N
$(UR)_m$ LEFT 1
SC = 0
ML # 1
ON
OFF
EXIT
STOP
SUBTR. OUTPUT ≠ 0 AND SUBTR. SIGN ≠ $U_{ms}$
N
Y
SUBTR. → U
1 → $R_{m47}$
0 → $R_{m47}$
SC + 1 → SC
SC = 47
N
Y
$(UR)_m$ LEFT 1
R → S
U → R
α
ON
OFF
S → U
-S → U
EXIT

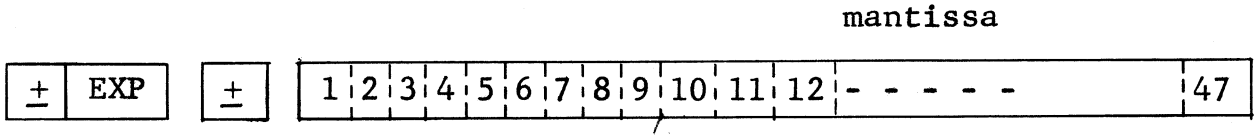
# XVI.

## BINARY POINT LOCATION AND FLOATING POINT ARITHMETIC

In programming fixed point calculations one of the major problems is decimal or binary point location. This problem is particularly difficult if, during a calculation, the number of required significant bits approaches the bit length of the mantissa. Various devices are used to assist in planning, but it should be emphasized that these methods are primarily methods of how to think about the numbers in memory.

### Binary Point Location

The simplest method is to fix the binary point at some arbitrary place in the middle of the word or at either end. For example, suppose we locate the binary point between the ninth and tenth bit in the mantissa of our word (that is, between bits number 15 and 16 in the word).




This would be the form in which all data and constants are entered into the problem and in which all answers and intermediate results appear. Note that no data or results may be larger than $2^{10} - 1 = \overset{1023}{\cancel{1024}}$ or smaller than $2^{-38} = 0.000\ 000\ 000\ 002$.

Since in this system all numbers in memory have the same decimal point, addition and subtraction can be carried out

without any preliminaries. If overflow were to occur, the sum would be larger than $2^{10} - 1$ and in this system nothing could be done. It is up to the coder to look at his problem and choose a location for his binary point such that all data and results of calculations will be less than the largest number possible.

In multiplication we have two numbers each of which has 38 places to the right of the point. By the usual rules the product (in U and R combined) will have 38 + 38 = 76 places to the right of the point. These 76 places include the 47 places in R and 29 places in U. Thus in the result the binary point can be placed (in thought) between the 18th and 19th bits in the mantissa (i.e., bits number 24 and 25 in the entire word). It is necessary to shift the combined mantissae of U and R left nine places after each multiplication to re-establish the proper binary point. Again any overflow is an error in this system.

In division there will be as many places to the right of the point in the quotient as the difference of the number of places in the fractional part of the dividend and the number in the divisor. Note that the dividend includes U and R. Thus to get 38 to the right of the point in the quotient (with 38 to the right in the divisor) we must arrange to have 76 in the dividend. The dividend is thus shifted right nine places before each division. This is almost the reverse of multiplication.

It is important to note that the actual electronic processes of add, multiply, etc., are independent of the choice

of the binary point. The machine does not have a binary point built in and it is entirely up to the coder to keep track of the point.

If we think of the binary point as being to the left of position 1 in the mantissa, the numbers all have 47 places to the right of the point. By examination of the above discussion one can see that in this case no shifting is required in multiplication and division. For this reason many people regard this location of the point as the natural one and emphasize that the computer only uses fractions as input data and only yields results in the form of fractions.

A slightly more general procedure would be to keep track of the binary point with the following rules. (One must always check for overflow and improper division.)

(1) In addition or subtraction, the binary point location of both numbers must be the same.

(2) In multiplication, the product has as many places to the right of the point as the sum of the number of places to the right of the point in the multiplier and multiplicand.

(3) In division, the number of places to the right of the point in the quotient is the difference of the number of places to the right of the point in the dividend and the number of places to the right in the divisor.

Note: The location of the point in the dividend and in the product is in respect to the U and R registers combinded, whereas all other quantities are regarded as single length numbers.

Floating Point Representation

At best the problem of keeping track of the binary point is time-consuming and annoying. At times it is extremely difficult to plan adequately, since it involves predicting the sizes of all numbers in the calculation. A system is needed that will indicate where the binary point is and instruct the machine to take account of this in doing operations on numbers. Such a system is a floating point system.

The basic idea is to write all numbers as a binary fraction times a scaling factor. Thus, if x is the actual number and $\bar{x}$ is the fraction in memory, we have the scale factor equation:

$$x = s\bar{x}$$

The object is to store the scaling factor in memory along with the fractional part $\bar{x}$.

In usual scientific applications, s is restricted to be a power of 10, but in a binary machine s is usually restricted to be a power of 2:

$$x = 2^q \bar{x}$$

and q is the quantity stored in memory together with $\bar{x}$. The

exponent q in this scheme is the same as the number of places to the left of the binary point. In such a procedure we think of all numbers in memory as having a binary point fixed at the left end of the word. If a number is shifted, the binary point does not move but rather the scale factor changes. This is obviously merely a change in viewpoint. The "lining up" of the binary point by shifting in the fixed point procedure is replaced by a "matching" of exponents, again by shifting, in the floating point procedure.

The disadvantages of such a system should also be mentioned:

(1) The number of bits used to hold the exponent in memory decreases the number of significant figures that can be carried in a number.

(2) The process of matching exponents (i.e., lining up the binary points) of two numbers before addition or subtraction is time-consuming.

(3) The problem of whether to restrict the operations to completely normalized numbers (i.e., numbers between 1/2 and 1) or not is introduced.

(4) The possibility of floating point overflow and underflow must be considered.

(5) Conventions to avoid time-consuming shifts must be adopted.

It is not necessary to use a power of 2 as the scaling factor. Suppose we consider:

$$s = (2^p)^q$$

where q is the variable exponent and p is a fixed integer. This form is chosen so that there will be a simple relation between shifting and the exponent q. Using such a scale factor is often referred to as using the number base $2^p$. In many ways this is convenient but it is important to remember that the machine is still a binary computer.

With this choice of s, we need a smaller range in q for a given range in s than in the previous system. Since only shifts of p bits at a time can be recorded, the fractional part of the number can vary over a wider range. In fact, if we maintain the least possible number of leading zeros in all the fractional parts (this is called normalized or standardized numbers) up to p-1 leading zeros can still occur.

$$2^{-p} \leq \bar{x} < 1$$

A large p means a greater possible range in s (which is of course the major object of a floating point system), faster arithmetic (since matching of exponents by shifting is less frequent and fewer shifts are needed in general), but a smaller possible number of significant figures (since p-1 leading zeros will almost certainly occur during the calculation).

As a compromise p = 8 has been chosen for the Rice Computer. As usual such a choice also introduces new complications

which will be discussed below. Thus we have:

$$x = (2^8)^q \bar{x} = (256)^q \bar{x}$$

$$-1 < \bar{x} < 1 ; \qquad 1 > |\bar{x}| \geq 1/256$$

$$-31 \leq q \leq 31$$

Normalization (or Standardization)

A floating point number is said to be in normal or standard form if the fractional part of the number, $\bar{x}$, has no more than 7 leading zeros (if $\bar{x}$ is positive or no more than 7 leading "ones" if $\bar{x}$ is negative). Note that this does not mean that the remaining 40 bits in the mantissa are significant. It may well be that none of the 40 bits are significant. The zero presents a special problem. A normalized or standard zero is a number whose mantissa is zero (i.e., all 47 bits zero) and whose exponent is plus zero (i.e. all 6 bits zero)

In the process of normalization a number is shifted left eight bits at a time (with a corresponding decrease in the exponent by one with each shift) until it is in standard form. In some cases the U and R registers are combined and this is noted in the description of the floating point arithmetic.

Normal Mode for Floating Point Arithmetic

When mode light 4 is zero (i.e., off), the machine will carry out floating point calculations according to the following scheme:

(A) Addition and Subtraction.

Addition can be characterized by the following sequences of steps, assuming that the two operands are in U and S.

(1) Examine [(Exp U) - (Exp S)] at output of the adder.

(a) Positive: Interchange (S) and (U), proceed to step (2).

(b) Negative: Proceed to step (2).

(2) (Exp S) → Exp R

(Exp U) + (Exp S) → Exp U

+ 1 → Exp S

(3) Examine sign of mantissa of U.

(a) Positive: clear mantissa of R to "plus zero"

(b) Negative: clear mantissa of R to "minus zero."

(4) Examine (Exp U).

(a) If (Exp U) $\leq$ -16, (S) → U, (Exp R) → Exp U, proceed to step (12).

(b) If (Exp U) $\geq$ -15, proceed to step (5).

(5) Test (Exp U)

(a) Exp U = 0, proceed to step (6)

(b) Exp U $\neq$ 0

(i) Shift UR right by eight

(ii) (Exp U) + (Exp S) → Exp U.

(iii) Repeat Step (5).

(6) Fix-point addition of (U) + (S)

(7) (Exp R) → (Exp U) ,

(a) Exp U = +0, proceed to end

(b) Exp U $\neq$ +0, proceed to step (8).

(8) Test overflow.

(a) No overflow: Proceed to step (9).

(b) Overflow

(i) Shift UR right by eight

(ii) (Exp U) + (Exp S) → Exp U → Exp R

(iii) If Exp U overflows set EXOV indicator, then proceed to end.

(9) Zero → Mantissa of S, Test Mantissa of U.

(a) U = 0, proceed to end of add

(b) U ≠ 0, proceed to step (10).

(10) Compare Sign U and Sign R

(a) $m_8$ of U = $m_8$ of R, proceed to step (12)

(b) $m_8$ of U ≠ $m_8$ of R

(i) Set $+ 1 \cdot 2^{-47}$ in mantissa of S

(ii) $m_8$ of U → $m_8$ of R, proceed to step (11)

(11) Test sign of U

(a) $m_8$ = + ; (U) - (S) → U Fixed point

(b) $m_8$ = - ; (U) + (S) → U Fixed point

(12) Test $m_1$ to $m_8$ of U

(a) $m_1$ → $m_8$ ≠ 0, proceed to end of add.

(b) $m_1$ → $m_8$ = 0,

(i) Shift UR left eight places

(ii) Exp (U) - Exp (S) → Exp U → Exp R.

(c) Test Exp (U) for underflow

(i) Exp (U) underflows, clear UR to zeros, proceed to end of addition

(ii) Exp (U) does not underflow: Repeat step (12)

(13) End of Floating Add.

(B) Normalization in Floating Point Addition.

No assumption is made about the initial operands - they may be normalized or unnormalized numbers.

The result of a floating point addition is always normalized with the following two exceptions:

(1) For complete cancellation of significant figures in U (i.e., when the result of the fixed point addition step is zero), the U register will be left with a zero mantissa and an exponent equal to Exp R. The R register will contain whatever was shifted out of U in the initial matching of exponents.

(2) When the exponents of both operands are "plus zero" the addition will be a fixed point operation with no renormalization. Thus the floating point addition order can be applied to fixed point numbers if one is careful to give the fixed point quantit a "plus zero" exponent.

Overflow and Underflow.

It has already been mentioned that an overflow condition is sensed by the machine whenever sign and overflow bits in either the exponent or mantissa of U differ in value, and that one can recognize the (10) bit combination as a case of underflow and (01)as overflow. Exponent underflow on floating point operations is sensed by the machine, and (U) and (R)are replaced by the zero word, so this is not a condition the programmer can detect. In addition, the exponent overflow indicator is used to detect a "spill" of a "1" bit on logical shifts although this is not truly a numerical overflow. The table summarizes the cases which may arise:

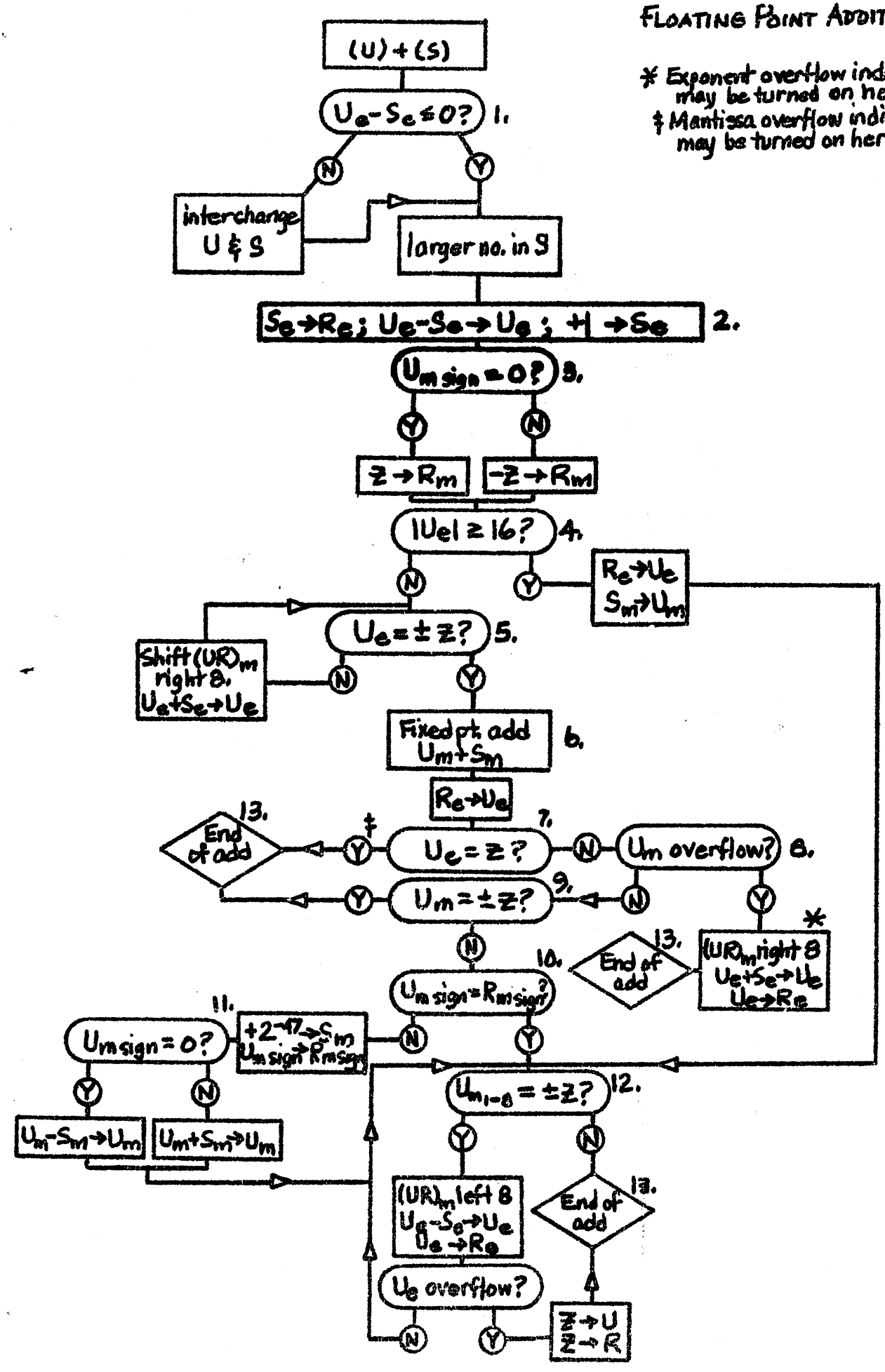
| Indicator | Condition | Cause | Type |
|---|---|---|---|
| Mantissa Overflow Indicator | ON | 1. Arithmetic left shift | Overflow or underflow |
| | | 2. Fixed point add | Overflow or underflow |
| | | 3. Comparison of fixed point numbers (exponents zero) | Overflow or underflow |
| | | 4. Add to Memory of fixed point no's. | |
| | OFF | 1. Mantissa overflow test | —— |
| Exponent Overflow Indicator | ON | 1. Floating point add, subtract, multiply, or divide | Overflow |
| | | 2. Comparison orders and add to memory | Overflow |
| | | 3. Logical left shift | Overflow |
| | OFF | 1. Exponent overflow test | ——— |

Note that overflow and sign bits act independently in arithmetic orders, but are regarded as a single bit (00 or 11) during logical operations in order to prevent spurious overflow conditions being sensed. For this reason, before any logical order (Class 5 or Class 4 logical shift or bit count) the machine ties the overflow bit equal to the sign bit in both the exponent and mantissa of U.

Lastly, it should be noted that field 1 codes which call for U, -U, |U|, or -|U| to be brought to U treat sign and overflow bits independently.

# Floating Point Addition

* Exponent overflow indicator may be turned on here.
‡ Mantissa overflow indicator may be turned on here.



(B) Floating Multiply

(1) $(U_e) + (S_e) \rightarrow U_e \rightarrow R_e$

(2) Test exponent overflow

(a) Overflow - Set EXOV indicator - proceed to end.

(b) Underflow - Clear UR to + 0 - proceed to end.

(c) No overflow - Proceed to step (3).

(3) Test sign of (S).

(a) (S) +, go to step (4).

(b) (S) -, complement (U) and (S), then go to step (4).

(4) $(S_m) \rightarrow R_m$

(5) $(U_m) \rightarrow S_m$, $-0 \rightarrow U_m$ if U is - ;

$+0 \rightarrow U_m$ if U is + .

(6) Test for zero at adder output.

(a) Zero; clear $U_m$ and $R_m$, proceed to end.

(b) Not zero; proceed to step (6).

(7) Multiply by repeated addition as in fixed point multiplication.

(8) $m_s$ of $U \rightarrow m_s$ of R.

(9) If mantissa of $U < 1/256$, the number in U and R combined is normalized.

(C) Floating Divide.

(1) Test Op 3 of divide instruction.

(a) Op 3 = 0, 3, 4, 5, 6, or 7, proceed to step (2).

(b) Op 3 = 1, clear $R_m$ to sign of $(U_m)$; proceed to step (2).

(c) Op 3 = 2, clear $U_m$ to sign of $(R_m)$; proceed to step (2).

(2) Compare sign of (U) and sign of (S).

(a) $U_s = S_s$, then will not complement quotient after division in step (10).

(b) $U_s \neq S_s$, $-(S) \rightarrow S$ and will complement quotient after division in step (10).

(3) $(U_e) - (S_e) \rightarrow U_e \rightarrow R_e$ and $-1 \rightarrow S_e$

(4) Compare $U_m$ and $S_m$.

(a) $|U_m| \geq |S_m|$, shift (UR) in right 8 and $(U_e) + 1 \rightarrow U_e \rightarrow R_e$.

(b) $|U_m| < |S_m|$, proceed to step (5).

(5) Subtracter output $\neq 0$ and sign of subtracter output $\neq U_{ms}$?

(a) Yes, $0 \rightarrow R_{ms}$, shift $(UR)_m$ left 1, set shift count SC = 0.

(b) No, test ML#1.

(i) On, go to end of divide.

(ii) Off, stop.

(6) Subtracter output $\neq 0$ and sign of subtracter output $\neq U_{ms}$?

(a) Yes, $0 \rightarrow R_{m47}$.

(b) No, $1 \rightarrow R_{m47}$. subtracter output $\rightarrow$ U.

(7) Advance SC by 1.

(8) SC = 47?

(a) Yes, proceed to step (9).

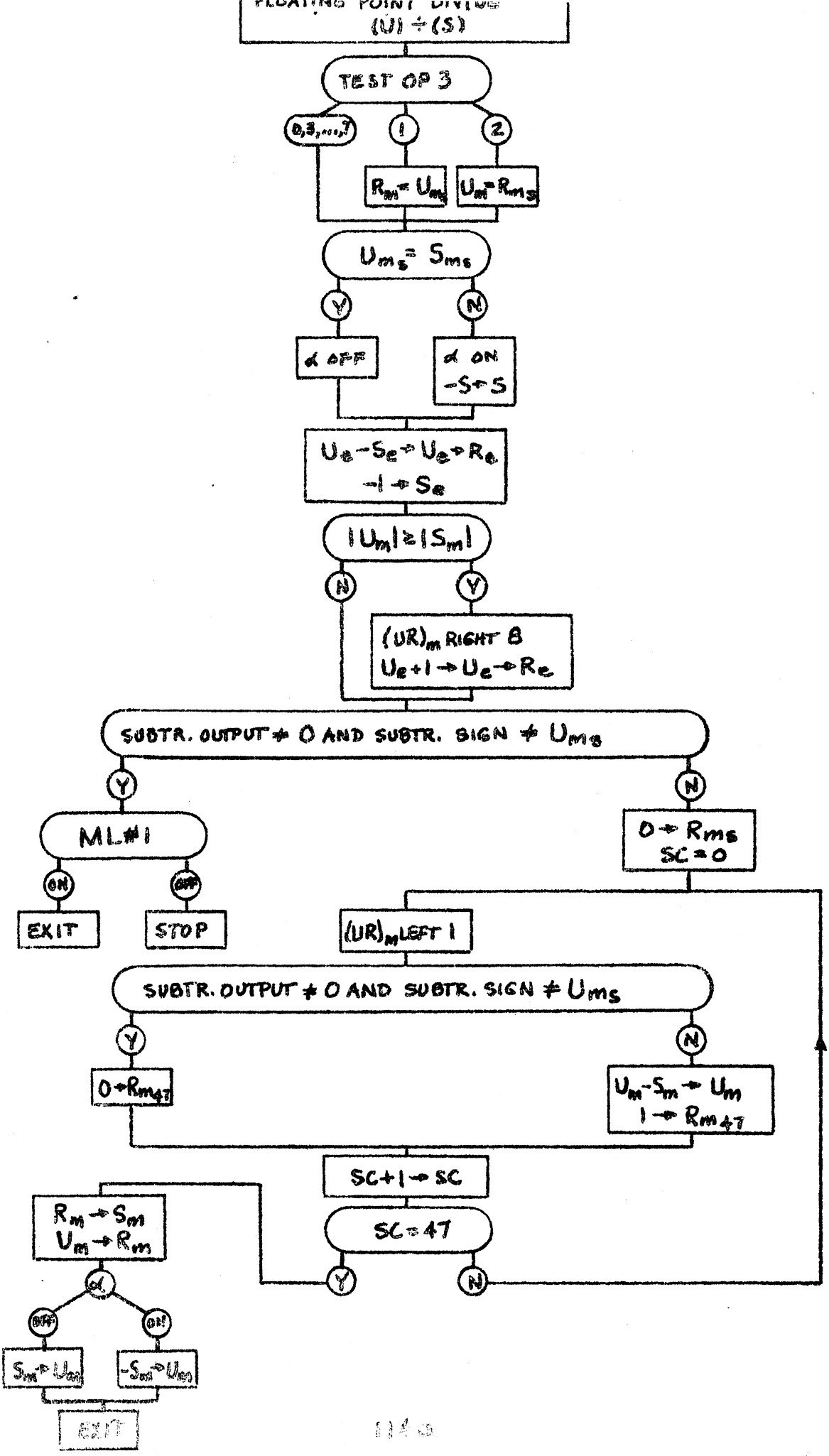
(b) No, (UR) arithmetic shift left 1; return to step (6).

(9) $(R_m) \rightarrow S_m$, $(U_m) \rightarrow R_m$.

(10) $(S_m) \rightarrow U_m$ or $-(S_m) \rightarrow U_m$ as indicated in step (2).

(11) End of divide.

After division the quotient is in U and the remainder is in R, both with the exponent of the quotient.


FLOATING POINT DIVIDE
(U) ÷ (S)
TEST OP 3
0,3,...,7
1
2
$R_m = U_m$
$U_m = R_{ms}$
$U_{ms} = S_{ms}$
Y
N
α OFF
α ON
$-S \to S$
$U_e - S_e \to U_e \to R_e$
$-1 \to S_e$
$|U_m| \geq |S_m|$
N
Y
$(UR)_m$ RIGHT 8
$U_e + 1 \to U_e \to R_e$
SUBTR. OUTPUT ≠ 0 AND SUBTR. SIGN ≠ $U_{ms}$
Y
N
ML#1
ON
OFF
EXIT
STOP
$0 \to R_{ms}$
$SC = 0$
$(UR)_m$ LEFT 1
SUBTR. OUTPUT ≠ 0 AND SUBTR. SIGN ≠ $U_{ms}$
Y
N
$0 \to R_{m47}$
$U_m - S_m \to U_m$
$1 \to R_{m47}$
$SC + 1 \to SC$
$SC = 47$
Y
N
$R_m \to S_m$
$U_m \to R_m$
α
OFF
ON
$S_m \to U_m$
$-S_m \to U_m$
EXIT

# APPENDIX 1

# ELECTRONICS OF THE COMPUTER

# I.

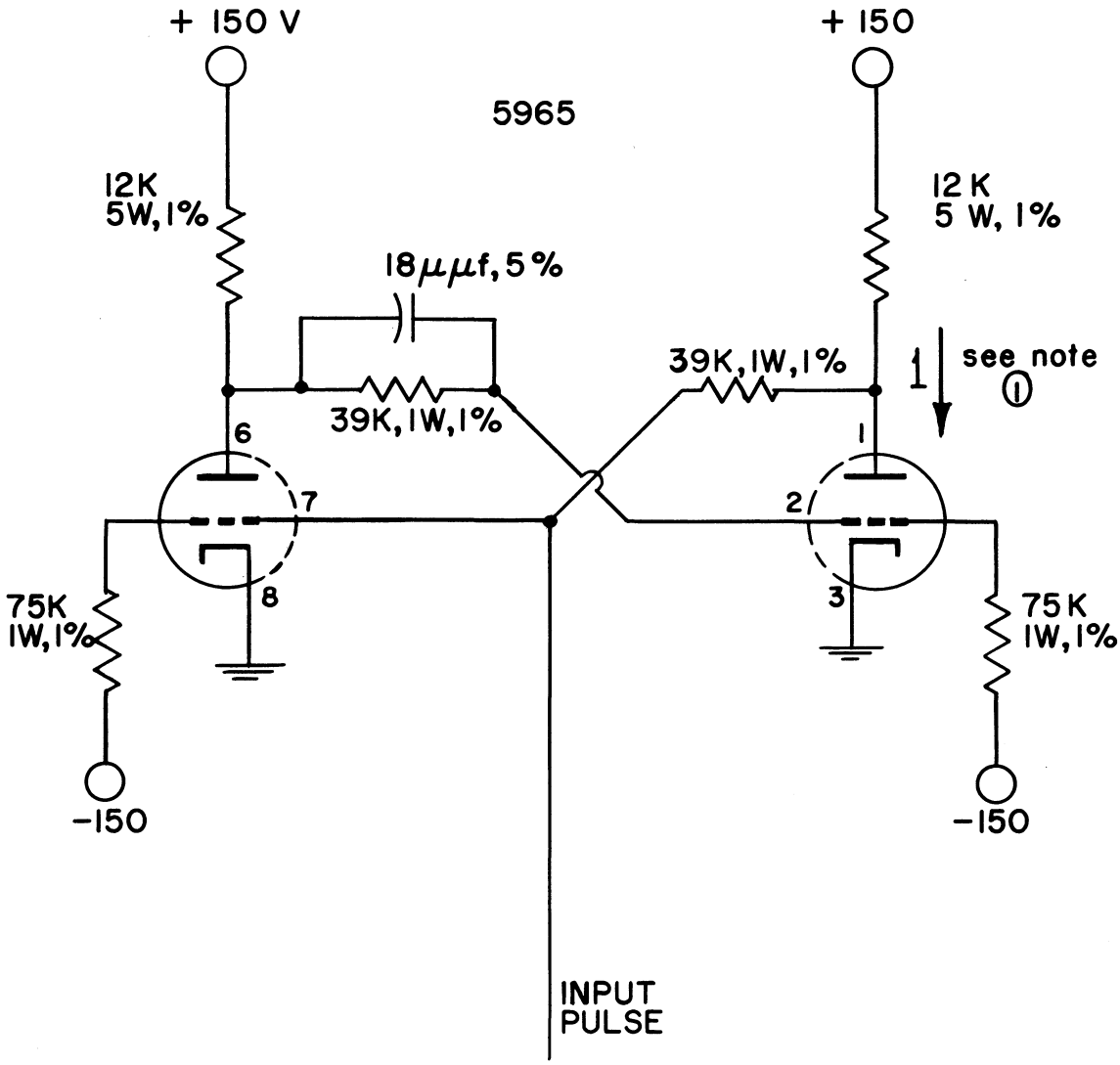
# ARITHMETIC UNIT

The arithmetic unit consists of seven registers, an adder, and gates (such as the "central dispatcher") for setting or transferring the contents of registers. Three of the registers (R, S, and U) are used in arithmetic manipulations and may be shifted left or right. The other four are a set of "T" registers which are used for temporary storage. The registers and adder are static circuits, i.e., voltages can have stable d.c. values in the absence of gating signals.

## Basic Circuitry

### Flip-Flop

The basic element of each register is a "flip-flop" circuit, often called a "bistable multivibrator," "binary," or "Eccles-Jordan circuit," Figure 1. This circuit is symmetrical in d.c. operation with conduction either through the right hand triode (defined as a "1" condition) or conduction through the left hand triode (defined as a "0" condition). The state of the flip-flop (FF) circuit is measured at the grid of the right hand triode, which drives the grid of a cathode follower to provide a low impedance decoupled output. Thus the state "1" is identified with zero volts output, the state "0" corresponds to about -20 volts output. By this definition, the COMPLEMENT




NOTE: ① THE STAGE IS CONSIDERED TO BE IN THE "1" STATE WITH ELEMENTS 1,2&3 CONDUCTING

② RESISTOR VALUES ARE THE SAME AS THE LOS ALAMOS COMPUTER, MANIAC II

FIGURE 1 BASIC FLIP-FLOP CIRCUIT ARITHMETIC UNIT

of the state of the FF can be measured at the grid of the left hand triode. In the U, R, and S registers the above method is used to read the state of the FF; the T registers are read in the plate circuit.

Since in arithmetic operations the number is 54 bits in length, each register has 54 flip-flops, each with provisions for reading its state and with provisions for setting its state. The latter is discussed in the following section.

Gating In

To store a bit in the FF requires the capability to set it to either a 1 or a 0 condition. Two gating schemes commonly used are the double-sided and the single-sided gates.

In the double-sided gating scheme the left hand triode is cut off to set in a 1 (by lowering its grid or the opposite plate), and the right hand tube is similarly cut off to set in a 0. This requires two gates: one capable of setting in 0's, the other to set in 1's. Thus for setting a number into a register from any of N different sources, there are required 2N gate connections to the register.

An alternative technique is to clear the entire register to 0, then set to 1 all those stages that are to have a 1 gated in. This still requires two gates, but one of them is common for all gating operations. Thus for transferring a number into a register from any of N different sources there are required N + 1 gate connections.

For a large number of gates, roughly half can be saved by using single-sided operation rather than double. However, there is a possible disadvantage if extra time is involved by first setting the FF to 0 before gating in the 1's. This can be avoided by setting in the 0's and 1's simultaneously, provided that the setting in of a 1 overrides the setting in of a 0. Here this is accomplished by driving the same grid with opposing signals that are at two different impedance levels.

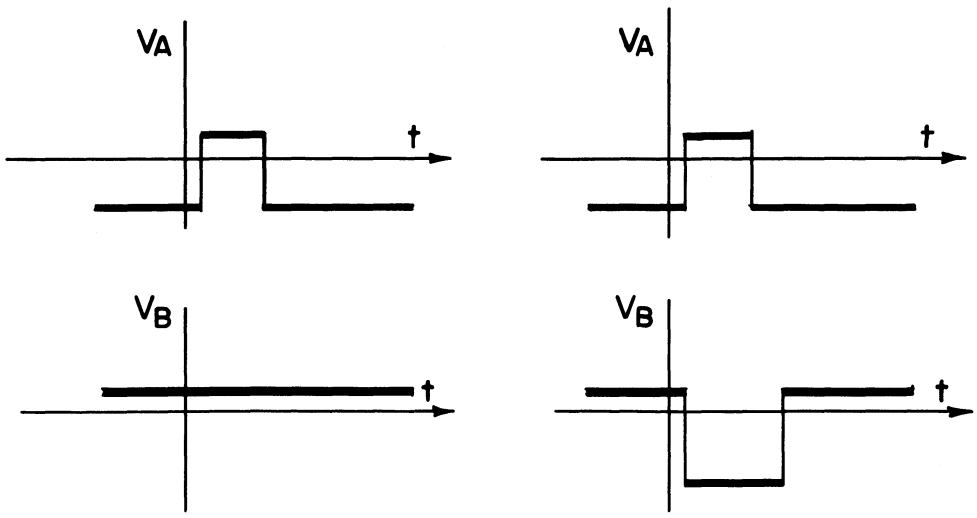
One possible circuit arrangement is the asymmetrical setting system shown in Figure 2: when clearing the FF stage to 0, the left triode grid is pulled up by the current, I, when switched by a positive pulse applied at A. When setting the FF stage to 1, the same grid is driven negative by a voltage pulse at B. Since in each case the left hand grid is being set to its proper value, no speed-up capacitor is needed from the opposite plate.

Accurate registration is not essential. It is required only:

(1) that pulse "A" be large enough to set the FF to 0 during the pulse duration;

(2) that pulse "B" go negative enough and last long enough to set the FF to 1; and

(3) that when the "B" pulse exists, it end after the "A" pulse.




VOLTAGE WAVE FORMS APPLIED TO POINT A AND POINT B TO CLEAR TO "0"

VOLTAGE WAVE FORMS APPLIED TO POINT A AND POINT B TO SET "1"

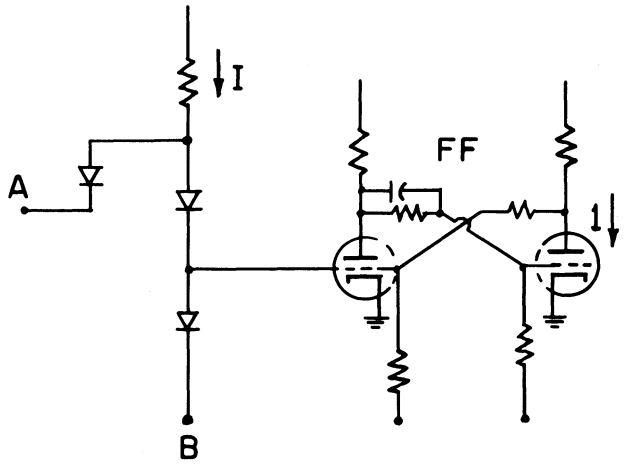



FIGURE 2 AN ASYMMETRIC SINGLE-SIDED GATE

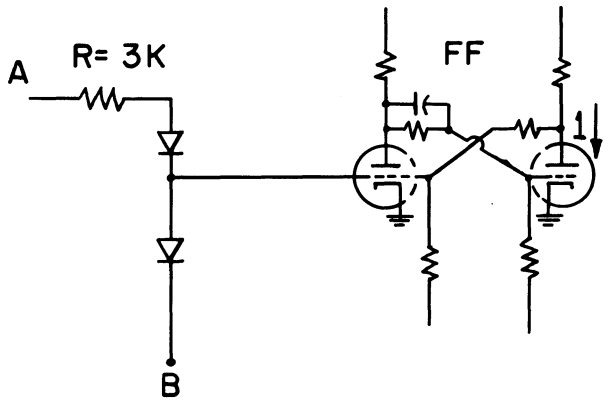



FIGURE 3 GATE CIRCUIT FOR THIS COMPUTER

A satisfactory approximation, which saves one diode per stage and some d.c. power, is the circuit in Figure 3. The "A" pulse need not start at -20 V, but a voltage sufficient to keep the left triode cut off. For the circuit pictured, -15 V is used.

It is possible to have several B inputs: see Figures 4 and 5. For the circuit of Figure 4 the back leakage current of the diodes in parallel may load the high-impedance grid circuit. The configuration of Figure 5 has the parallel diodes shunted by the low impedance of a conducting diode "d." The quiescent voltage of the "B" lines is about +5 volts, enough to prevent noise and "hash" on the lines from setting a 1 into the FF. During a 0 state when the left triode is caught on grid current, the barely-conducting diode "b" does not divert much of the grid current. (Diode "b" has a high impedance at the voltage level of $E_{grid}$ minus $E_{diode\ "d"}$.)

The FF stage then has the appearance of Figure 5. The 18 mmf capacitor flattens the response of the crossover divider. Whereas the grid of the left triode is limited by the voltage at "A" to a swing from 0 to -15 V, the grid of the right triode will move between 0 and -20 volts. This grid is connected to a cathode follower for a decoupled low impedance output.

Gating Out

The usual output from a register is dynamic; a 1 output

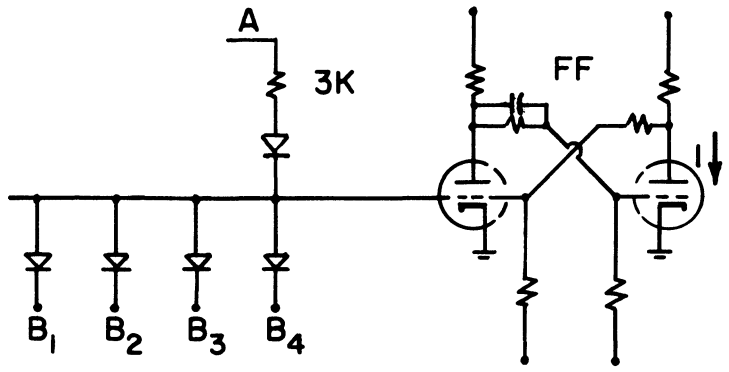



FIGURE 4 A CIRCUIT FOR MULTIPLE INPUT GATES

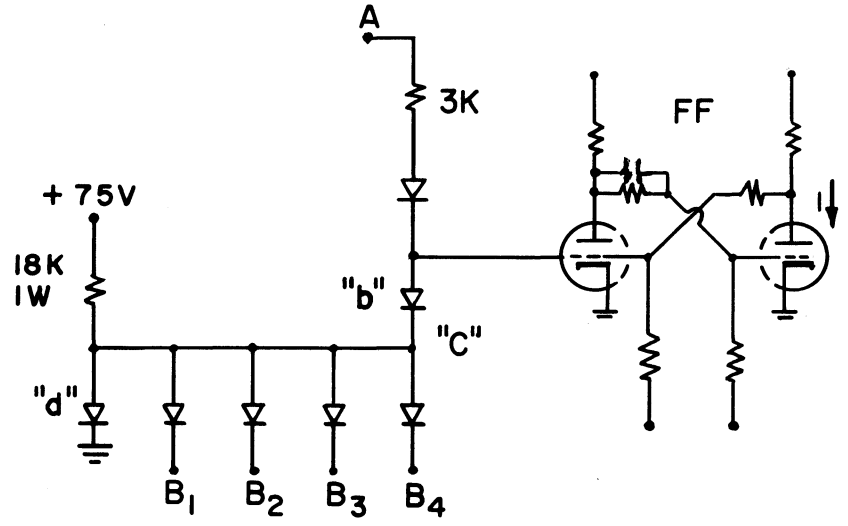



FIGURE 5 A BETTER CIRCUIT FOR MULTIPLE INPUT GATES

is a pulse on the secondary of a pulse transformer whose primary is in the plate of a 6197 pentode amplifier, see Figure 6. The pentode can be driven on from cut-off by a READ pulse when the FF is in a 1 state so a pulse or no pulse indicates, respectively, 1 or 0.

The AND gate for the 1 state and the READ pulse uses a delay line, as shown in Figure 6. The 15-volt positive READ pulse applied to the shield of the delay line either turns on the pentode or does not, depending upon the voltage of the FF-controlled cathode follower.

The delay line is used only in those registers where temporary storage is required by some gating operations: gates such as shifts and complements, which erase the information being used to control the gating. If the state of the FF is changed, the right end of the delay line is independent of that change for .4 microsecond.

The damping winding has a damping resistor and a diode whose polarity allows damping only after the pulse, during the back-kick. Sometimes this winding is used for additional purposes in COMPLEMENT and EXTRACT operations.

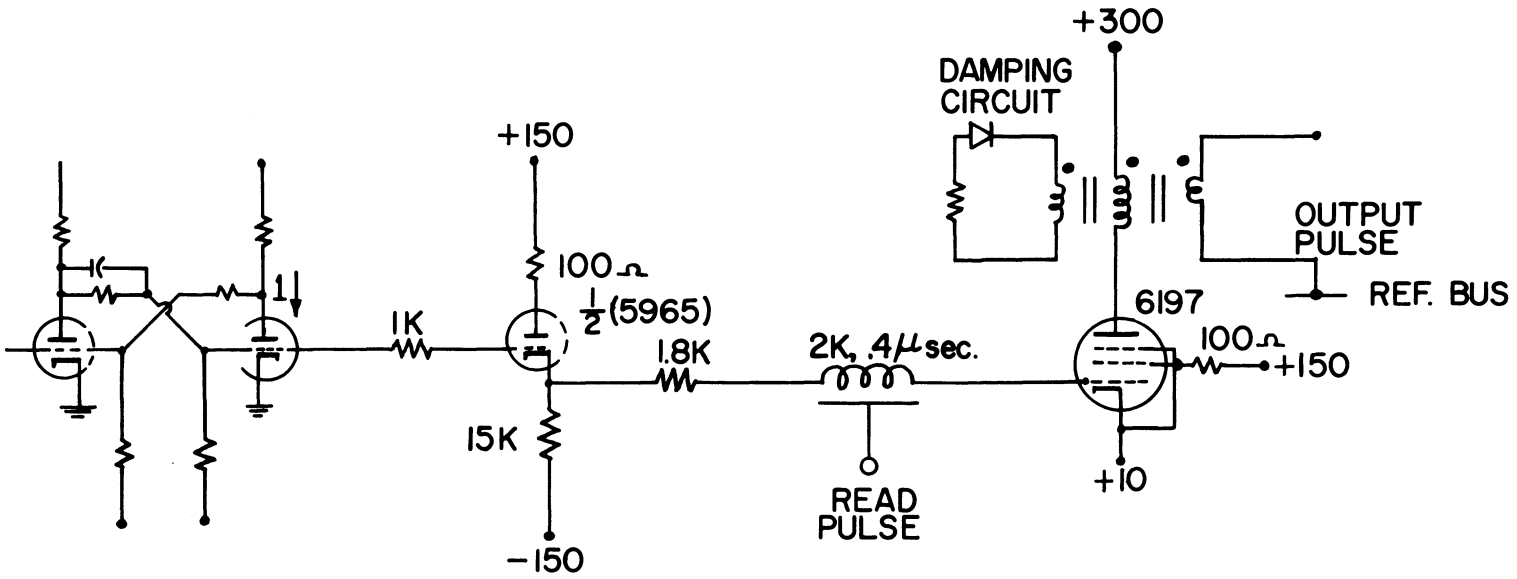


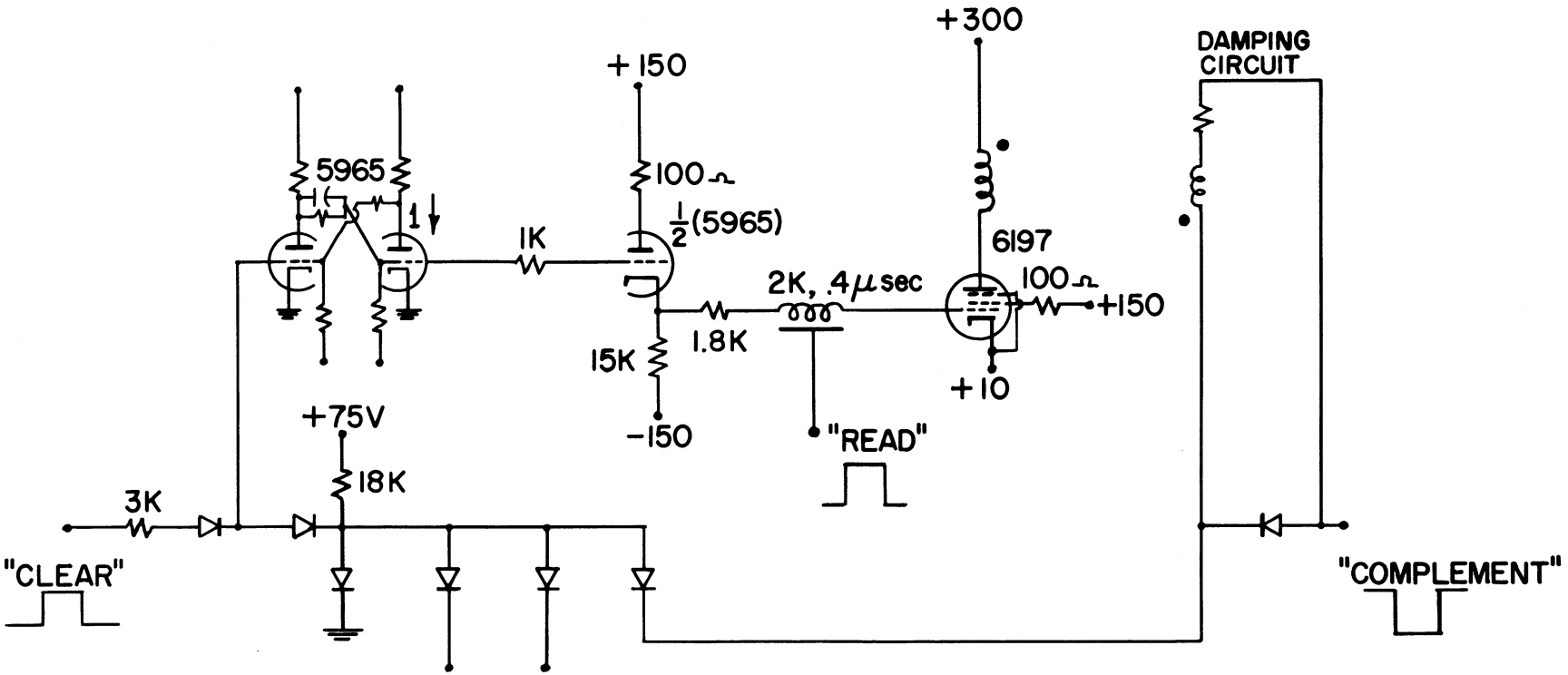

FIGURE 6 THE DYNAMIC OUTPUT CIRCUIT FOR REGISTERS U, S AND R

Complement

Two of the registers to be discussed, S and U, have provision for replacing the number in the register by its complement: replacing each 1 by a 0 and conversely. This scheme requires the 3 coincident pulses shown on the schematic, Figure 7. At the delay line shield there is applied a positive pulse to "READ" the contents of the register. Simultaneously, there is a positive pulse at the FF grid to "CLEAR" the register to zero, and a negative "COMPLEMENT" gate is applied to the damping bus bar.

If the number that was in the register was zero, the initial condition on each end of the delay line was -20 volts. Thus the READ pulse of +15 volts does not turn on the pentode driver, thus no voltage is developed at the transformer secondary, and the negative COMPLEMENT gate sets a 1 into the FF (overriding the CLEAR pulse).

If the number that was in the register was 1, the initial condition at the right end of the delay line is about +2 volts and will remain at +2 volts for .4 microseconds although the register was CLEARED to zero. Thus the READ pulse of +15 volts drives the grid positive, turning on the pentode hard. So at the secondary there is developed across the damping diode an output of approximately +35 volts, which inhibits the COMPLEMENT




"CLEAR", "READ", AND "COMPLEMENT" GATE PULSES ARE SIMULTANEOUS

FIGURE 7 COMPLEMENT GATE CIRCUIT

pulse in the manner indicated in Figure 8. Thus a 1 condition has caused a positive ouput which overrides the would-be input gate, resulting in the 0 condition which was just previously set by the CLEAR pulse.

For proper operation, this circuit requires that the overriding output pulse be well shaped: it must be greater in amplitude and duration than the COMPLEMENT pulse, and it must have no negative undershoot. The diode across this winding performs the dual purpose of damping the undershoot and shaping the override pulse for the complement gate.

Extract

The EXTRACT gate is used in applications where it is desirable to select or to replace a portion of the number. The operation performed is that of gating (S)$\longrightarrow$U when the corresponding bit in R holds a 1; otherwise, (U)$\longrightarrow$U.

The circuitry uses two inputs to U (two of the multiple input gates described previously); one input is determined by S AND R, the other results from U INHIBITED BY R.

The first input uses a winding on the S pulse transformer in series with a similar winding on the R transformer, as an input to U. The other end of this double winding is returned to a bus held at about plus 25 volts. Thus a 1 in S AND a 1 in R is required to set a 1 in U. Otherwise that bit in U is cleared to 0 by the CLEAR gate.

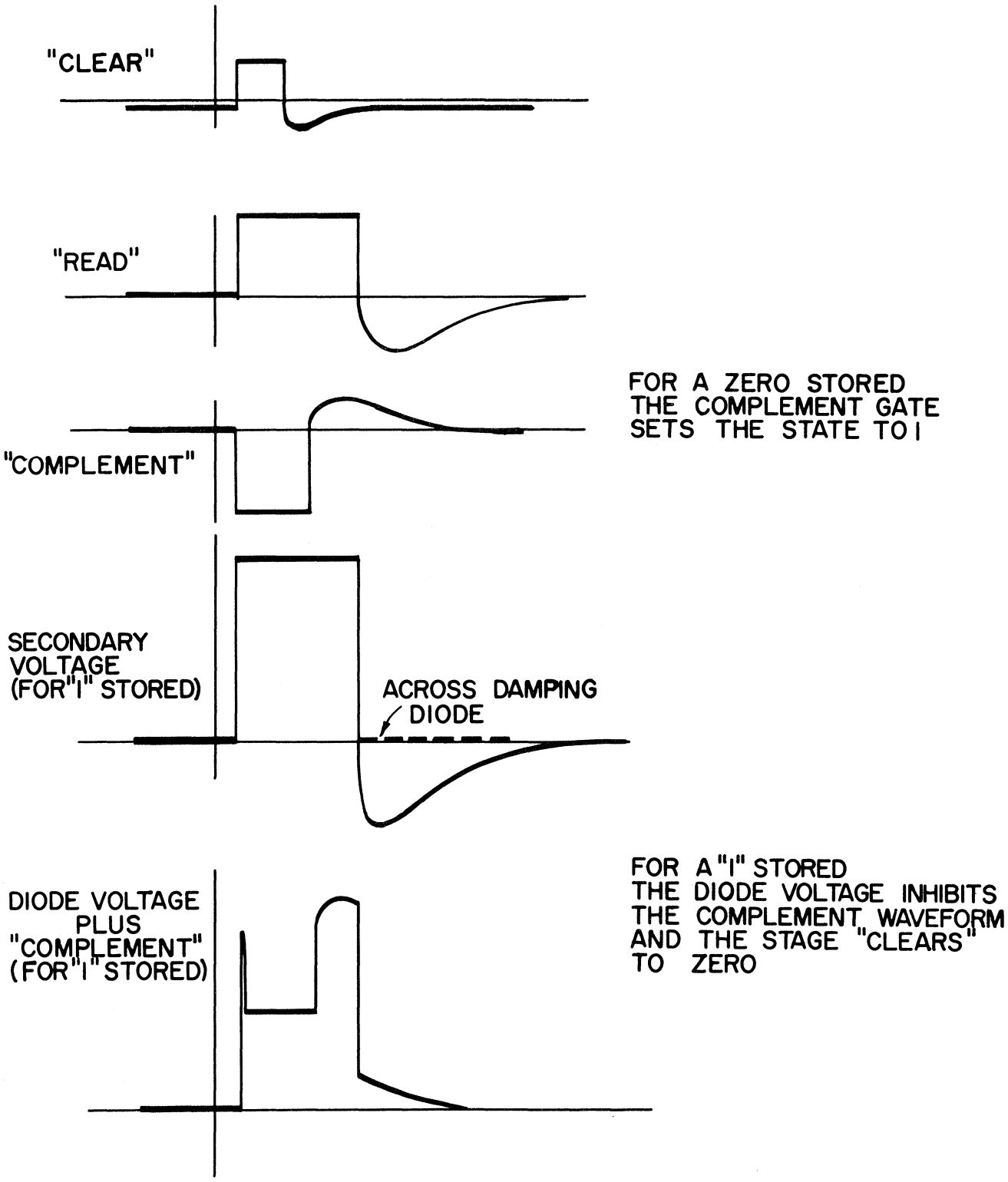



FIGURE 8 VOLTAGE WAVEFORMS IN THE COMPLEMENT CIRCUIT

The second input uses a clockwise 2-turn winding on the U transformer in series with a counterclockwise 5-turn winding on the R transformer. Thus a pulse from U will set a 1 back into U unless R also has a 1, in which case the positive pulse from R will INHIBIT the pulse from U. Since the pulse from R must be large and must have no undershoot, the voltage on the R damping diode is used in a manner similar to the COMPLEMENT operation described in the preceding section.

## Arithmetic Unit

### U Register

Since the U register is utilized in all arithmetic operations, it has several possible inputs. Input signals can be sent from the adder, either directly or shifted left one stage or shifted right one stage; two extract inputs are available so that a bit can be gated into U from either U or S, depending upon the state of the corresponding bit in R; or a number from a variety of places can be gated into U via the central distributor. Or U itself can be gated back into U, shifted 1 or 8 places left or right; or the complement of U can be gated into U.

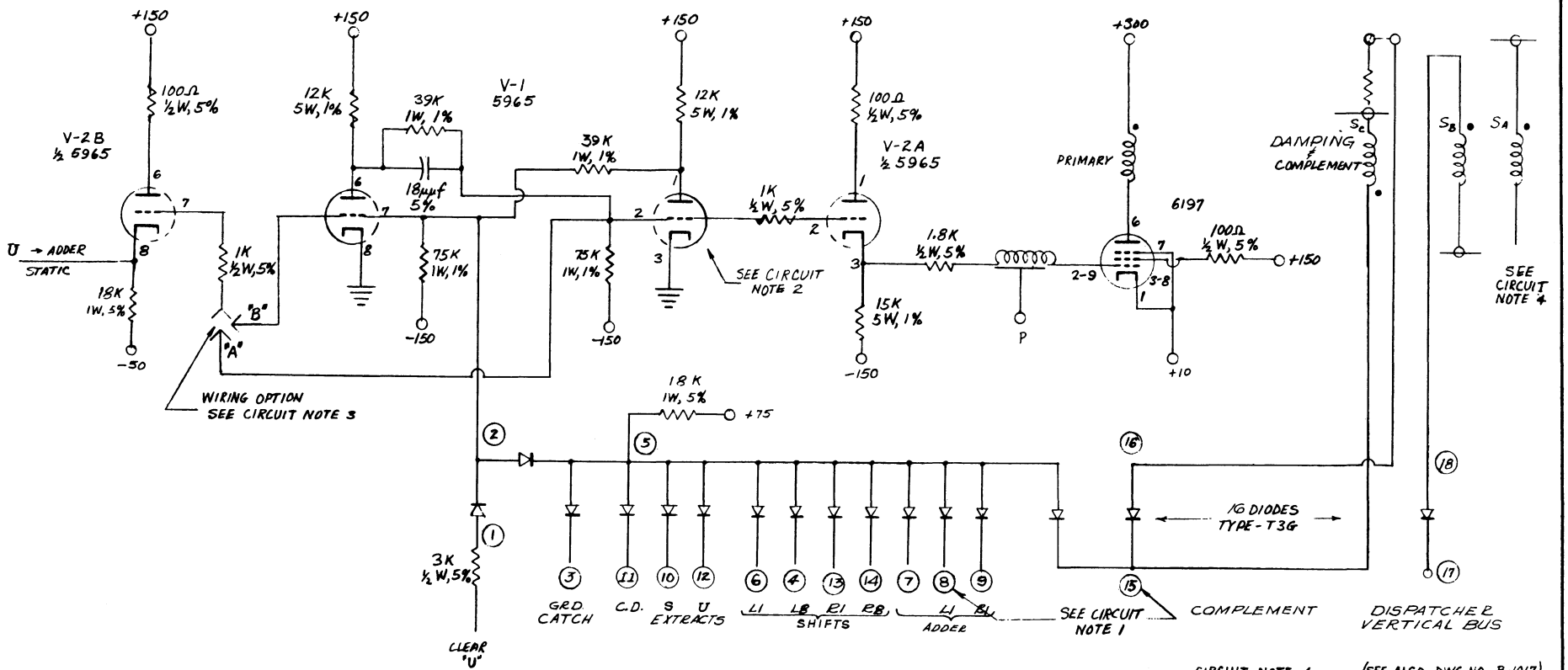
Each input to U is a negative pulse through a diode, as described earlier in this manual. Each input is derived from two turns of a pulse transformer, wound in a (clockwise) direction

so as to pulse negative from a reference voltage bus. An input is gated in from any particular location only when the voltage of the corresponding reference bus is pulsed from its usual +25 volts down to about +5 volts in coincidence with the negative 20-volt pulse from the pulse transformer.

The secondary windings of the pulse transformer (outputs from U) include a complement winding, an extract winding, an output to the central dispatcher, and the four shift windings.

In addition to the pulse outputs from U there is a static output connection to the adder.

As shown by the Schematic C-1052, the static output to the adder is from a cathode follower (V-2B) whose input alternates from stage to stage between the two grids of the flip-flop: on even-numbered stages the cathode follower is connected to read the number stored in the flip-flop, i.e., zero volts indicates a 1; on odd stages the cathode follower reads the complement of the stored number, i.e., zero volts indicates a 0. The purpose of alternating the connections is to minimize the carry time, discussed in the subsequent adder section. The supply voltage of minus 50 volts for this cathode follower is for protection of the adder diode in case the tube V-2 is pulled out.

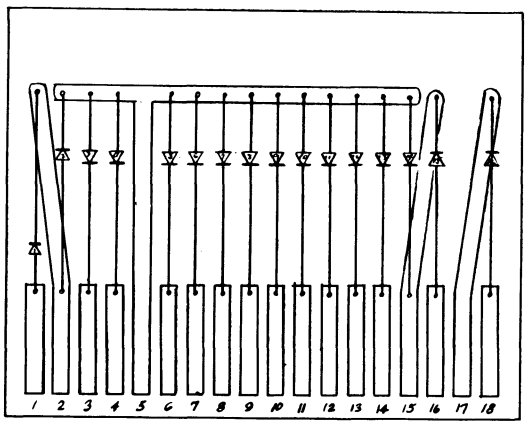



CIRCUIT NOTE. 4 (SEE ALSO DWG. NO. B-1017)

WINDINGS OF PULSE TRANSFORMER

| NO. | HORIZ. BUSS | TURNS | COLOR | CONNECTION |
|---|---|---|---|---|
| 1 | | 28 | BLUE | PRIMARY |
| 2 | | -5 | WHITE | DAMPING & COMPLEMENT |
| 3 | | -2 | RED | TO CENTRAL DISPATCHER |
| 4 | | +2 | BLUE | SHIFT L1 |
| 5 | | +2 | SEE NOTE 5 | SHIFT L8 |
| 6 | | +2 | YELLOW | SHIFT R1 |
| 7 | | +2 | SEE NOTE 5 | SHIFT R8 |
| 8 | | +2 | BROWN | EXTRACT [U] → U |

DIODE BOARD (SHOWN FOR REFERENCE ONLY)
PHYSICAL LAYOUT (SEE DWG. NO. A-1060)




CIRCUIT NOTE: 1- NUMERALS WITHIN CIRCLES INDICATE TERMINALS OF DIODE BOARD.

2- THE STAGE IS CONSIDERED TO BE IN THE "1" STATE WITH VACUUM TUBE V-1, ELEMENTS 1, 2 & 3 CONDUCTING.

3- WIRING OPTION "A" IS TO BE APPLIED ON STAGES NO. 2, 4, 6 AND 8. WIRING OPTION "B" IS TO BE APPLIED ON STAGES NO. 1, 3, 5 AND 7. (VIEW CHASSIS FROM WIRING SIDE WITH V-1 BELOW V-2; COUNT FROM LEFT)

4- SEE CHART TO THE RIGHT

5- COLORS OF WINDINGS FOR SHIFTS OF 8 FOR EACH STAGE ARE AS FOLLOWS:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| L-8 | BROWN | RED | YELLOW | GREEN | BLUE | VIOLET | GREY | WHITE |
| STAGE NO. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| R-8 | RED | YELLOW | GREEN | BLUE | VIOLET | GREY | WHITE | BROWN |

| NO. | DATE | BY |
|---|---|---|
| 1 | 3/22/58 | TMm |
| 2 | 9/13/58 | RA |

RICE INSTITUTE COMPUTER

ARITHMETIC UNIT
"U" REGISTER. SCHEMATIC

DRAWN BY 2-27-58 JKW | SCALE NONE | MATERIAL

CHK'D | DATE | APP'D | DRAWING NO. C 1052

ISSUED BY 1050

S Register

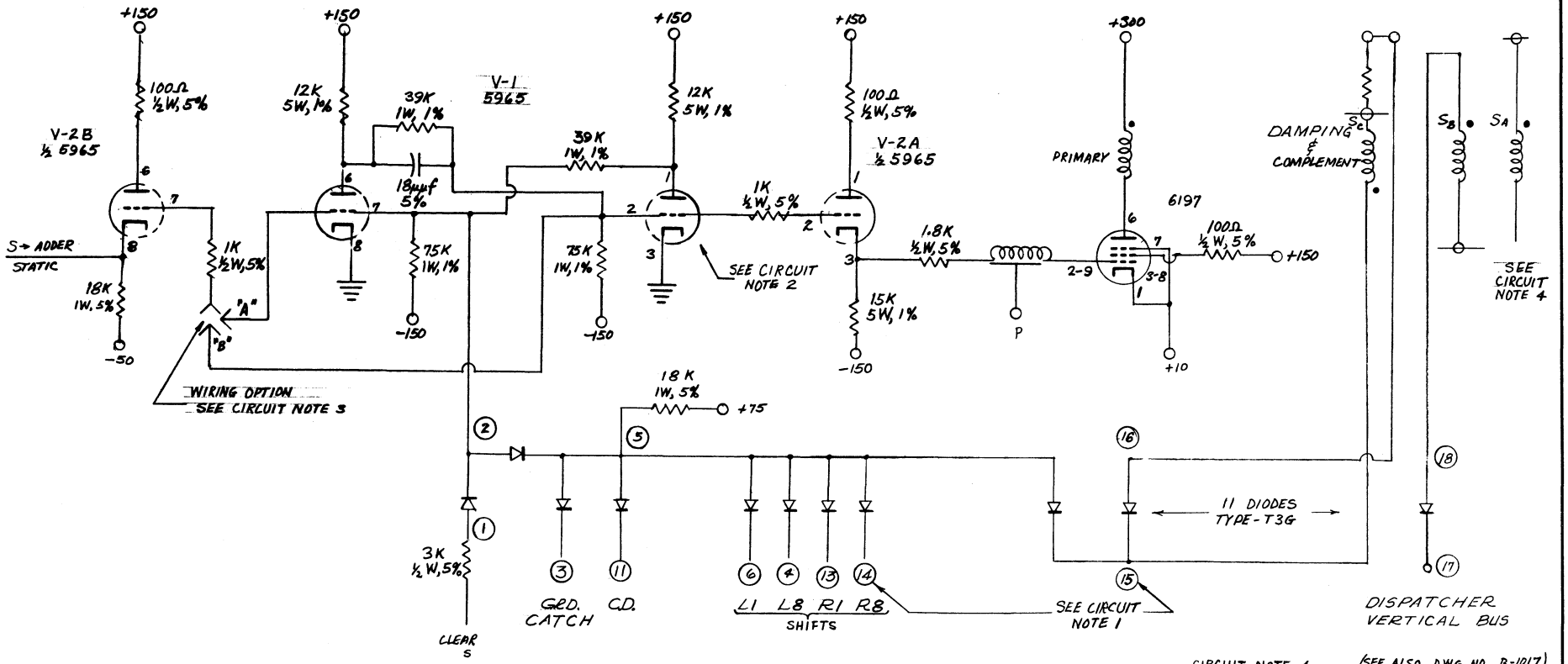
The S register (see Schematic C-1037) looks very much like the U register: the basic flip-flop circuit with a pulse tube output and with a static output to the adder. It is also similar with circuitry for complementing, shifting left or right 1 or 8 stages, and for gating into or from the central dispatcher.

However, the extract function uses merely a 2-turn winding on the output transformer of S (to accomplish (S)$\longrightarrow$ U if R is 1). Also the output to the adder alternates opposite to U, such that the complement of S is added to U and vice versa. This is discussed in greater detail in the adder section.

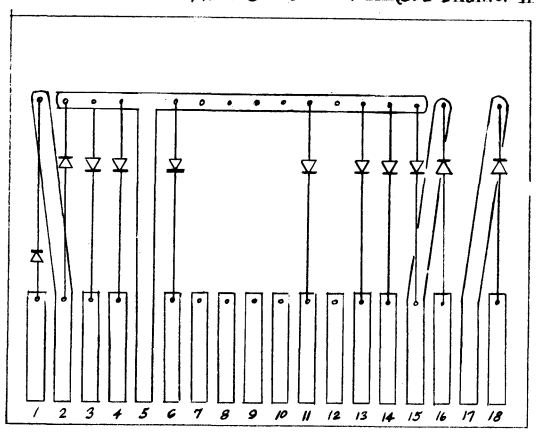
R Register

The R register (see Schematic C-1022) is similar to the U and S registers: the same basic flip-flop circuit with pulse tube output and diode OR inputs. It can be shifted left or right in a similar manner as they, and can be gated into or from the central dispatcher.

This register is unlike U or S in having no connection to the adder and in having no COMPLEMENT gate. The damping winding, used as the complement gate INHIBITOR in S and U, is used to INHIBIT (U)$\longrightarrow$U in the EXTRACT function. A separate winding on the transformer, in series with the EXTRACT winding on S, is for the function: (S)$\longrightarrow$U when R is 1.




DIODE BOARD (SHOWN FOR REFERENCE ONLY)
PHYSICAL LAYOUT (SEE DWG. NO. A-1046)




CIRCUIT NOTE: 1- NUMERALS WITHIN CIRCLES INDICATE TERMINALS OF DIODE BOARD.

2- THE STAGE IS CONSIDERED TO BE IN THE "1" STATE WITH VACUUM TUBE V-1, ELEMENTS 1, 2 & 3 CONDUCTING.

3- WIRING OPTION "A" IS TO BE APPLIED ON STAGES NO. 2, 4, 6 AND 8. WIRING OPTION "B" IS TO BE APPLIED ON STAGES NO. 1, 3, 5 AND 7. (VIEW CHASSIS FROM WIRING SIDE WITH V-1 BELOW V-2; COUNT FROM LEFT)

4- SEE CHART TO THE RIGHT

5- COLORS OF WINDINGS FOR SHIFTS OF 8 FOR EACH STAGE ARE AS FOLLOWS:

| STAGE NO. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| L8 | BROWN | RED | YELLOW | GREEN | BLUE | VIOLET | GREY | WHITE |
| R8 | RED | YELLOW | GREEN | BLUE | VIOLET | GREY | WHITE | BROWN |

CIRCUIT NOTE 4 (SEE ALSO DWG. NO. B-1017)

WINDINGS OF PULSE TRANSFORMER

| NO. | HORIZ. BUSS | TURNS | COLOR | CONNECTION |
|---|---|---|---|---|
| 1 | | 28 | BLUE | PRIMARY |
| 2 | | -5 | WHITE | DAMPING & COMPLEMENT |
| 3 | | -2 | RED | TO CENTRAL DISPATCHER |
| 4 | | +2 | BLUE | SHIFT L1 |
| 5 | | +2 | SEE NOTE 5 | SHIFT L8 |
| 6 | | +2 | YELLOW | SHIFT R1 |
| 7 | | +2 | SEE NOTE 5 | SHIFT R8 |
| 8 | | +2 | BLACK | EXTRACT (S) → U |

| REVISIONS | | | |
|---|---|---|---|
| NO. | DATE | BY | |
| 1 | 5/22/58 | [illegible] | |
| 2 | 9/12/58 | RA | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

RICE INSTITUTE COMPUTER

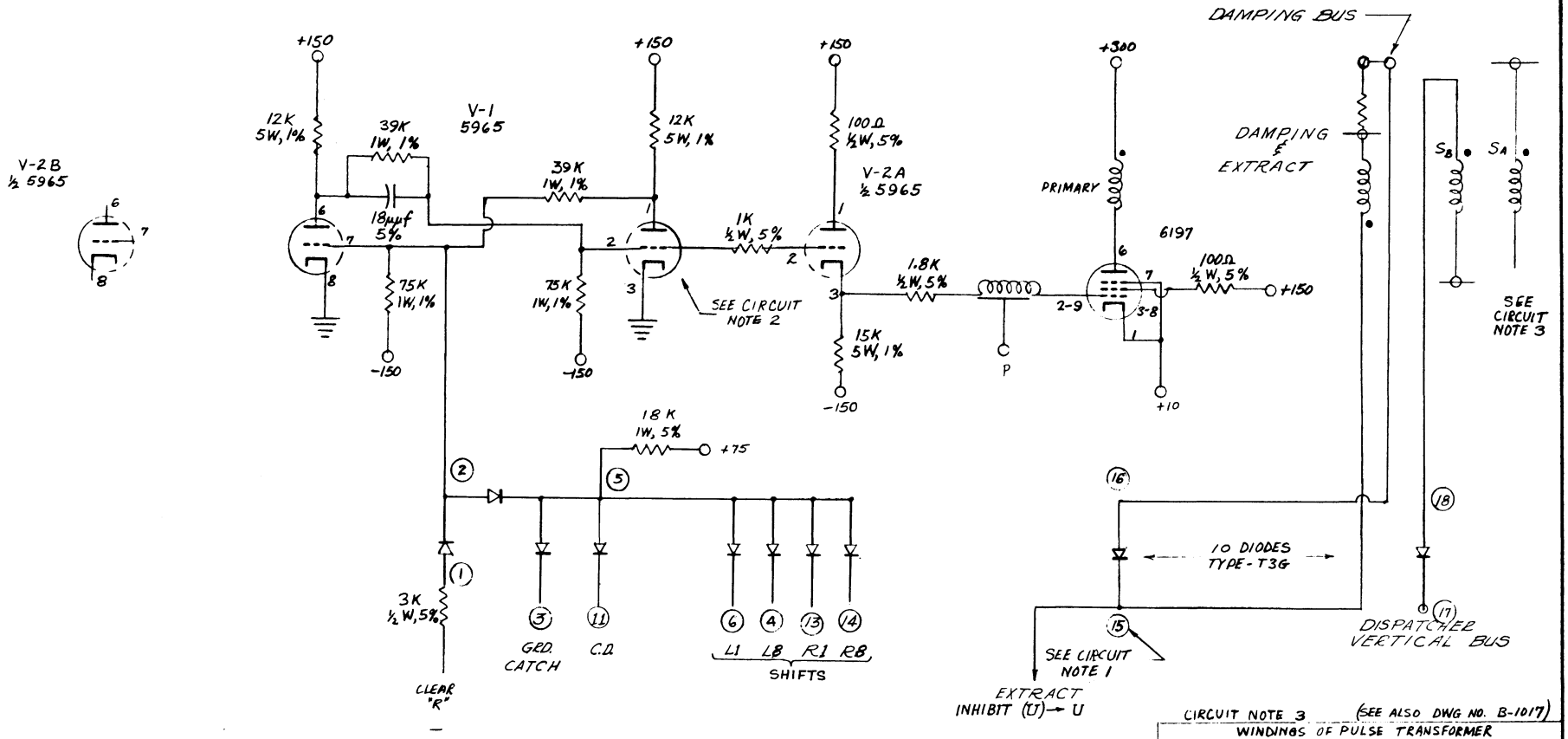
ARITHMETIC UNIT
"S" REGISTER SCHEMATIC

DRAWN BY 2-27-58 JKW | SCALE NONE | MATERIAL

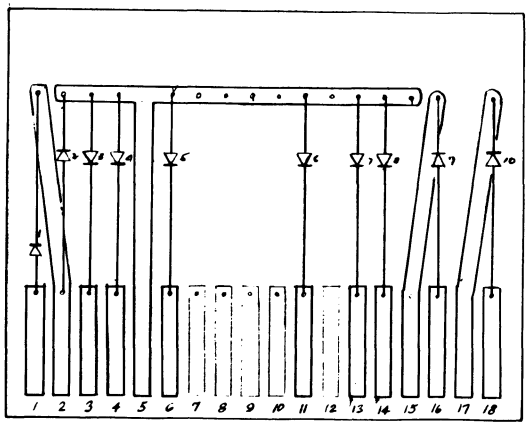
CHE'D | DATE

PART OF 1035 | APP'D

DRAWING NO. C 1037




DIODE BOARD (SHOWN FOR REFERENCE ONLY)
PHYSICAL LAYOUT (SEE DWG. NO. A-1032)




CIRCUIT NOTE: 1- NUMERALS WITHIN CIRCLES INDICATE TERMINALS OF DIODE BOARD.

2- THE STAGE IS CONSIDERED TO BE IN THE "1" STATE WITH VACUUM TUBE V-1, ELEMENTS 1, 2 & 3 CONDUCTING.

3- SEE CHART TO THE RIGHT

4- COLORS OF THE WINDINGS FOR SHIFTS OF 8 FOR EACH STAGE ARE AS FOLLOWS:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| L8 | BROWN | RED | YELLOW | GREEN | BLUE | VIOLET | GREY | WHITE |
| STAGE NO. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| R8 | RED | YELLOW | GREEN | BLUE | VIOLET | GREY | WHITE | BROWN |

CIRCUIT NOTE 3 (SEE ALSO DWG NO. B-1017)

WINDINGS OF PULSE TRANSFORMER

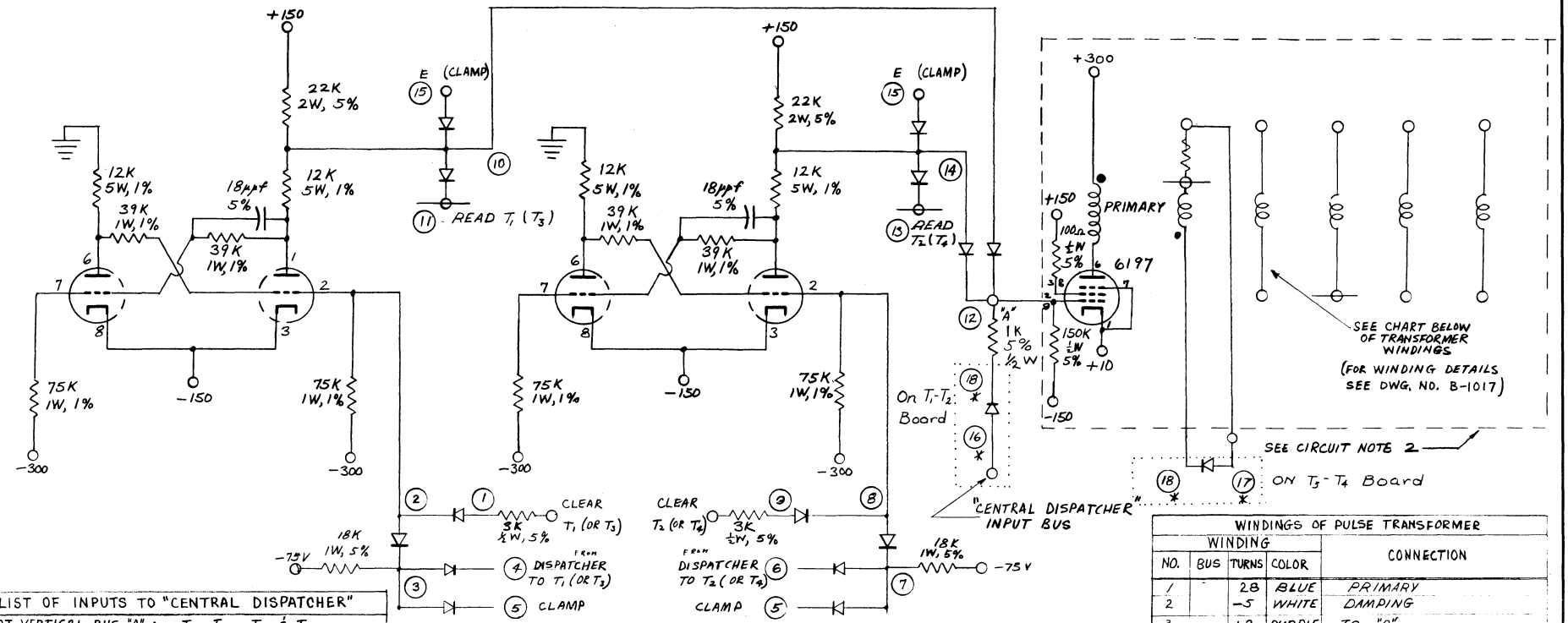
| NO. | WORTZ BUSS | TURNS | COLOR | CONNECTION |
|---|---|---|---|---|
| 1 | | +28 | BLUE | PRIMARY |
| 2 | | -5 | WHITE | DAMPING & EXTRACT |
| 3 | | -2 | RED | TO CENTRAL DISPATCH |
| 4 | | +2 | BLUE | SHIFT L1 |
| 5 | | +2 | SEE NOTE 4 | SHIFT L8 |
| 6 | | +2 | YELLOW | SHIFT R1 |
| 7 | | +2 | SEE NOTE 4 | SHIFT R8 |
| 8 | | +2 | BLACK | EXTRACT [GATE (6)→U] |

| REVISIONS | | | |
|---|---|---|---|
| NO. | DATE | BY | |
| 1 | 5/22/58 | [illegible] | |
| 2 | 9/13/58 | [illegible] | |

RICE INSTITUTE COMPUTER

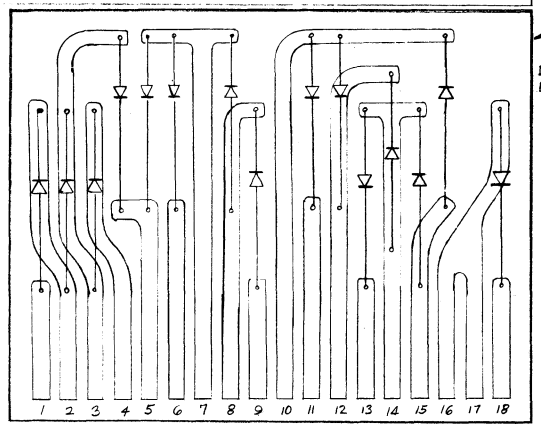
ARITHMETIC UNIT
"R" REGISTER SCHEMATIC

| | | | |
|---|---|---|---|
| DRAWN BY 2-27-58 JKW | SCALE NONE | MATERIAL | |
| CHK'D | DATE | DRAWING NO. C 1022 | |
| PART OF 1020 | APP'D | | |




| WINDINGS OF PULSE TRANSFORMER | | | | |
|---|---|---|---|---|
| WINDING NO. | BUS | TURNS | COLOR | CONNECTION |
| 1 | | 28 | BLUE | PRIMARY |
| 2 | | -5 | WHITE | DAMPING |
| 3 | | +2 | PURPLE | TO "A" |
| 4 | | +2 | GREEN | TO "S" |
| 5 | | +2 | RED | TO "U" |
| 6 | | +2 | GREY | TO $T_1$ |
| 7 | | +2 | YELLOW | TO $T_2$ |
| 8 | | +2 | BROWN | TO $T_3$ |
| 9 | | +2 | BLACK | TO $T_4$ |
| 10 | | +1 | YELLOW | TO MEMORY |
| 11 | | +2 | GREY | TO INSTRUCTION REGISTER |
| 12 | | -1 | BROWN | TO PRINTER |

| LIST OF INPUTS TO "CENTRAL DISPATCHER" |
|---|
| TO SHORT VERTICAL BUS "A": $T_1$, $T_2$, $T_3$ & $T_4$ |
| TO LONG CENTRAL DISPATCHER INPUT BUS: |
| U, S, R |
| CONTROL ADDER |
| MEMORY OUTPUT |




DIODE BOARD PHYSICAL LAYOUT (SHOWN FOR REFERENCE ONLY) (SEE DWG. NO. A-1074)

DIODE SIDE

CIRCUIT NOTE: 1- NUMERALS WITHIN CIRCLES INDICATE TERMINALS OF DIODE BOARD

2- ALL CIRCUIT COMPONENTS WITHIN THE BROKEN LINE BOX ARE COMMON TO DIODE & FLIP-FLOP CHASSIS T(1 & 2) AND T(3 & 4). T(1 & 2) AND T(3 & 4) ARE CONNECTED THRU COMMON POINT "A" OF TUBE 6197. TUBE 6197 ALSO SERVES AS "CENTRAL DISPATCHER". SEE CHART TO THE LEFT FOR LIST OF INPUTS.

3- THE CIRCUIT IS THE SAME FOR T(1 & 2) AND T(3 & 4) EXCEPT FOR "CLEAR" AND INPUTS TO DIODE BOARD INDICATED BY ASTERISK (*).

4- THE STAGE IS CONSIDERED TO BE IN THE "1" STATE WITH ELEMENTS 6-7-8 CONDUCTING

| REVISIONS | | | |
|---|---|---|---|
| NO. | DATE | BY | |
| 1 | 5/22/58 | JKW | |
| 2 | 9/12/58 | RA | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

RICE INSTITUTE COMPUTER

ARITHMETIC UNIT "T" 1-4 REGISTER SCHEMATIC

| DRAWN BY | JKW | SCALE | NONE | MATERIAL |
|---|---|---|---|---|
| CHK'D | | DATE | 3-14-58 | DRAWING NO. C-1067 |
| PART OF | | APP'D | | |

T Registers

These registers are similar to the other arithmetic registers U, S, and R only in the basic flip-flop circuit and in the diode OR input circuitry. They are not used in arithmetic operations, therefore do not have temporary (delay line) storage or gates for shifting or complementing. They are used merely for fast-access storage of four words; their only inputs and outputs are via the central dispatcher.

As shown by Schematic C-1067, the supply voltages for the T flip-flops are 150 volts more negative than the other registers. The plate supply is at ground potential for the convenience of the output gate: a 22 K resistor and a pair of diodes. The CLAMP voltage, connection 15, is at ground potential; the READ bus, connection 11 or 13, is ordinarily at ground except when a 15-volt positive pulse is applied to the proper bus to read a particular one of the T's. The read operation is as follows.

If the T register is in a 0 state, defined as conduction in the right hand triode of the FF, the CLAMP diode conducts about 2 milliamperes and its cathode is at ground potential regardless of whether a positive READ pulse is applied. Thus the central dispatcher pentode remains cut off. If the FF is in a 1 state, the right hand triode is cut off and thus does not pull current down through the CLAMP diode as above. Current

flow for this state is through the READ diode. A 15-volt positive pulse to its cathode allows the grid of the central dispatcher to be driven well into saturation by the current through the 22 K resistor from +150 volts. Thus the central dispatcher is turned on by a 1 AND a READ pulse.

Central Dispatcher

The central dispatcher, or distributor, serves as the output tube for the T registers. As its name implies, it also serves as the route by which information is gated from one part of the computer to another. The savings in components can be illustrated by the following example: if there are N locations which need to be connected to one another by both input and output gates, there will be required 2N(N-1) gate circuits. If instead each location is connected only to a central distributor by input and output, almost the same flexibility results from 2N gate circuits. This is an appreciable simplification, particularly if N is large. Although not all number locations need to be connected to one another, use of the distributor makes possible a more flexible arrangement than otherwise would be feasible. The equivalent N (for the above example) is 10 or so for this computer.

In addition to inputs from $T_1$ through $T_4$, inputs come to the distributor from several other locations via the central dispatcher bus. These locations include U, S, and R in the

arithmetic unit, the address adder in the control unit, and the memory. Each of these inputs is a positive pulse from a transformer secondary, gated onto the dispatcher bus by pulsing a reference bus--the usual gating technique at register outputs.

Each output from the central dispatcher requires a secondary on the pulse transformer. Thus 7 secondaries are required for the arithmetic unit and one each for the instruction register, memory, and high-speed line printer.

To gate from one location to another via the central distributor requires 3 coincident pulses. For example, to gate from R to $T_1$, there must be a READ (R) pulse, a pulse to gate the output of R to the dispatcher bus, and a pulse to the central distributor to gate its output into $T_1$.

Adder

The adder circuitry is based on a combination of two common techniques: the Kirchhoff adder and the diode switch.

As illustration of the principle, consider Figure 9 in which the diodes are represented as switches. In discrete steps these switches control the conductance from the summing point to the minus supply, thereby controlling the potential of the summing point to determine whether the vacuum tube is conducting or cut off. By choosing different values for resistors R and $R_1$, the tube may be made to cut off for any condition of 1, 2, or all three switches closed. An advantage of using

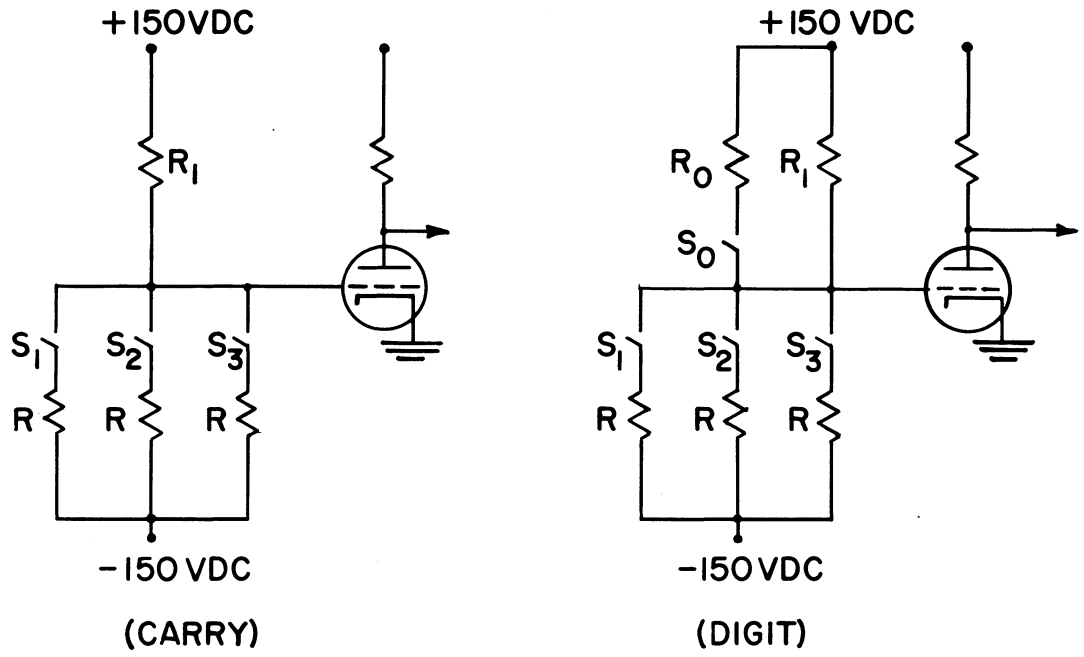



FIGURE 9 KIRCHHOFF ADDERS USING SWITCHES

this technique is that several milliamperes of current are readily switched in, charging to a new voltage the stray capacitance at the summing point.

The triode used in the adder, the 5965, has only about 5 volts grid swing between cutoff and saturation. For this small region of operation it is a convenient and valid approximation to represent the switches as controlling currents of $\frac{150}{R}$ or $\frac{150}{R_1}$ rather than impedances.

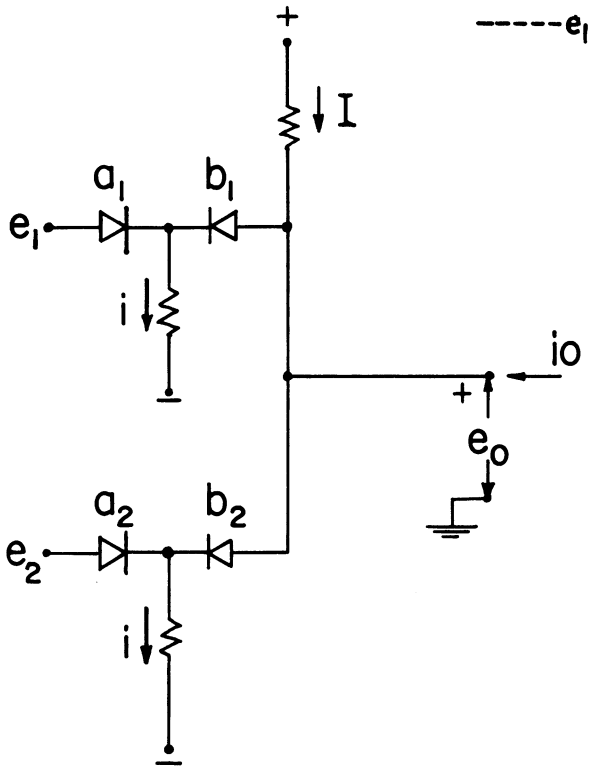
An example of the performance of a circuit of this type is shown in Figure 10. Diodes are assumed to be ideal; for simplicity only two adder elements are shown. The graph shown is the characteristic, output current versus output voltage, which might be obtained by supplying an external current source and measuring the resulting voltage. Conditions for the characteristics are: both inputs at zero; $e_1$ at zero and $e_2$ at -E; and both inputs at -E (merely the first characteristic shifted in voltage). Since usual circuit operation has the output point connected to ground only through a few micro-micro-farads of stray capacitance, the only stable value of d.c. output current is zero. For the values of current $\frac{I}{2} < i < I$, the output remains at $e_0 = 0$ for either or both inputs at zero volts. The operating point for both inputs at -E volts is $e_0 = -E$. By changing the value of the bias current I, this sample circuit can be made to switch operating points for only one input of -E,

| STATES OF DIODES $a_1, a_2, b_1, b_2$ | | |
|---|---|---|
| STATE | CONDUCTING | OPEN |
| A | $a_1\ a_2$ | $b_1\ b_2$ |
| B | $a_1\ b_1\ b_2$ | $a_2$ |
| C | $b_1\ b_2$ | $a_1 a_2$ |

—— $e_1 = e_2 = 0$ : ONLY POINT*1 IS STABLE

--- $e_1 = 0, e_2 = -E$ : ONLY POINT*1 IS STABLE

-----$e_1 = e_2 = -E$ : ONLY POINT*2 IS STABLE



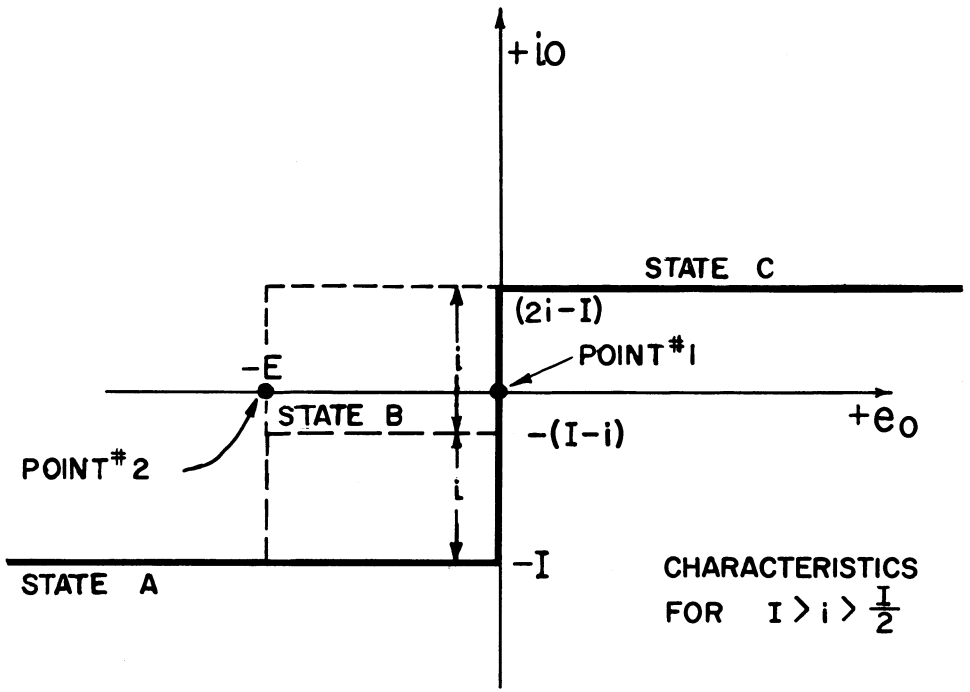
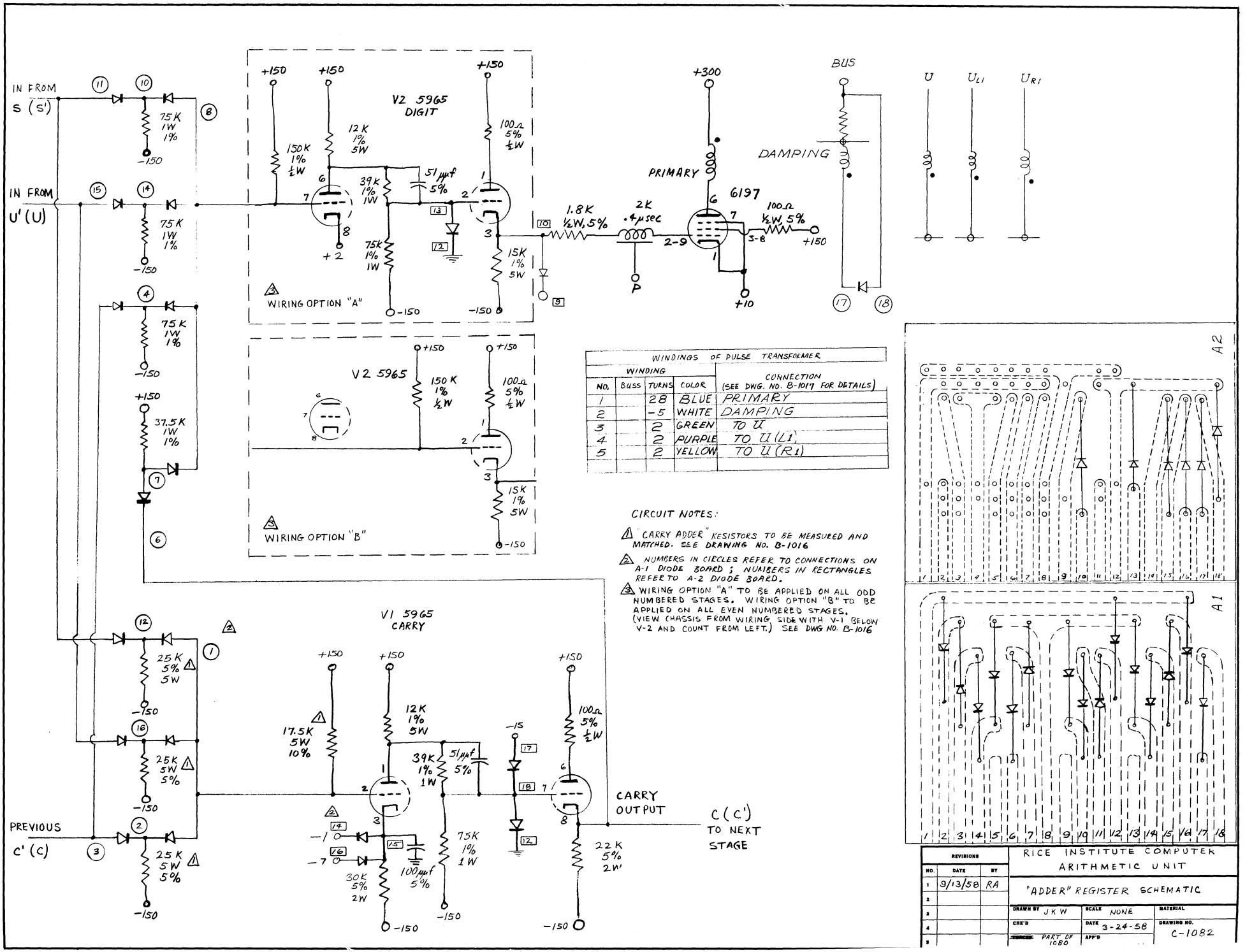




FIGURE 10 A KIRCHHOFF ADDER USING DIODE SWITCHES

or for three or more inputs of -E. This is equivalent to moving the characteristic up or down relative to the origin. The size of the increment of current on either side of the operating point determines the rate of charge of stray capacitance, thus the speed of changing from one operating point to another.

As a review of the purpose of the adder, note that each adder stage has 3 inputs: the S register, the U register, and the carry from the previous adder stage. Each stage has 2 outputs: the digit output and the carry output. These must be, respectively, in the states 0, 1, 0, 1 and 0, 0, 1, 1 for 0, 1, 2, or 3 input 1's. This is apparent from the binary equivalents of numbers 0 through 3.

The carry circuit, see Schematic C-1082, has 9 ma bias current and 6 ma at each input. Input voltages, at either zero volts or -15, are from cathode followers which can supply several milliamperes. If either 2 or 3 inputs are at -15 volts, the output junction is at -15 volts. If 2 (or 3) inputs go to zero volts, the output junction is driven up to zero potential by 3 (or 6) milliamperes. If there is 20 mmf of stray capacitance, it can be charged through the 15-volt transition by 3 ma in 0.1 microsecond. Slight inequalities in voltages, diodes, or resistors may cause unequal distribution of currents, but the switching should not be affected. (Actually, if there are




| WINDINGS OF PULSE TRANSFORMER | | | | |
|---|---|---|---|---|
| WINDING NO. | WINDING BUSS | WINDING TURNS | COLOR | CONNECTION (SEE DWG. NO. B-1017 FOR DETAILS) |
| 1 | | 28 | BLUE | PRIMARY |
| 2 | | -5 | WHITE | DAMPING |
| 3 | | 2 | GREEN | TO U |
| 4 | | 2 | PURPLE | TO U (L1) |
| 5 | | 2 | YELLOW | TO U (R1) |

CIRCUIT NOTES:

1. "CARRY ADDER" RESISTORS TO BE MEASURED AND MATCHED. SEE DRAWING NO. B-1016
2. NUMBERS IN CIRCLES REFER TO CONNECTIONS ON A-1 DIODE BOARD; NUMBERS IN RECTANGLES REFER TO A-2 DIODE BOARD.
3. WIRING OPTION "A" TO BE APPLIED ON ALL ODD NUMBERED STAGES. WIRING OPTION "B" TO BE APPLIED ON ALL EVEN NUMBERED STAGES. (VIEW CHASSIS FROM WIRING SIDE WITH V-1 BELOW V-2 AND COUNT FROM LEFT.) SEE DWG NO. B-1016

| REVISIONS | | | RICE INSTITUTE COMPUTER ARITHMETIC UNIT | | |
|---|---|---|---|---|---|
| NO. | DATE | BY | "ADDER" REGISTER SCHEMATIC | | |
| 1 | 9/13/58 | RA | | | |
| 2 | | | DRAWN BY J K W | SCALE NONE | MATERIAL |
| 3 | | | CHK'D | DATE 3-24-58 | DRAWING NO. C-1082 |
| 4 | | | PART OF 1080 | APP'D | |
| 5 | | | | | |

3 different values of input voltage, the output will correspond to the middle potential.)

Thus the carry circuit has the required response 0, 0, 1, 1 for 0, 1, 2, or 3 input 1's, respectively.

The digit circuit, as indicated by the schematic, has input currents of 2 ma with only 1 ma of bias current. Thus, a single input 1 will cause a change of voltage at the output. For two input 1 conditions the output from the carry circuit switches 4 ma additional bias into the digit circuit, resulting in another 0 output from the digit adder. For three input 1's, the 6 ma input current exceeds the now 5 ma bias to cause an output 1 condition. Thus the digit output has the required 0, 1, 0, 1 for inputs of 0, 1, 2, or 3 1's.

To reduce the "carry" time (since the carry may propagate through several stages) there are 3 circuit considerations: the higher values of switching current described above; only one inversion; and a cathode catch on the adder inverter.

The attenuation in the adder requires voltage amplification in the carry circuit because a particular carry may propagate through several stages. Since a single amplification is more desirable than two, because of the delay, an inversion occurs in every stage: at the input to odd-numbered stages, as shown in the schematic, Drawing C-1082, a 1 corresponds to -15 volts and a 0 corresponds to zero volts; at the carry output

(thus at the input to even-numbered stages) a 1 corresponds to zero volts and a 0 to -15 volts due to the inversion; and at the digit output a 1 always is indicated by zero volts (and a 0 by -15 volts) so that a READ pulse at the delay line shield will turn on the pulse tube. Thus, the output of the digit adder requires inversion only in odd stages. The inputs to odd-numbered stages come from S and the complement of U; inputs to even stages come from U and the complement of S; the whole effect is thus subtraction of the number in S from the number in U.

The time for carry propagation is reduced also by the diode latches on the cathode of the carry inverter. When the grid of the inverter is at zero volts, the cathode conducts through the upper diode. When the adder switches states to -15 volts output, the tube is cut off quickly by the initial 5-volt swing because its cathode is held temporarily at its potential by the cathode capacitor. For the static condition of -15 volts at the grid of the inverter, the tube is cut off because its cathode is caught at -7 volts by the lower diode. As the input is switched toward zero volts, the tube begins to conduct more quickly than if it were grounded. Since the cathode is held temporarily at -7 volts, the initial portion of the transient turns on the tube and thereby switches the output. As the input transient continues to rise, reaching zero

volts, the cathode potential follows until caught by conduction of the upper diode.

The output circuit from the digit portion of the adder register, as indicated in Schematic C-1082, is the same type as registers U, R, and S. The output pulse from the adder can go only to the U register, but can be shifted left or right one stage.

A static output is available through a diode on the same cathode that drives the delay line. This output is used in a logical OR to detect multiple zeros, as described elsewhere in this manual.

# APPENDIX 3

## A SAMPLE ROUTINE

# SAMPLE ROUTINE

The following matrix routine, written in the machine language of the Rice Computer, is intended to illustrate some of the coding features outlined in the manual. The resultant matrix, as described, could be obtained in a number of different ways; the techniques used are not intended to minimize running time or storage space. In fact, more orders have been introduced than are needed in order to illustrate certain features of the order code.

The write-up is similar to that which would be on hand for the use of the coder at the computer site. The five octal triads of the address (M) field are represented symbolically when they refer to an address rather than a number, and each order is assigned a symbolic location in memory.

## MATRIX ADD OR SUBTRACT SUBROUTINE

Purpose: To add a matrix B to a matrix A or to subtract B from A, where both A and B are of size m x n and their elements are floating point numbers.

Method: Before entering this routine, the elements of each matrix must be located sequentially in memory either by rows or by columns; the elements of the resultant matrix C are stored in the same way.

Usage:

Range: $(256^{-31})(2^{-47}) \leq$ element of matrix C $\leq (256^{31})(1-2^{-47})$

Calling sequence:

| location | F1 | F2 | F3 | F4 | comments |
|---|---|---|---|---|---|
| a | 00 | 01000 | 00 | 4000 MAS00 | transfer to MAS |
| a+1 | 00 | 00000 | 00 | 0000← A → | |
| a+2 | 00 | 00000 | 00 | 0000← B → | |
| a+3 | 00 | 00000 | 00 | 0000← C → | |
| a+4 | 00 | 00000 | 00 | 0000← S → | |
| a+5 | 00 | 00000 | 00 | 0000← m → | |
| a+6 | 00 | 00000 | 00 | 0000← n → | |
| a+7 | -- | ----- | -- | ---- ----- | error return |
| a+8 | -- | ----- | -- | ---- ----- | normal return |

where

A = location in memory of first element of matrix A.

B = location in memory of first element of matrix B.

C = location in memory of first element of matrix C.

S = 0 to add B to A,
1 to subtract B from A.

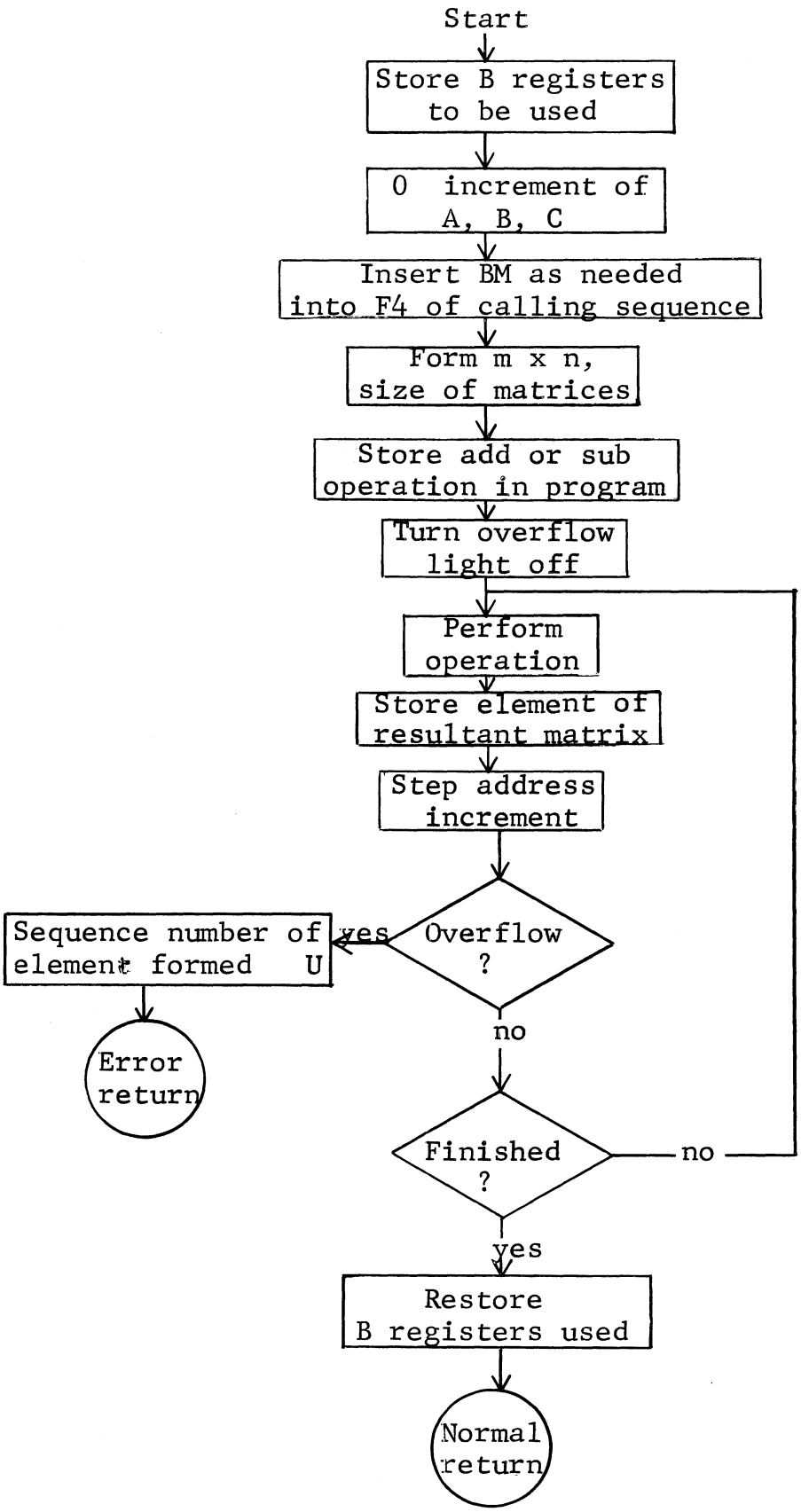
m = number of rows in matrices.

n = number of columns in matrices.

Note: Exit is made to error return if the sum of two elements exceeds the given range, at which time the sequence number of this element is immediately available in the U register.

$B_1$, $B_2$, $B_3$, $B_4$, $B_5$, and $B_6$ are restored to their original status upon the normal return.

The T registers, $T_4$, $T_5$, $T_6$, and $T_7$ are used in this routine.


Start
Store B registers to be used
0 increment of A, B, C
Insert BM as needed into F4 of calling sequence
Form m x n, size of matrices
Store add or sub operation in program
Turn overflow light off
Perform operation
Store element of resultant matrix
Step address increment
Overflow ?
yes
Sequence number of element formed U
Error return
no
Finished ?
no
yes
Restore B registers used
Normal return

Listing:

| location | F1 | F2 | F3 | | F4 | comments |
|---|---|---|---|---|---|---|
| MAS00 | 41 | 21000 | 00 | 4000 | MAS24 | store $B_1$ |
| MAS01 | 42 | 21000 | 00 | 4000 | MAS25 | store $B_2$ |
| MAS02 | 47 | 44100 | 42 | 4000 | 00000 | $0 \rightarrow B_1$; PF1$\rightarrow B_2$ |
| MAS03 | 00 | 51000 | 00 | 0004 | 00000 | fetch A |
| MAS04 | 01 | 10000 | 04 | 0000 | MAS27 | insert BM; store in $T_4$ |
| MAS05 | 00 | 51000 | 00 | 0004 | 00001 | fetch B |
| MAS06 | 01 | 10000 | 05 | 0000 | MAS27 | insert BM; store in $T_5$ |
| MAS07 | 00 | 51000 | 00 | 0004 | 00002 | fetch C |
| MAS08 | 01 | 10000 | 06 | 0000 | MAS28 | insert BM; store in $T_6$ |
| MAS09 | 00 | 51000 | 00 | 0004 | 00004 | fetch M |
| MAS10 | 01 | 10200 | 17 | 0004 | 00005 | m x n$\rightarrow T_7$ |
| MAS11 | 00 | 51000 | 00 | 0004 | 00003 | fetch S |
| MAS12 | 01 | 06060 | 00 | 4000 | 00000 | if odd, skip by one |
| MAS13 | 00 | 51000 | 60 | 0000 | MAS29 | if even, fetch add oper.;skip |
| MAS14 | 00 | 51000 | 00 | 0000 | MAS30 | if odd, fetch sub operation |
| MAS15 | 01 | 20000 | 00 | 4000 | MAS18 | store in operation step |
| MAS16 | 00 | 01300 | 00 | 4000 | MAS17 | turn off exp overflow indic. |
| MAS17 | 00 | 51000 | 00 | 4400 | 00004 | fetch element of matrix A |
| MAS18 | 00 | 00000 | 00 | 0000 | 00000 | add or sub element of matrix B |
| MAS19 | 01 | 20000 | 00 | 4400 | 00006 | store as element of matrixC |
| MAS20 | 00 | 02700 | 61 | 4000 | 00000 | if no overflow,skip;step $B_1$ |
| MAS21 | 41 | 01000 | 00 | 4004 | 00006 | $B_1\rightarrow$U;transfer to error ret. |
| MAS22 | 41 | 02050 | 00 | 0000 | 00007 | if $B_1$=mxn, skip by one |
| MAS23 | 00 | 01000 | 00 | 4000 | MAS17 | loop back |
| MAS24 | 00 | 44100 | 00 | 4000 | 00000 | restore $B_1$ |
| MAS25 | 42 | 44200 | 47 | 4000 | 00000 | $B_2\rightarrow$PF1; restore $B_2$ |
| MAS26 | 00 | 01000 | 00 | 4200 | 00007 | transfer to normal return |
| MAS27 | 00 | 00000 | 00 | 0002 | 00000 | |
| MAS28 | 00 | 00000 | 00 | 4002 | 00000 | |
| MAS29 | 01 | 10400 | 00 | 4400 | 00005 | add operation |
| MAS30 | 01 | 10500 | 00 | 4400 | 00005 | sub operation |

The reader is strongly urged to undertake the exercise of rewriting the foregoing program reducing the number of instructions necessary to accomplish the same task.