BROOKHAVEN NATIONAL LABORATORY

APPLIED MATHEMATICS DIVISION

INTERNAL REPORT

Merlin *

by

Y. Shimamoto

September 1959

* Work peformed under the Auspices of the U.S. Atomic Energy Commission

Acknowledgment

The construction of Merlin was started in the fall of 1956 as a joint project of the Instrumentation Division and the Applied Mathematics Division. W. Higinbotham, M. Graham (now at the Rice Institute), R. Spinrad, M. E. Rose and Y. Shimamoto were involved in the initial planning. Subsequently, G. L. Miller, H. Pate, G. Schwender and R. Kenschaft joined the engineering staff, while D. A. Ravenhall and S. S. Rideout participated in the project for the Applied Mathematics Division.

The original plan was to construct a computer similar to the Los Alamos MANIAC II and we are grateful to Professor N.C. Metropolis, Dr. R. B. Lazarus, Mr. J. H. Richardson, Mr. W. Orvedahl and Mr. W. Spack, Jr., for the many hours they spent to acquaint us with the details of that computer. We have also benefited from numerous discussions with Dr. J. Pasta and Professor A. H. Taub.

Contents

1. INTRODUCTION

This is a provisional report on Merlin, a high speed electronic computer presently under construction at Brookhaven National Laboratory. We shall be concerned primarily with presenting only information related to the operational aspects of Merlin as are needed by a programmer and shall leave out details as to how each operation is performed. It is hoped that the description presented here is sufficient to enable the expert reader to code in the computer language immediately. For the less experienced, however, the present description will serve as an introduction to the description of the Merlin Programming System which will be made available in the very near future. Merlin as described here by no means represents the final computer, for it is anticipated that some minor modifications as well as additions will be made sometime after the computer goes on air.

Merlin is a high speed, general purpose, binary digital computer with a random access electrostatic memory. It operates on fixed or floating point numbers and contains many of the features existing in the Los Alamos MANIAC II, after which is has been modeled. (The reader familiar with the MANIAC II will thus recognize the fact that the nomenclature adopted for the various computer components in MANIAC II are similarly used in Merlin whenever applicable). It operates in a parallel and highly asynchronous mode and thus is not controlled by a clock.

Merlin, like many other similar type computers, is made up of five operational sections as shown in Fig. 1.

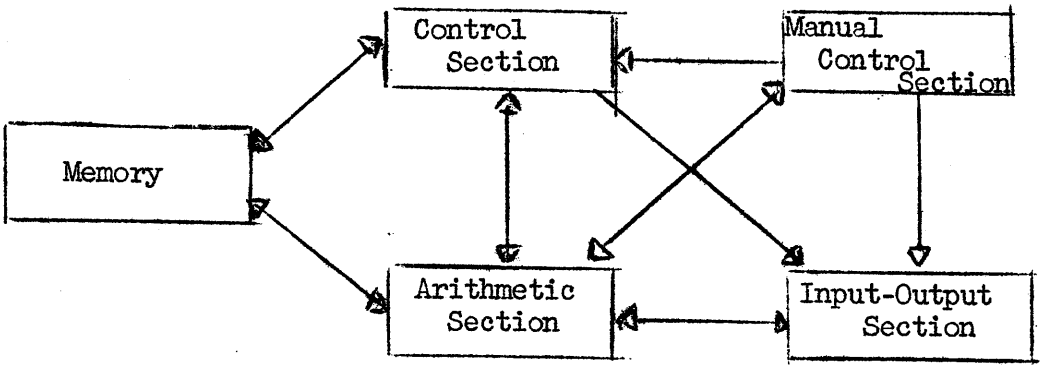


Fig. 1
Functional Section of Merlin

The fast access memory, denoted as "MEMORY" in Fig. 1, may be regarded as the major storage place for all information needed in the computer operation. Initially it will be able to hold 8,192 words, a word consisting of 48 binary digits. The manner in which the information must be presented to Merlin, i.e., the information format, will be described in Section 2. Since the manner in which the memory operates will not be relevant throughout our discussion, its description will be postponed until the last section.

The bulk of Merlin operation takes place in the Arithmetic and Control Sections, and operations which take place in these computer sections are described in Section 3. The understanding of Sections 2 and 3 is essential for the understanding of the remaining sections of this report.

The Merlin Input-output section consists of a paper tape system, a fast on-line printer, and eventually a magnetic tape system. The manner in which these devices are to be used will be described in Section 4.

The Merlin vocabulary, or to be more precise, those instructions which will be available to the user when Merlin goes on air, is described in Section 5.

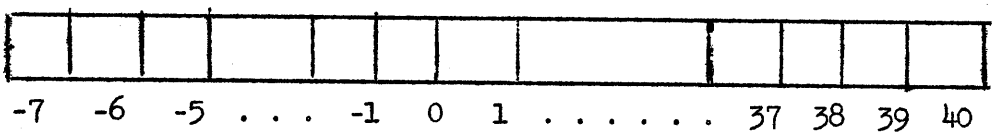
During the design stage of Merlin, it was felt that ample facilities to allow human intervention should be provided to make it useful as a research tool. The means of manually controlling the computer operation finally adopted for Merlin is described in Section 6. For later reference it should be mentioned at this point that a switch which allows the computer operation to be performed either in the automatic or manual mode is provided on the Control panel.

Finally it is still premature to discuss the final operational speed of Merlin. It can nevertheless be said that we expect a speed appropriate to (1) a memory cycle of 10 μsec, (2) a basic add time of 3 μsec, and (3) a shift time of approximately 2 μsec.

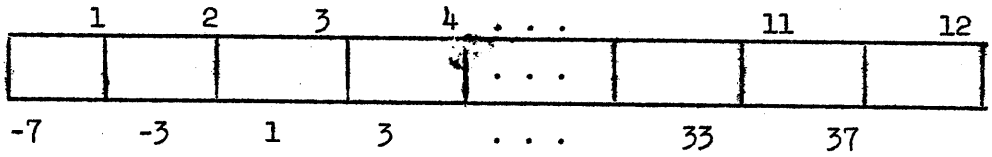
2. Information Format

The basic unit of binary information of fixed length which a binary computer treats collectively is defined as a computer word. In Merlin a word consists of 48 binary digits (or bits). These words will in general be identified by the location in which they are stored, either permanently or temporarily. Letting W for the moment represent one of the locations in which a word may be stored, we shall use the convention of using the parentheses to denote "the word located at W" or the "contents of W" by writing (W).

Each bit (or stage) of W is assigned a number which will be used whenever reference must be made to a specific bit or bits as follows:



Bit number k (k= -7, -6 ..0..40) of a word in W will therefore be denoted by (kW). For ease in handling a machine word outside of the computer, and to facilitate input-output operations, a word may also be broken down into tetrads, i.e., groupings of four bits each, and the position of each tetrad in the word is designated a number as follows:



When a word is expressed in terms of tetrads, the hexadecimal (base 16) notation will be used:

Binary	Hx.	Binary	Hx.
0000	0	1000	8
0001	1	1001	9
0010	2	1010	a
0011	3	1011	b
0100	4	1100	c
0101	5	1101	d
0110	6	1110	e
0111	7	1111	f

Thus, e.g., a word containing a "1" in all 48 bits may be expressed as fffffffffff, while a word containing a "1" in every other bit beginning with "0" in the -7 bit may be expressed as 555555555555. By a Hexadecimal Digit (or Character) we shall mean one of the characters 0 through f expressed in the binary form given above.

In order to accommodate input-output of alpha-numeric characters, a word may also be decomposed into groupings of six bits each (hexads). This mode of expressing a machine word is used only for input-output operations and the reader is referred to Sec. 4 for details.

A word may be an instruction, an operand (logical or numerical), or both, according to the use which is made of it. When a word is regarded as a numerical operand, it is assumed by the computer that the number N is expressed in its binary representation with base $2^8 = 256$, i.e.,

$$N = 2^{8e}\, x \doteq (e,x),$$

with bits -7 through -4 holding the magnitude of the exponent e; bit -3, the sign of e; bit 0, the sign of the fractional part x; and bits 1 through 40, the magnitude of x.

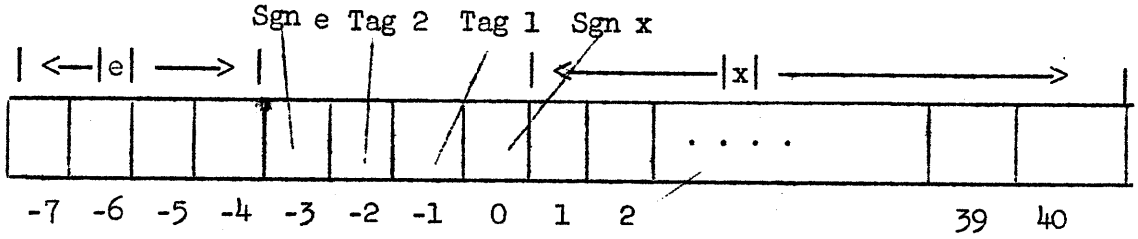


Fig. 2
Arithmetic Information

The binary point is assumed to lie between bits 0 and 1; therefore, $|x|$ lies in the range $0 \leq |x| < 1-2^{-40}$. With the number representation used, the range of a non-zero numeric operand is $2^{-160} \leq N < 2^{120}$ or approximately $4\times10^{-49} < N < 10^{36}$.

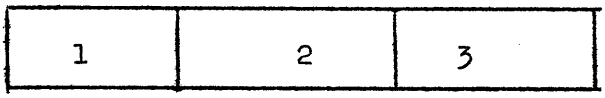
A "0" in the sign bit position is interpreted by the computer to denote a + sign, while a "1" is interpreted as a - sign.

In Merlin a distinction need not be made between fixed or floating

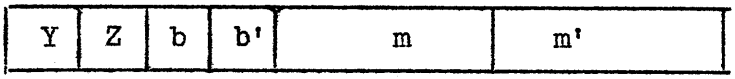
point numbers, although a number with a zero exponent (e = o) may be considered equivalent to a fixed point number. While this implies relinquishing 7 bits which otherwise might have been used to increase the significance of a fixed point number, the resulting uniformity and saving in computer hardware more than compensate the loss of significance.

During the design stage of Merlin, it was felt that the facility to tag or mark a computer word could lead to certain programmatic advantages. Bits -1 and -2 of a numerical operand, referred to as Tag 1 and Tag 2 respectively, are reserved for this purpose. The manner in which Merlin treats a tagged operand will be described in the subsequent section.

Merlin is a one instruction per word computer. An instruction word is divided into groupings of 16 bits each called dioctads.



The contents of each of the second and third dioctads of an instruction may be treated collectively as single entities, while the 4 tetrads comprising dioctad 1 must be treated separately. For reasons which will become obvious later, the notation Y Z b b' m m' will be used throughout to denote the contents of the tetrads and dioctads of an instruction, with the following tetrads or dioctads of a machine word in mind:



The two tetrad combination denoted by YZ corresponds to the order or command part of an instruction and defines for the computer the significance of the remaining tetrads b,b' and dioctads m,m'. Thus, the interpretation of the contents of b,b', m and m' of an instruction word will differ according to the hexadecimal digit assignments of Y and Z. It is sufficient at the present stage of discussion to remark only that a Merlin instruction belongs to one of two classes. Those not making reference to a memory address

belong to Class "0" and are characterized by order tetrads having a "0" in the 2nd bit of the tetrad Z. (i.e., the -2 bit). Instructions making reference to memory belong to Class "1" and will have a "1" in the 2nd bit of Z.

3. Arithmetic and Control Sections

Merlin operates through internally stored programs which, together with the operends, are stored as words in a fast access electrostatic Memory. The memory, whose description is given in Sec. 7, will hold (initially) 8,192 words with an average access time of 6 μ seconds. The instructions comprising the program are generally stored in consecutive locations in the memory, and the proper sequencing as well as the execution of the instructions are performed by the CONTROL Section.

The bulk of the programmed operation is performed either in the Arithmetic Section or the Address Arithmetic Unit of the Control Section. The registers in the former are 48 bits long, while those in the latter are 16 bits long.

3.1 Arithmetic Section

The Arithmetic Section consists of three shifting registers called the R, S, and U Registers, four nonshifting registers called the T1, T2, T3 and T4 Registers, and the 40 bit Magnitude and 4 bit Exponent Magnitude Adders. The magnitude parts (bits 1 through 40) as well as the exponent magnitude parts (bits -7 through -4) of the U and S Registers are statically connected to the Magnitude Adder and Exponent Adder, respectively. Each stage (or bit) of the adder contains the sum of the corresponding bits of U and S and the carry resulting from the preceding stage. The outputs from the adders can be gated, i.e., transferred, to the U Register. Thus the Universal Register, U, acts as the accumulator and receives the result of addition or subtraction. In multiplication it contains the multiplicand and the higher order product; in division the higher order divident and quotient. The Remainder Register, R, holds the low order product and divident, the remainder, and the extract pattern. It can also receive a word directly from the electrostatic memory. The Storage Register, S, in addition to holding the addend and subtrahend, communicates directly with the electrostatic memory and through a six bit buffer called the Feeder Stage with all input-output equipment. It has the facility of giving as an output the 1's complement (or reflection) of its contents; its 2nd dioctad is connected to the Address Arithmetic Unit of the Control Section.

The four nonshifting registers T1, T2, T3 and T4 provide extra

fast (1μs) storage for operands and the results of arithmetic operations; therefore, the operations involving the T registers constitute an integral part of many of the arithmetic instructions. The T registers are addressed by placing their respective numbers (1,2,3, or 4) either in the 4th tetrad of the first dioctad of such instructions, the 2nd dioctad or both. (c.f. Sec. 5) The U, R and S Registers, however, are not addressable in this sense. The gating scheme of the arithmetic registers is shown in Fig. 3.

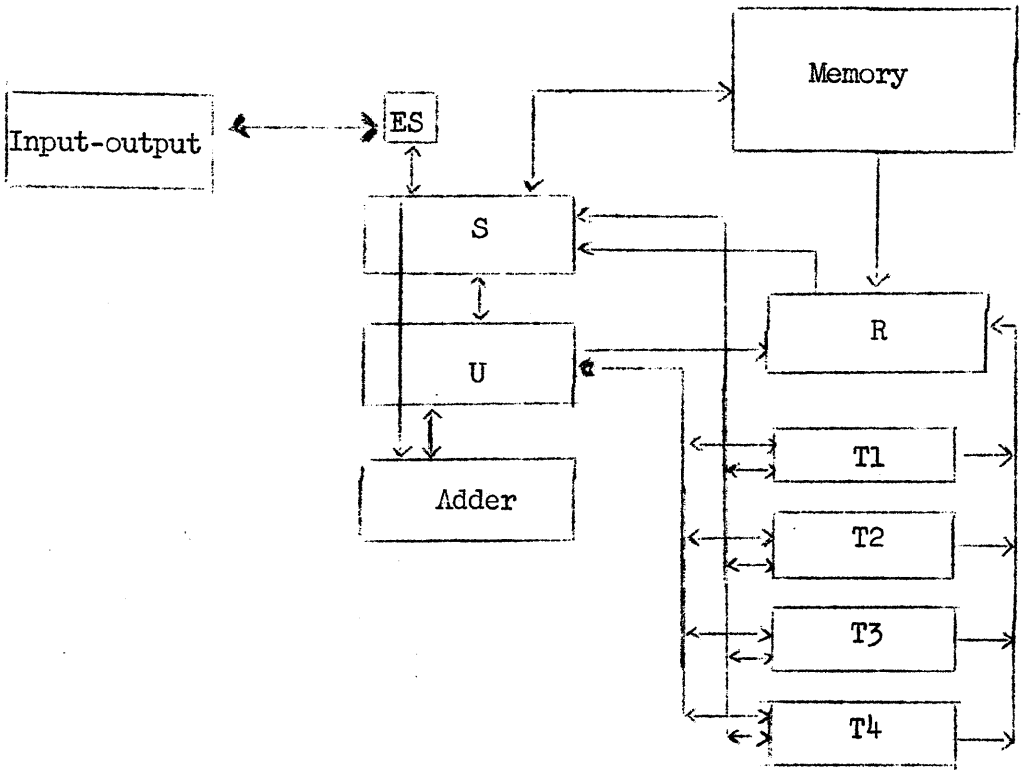


Fig. 3
Gating Scheme of Arithmetic Section

3.2 Address Arithmetic Unit

The Address Arithmetic Unit of the Control Section is centered around the 16 bit Address Adder. Here, unlike the Arithmetic Adder, the inputs are not fixed but must be selected from several possible 16 bit registers by the Control Section, depending on the operation required. All but two of these registers, hereafter referred to as the Short Registers, which can provide inputs to the adder, can also be selected to receive outputs from the adder. The Short Registers are divided into two groups and the Address Adder can accept as inputs only one member from each group at a time. These two groups are referred to as the Upper and Lower groups (See Fig. 4) based on the manner in which the short registers are physically situated with respect to the address adder. The registers in the upper group are I_2, I_3, Control Counter (CC), Manual Address (MA), and Pathfinder (PF). Those in the lower group are B1, B2, B3, B4, B5, B6, Breakpoint Address, (BA), Sense (SN), and Tag Transfer Address (TTA). Note that there is no gating existing from the Address Adder to the Manual Address and the Breakpoint Address. The function of each of the short registers will now be described.

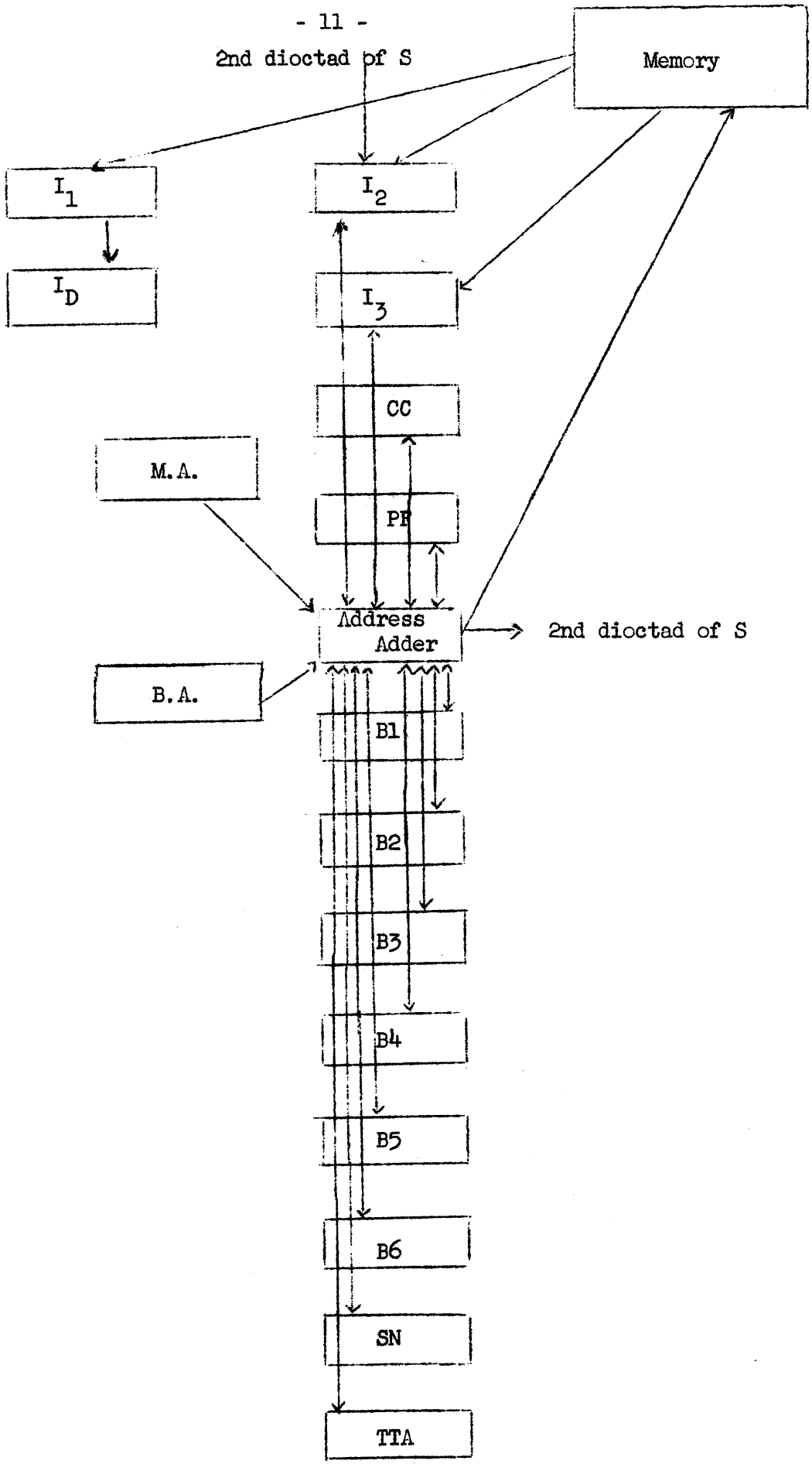


Fig. 4
Gating Scheme of Control Register

I_2 and I_3

The Short Registers I_2 and I_3 together with another Short Register called I_1, which is not connected to the address adder, collectively comprise the Instruction (or I) Register where an internally stored instruction is brought for execution. The subscript j of the registers I_j refers to the three dioctads of an instruction. As will be described in the section on vocabulary, these registers, in general, will contain the following types of information:

(a) Information, such as the order tetrads YZ, which are needed throughout the perform cycle of an instruction, are grouped together in I_1.

(b) That part of the instruction which need not be retained throughout the perform cycle, such as addresses of operands, number of shifts to be performed, etc., are contained in I_2 and I_3.

In Merlin, an instruction to be performed is brought to the I register not upon completion but during the perform cycle of the previous instruction to minimize the instruction fetch time. To achieve this time saving, the contents of I_1, which contains information required by the individual Order Control* throughout a perform cycle, is gated to a 16 bit decoding register, I_D, just before the instruction is executed. The entire I Register is thus free to receive the next instruction in the sequence of the stored program as soon as the contents of I_2 and I_3 have been utilized by the individual order controls.

In the Address Arithmetic Section, only the I Register as a whole or I_2 and I_3, separately, can receive a word or the 2nd or 3rd dioctads

* An order control of an instruction refers to the circuitry provided in the Control Section for the execution of that instruction.

respectively, of a word directly from memory. The loading of the remaining Short Registers, when provided by an instruction, is achieved by bringing the 2nd (or 3rd) dioctad of a word in memory first to I_2 (or I_3). The contents of I_2 (or I_3) are then gated to the Address Adder, which in turn dispatches the information to the specified locations. As indicated in Fig. 4, I_2 can also receive information directly from the 2nd dioctad of the S Register.

B1, B2, B3, B4, B5, B6

Six B Registers, B1, B2, B3, B4, B5, and B6 are provided for automatic address modification of the contents of I_2 and I_3 of all Class 1 instructions. In these instructions, the 3rd and 4th tetrads of I_1 are used to specify the B Registers whose contents are to be added to the number contained in I_2 and I_3, respectively, for the effective memory address. Thus in an instruction involving a memory address with a number 5, say, in the 3rd tetrad of I_1 and a number m in the 2nd dioctad, (--5-m-), the computer will interpret the effective memory address involved in the execution of the instruction to be not m, but X = m + (B5). The B Registers can be loaded only programmatically, and instructions are provided also to store their contents in memory.

Control Counter (CC)

The memory address of the instruction brought to the I Register to be performed is located in the Control Counter. The number contained in the control counter is increased by one when the control section enters the perform cycle of the instruction contained in the I Register. This operation will hereafter be referred to as "counting the CC by 1". Thus, when an instruction from memory address α is being performed, the control counter will read $\alpha + 1$, provided, of course, that this number had not been changed by the instruction itself.

Pathfinder (PF)

The contents of the Control Counter may be changed either programmatically or manually to initiate a new sequencing of instructions beginning with any desired memory address. Any operation which effectively changes the contents of CC, other than that of counting CC by 1 (or by 2 for skip instructions - see vocabulary) will be referred to as a transfer of control. Before a transfer of control is effected, the contents of CC, which is the address of the next instruction which would have been executed had the transfer of control not taken place, is automatically gated via the Address Adder to the Pathfinder for temporary storage. An instruction is provided which inserts the contents of the PF into the 2nd dioctad (corresponding to the address part of a transfer of control instruction) of any word in memory. Thus, having transferred control to a subroutine, one may return to the main routine at the exit point with another transfer of control instruction, provided, of course, that the address part of this instruction contains the contents of the PF.

Sense Register (SN)

The Sense Register (or lights) provide storage for 16 independent bits. Each of the lights may be manually set, independent of the others, to the one (on) or zero (off) position by the operator while Merlin is in operation, and the state of these lights is displayed on the Control Panel. Each or any combination of them may also be set either way programmatically by a single instruction, in which case each of the 16 bits of I_2 is used to address one sense light (the manner of addressing the sense lights is described in Sec. 5).

Tag Transfer Address Register (TTA)

As mentioned earlier, the -2 and -1 bits of a word, when regarded as an operand, may be used as tags or markers. In order to facilitate the use of one of the tag bits, automatic detection of Tag 2 (i.e., the -2 bit) is incorporated in all arithmetic instructions and a few non-arithmetic instructions involving an operand from memory. The order

controls of these instructions are so constructed that, as soon as an operand is brought to the S Register from memory, the state of the -2 bit of S (-2S), is noted and remembered by the control section. Irrespective of the state of the tag bit in S, the computer will proceed to execute the instruction. Upon completion of an instruction involving an operand with tag 2 = 1, however, the computer may proceed in several ways depending on the state of sense lights 15 and 16. If sense light 15 is on, the computer will stop before entering the perform cycle of the next instruction. If sense light 16 is on, the contents of CC will be gated to the Pathfinder and replaced by the contents of the Tag Transfer Address Register (TTA), thus resulting in an automatic transfer of control. If both sense lights 15 and 16 are on, the computer will stop after the transfer of control is effected while, if neither is on, the computer will proceed in the usual manner.

Special instructions are also provided to detect a Tag 1 or Tag 2 condition of a word in U.

The TTA register can be loaded only programmatically.

Manual Address (MA) and Manual Breakpoint (MBP) Registers

In addition to the short registers discussed above, there are two additional 16 bit registers, the Manual Address Register (MA) and the Breakpoint Address Register (BA) which can be set only manually to provide inputs to the Address Adder.

The Manual Address Register can be set with a memory address by means of a set of 16 switches, the Manual Address switches, on the Control Panel (see Sec. 6) to provide multipurpose input to the adder. The gating of the contents of this register to the Address Adder is activated either manually by depressing a specific push button on the control panel or, in one special case, automatically. Except for the two cases described below, which correspond to transfer of control, the CC and PF are not affected by operations involving this register.

One function of the Manual Address Register is the manual setting

of the Control Counter. This is done by first setting the Manual Address Switch with the desired address and then depressing the "CC Manual Load" push button on the Control Panel. The "CC Manual Load" button is operative only when the Operation Switch on the console is in the Manual position. When it is depressed, the contents of the control counter (the old address) is first stored in the Pathfinder, the contents of MA is gated to CC via the address adder, and the computer proceeds to fetch the newly addressed instruction from memory and places it in the I Register for execution.

Merlin is provided with both programmatic and manual breakpoints. Bit 1 of all instructions is reserved for the former case, and an instruction being performed with a "1" in this bit position will be regarded by the Control Section as being breakpointed. For the latter case, a set of 16 switches called the "BRKPT ADDRESS" is provided on the Control Panel for setting up the address of the instruction that is to be breakpointed manually. When the instruction located at the address specified by the Manual Breakpoint switches is brought to the I Register, the control will regard this instruction as being breakpointed.

There are two 3-way Breakpoint switches, one for each type of breakpoint, on the Control Panel allowing the operator to instruct the control section as to the desired course of the computer operation upon completion of a breakpointed instruction. If the switch corresponding to the type of breakpoint is in the position marked "IGNORE", the computer will proceed as though the instruction were not breakpointed, If set in the position marked "STOP", the computer will stop and await human intervention. The 3rd position marked "TRANSFER", will bring about an automatic transfer of control to the address set in the Manual Address Register. It should be noted that there is only one Manual Address Register for the two types of breakpoint transfers. There are two important exceptions to the rule for breakpoint transfer as described above:

(1) A breakpoint transfer on all unconditional and successful conditional transfer of control instructions will be ignored by the computer.

(2) Breakpoint transfers on instructions involving a tagged operand resulting in a tag-transfer-of-control will be ignored.

3.3 Computer Control Conditions

Exponent Spill and Overflow

As discussed above, the normal sequencing of instructions from consecutive memory addresses may be altered not only by operations involving the interaction of special registers and switches, (as, e.g., manual breakpoint transfer) but also by transfer of control instructions. The transfer of control via an instruction may be accomplished conditionally or unconditionally. In the former case, the condition may be either the presence or absence of certain information in a computer word contained in the U register, or the state of certain internal switches which were set up automatically during the perform cycle of certain instructions. There are two such internal switches in Merlin, namely, the Overflow Switch and Positive Exponent Spill Switch. Their state can be interrogated programmatically only by conditional transfer of control instructions.

The Overflow Switch (OVFL) is automatically turned on whenever the magnitude of the result of a fixed point operation, including round, is greater than one, or when a "1" is lost during certain left shift operations (c.f. Sec. 5). A light marked "OVFL" on the Control Panel is turned on at the same time. This switch (and light), if in the "on" position, may be turned off programmatically by the "transfer of control on overflow" instruction. A manual means of turning off the overflow switch is also provided on the Control Panel.

The Positive Exponent Spill switch (EXP SPILL +) is turned on whenever the exponent of a number becomes greater than 15 in all floating point operations. A light marked"EXP SPILL + " on the control panel is turned on at the same time. The "EXP SPILL + "

switch and light may be turned off programmatically by the "transfer of control on positive exponent spill" instruction or manually by a push button on the console.

It is clear from the foregoing discussion that the setting of the special registers (such as the manual breakpoint registers) or switches (such as the overflow switch), defines a computer state (or condition) which may result in operations not explicitly asked for by the different individual instructions. These different special conditions have the common property of persisting through several perform cycles until changed manually and/or programmatically and, when called into play, of affecting the sequencing of the stored program or stopping the computer. Note that loading of the B Registers, Pathfinder, Tag Transfer Address Register and Sense Register (bits 1 through 14) are excluded from our definition of setting computer conditions, since the programmatic operations involving them constitute an integral part of an instruction. To complete our discussion on computer conditions we must also mention the "Negative Exponent Spill Switch" and the "Do Mode Switch".

Negative Exponent Spill (EXP SPILL -)

The situation which arises when the exponent of a result in any floating point operation becomes less than -15 is defined as Negative Exponent Spill (EXP SPILL -). Merlin treats negative exponent spill differently from the case of positive exponent spill. A switch marked "EXP SPILL -" is provided on the operator's console to set the computer either in the "Allow" state or "Stop" state. When negative exponent spill occur, irrespective of the state of the "EXP SPILL -", the computer will automatically reset the exponent and magnitude parts of the number involved to -15 and 0, respectively. With the switch in the "Stop" position however, Merlin will automatically stop after the perform cycle of the instruction resulting in negative exponent spill. Note that the function of the "EXP SPILL -" is analogous to the 3-position breakpoint switches or to sense lights 15 and 16 in that it serves to place the computer in certain "states of readiness".

Do Mode Switch

A special instruction called the "Do" instruction, which allows a programmer to have any instruction following it to be performed any desired number of times in succession, is contained in the Merlin vocabulary. When the "Do" instruction is executed Merlin is said to be in the Do Mode of operation and an internal "Do Mode switch" is turned on. Since the "Do" mode affects the operation of fetching the next instruction, it must be regarded as a computer condition. A light marked "DO MODE" on the Control Panel is also turned on when the computer enters the do mode. The "DO MODE" light is turned off automatically when Merlin returns to its normal mode of operation.

3.4 Control Priorities

Clearly, situations will arise during the course of computer operation where more than one of the "transfer conditions" will be satisfied while an instruction is being performed. The course taken by the computer in such cases can be summarized as follows:

(1) When the condition for automatic tag transfer is satisfied:

and

(a) the condition for breakpoint transfer is also satisfied:

The tag transfer is effected and the breakpoint transfer is ignored.

(b) the instruction is in the "Do Mode":

The tag transfer is effected and the condition for do mode is cleared by the computer.

(2) When the breakpoint transfer condition is satisfied by an instruction in the "Do Mode":

The breakpoint transfer is effected
at the end of the "Do Mode" and not
after each perform cycle of that instruction.

We also have the following situation involving special orders.

(3) The instruction to be performed is a "Do" order

and

(a) the breakpoint transfer condition is satisfied.
The breakpoint transfer is ignored,
except when the number of "Do's"
indicated is zero in which case the
transfer of control is effected.

(b) the computer is already in the "Do" mode:
Let us assume that the first "Do"
instruction was in memory location α,
and the second in $\alpha+1$. The computer
will then stop with the contents of
memory location $\alpha+2$ (and not $\alpha+1$) in
the I-Register.

(4) The instruction is an unconditional or successful conditional transfer of control order,

and

(a) condition for breakpoint transfer is also satisfied:
The breakpoint transfer is ignored.

(b) the computer is in the "Do Mode":
The transfer of control is effected, and
the computer comes to a halt.

All instructions involving skips (c.f. Sec. 5) are treated differently from transfer of control instructions in that the contents of the control counter are not saved in the Pathfinder for these orders. Thus the considerations which apply to all other normal instructions, i.e., those not covered by the special situations discussed above, are equally valid for skip instructions.

4. Input-Output Section

The Merlin input-output section consists of a paper tape system, a high speed on-line printer, and a control console typewriter. A magnetic tape system is presently being designed and punch cards may be considered in the future. Although the control typewriter may be used programmatically, it is anticipated that its use will be restricted to program debugging and manual intervention. Therefore the present section will be devoted primarily to the descriptions of the paper tape system and fast printer, and the discussion on the use of the console typewriter will be postponed until the section on Manual Control. A description of the magnetic tape will be made available at some future date.

4.1 Paper tape system

Input-output operations involving the paper tape system (and the console typewriter) make use of the S Register via a six bit buffer called the feeder stage. A paper tape reader is used for the input operation, while a paper tape punch is uded for the input operation. The gating scheme which exists in Merlin for these operations is illustrated in Figure 5.

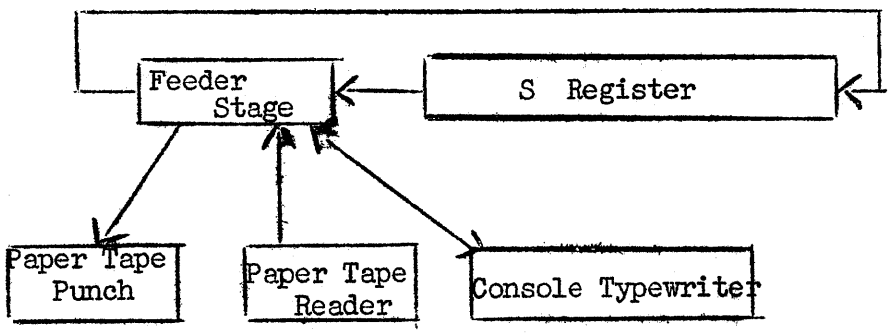
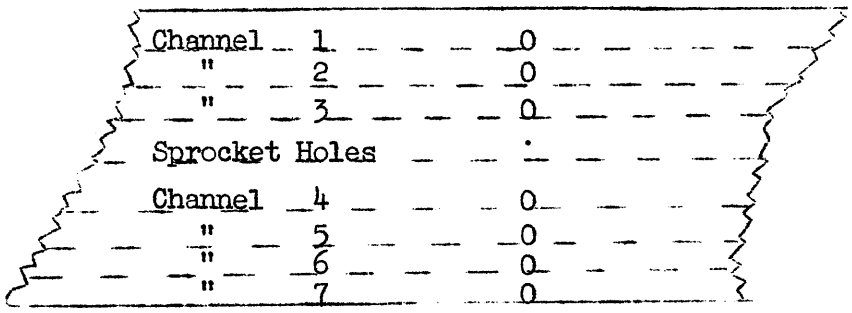


Fig. 5
Gating Scheme of Paper Tape System and Console Typewriter

Both the paper tape reader and punch can handle only six bits of information (called a hexad or a character) at a time. Thus paper tape input-output operations take place serially, one character at a time. During input operation, all six bits or the rightmost four bits of a character arriving at the feeder stage are shifted left into the six or four rightmost bits of the S Register, respectively, as indicated in Fig. 5. As will be explained below the four bit shift takes place during manual load of paper tape to facilitate input operation of information expressed in terms of the hexadecimal characters. During output operation, the leftmost six bits of the S Register are shifted into the feeder stage to be punched.

Paper Tape

The paper tape reader and punch use 7/8 inch paper tape. Seven hole positions called channels are spread across the breadth of the tape as shown below.



The various hole combinations which are punched on channels one through six will be called character codes. A hole is regarded as a binary "1".

A character code may be punched on a paper tape by depressing one of the keys of the tape preparation typewriter. Alternatively, a punched paper tape may be fed into the typewriter for a printed output, in which case each character code selects the same key which was used to give it as an output during the punching process to be activated. Thus each character code may be identified either by a printed character or a typewriter function such as tab, carriage

return, etc. In addition, three character codes are given the names, "End of word", "End of Block" and "Delete" codes since they are sometimes used for the purpose of controlling paper tape input operations.

Channel seven of the paper tape is used for parity information, which is checked in paper tape read operations. It is punched whenever the six character-channels contain an even number of holes or none at all. Since tape prepared by automatic data recording systems may not contain parity information, the computer control panel has been equipped with a Parity Switch which can be set so that parity checking is inhibited, making the contents of channel seven of no consequence.

The character codes (with the parity bit) adopted for Merlin are listed in Table I. The six-channel character codes listed there are also used for the console typewriter. It is to be noted that the tape preparation equipment punches only 49 of the possible 63 six-channel combinations, and utilizes a zero which contains no punching in channels one through six and parity punching in channel seven.

When parity information is not punched along with the character, zero becomes a problem. Without the parity hole, zero appears as a blank and difficulty arises in distinguishing it from an empty space on the tape. Data recording systems which do not punch parity information must, therefore, provide a special character code for identifying true zeros.

Tape is prepared on a Flexowriter which is a heavy duty electric typewriter equipped with a paper tape reader and punch. Figure 6 shows the layout of the keyboard and control switches. Besides manual tape punching, these machines permit reading and printing a paper tape or reading and duplicating one. Printing always accompanies manual or automatic punching.

In all but one case, the function of each Flexowriter control switch is explained by its caption. The "Conditional Stop" switch permits a conditional halt of the reading operation each time an End of Block code(:) is read. When the switch is depressed the halt takes place. The end of block character is printed whether or not a halt occurs.

Table I

Character Codes for Paper Tape System and Console Typewriter

CHARACTER CODE	CHARACTERS		CHARACTER CODE	CHARACTERS	
7654321	L.C.	U.C.	7654321	L.C.	U.C.
1000000	0	)	0011010	q	Q
0000001	1	->	1011011	r	R
0000010	2	<-	0011100	s	S
1000011	3	↑	1011101	t	T
0000100	4	↓	1011110	u	U
1000101	5	=	0011111	v	V
1000110	6	|	0100000	w	W
0000111	7	/	1100001	x	X
0001000	8	*	1100010	y	Y
1001001	9	(	0100011	z	Z
1001010	a	Λ	0101010	+	+
0001011	b	B	1101011	-	-
1001100	c	C	0101100	,	,
0001101	d	D	1101101	.	<
0001110	e	E	0101111	√	≤
1001111	f	F	1110101	space	
0010000	g	G	0111000	Lower Case	
1010001	h	H	1111001	Upper Case	
1010010	i	I	1111010	Tab	
0010011	j	J	0111011	C.R.	
1010100	k	K	1111100	; (End of word)	;
0010101	l	L	1111101	: (End of block)	:
0010110	m	M	1111111	delete	
1010111	n	N			
1011000	o	θ			
0011001	p	P			

L.C. Lower Case

U.C. Upper Case

Figure 6. Merlin Flexowriter Keyboard.

PAPER TAPE READER

A Ferranti model TR-2 photo-electric paper tape reader reads the seven channels paper tape at a speed of 200 characters per second without halting tape motion between characters. Power is removed from the reader motor whenever tape motion ceases for ten seconds or more. Power is restored when the next read operation commences.

Tape sprocket holes are sensed and used to control read timing. Each hole indicates a character and initiates a read cycle which transmits photo-electrically sensed character information to the feeder stage and parity information to the control section. The cycle is complete when the reader circuitry has been prepared for the next character.

Merlin has been supplied with two methods for entering information from paper tape. The first is the Read Hexad instruction which causes one six bit character to be read from tape and stored alone in a specified memory location. The instruction will proceed in one of two ways depending on the setting of the parity switch:

(a) Parity Switch in Normal Position

All characters entering the feeder stage undergo a parity check, and the result is recorded by the order control. The contents of the feeder stage are then detected by the control for the "delete" and "end of block" codes. If the character is a "Delete", the feeder stage awaits the arrival of the next character for processing; otherwise the character is shifted into the S Register in the manner described above for a memory store. The "end of block" code is detected because it is used to stop the paper tape reader.

Upon completion of the memory store, the instruction control will skip one instruction if the character has passed the parity check. Otherwise the parity failure light on the control panel is turned on and the next instruction in sequence of the stored program will be performed.

(b) Parity Switch in Ignore Position

All characters entering the feeder stage are accepted for transmittal to the rightmost bits of the S Register for a memory store. Upon completion of this store, the computer will skip one instruction.

Since 5 milleseconds are required between two successive hexad reads, while the average multiply time of Merlin is expected to be approximately 130 μsec., clearly one can utilize the time between reads for computation. In the event the next hexad should arrive at the feeder stage during a between-read-computation, a light marked "Hexad Lost" on the control panel (with a manual reset button) is turned on and both the computer and paper tape reader are automatically stopped.

The second method of paper tape input to Merlin is the Manual Load operation which has been provided to simplify initial loading of machine language routines and programs. Manual Load is a manually initiated input operation in which successive groups of paper tape characters (assumed to be hexadecimal characters) are stored in consecutive memory locations starting from the one whose address is set in the control panel "Manual Address" switches. For this operation the control panel "Operation" switch must be in the "Manual" position and, regardless of the state of the parity switch, all characters entering the feeder stage undergo parity check. The operation is initiated by depressing the "Manual Load" button on the control panel.

During manual load, the "end of word" and "end of block" codes are used for control purposes and are not regarded as part of the input information. All other character codes (except, of course, the "delete" code) are treated as tetrad characters. Thus as they arrive at the feeder stage, the entire feeder stage is shifted left by two bits first to discard the two unwanted bits before the remaining tetrad information is shifted into the rightmost tetrad of the S Register. The entire S Register may thus be loaded from the right with up to twelve hexadecimal characters. Each loading process is

terminated on the arrival of the "end of word" code in the feeder stage, at which time the contents of the entire S Register are stored in memory and S is cleared. The "end of word" code which terminates the last character group in the paper tape must be followed by an "end of block" code which is used to stop the paper tape reader.

Should a parity failure occur during manual load, the paper tape reader is stopped before the character involved is shifted into the S Register and the parity failure indicator on the control panel is turned on. The memory address in which the contents of the S Register are to be stored will be given in the second dioctad of the I Register which is displayed in the control panel.

PAPER TAPE PUNCH

Paper tape output is achieved through the use of a 60 character per second teletype motorized punch and a Punch Hexad instruction. During execution of this instruction, the entire contents of the addressed memory location is read into S where only the six leftmost bits are shifted into the Feeder Stage. Once the output character is in the Feeder Stage, the control section determines whether a parity bit must be punched to create odd-parity. The character along with parity information is then transmitted to the paper tape punch and recorded in the output tape.

4.2 Fast Printer

The Fast Printer is an on-line device which provides printed outputs under the control of an internally stored program. It prints one line of characters at a time, a line being either 48 or 96 characters long. There are 64 different characters available in the Merlin fast printer, and they are listed in Table II.

The speed of the fast printer is achieved through the nature of the printing mechanism and the practice of printing all characters

appearing on a single line in a parallel manner in one print cycle rather than in the serial manner of a typewriter. This parallel mode necessitates the use of 96 printing mechanisms, one for each character position (column) on the printed line. Each mechanism consists of a rotating print wheel bearing 64 characters of type on its periphery and a solenoid actuated hammer which strikes the paper and ink ribbon against the type. Characters are selected by timing the action of the hammer to coincide with the arrival of the desired character at the printing position opposite the hammer. The 64 characters which comprise each circle of type on the print drum occur in the order given in Table II. The character "zero" has been established as the reference point for the start of each print cycle and is the first character to pass the print hammer following the start of execution of a print instruction.

For the selection of a character as well as the column of a print line on which that character is to be printed, control information called a print matrix must be comstructed in memory. For a 48 character print, the print matrix consists of either 32 or 64 consecutive words. The relative position of each word in the print matrix serves to select a character occupying the corresponding position in the print wheel, beginning with the character "zero". Thus there is a one-to-one correspondence between the characters in the print wheel and words in the print matrix. The 32 word print matrix is to be used when all of the characters desired in a print cycle occupies the first half, i.e., "0" through the character "," of the print wheel.

The 48 bits of a word in the print matrix governs the selection of the column on which the character corresponding to the position of the word in the print matrix is to be printed. For this purpose the bit positions are placed in a one-to-one correspondence with the columns of the printed line, beginning with the (-7) bit of the print matrix word which addresses the leftmost or first column of

Table II (a)

Print Characters on First Half of Print-Wheel

	Symbol	Sequential Position	Binary Code	Comments
	0	0	00 0000	
	1	1	00 0001	
	2	2	00 0010	
	3	3	00 0011	
	4	4	00 0100	hexadecimal
	5	5	00 0101	characters
	6	6	00 0110	
1st	7	7	00 0111	
half	8	8	00 1000	
circum-	9	9	00 1001	
ference	a	10	00 1010	
	b	11	00 1011	
	c	12	00 1100	
	d	13	00 1101	
	e	14	00 1110	
	f	15	00 1111	
	+	16	01 0000	
	-	17	01 0001	
	/	18	01 0010	
	.	19	01 0011	plotting symbols
	.	20	01 0100	
	x	21	01 0101	
	x	22	01 0110	
	△	23	01 0111	
	△	24	01 1000	
	*	25	01 1001	
	|	26	01 1010	
	(	27	01 1011	
	)	28	01 1100	
	✗	29	01 1101	multiplication symbol
	=	30	01 1110	
	,	31	01 1111	

Table IV (b)

Print Characters on Second Half of Print Wheel

	Symbol	Sequential Position	Binary Code	Comments
	A	32	10 0000	Upper case alphabet
	B	33	10 0001	
	C	34	10 0010	
	D	35	10 0011	
	E	36	10 0100	
	F	37	10 0101	
	G	38	10 0110	
	H	39	10 0111	
2nd	I	40	10 1000	
half	J	41	10 1001	
circum-	K	42	10 1010	
ference	L	43	10 1011	
	M	44	10 1100	
	N	45	10 1101	
	O	46	10 1110	
	P	47	10 1111	
	Q	48	11 0000	
	R	49	11 0001	
	S	50	11 0010	
	T	51	11 0011	
	U	52	11 0100	
	V	53	11 0101	
	W	54	11 0110	
	X	55	11 0111	
	Y	56	11 1000	
	Z	57	11 1001	
	<	58	11 1010	
	≤	59	11 1011	
	↑	60	11 1100	
	←	61	11 1101	
	→	62	11 1110	
	↓	63	11 1111	

the printed line. A "1" in any of the 48 bits of a print matrix word addresses the column of the printed line on which the character corresponding to the word is to be printed. As each row of characters of the print wheel goes by for a print, the corresponding word in this print matrix is consulted in the S Register.

If a 96 character print is desired, a second print matrix addressing the 49th through 98th columns of a print line must be prepared. The corresponding words of this second matrix are consulted in the R Register at the same time the words of the first matrix are consulted in the S Register.

Four print instructions, Long Print-Full, Long Print-Half, Short Print-Full and Short Print-Half provide a choice of half or full line printing and a selection of 32 or 64 distinct characters.

The Short Print instructions are used when only characters on the first half periphery of the drum (0,1,2,3...a,b...f.....=,') are required. Printing is completed during the first half revolution of the drum and the paper is spaced during the second. Because all activity surrounding the printing of a line is completed in one revolution, the maximum printing rate of 600 lines per minute is achieved.

Long Print instructions make available the full set of 64 characters. With these instructions the print cycle occupies an entire drum revolution necessitating the use of the following revolution for paper spacing. Since one complete revolution of the print drum is required per printed line, the printing rate is only half that of the Short Print or 300 lines per minute.

The Short Print-Half and Long Print-Half instructions permit printing on only the left half of a line, i.e., columns 1-48. These instructions have been included to simplify programming when 48 or fewer columns can satisfy output requirements. Only one print matrix is required whose starting address must be specified in the instruction.

The two instructions Short Print-Full and Long Print-Full provide for printing in any or all of the 96 columns of the line. Two print matrices are required with this instruction. The address of the first row of the "S" matrix as well as the address of the first

row of the "R" matrix must now be specified with these instructions. The manner in which two memory addresses may be specified in a single instruction will be described in the next section.

5. Order Structure and Vocabulary

5.1 Order Structure

In this section the order structure and current listing of Merlin instructions will be discussed. It is of course anticipated that additional instructions, dictated by experience and utility, will become available as time goes on.

The convention of using the parentheses to denote "the contents of" or "the information stored at", as described previously, will be used in the following discussion. Thus we shall mean, e.g. by (U), the word in the U Register, or by (MU) the magnitude part of a number in the U Register. Furthermore, subscripts will be used to distinguish the three dioctads of a word. Thus a word in memory location X, say, which is denoted by (X), may also be expressed as $(X_1\ X_2\ X_3)$.

A Merlin instruction (excluding magnetic tape instructions) belongs to one of two classes depending on whether or not it makes reference to a memory address for an operand. Those which require a memory address are said to belong to Class 1 and are distinguished from those which do not, or Class 0 instructions, by the presence of a "1" in the -2 bit position. (It will be recalled that the -2 bit of a word is also the tag 2 position. This fact permits automatic detection of Class 1 instructions in the Arithmetic Section).

Whenever possible, instructions which have some property in common are grouped together. These groupings are identified by one of the hexadecimal digits 9 through f in the first tetrad (bits -7 to -4) of an instruction as follows:

1st tetrad

9 : Input-output instructions (non-magnetic tape)

A : Arithmetic instructions (add and multiply)

B : B Registers and Sense Register instructions

C : Transfer of Control instructions

D : Divide

E : Operand modification instruction

F : Logical instructions

It should be mentioned that communication type instructions involving fetching and storing of operands have not been grouped together and are regarded as belonging to several of the groups listed above. Orders related to the use of magnetic tape will be incorporated at a later date.

Instructions within a group are distinguished by a hexadecimal digit in the 2nd tetrad position. From the definition of an instruction class, each group can thus have up to eight Class 0 instructions and up to eight Class 1 instructions. The first two tetrads of an instruction thus define the order (operation) desired, although more information such as the location of the operand, etc., must still be specified. These two tetrads will be referred to as the order tetrads of an instruction, and the letters YZ will be used whenever reference must be made to them.

The significance of the information contained in the other bit positions of an instruction will differ and depend on the contents of the order tetrads with the exception of the breakpoint bit discussed in Sec. 3 and bit 5 which is reserved for a possible future use. In general several bits will be required for a specific control information. Thus the required control information will be specified in bits (2-4), bits (6-8), the second dioctad, or the third dioctad. Since there is no danger of ambiguity, we shall, for ease in exposition, group the breakpoint bit with bits (2-4) and the unused bit 5 with bits (6-8) and refer to them collectively as the third and fourth tetrads respectively.

Each of the control information tetrads and dioctads may give information independently, or in conjunction with one of the others. Thus for example, in a class 1 instruction involving two memory addresses, the third (fourth) tetrad is used to address the B-Register whose contents are to be added to the number in the second (third) dioctad for the effective first (second) memory address. We may express this fact by saying that these instructions have an order structure of the form XY b b' m m'. Similarly other class 1 instructions involving only one memory address may be expressed as X Y b - m - where the dash is used to indicate the fact that the information contained in the corresponding tetrad or dioctad of the instruction is ignored by Merlin.

In addition to the two possible interpretations of the control information tetrads and dioctads denoted in the above example by (- - b - m -) and (- - - b' - m'), seven others are currently used. Each of them will be distinguished from the others by the introduction of distinct symbols in the appropriate tetrads or dioctads, as were b, b', m, m' in the above example, in the vocabulary. Before discussing these basic interpretation schemes and the notation used to express them in the vocabulary, it may perhaps be worth while to mention that every Merlin instruction assumes one of the following forms:

Class 1: X Y b b' m m'

X Y b - m -

X Y b N m -

X Y b B m -

Class 0: X Y - - - -

X Y - L' L -

X Y - B α -

X Y - B α β

X Y - - α -

X Y - - L -

X Y - - T -

X Y - L - -

The symbols appearing above are to be interpreted in the following manner in the vocabulary:

(1) - - b - m -

All Class 1 instructions will appear with this combination which is to be interpreted as follows: If the hexadecimal digit 0 (or 8 in the case of a breakpointed instruction)* appears in the 3rd tetrad, i.e., b = 0, the contents of the 2nd dioctad, m, will be regarded by the Control Section as representing an absolute memory address. If b assumes any one of the hexadecimal digits 1 through 6 (or 9 through e for breakpointed instructions), the Control Section will add the contents of index register Bb to m to obtain an effective memory address, X = m + (Bb). Thus the allowed hexadecimal characters in the 3rd tetrad for this case are restricted to 0 through 6 (or 8 through e). There are no restrictions on the values of m. (See, however, the section on memory).

(2) - - - b' - m'

This combination is used in certain class 1 instructions which

* The case of program breakpointed instructions will be treated separately only in this paragraph. In the remaining paragraphs, our discussion will be based on cases only where bit 1 is a zero.

require a second memory address. As in the above case, we have $X' = m' + (Bb')$ with obvious interpretation.

(3) - - - L - -

(4) - - - - L -

The L notation appears in certain class 0 instructions to specify the U Register, if the tetrad or dioctad indicated by L contains an "0", or one of the four T Registers, if they contain the digits 1 through 4, i.e., Register Ti for L = i. Thus the range of L is 0 through 4. An L appearing in the fourth tetrad serves to address a register to be used for storage (i.e., a receiver) while an L appearing in the second dioctad addresses a register for an operand. A prime, e.g., in the combination - - - L' L - is used to emphasize the fact that two different T Registers may be used in the operation.

(5) - - - b N m -

Certain class 1 instructions not requiring a second memory address are expressed with an N instead of a b' (and m' ignored). In these instructions, the function of the information in the fourth tetrad is to specify either the U or one of the four T Registers, in which case the rule above for the case of - - - L - - is applicable, or to specify the memory address X = (b) + m as a storage place for an operand. In the latter case, the hexadecimal character 5 must appear in the fourth tetrad. Thus the range of N is 0 through 5.

(6) - - - - T -

A T appearing in the 2nd dioctad of class 0 instructions indicates that one of the four T Registers must be specified. The situation here is analogous to - - - - L - discussed above, except that the use of the character 0 which is allowed in - - - L - to specify the U Register is now prohibited.

(7) - - - B - -

Whereas the b' appearing in a class 1 instruction specifies the B Register whose content is to be added to the number in m', a B appearing in the fourth tetrad of either class 0 or class 1 instructions

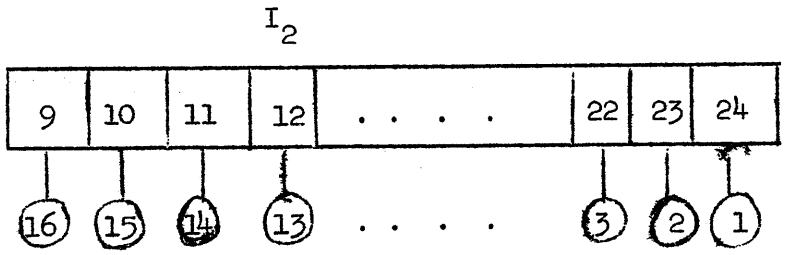
addresses the B Registers involved in the operation specified by the order tetrads X Y.

(8) - - - - α -

(9) - - - - - β

A small Greek letter appearing in the 2nd or 3rd dioctads denotes operands in the case of instructions involving the B registers, or number of operations to be performed in skip or shift instructions. The allowed range of α or β are given with each of the instructions in the vocabulary.

In all instructions involving the sense register, each of the 16 bits of the 2nd dioctad are placed in correspondence with one of the sense lights, and each of the sense lights is addressed by placing a "1" in the bit of 2nd dioctad corresponding to it.



More than one sense light may thus be addressed simultaneously. The configuration in the second dioctad used to address the sense lights will be denoted also by α.

Finally, Merlin will regard as "illegal" all those order tetrads listed in the vocabulary in which the control information tetrads and dioctads are left as blanks. On entering the perform cycle of these "illegal instructions", the computer will automatically stop without giving any special indication of the nature of the stoppage. Legal stops take place at the end of a perform cycle and an appropriate indication is given on the control panel.

5.2 Vocabulary

A listing of Merlin instructions is presented here with a brief description of each group. In the comment column of the listing the following abbreviations are used:

tag : tag 2 is tested
es : exponent spill is possible
mo : magnitude overflow is possible

9 GROUP

Instruction	Description
90 - - - -	Halt
91	
92	
93 - - - -	Stop paper tape feed on reader
94 b - m -	Paper tape read. Read 1 hexad character into the rightmost 6 bits of X. Skip next instruction unless parity failure occurs.
95 b - m -	Paper tape punch. Punch 1 hexad character from the leftmost 6 bits of X.
96	
97	
98 - - - -	Flexowrite, tetrad mode. Type the contents of S on console typewriter in tetrad mode.
99 - - - -	Flexowrite, hexad mode. Type the contents of S on console typewriter in hexad mode.
9a	
9b - - - -	Advance paper one row on fast printer.
9c b - m -	Short Print-Half. Print cols. 1 → 48 of fast printer, using 32 word print matrix starting at X.
9d b b' m m'	Short Print-Full. Prints cols. 1 → 96 of fast printer, using two 32 word print matrices starting at X and X'.
9e b - m -	Long Print-Half. Uses 64 word print matrix starting at X.
9f b b' m m'	Long Print-Full. Uses two 64 word print matrices starting at X and X'.

A Group

The A Group includes all addition and subtraction (both fixed and floating point) and multiplication instructions. In this group (U) is always one of the operands. Since initial normalization is not necessary in multiplication, a distinction between fixed and floating point multiplication is not required. In orders a8 through af the magnitude of R, (MR), is regarded as an extension of the magnitude of U, (MU). The combined magnitudes are represented by UR and the final exponent in U (EU) will also appear in ER, the exponent part of R. If, during a normalization operation, the 8 most-significant bits of MU contain only 0's, UR is shifted left 8 bit positions and the exponent parts EU and ER are decreased by 1. In the round operation, the most significant bit of MR is checked and if it contains a "1", a "1" is added to the least significant bit position of MU. The subscripts n and r are used to indicate normalize and round operations.

A Group

Instruction YZ b b' m m'	Description	Comment
a0 - 1' L -	(U) + (L) → U and L', fixed point	mo
a1 - L' L -	(U) + (L) → U and L', floating point	es
a2 - L' L -	(U) - (L) → U and L', fixed point	mo
a3 - L' L -	(U) - (L) → U and L', floating point	es
a4 b N m -	(U) + (X) → U and N, fixed point	mo,tag
a5 b N m -	(U) + (X) → U and N, floating point	es, tag
a6 b N m -	(U) - (X) → U and N, fixed point	mo, tag
a7 b N m -	(U) - (X) → U and N, floating point	es, tag
a8 - L' L -	(U) . (L) → UR' (U) → L'	es
a9 - L' L -	(U) . $(L)_n$ → UR, (U) → L'	es
aa - L' L -	(U) . $(L)_r$ → UR, (U) → L'	es
ab - L' L -	(U) . $(L)_{n+r}$ → UR, (U) → L'	es, mo
ac b N m -	(U) . (X) → UR, (U) → N	es, tag
ad b N m -	(U) . $(X)_n$ → UR, (U) → N	es, tag
ae b N m -	(U) . $(X)_r$ → UR, (U) → N	es, tag
af b N m -	(U) . $(X)_{n+r}$ → UR, (U) → N	es,mo, tag

B Group

The B Group includes all instructions which address the B-Registers, involve the sense registers or affect the contents of the tag transfer register. This group contains, in addition, two instructions which provide for the transfer of information between memory and the U register. (U_2) or (X_2) refers to the contents of the 2nd dioctad of the U register or memory word at address X = m + (Bb). Note, in particular, that the order "bc" enables the loading of U from memory without interrogating the tag-2 bit.

B GROUP

Instruction	Description
b0 - B α -	$\alpha \rightarrow B$
b1 - B α -	$\alpha + (B) \rightarrow B$
b2 - B α β	$\alpha + (B) \rightarrow B$; then skip the next instruction if $(B) \geq \beta$
b3 - B - -	$(U_2) \rightarrow B$
b4 b B m -	$(X_2) \rightarrow (B)$
b5 b B m -	$(X_2) + (B) \rightarrow B$
b6 b B m -	$(X_2) + (B) \rightarrow B$; then skip the next instruction if $(B) \geq (X_3)$
b7 b B m -	$(B) \rightarrow X_2$
b8 - - α -	Set all Sense Lights addressed by α to zero (each sense light is addressed by a one in the corresponding bit position in α)
b9 - - α -	Set all Sense Lights addressed by α to one
ba - - α -	Skip the next instruction if all Sense Lights addressed by α contain ones
bb - - α -	Skip the next α instructions
bc b - m -	$(X) \rightarrow U$
bd b - m -	$(X_2) \rightarrow$ Tag Transfer Address Register
be b - m -	$(U) \rightarrow X$
bf b - m -	$(A) \rightarrow X_2$

C Group

Transfer of control instructions make up the class 1 members of this group. Of the eight class 0 members, four are special orders with obvious meanings and four are related either to tagging or untagging an operand in the U register.

C GROUP

Instruction	Description	Comments
c0 - - - -	$\sqrt{\lvert U \rvert}$ -> U	[redacted]
c1 - - - -	Normalize (UR)	es
c2 - - - -	Round (U)	mo
c3 - - - -	Normalize (UR) and Round	es, mo
c4 b - m -	Transfer control to X, unconditional	
c5 b - m -	Transfer control to X on positive, i.e., if (0U) = 0	
c6 b - m -	Transfer control to X on overflow, i.e., if the overflow light is on	
c7 b - m -	Transfer control to X on zero magnitude, i.e., if (MU) = 0	
c8 - - - -	0 -> Tag 1 bit of U	
c9 - - - -	0 -> Tag 2 bit of U	
ca - - - -	1 -> Tag 1 bit of U	
cb - - - -	1 -> Tag 2 bit of U	
cc b - m -	Transfer control to X on positive exponent spill, i.e. if exponent spill light is on	
cd b - m -	Transfer control to X on Tag 1, i.e., if (-1U) = 1	
ce b - m -	Transfer control to X on Tag 2, i.e. if (-2U) = 1	
cf b - m -	Transfer control to X on logical zero, i.e. if (U) = 0	

D Group

The instructions in this group are concerned with division, the movement of operands from R and S to memory and the fast access registers, and the movement of 16 bit (address length) operands from PF, U_2 and U_3 to memory. An instruction for loading R from memory is also included. In the divide orders a long numerator, i.e. UR, is used. The rounded quotient will appear in U, the unrounded quotient in S, and the remainder in R. In a floating division requiring "normalization", the long numerator is shifted right, eight bits at a time, until its magnitude part become less than the magnitude part of the denominator, i.e. (MS).

D GROUP

Instruction	Description	Comments
d0 - - T -	(UR) ÷ (T), fixed point	mo
d1 - - T -	(UR) ÷ (T), floating point	es, mo
d2 - L - -	(S) → L	
d3 - L - -	(R) → L	
d4 b - m -	(UR) ÷ (X), fixed point	mo, tag
d5 b - m -	(UR) ÷ (X), floating point	es, mo tag
d6 b - m -	(S) → X	
d7 b - m -	(R) → X	
d8		
d9		
da		
db		
dc b - m -	(PF) → X_2	
dd b - m -	(U_2) → X_2	
de b - m -	(U_3) → X_3	
df b - m -	(X) → R	

E Group

This group consists of operand-modifying instructions, several two memory address instructions and the "Do" instruction, eb. In the cumulative multiply order, ed, the product is always rounded before the sum is taken. The Do order operates α times (mod 256). When $\alpha = 0$, the instruction following the Do order is skipped.

E GROUP

Instruction	Description	Comments
e0 - L' L -	(L) → U and L'	
eL - L' L -	|(L)| → U and L'	
e2 - L' L -	-(L) → U and L'	
e3 - L' L -	- |(L)| → U and L'	
e4 b N m -	(X) → U and N	tag
e5 b N m -	|(X)| → U and N	tag
e6 b N m -	- (X) → U and N	tag
e7 b N m -	- |(X)| → U and N	tag
e8 - - L -	(L) → R	
e9		
ea		
eb - - α -	Do the next instruction α times, (mod 256) increasing the effective left and right address by (B6) and (B5) respectively, after each perform	
ec b b' m m'	(X) → X'	tag
ed b b' m m'	(X) . (X') + (T4) → T 4	es, tag
ee		
ef		

F Group

The logical instructions (shift, extract, complement, etc.) of Merlin are contained in the F Group. In the shift instructions, which are described diagrammatically in Fig. 7, only the 7 right-most bits of α are used (mod 128). In the extract order those bits of U which correspond to the bit positions of R which contain a "1" are replaced by the corresponding bits of X. For instance, if bit 9 of R contains a 1 then bit 9 of U will be replaced with bit 9 of X.

The remaining logical orders may be summarized as follows

Complement:

$$1 \rightarrow 0,\ 0 \rightarrow 1.$$

Union:

$$0 \cup 0 = 0,\ 1 \cup 0 = 0 \cup 1 = 1 \cup 1 = 1.$$

Intersection:

$$0 \cap 0 = 1 \cap 0 = 0 \cap 1 = 0,\ 1 \cap 1 = 1.$$

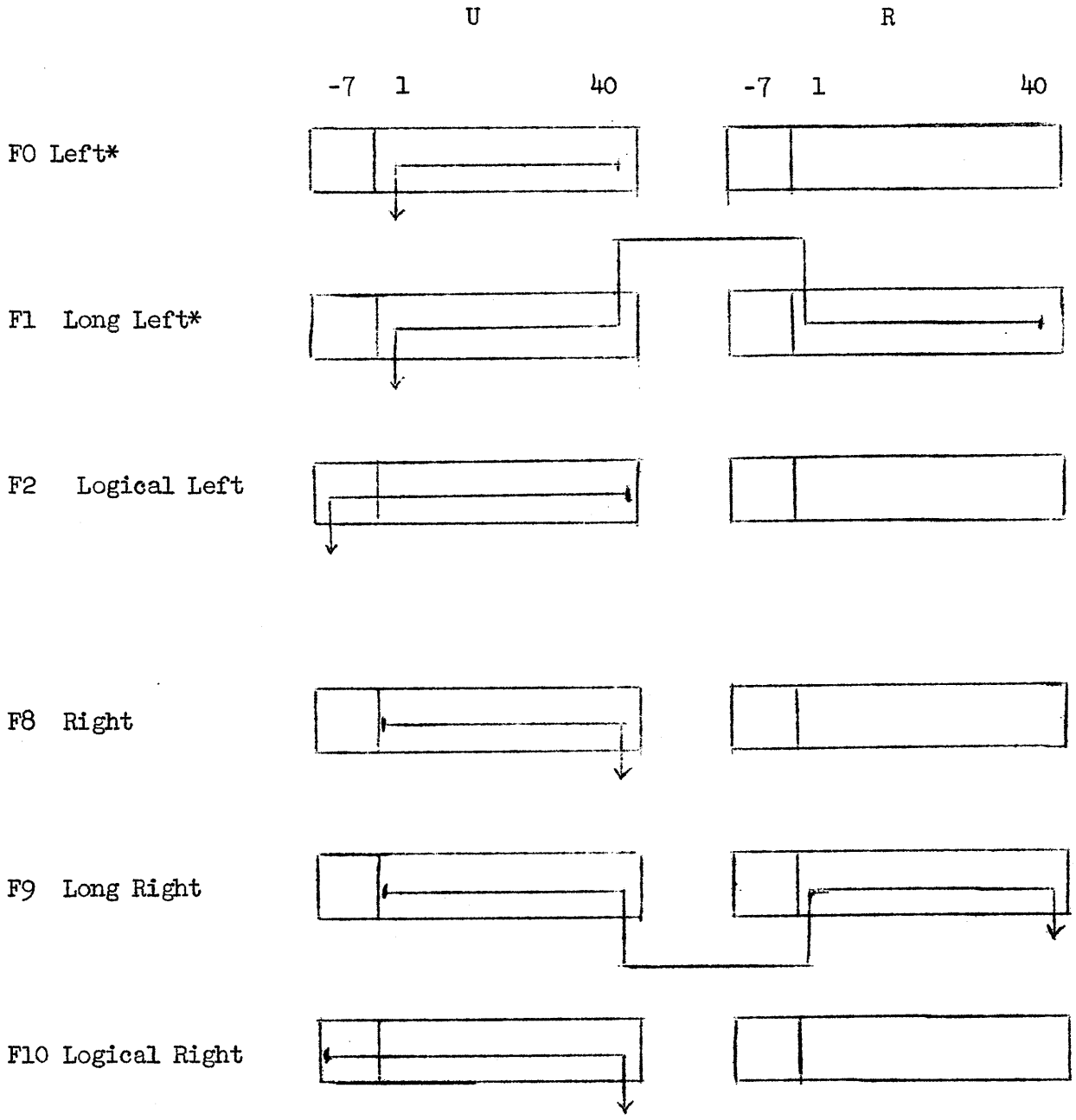
Symmetric difference:

$$0 \oplus 0 = 1 \oplus 1 = 0,\quad 1 \oplus 0 = 0 \oplus 1 = 1.$$

If we denote corresponding bits of U, R and X by u, r, and x and a complement by a prime, the extract order may be expressed as

$$(u \cap r') \oplus (r \cap s) \rightarrow u.$$

Fig. 7
Shift Orders (Mod 128)



*: can result in overflow

F GROUP

Instruction	Description	Comments
f0 - - α -	Left shift (MU) α bits (mod 128);	mo
f1 - - α -	Long left shift (MUR) α bits (mod 128);	mo
f2 - - α -	Logical left shift α bits (mod 128)	
f3 - B - -	Shift (1-40U) to left until (1U) ≠ 0, counting B by 1 after each shift	
f4 b - m -	Union (U) of (U) and (X)	
f5 b - m -	Extract	
f6 b - m -	Symmetric difference ($\oplus$) of (U) and (X)	
f7 b - m -	Intersection ($\cap$) of (U) and (X)	
f8 - - α -	Right shift (MU) α bits (mod 128)	
f9 - - α -	Long right shift (MUR) α bits (mod 128)	
fa - - α -	Logical right shift (U) α bits (mod 128)	
fb - - - -	Complement (U)	
fc b - m -	Complement (X) → U	
fd		
fe		
ff		

6. Manual Control

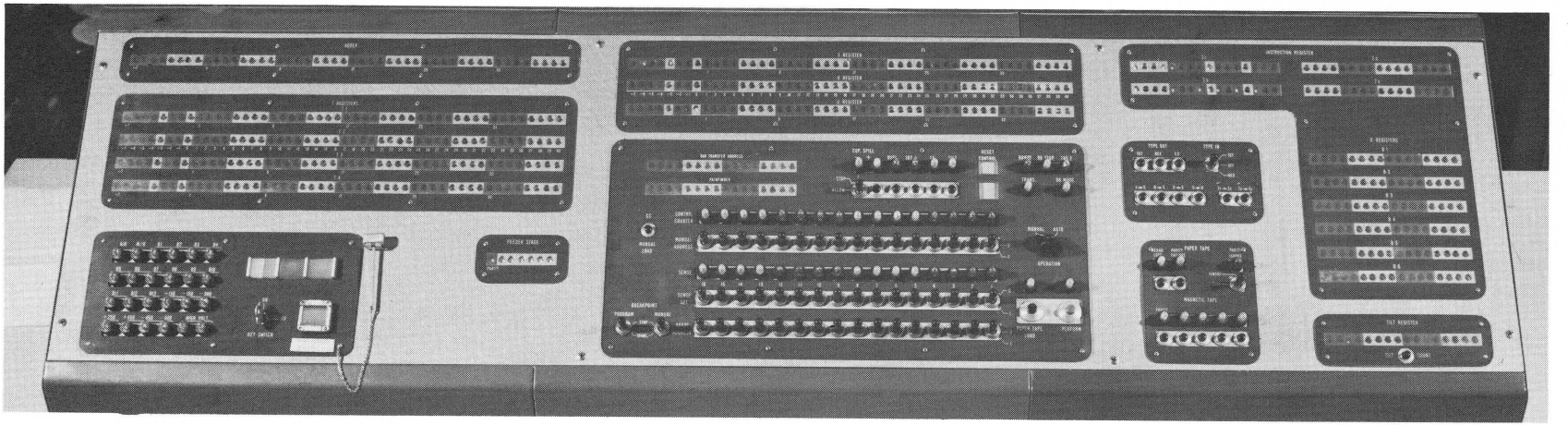
Manual access to Merlin can be gained via the Control Console which consists of the Control Console Panel and the Console Typewriter.

6.1 Control Console Panel

The Control Panel (see Figs. 8, 9, 10, 11) is made up of display lights as well as various switches and push buttons. The contents of the individual arithmetic registers as well as the short registers of the address arithmetic unit are always exhibited on the display lights. Some of the computer conditions are also displayed on the panel by individual lights, and some of these single display lights may be turned off manuall by the operator. The switches are used to set up computer conditions or to provide manual inputs to some of the registers, while the push buttons are used to initiate computer operations which can only be performed in the manual mode. The console is also connected to the paper tape reader.

The section on the panel marked "S Registers", "R Registers", "U Registers", "Adder", "T Registers", "PF", "TTA", "Feeder Stage", all denote display lights. In the center section of the panel, (Fig. 10), the 16-light rows indicated by "Control Counter" and "Sense" also denote display lights. The rows indicated by "Manual Address", "Sense Set" and "Brkpt Address" correspond to the respective switches already mentioned in Sec. 3. To the left of the breakpoint switches are located the two three-position toggle switches for the program and manual breakpoints. To the left of the "Control Counter" lights is found the "CC Manual Load" button.

To the right of the "Control Counter" lights is found the Automatic-Manual Switch. When this switch is in the "Automatic" position, the computer will automatically step through the internally stored program, stopping only when one of the stop conditions in the computer is satisfied. When the switch is in the "Manual" position the computer will stop after each perform cycle, i.e. just before performing the instruction contained in the "I" Register. This instruction is displayed by the rows



SECTION A SECTION B SECTION C

Figure 8. Merlin Control Panel.

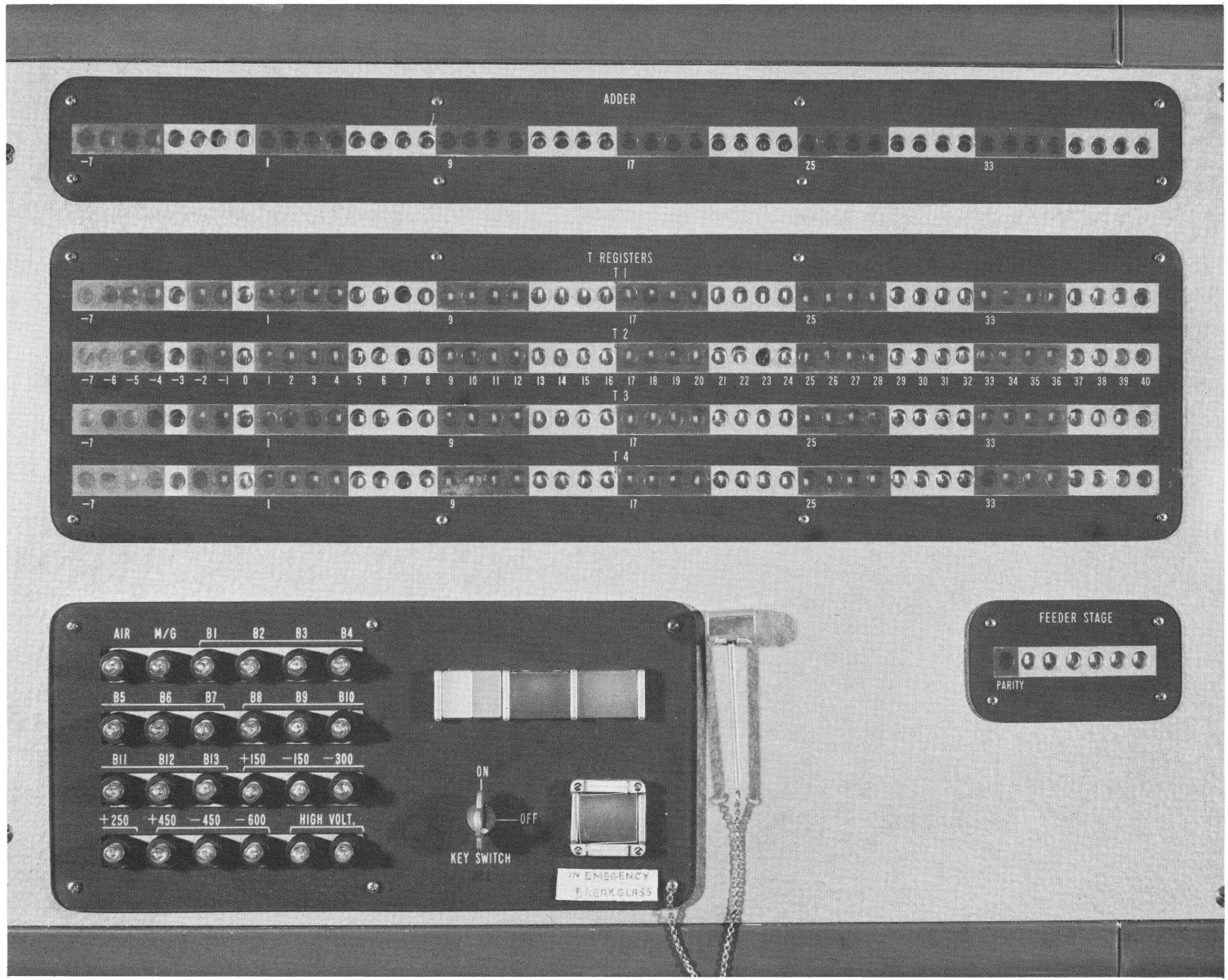


Figure 9. Section A of Merlin Control Panel.

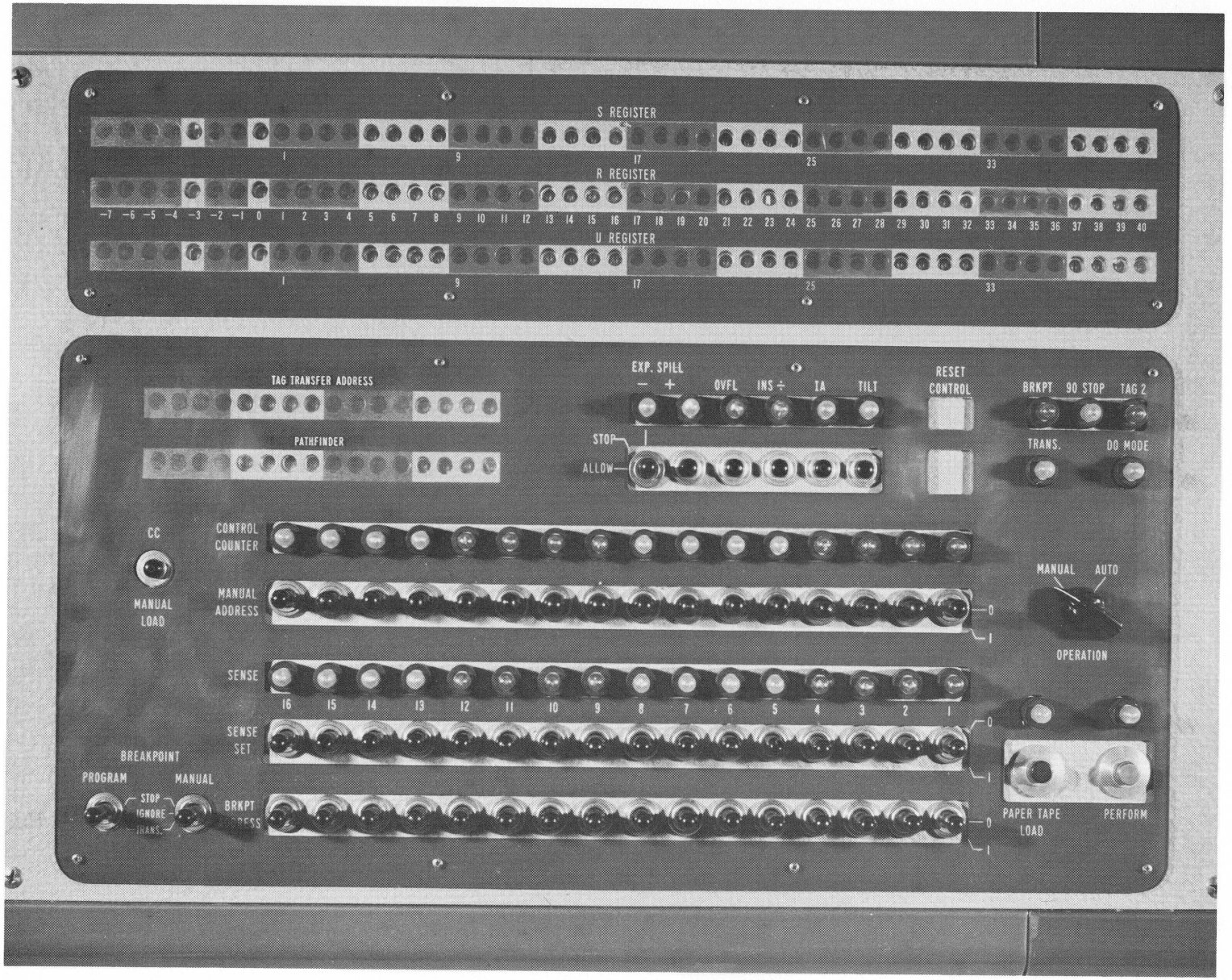


Figure 10. Section B of Merlin Control Panel.

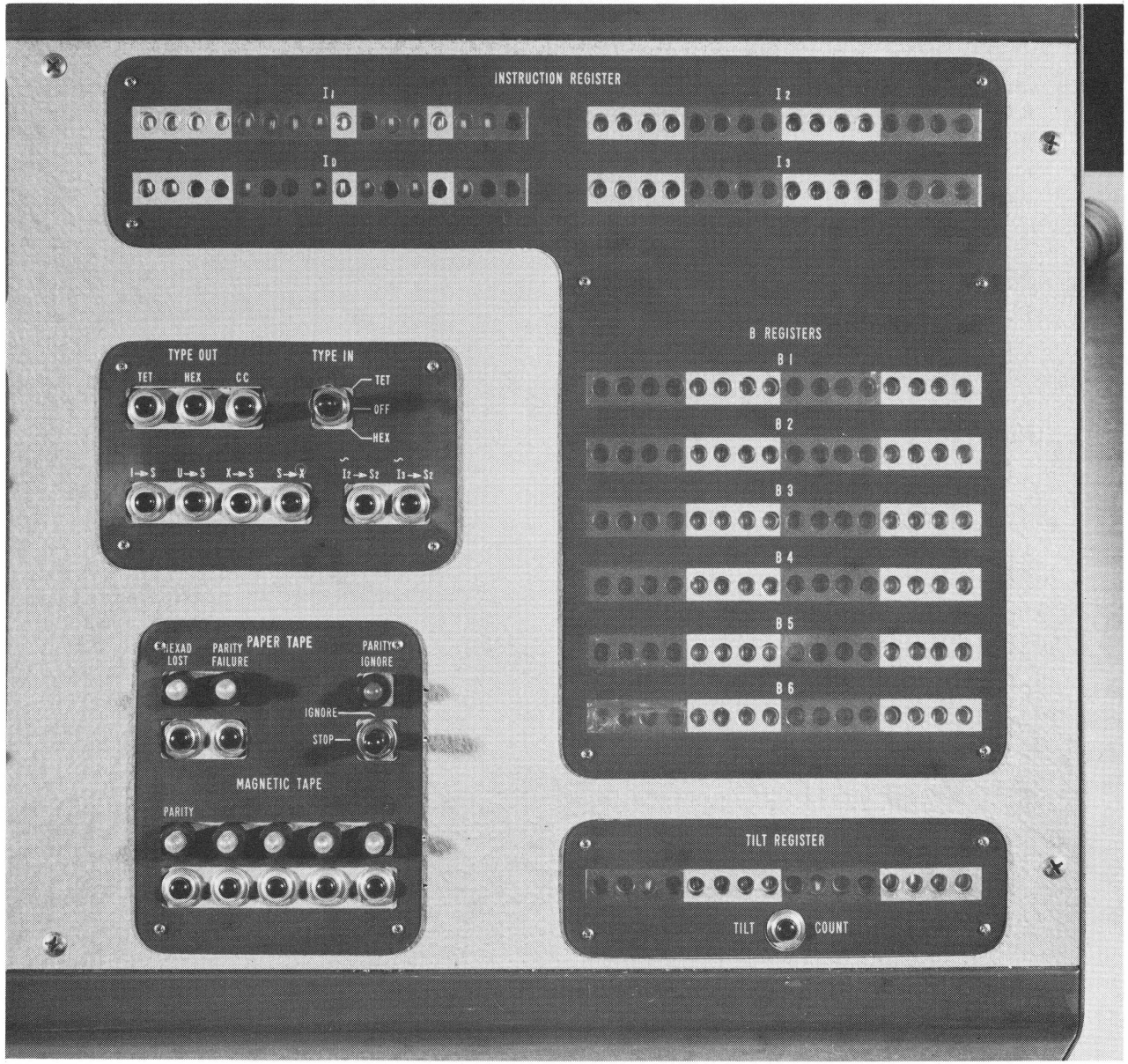


Figure 11. Section C of Merlin Control Panel.

of lights indicated by I_1, I_2, and I_3 (Fig. 11). The contents of the 16 lights indicated by I_D is the order dioctad of the instruction which had been performed by the computer before it stopped. Note that, at this point, the Control Counter will contain the memory address of the instruction to be performed.

The execution of the instruction in the I Register is initiated by depressing the "Perform" button* (Fig. 10). As long as the "Operation" switch is in the manual position, one must manually initiate the perform cycle of every new instruction which is brought to the I Register in this manner. The computer, however, may be made to operate in the automatic mode by turning the Operation switch to the Automatic position and by depressing the "Perform" button once.

The section of the panel to the right of the PF and TTA Register display lights, (Fig. 10) consists of lights which are turned on whenever some of the computer conditions are either established or satisfied. These lights should be consulted whenever the computer in the "Automatic" mode stops in order to determine not only the nature of the stoppage, but also the state of the stored program as described below:

(1) Tag 2 Light: This light is turned on whenever a tagged operand (i.e. a number with a "1" in the Tag 2 position) is brought to the Arithmetic Section from memory in all instructions which sence the tag 2 bit. It is turned off by the computer on entering the perform cycle of the next instruction.

(2) Do Mode Light: This light is turned on by the "Do" order, and turned off automatically when the computer leaves the "Do" mode. It should be mentioned again that a breakpoint stop on an instruction which is being repeated will be honored only after the last "Do". Thus logically the computer can be stopped with this light turned on only if the "Do" instruction itself is breakpointed for a stop.

* A special indicator which is located above this push buton (Fig. 10) is turned on whenever a manual perform may be initiated.

(3) 90 Stop Light: This light is turned on whenever the computer is stopped by the "90" instruction to indicate that the stoppage is a legitimate one. It is turned off automatically when the Perform button for the next instruction is depressed.

(4) Breakpoint Light: (BRK.PT.) This light is turned on whenever the conditions for a breakpoint stop (either program or manual) are satisfied.

(5) Transfer Light: (Trans) This light is turned on whenever a transfer of control as described in Sec. 3 is effected. It is turned off automatically by the computer.

(6) Negative Exponent Spill Light: (EXP SPILL -) This light is turned on whenever a negative exponent spill occurs in a floating point or normalize operation. The computer automatically clears the number as defined in Sec. 3 and proceeds automatically to the perform cycle of the next instruction if the toggle switch which is located below the light is in the "allow" position. Otherwise the computer will come to a halt. This light is turned off automatically on entering the perform cycle of the next instruction.

(7) Positive Exponent Spill Light: (EXP SPILL +) This light is turned on whenever a positive exponent spill occurs in a floating point operation. The light can be turned off manually by a push button or by the transfer of control on positive exponent spill instruction.

(8) Overflow Lights (OvFl): This light is turned on whenever an overflow occurs in a (fixed point operation or a left shift operation). It can be turned off manually, or by the transfer of control on overflow instruction.

(9) Insignificant Divide Lights (INS ÷): An insignificant divide light is turned on whenever a floating divide operation requires five or more 8 bit shifts in the "normalization" of the numerator. (c.f. Sec. 5). It does not define a computer condition in the sense of Sec. 3 and this light is turned on "for information only". The light must be turned off manually.

(10) Tilt Light: The tilt light is turned on whenever a parity failure occurs in a word in memory. The computer is stopped on all tilts, and the address of the memory word involved will be displayed on the 16 bit tilt register.

6.2 Console Typewriter

One can manually type information into the computer by means of the Console Typewriter. This typewriter is provided with 42 keys and, with the Operation switch in the Manual position, a six bit character code described in Sec. 4, can be generated in the "Feeder Stage" whenever any one of them is depressed. The contents of the Feeder Stage is then automatically shifted into the rightmost bit positions of the S-Register according to the setting of a special switch labeled "Flex In" on the console panel, (Fig. 11).

With the "TYPE IN" switch in the "TET" (for tetrad) position, the control section will drop the two leftmost bits in the Feeder Stage and shift the remaining 4 bits into bits 37-40 of the S Register. The S Register is not cleared to zero before the shift of 4 takes place; thus one is able to type in successively and load the S Register with 12 hexadecimal characters. When the "Flex In" switch is in the "HEX" position, the entire contentsof the Feeder stage is shifted into bits 35-40 of the S Register as soon as the Feeder stage is loaded by depressing one of the keys of the console typewriter. With this setting one may load the S Register with eight hexad characters. When the console typewriter is not in use, the "Flex In" switch must be returned to the "off" position.

The contentsof the S Register may be manually transferred to any

memory location by depressing the "S → X" push button, (Fig. 11). This button is operative only when the Operation switch is in the Manual position, and the address of the memory location involved in this operation must be specified by the Manual Address Switches. The operation takes place without affecting the control counter.

With the Operation switch in the Manual position, one can also bring the contents of any memory location to S by setting the address of the desired memory word in the Manual Address Switch and then depressing the X → S push button. The contents of the I and U registers can similarly be brought to the S Register by depressing the "I → S" and "U → S" push buttons. Here again, these buttons are operative only when the Operation switch is in the manual position.

The contents of the S Register may be typed out on the console typewriter either in the tetrad or hexad mode by depressing the button marked "TET" or "HEX" under the heading "TYPE OUT" on the console panel, (Fig. 11). The Operation switch must also be in the manual position for these operations. The operations involved here are clearly the inverse of the "Flex In" operations mentioned above.

In addition to the manual type out of the S Register, a push button marked "CC" is provided in the "Type Out" section for manually typing out the contents of the Control Counter. The output in this case will always be in the tetrad mode and, since the S Register is used, it should be warned that information contained in that register will be destroyed during this operation. The output will always be in the form OOOOXXXXOOOO, with the four X's corresponding to the desired information. The (CC) button is operative only when the Operation switch is in the manual position.

To aid the operator in either reading or recording the effective memory address of class "1" instruction when the Operation switch is in the manual position, two buttons labelled $\tilde{I}_2 \rightarrow S$ and $\tilde{I}_3 \rightarrow S$ are also provided on the panel. When the $\tilde{I}_2 \rightarrow S$ button is depressed, the contents of I_2 is added to the contents of the B Register specified by the third tetrad of I_1 and the result is gated to the second dioctad of the S Register. When the $\tilde{I}_3 \rightarrow S$ button is depressed, the contents of I_3 is

added to the contents of the B Register specified by the fourth tetrad of I_1, and the result gated to S_2. The result of these additions may then be typed out by depressing the "TET" button.

The lights and switches connected with the paper tape system are found to the right of the B Register lights in Fig. 11. The push button which initiates manual paper tape load, however, is located to the left of the "Perform" button in Fig. 10.

All other switches and lights appearing in the panel which have not been discussed are either reserved for the engineers or not in use currently.

7. Memory Section *

The operation of the electrostatic memory depends on the fact that a charge placed on an insulating surface cannot move. Thus information may be stored by converting it to a charge pattern, of some suitable form, on the surface of a good electrical insulator. In the Brookhaven machine the insulating surface is mica and the charge is deposited by means of an electron beam.

Fig. 12 shows diagrammatically the construction of one of the "barrier grid" storage tubes. The tube derives its name from the fine wire mesh covering the storage surface. The operation of the tube can be understood by considering the fact that directing the electron beam to any point of the storage surface is equivalent to connecting that point to ground. The "writing" process is as follows:

I. With the beam off, the backplate is raised to a potential of +20 V from ground. As a consequence the entire front surface of the insulator will also be at (approximately) this potential.

II. The deflection plates are now "indexed" to some point on the storage surface and the beam is switched on. That point, and only that point, is thereby returned very nearly to ground potential. The beam is now switched off.

III. The backplate is returned to ground.

Now clearly the entire front surface of the insulator is once more at ground potential except for the one spot that was bombarded by the electron beam. That spot is at a potential of roughly -20 V, and the negative charge at that location represents the stored information.

Similarly, the writing process may be repeated employing a negative backplate excursion, resulting in the storage of a positive charge. In the machine this "bipolar" operation is employed to improve the discrimination between "ones" and "zeros."

The "reading" process depends on the fact that with the backplate at ground potential, the current that flows to a particular spot on the

* In response to several inquiries regarding the operation of the Memory Section, Dr. G. L. Miller of the Instrumentation Division has kindly prepared this Section.

target, on indexing the beam to that location, will depend on the charge stored at that spot. It is clear, however, that reading is destructive in that it tends to obliterate the charge information at the point read. Steps I, II, III are therefore repeated immediately following a read to restore the information to its original state. This process of reading and rewriting constitutes a "regeneration."

Address Generation

For use in a machine the memory must be capable of storing and reading out words belonging to given memory "addresses." In the Merlin, the memory is split up into four banks of 48 tubes each. In each bank the tubes are arranged in order, -7 through 40, to correspond to the bits of a machine word. The 16 bit memory address employed in the machine is used as shown in Fig. 13. Starting from the right, the first 13 bits are used to generate a "raster" of 8192 dots on the storage tube surfaces, the next two bits are used to select the appropriate 1|4 of the memory, and the sixteenth bit is disregarded.

In writing or reading, all the 48 tubes in the selected quarter of the memory are operated in parallel; i.e. all the bits of a given word are read in or out simultaneously.

Readaround

A defect of this type of memory is the fact that writing or reading at a given location tends to degrade the information stored at adjacent locations. (This is primarily a consequence of the Gaussian tail of the electron beam radial current density distribution.) As a result there is a certain maximum number of times that any given spot may be addressed before it causes a failure at an adjacent spot. This number is called the "readaround ratio."

In order to mitigate this adjacent spot effect, the entire contents of the memory are constantly regenerated. A fetch or a store operation is an interruption of this regeneration cycle, and logical interlocks are provided to ensure that access is gained only in the appropriate intervals.

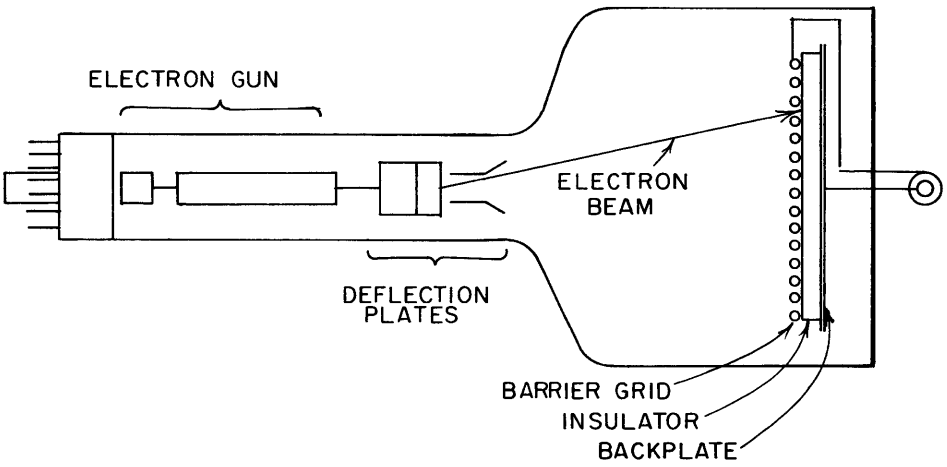


Figure 12. Radechon Storage Tube.

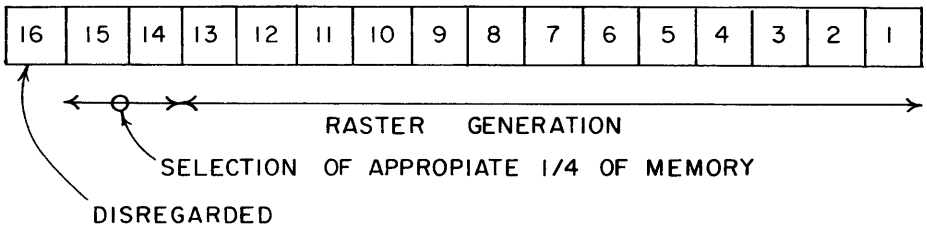


Figure 13. Organization of Memory Address.

Parity Check

On storing a word in memory the number of "ones" contained in the word is noted; if this number is odd, then a 1 is stored in a 49th barrier grid tube at the appropriate address. Otherwise, a zero is stored at that address. The extra bit stored for each word is the "parity" bit for that word. Constant check is kept during regeneration, and on readout, to ensure that the parity of the word agrees with its parity bit. In the event that they do not agree, it is an indication of a memory failure. This failure is indicated by the "tilt" light on the Console Panel; simultaneously the address of the faulty word is displayed in the tilt register.

Logical Organization

From the foregoing description, it will be clear that the memory must contain the following elements:

1. A register to contain the address of the word to be consulted. This is the Memory Address Register, M.A.R. In addition, there is an identical register, the Temporary Address Register, which is used to save the memory address involved in a store operation to make possible the irregular fetch of instructions described in Sec. 3.

2. A digital-to-analogue converter to generate deflection voltages for the storage tubes from the last 13 bits of the address.

3. A register to contain the address of the word being regenerated when the memory is not being consulted, but is only regenerating. This is the Regeneration Address Scaler.

4. Parity generating and checking circuitry.

5. A Tilt Register to contain the address of a failed word.

6. Timing and logical interlocks to allow the meshing of the asynchronous arithmetic part of the computer to the synchronous memory.

These elements are shows interconnected in a schematic form in Fig. 14.

Performance

Average access time is 6 μs. Memory capacity is 8192 words per quarter, and the entire contents of the memory are regenerated every 65 milliseconds.

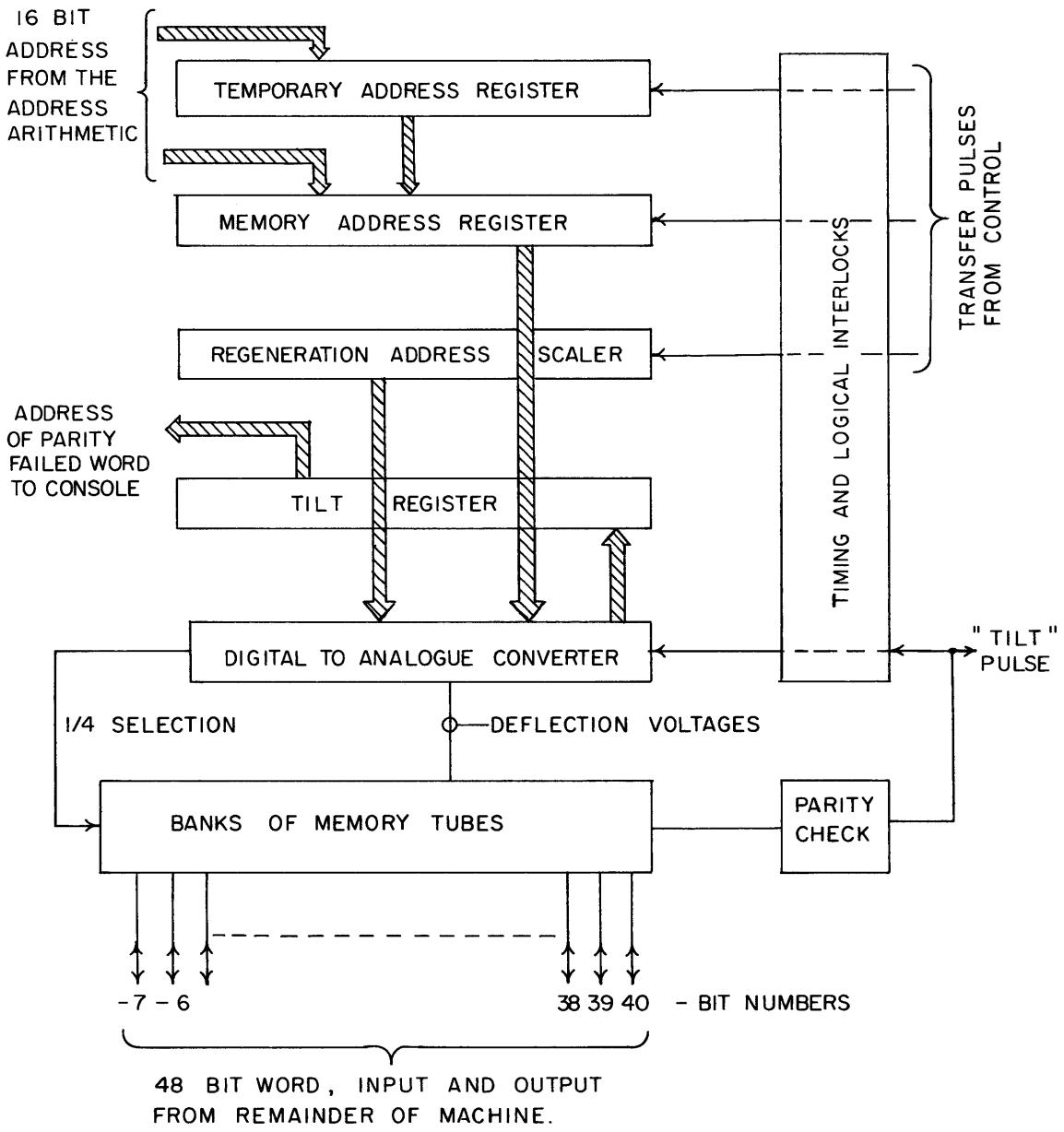


Figure 14. Memory Block Diagram.