Division 6 – Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: The Lincoln TX-2 Computer

To: Distribution List

From: W.A. Clark, J.M. Frankovich, H.P. Peterson, J.W. Forgie, R.L. Best, K.H. Olsen

Date: 1 April 1957

Approved: _____
W.N. Papian

Abstract: The information contained in this report was presented at the Western Joint Computer Conference held at Los Angeles, California, in February 1957. The papers given form the sections of this memorandum and are as follows:

THE LINCOLN TX-2 COMPUTER DEVELOPMENT
by Wesley A. Clark

Construction of the TX-2 computer at the Lincoln Laboratory of MIT is part of the Lincoln program for the study and development of large-scale digital computer systems. The Lincoln TX-2 incorporates several new developments in high-speed transistor circuits, large capacity magnetic-core memories, and flexibility in machine organization and is designed to work efficiently with many input-output devices of different types. In the course of development of the TX-2, Lincoln has constructed a small self-checking multiplier system which is on life test, and a complete, though skeletal, general-purpose computer known as the TX-0 which is now in operation.

A FUNCTIONAL DESCRIPTION OF THE LINCOLN TX-2 COMPUTER
by John M. Frankovich and H. Philip Peterson

The Lincoln TX-2 computer is a general-purpose, binary parallel machine with a code of 64 single-address instructions and 64 index registers. The design provides for a random-access memory of 260,000 36-bit words. The instruction code includes the usual arithmetic and logic operations executed at a peak rate of 160,000 36-bit additions per second, with several interesting

The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No. AF 19(122)-458.

variants. A unique feature of the central computer is its ability to deal with operands in one 36-bit, one 27- and one 9-bit, two 18-bit, or in four 9-bit configurations. These configurations are specified by each instruction -- a feature which permits the 9-bit quarters of the arithmetic element to be connected in various ways to the corresponding quarters of the memory. Control is exercised over the activity of the quarters during the execution of the instruction.

THE LINCOLN TX-2 INPUT-OUTPUT SYSTEM
by James W. Forgie

The Lincoln TX-2 computer design uses the "multiple-sequence program technique" to permit the concurrent operation of a number of input-output devices. A stored program (instruction) counter is associated with each input-output device. The programs referred to by these counters time-share the hardware of the central computer, giving attention to the associated input-output devices as required. A priority system ranks the devices according to speed and type for efficient operation with a minimum of programming restrictions. The multiple-sequence program technique provides an environment in which buffer storage may be considerably reduced at a small cost in machine speed within the limits set by peak and average data rate considerations.

MEMORY UNITS IN THE LINCOLN TX-2
by Richard L. Best

There are three random-access core memories in TX-2 -- all of which may be operated independently and concurrently. Two of these are for conventional storage of data and instructions. The third is used as a file of index registers and program counters. The three types of core memories in TX-2 are described. The largest memory contains 65,536 words 37 digits long, and has a full cycle time of 6 1/2 μsec. The next largest is entirely transistor driven, contains 4,096 words, 37 digits long, and has a 6-μsec cycle time. The smallest and fastest contains 64 words, 19 digits long, and uses external selection and two cores per bit to achieve a "read" cycle time of 1 μsec and a "write" cycle time of 3 μsec. Ferrite cores only 0.050 inches O.D. are used in the two smaller memories.

TX-2 CIRCUITRY
by Kenneth H. Olsen

Only two basic transistor logic circuits are used in TX-2. Surface-barrier transistors in saturated emitter-follower circuits are grouped in parallel for "AND" "OR" operations; groups of inverter circuits are connected in series, parallel, or series-parallel to perform more complicated logic operations. The

TX-2 flip-flop is assembled from saturated inverters and emitter-followers and incorporates enough amplification so that it appears to the logic designer as a simple switch over a wide range of loads.

Circuit tolerance to variations in transistor and other component characteristics, in temperature, supply voltages, and noise was studied in detail and designs were adjusted to minimize the effects of them. The study led to the selection of voltage sensitive parameters for indicating the deterioration of components with age and became the basis of the marginal checking system.

ACKNOWLEDGMENT

We are indebted to Mr Frank P. Hazel and Miss June Karlson for their assistance in editing; to Mrs. Barbara Clark, Miss Carole Olson, Miss Elaine Tonra and Miss Ruth McDonald for their typing of the manuscripts; and to Miss Alice Griffin and her associates for the preparation of the many figures.

Distribution List

Division 6

P.R. Bagley
R.L. Best
S. Bradspies
J.H. Burrows
W.A. Clark
L.B. Collins
J.W. Forgie
J.M. Frankovich
S. Goldberg
L.L. Holmes (Barta)
N.T. Jones
G. Marnie
K.H. Olsen
R.B. Paddock
W.N. Papian (3 copies)
H.P. Peterson
E.W. Pughe
A. Rowe (Barta)
W.F. Santelmann
C.A. Zraket

Non-Division 6

A.V. Nedzel
E.D. Thomas
Major H.B. Farmer
R. A. Nelson

Non-Lincoln (inside addresses)

N.E. Trieste (IBM)
A.R. Marshall (RAND)
Major S. Pierce (LRP)

Non-Lincoln (outside addresses)

Miss R. Rita Balogh
IBM, Military Products Div.
Kingston, New York

Carlo Bocciarelli
Philco Research Division
C and Tioga Streets
Philadelphia, Pennsylvania
(2 copies)

Mr. Arnold A. Cohen
Remington Rand UNIVAC
St. Paul 16, Minnesota

Mr. Abraham Katz
Commercial Electronic Products
RCA
Camden, New Jersey

Mr. Steve Levy
Lansdale Tube Co.
Church Rd.
Lansdale, Pennsylvania

Mr. Louis C. Murphy
Marchant Research
717 Los Palos Drive
Lafayette, California

Mr. A.A. Perez
National Cash Register Co.
Electronics Division
1401 East el Segundo Blvd.
Hawthorne, California

Mr. Kenneth M. Rehler, Mgr.
Electronic Control Systems, Inc.
2136 Westwood Blvd.
Los Angeles 25, California

Table of Contents

I. THE LINCOLN TX-2 COMPUTER DEVELOPMENT 1

A. Introduction 1
B. History 1
C. Design Objectives 4

II. A FUNCTIONAL DESCRIPTION OF THE LINCOLN TX-2 COMPUTER 7

A. Introduction 7
B. General Structure of TX-2 7
C. Memory Element 9
D. Control and Indexing 11
E. Arithmetic Element 13
F. AE Circuits 15
G. System Timing 16
H. Configuration Control 17
I. Operation Code 29
J. Instruction Times 31
K. Summary 31

III. THE LINCOLN TX-2 INPUT-OUTPUT SYSTEM 37

A. Introduction 37
B. The Multiple-Sequence Program Technique 37
C. Multiple-Sequence Operation in TX-2 38
D. The TX-2 Input-Output Element 39
E. Input-Output Instructions 41
1. rdn and rds 41
2. ios 41
F. Sequence-Changing and Operation of the Sequence-Selector 43
G. Interpretation of the Break Bit 44
H. Interpretation of the Dismiss Bit 44
I. Starting a Multiple-Sequence Computer 45
J. The Arithmetic Element in Multiple-Sequence Operation 45
K. Conclusions 46

IV. MEMORY UNITS IN THE LINCOLN TX-2 49

A. Introduction 49
B. S-Memory (65,536 Words) 49
C. T-Memory (4096 Words) 51
1. Mechanical Features 52
2. Selection Circuits 52
3. Digit Circuits 53
D. Memory 56
1. Operating Principle 56
2. Selection Circuits 60
3. Read-Write Drivers 61

Table of Contents (Continued)

4. Digit Circuits 63
5. Modes of Operation 63
E. Acknowledgment 64

V. TX-2 CIRCUITRY 65

A. Circuit Configurations 65
B. Flip-Flop 68
C. Marginal Checking 71
D. Packaging 71
E. Conclusion 73

List of Illustrations

Fig. 1	The Lincoln TX-2 Computer	1
Fig. 2	Steps in the Lincoln TX-2 Development Program	4
Fig. 3	TX-2 System Schematic	8
Fig. 4	TX-2 Memory Element - Showing method of using two address registers and two buffer registers to simultaneously operate two out of four memories	10
Fig. 5	TX-2 Instruction Word Structure	11
Fig. 6	TX-2 Program Element Showing paths enabling index adding and storing and loading program counter from the index memory and the exchange element	12
Fig. 7	Circuits and Transfer Paths (General) of any TX-2 Arithmetic Element Forms	14
Fig. 8(a)	Consecutive Load Type Instructions; Instructions and Operands in Different Memories	18
Fig. 8(b)	Consecutive Store Type Instructions	18
Fig. 8(c)	Instruction and Operand in Same Memory	19
Fig. 8(d)	Change Sequence	19
Fig. 9	TX-2 Configuration Selection	20
Fig. 10(a)	TX-2 Configuration; Quartering Permutation Paths and Activity Flip-Flops Shown	22
Fig. 10(b)	TX-2 Configuration; Paths In Exchange Element	23
Fig. 11(a)	i^{th} Quarter Coupling Units	24
(b)	Coupling Unit Connections	24
Fig. 11(c)	TX-2 Configuration; Arithmetic Elements and Operand Word Structures	26
Fig. 11(d)	TX-2 Shift Path Arrangements	27
Fig. 12	TX-2 Configurations; Areas of Activity During Execution of Instruction Shown Shaded. Effect of AE Couplings Illustrated by Juxtaposition of Quarters	28
Fig. 13	TX-2 Sequence Selector Stage	40
Fig. 14	TX-2 In-Out Element	42
Fig. 15	S-Memory; Coincident-current, Magnetic-Core Unit	49
Fig. 16	Block Diagram, S-Memory	50
Fig. 17	Block Diagram, T-Memory	51

Fig. 18	Timing, T-Memory	52
Fig. 19	Pluggable Array, T-Memory	52
Fig. 20	One Channel, Emitter follower and Inverter AND gate, T-Memory	53
Fig. 21	One Channel, Selection Line Driver, T-Memory	54
Fig. 22	Read-Write Driver (2 needed), T-Memory	54
Fig. 23	Digit-Plane Driver, T-Memory	55
Fig. 24	Sense Amplifier, T-Memory	56
Fig. 25	Winding Configuration, X-Memory	57
Fig. 26	Timing Diagram, X-Memory	57
Fig. 27	64 x 38 Memory Plane, X-Memory	58
Fig. 28	Section of Memory Plane, Enlarged, X-Memory	58
Fig. 29	Block Diagram, X-Memory	59
Fig. 30	Register Selection Circuit, X-Memory	60
Fig. 31	Read Driver, X-Memory	61
Fig. 32	Write Driver, X-Memory	62
Fig. 33	Digit Driver, X-Memory	62
Fig. 34	Sense Amplifier, X-Memory	63
Fig. 35	Emitter Follower	65
Fig. 36	Parallel Emitter Follower	65
Fig. 37	Inverter	66
Fig. 38	Parallel Inverters	66
Fig. 39	Series Inverters	66
Fig. 40	TX-2 Carry Circuits	67
Fig. 41	Turn-Off Time	67
Fig. 42	TX-2 Flip-Flop	68
Fig. 43	Flip-Flop Waveforms	69
Fig. 44	Trigger Sensitivity	69
Fig. 45	τ Margins	70
Fig. 46	β Margins	70
Fig. 47	-10 Volt Supply Margins	72
Fig. 48	-3 Volt Supply Margins	72
Fig. 49	Temperature Margins	72
Fig. 50	Pulse Margins	72
Fig. 51	TX-2 Plug-in Unit	73
Fig. 52	TX-2 Back Panel	73

I. THE LINCOLN TX-2 COMPUTER DEVELOPMENT

A. Introduction

The TX-2 is the newest member of a growing family of experimental computers designed and constructed at the Lincoln Laboratory of MIT as part of the Lincoln program for the study and development of large-scale, digital computer systems suitable for control in real time. Although, in general characteristics and design philosophy, it owes a great deal to its predecessors, Whirlwind I and the Memory Test Computer, the Lincoln TX-2 incorporates several new developments in components and circuits, memories, and logical organization. It is the purpose of this paper to summarize these new features and to give some idea of the historical development and general design objectives of the TX-2 program. Fig. 1 shows TX-2 in its present development stage.

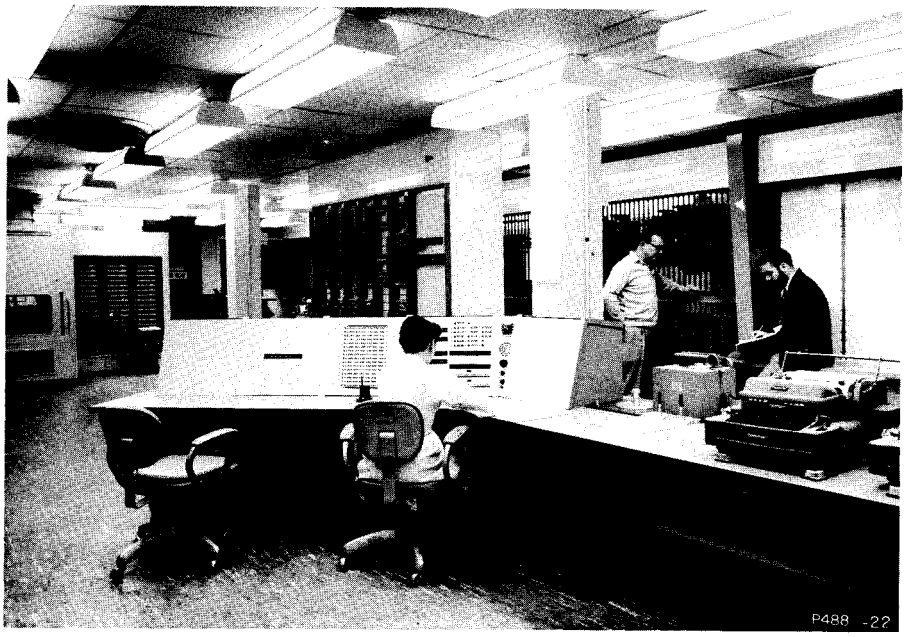

Fig. 1 The Lincoln TX-0 and TX-2 Computers

Foreground: TX-0 console
Middle center: TX-0 central computer frame
Right rear: Partially completed TX-2 frame showing plug-in unit construction
Left rear: The 256 x 256 memory

B. History

With the development by Lincoln and IBM engineers of the SAGE computer for air defense, real-time control computer systems had

reached an impressive level of size, sophistication, and complexity. The highly successful 64 x 64 coincident-current, magnetic-core, memory array was in operation in the Memory Test Computer which had given up its earlier 32 x 32 array to Whirlwind. Vacuum tubes abounded in all directions. It was apparent that the further advances in system design which could be made by increasing memory size, eliminating vacuum tubes wherever possible, and organizing input-output buffering, control, and communications into more efficient forms, would be well worthwhile.

The development of a 256 x 256, switch-driven, magnetic-core memory array was begun and the Philco surface-barrier transistor made its appearance. After some very promising bench experiments with flip-flops and logic circuits, it became apparent that this transistor was potentially well-suited to use in large-scale systems and warranted further study. Accordingly, plans were laid for a succession of experimental digital systems of increasing size and complexity which would make possible the development and evaluation of circuits using the surface-barrier transistors, and which would lead to a computer of advanced design that would be capable of making efficient use of the 256 x 256 memory.

A double-rank shift register of eight stages and containing about 100 transistors was constructed and put on life-test in April 1955. It has since been circulating a fixed pattern almost continuously with no known errors and no natural transistor failures.

As the next step, it was decided to build a small, high-speed, error-detecting multiplier and incorporate marginal checking and other system features. The value of a multiplier as a preliminary model had been well demonstrated by the 5-digit system built during Whirlwind's early development. The shift, carry, count, and complement operations, under closely controlled timing conditions, were felt to be representative of all of the operations in the manipulative elements of the type of computer planned. Accordingly, an 8-bit system using 600 transistors was designed and completed in August 1955 and has been in nearly continuous operation since. Operating margins are periodically checked, and in steady state operation, the multiplier's error-rate has been about one every two months or one error per 5×10^{11} multiplications at 10^5 multiplications per second. Most of these errors appear to have been caused by cracks in the printed wiring which open intermittently.

During this period, a better idea of the general characteristics of the projected computer began to develop and the engineers who were designing the 256 x 256 memory were encouraged to think in terms of a word of 36 bits. The notion of a logically separate

input-output processor was examined and rejected in favor of a minimum buffering scheme in which data is transferred directly to and from the central memory of the computer. The possibility was recognized of programming these transfers by means of additional program sequences and associated program counters, thus taking advantage of the extensive facilities of the central machine itself for processing input-output data.

It was realized that another development step was desirable before attempting such an elaborate 36-bit system. The 8-bit multiplier had produced a certain measure of confidence and familiarity with circuits, packaging, and techniques of logical design, but there remained the problems associated with communicating with memory units and input-output equipment operating at vacuum-tube levels over relatively large distances from a central machine which operated at transistor levels. It appeared that the memory development, which had now entered the construction phase, would also benefit by a preliminary evaluation of the 256 x 256 array and its switching, timing, and noise problems in an operating computer of some kind, possibly with a reduced word length. It was, therefore, decided to design and build next a simple machine - in fact, the simplest reasonable machine - in order to bring about an early intermediate closure of the various efforts within the program.

After some thought about the various possible minimal machines, a design was completed in which the word length would be 18 bits - a graceful half of the projected final form. We began to refer to this computer as the TX-0 and to the projected machine as the TX-2. Because the 256 x 256 memory array required 16 bits for complete addressing, the single-address instruction word of the TX-0 was left with 2 bits in which to encode instructions. The particular set of instructions chosen included three which required a memory address (add, store, and conditional jump) and one which did not. In this last instruction, the remaining 16 bits were used to control certain necessary and useful primitive operations such as clearing and complementing the accumulator, transferring words between registers, and turning on and off input-output equipment.

The TX-0, equipped with a Flexowriter, a paper-tape reader, and a cathode-ray tube display system was completed, except for the memory, in April 1956. Twenty planes of the 256 x 256 memory array were installed the following August and the TX-0, now containing about 3600 transistors and 400 vacuum tubes, began to function as a complete computer. Since that time, it has been used to run a variety of testing and demonstration programs, and a symbolic address compiler and other utility programs have been constructed and are currently in use.

Not only has the TX-0 served the evaluational purposes for which it was built, but it has also demonstrated an effectiveness as a

usable computer that is somewhat surprising in view of its simplicity. Its relatively high speed of about 80,000 instructions per second and its 65,536-word memory compensate in large measure for the limitations of its instruction code and logical structure.

With the successful completion of the TX-0, the final steps in the development were undertaken in packaging, circuit refinement, and logical design of the TX-2. A great deal had been learned about the performance of the transistors and memory, the types of logical circuits which are practical, techniques of marginal checking, and the lesser system problems such as color scheme selection and the proper location of pencil sharpeners. As design work progressed, the TX-2 took form as a system of about 22,000 transistors and 600 vacuum tubes. It is an interesting fact that at each step of the development since the shift register, the number of transistors involved was about 6 times the number in the preceding step. This is graphically shown in Fig. 2. At the time of writing approximately 16 million transistor-hours have accumulated in the shift register, multiplier, and TX-0. There have been two natural deaths and a dozen or so violent ones, primarily due to contact shorting with clip leads and probes.

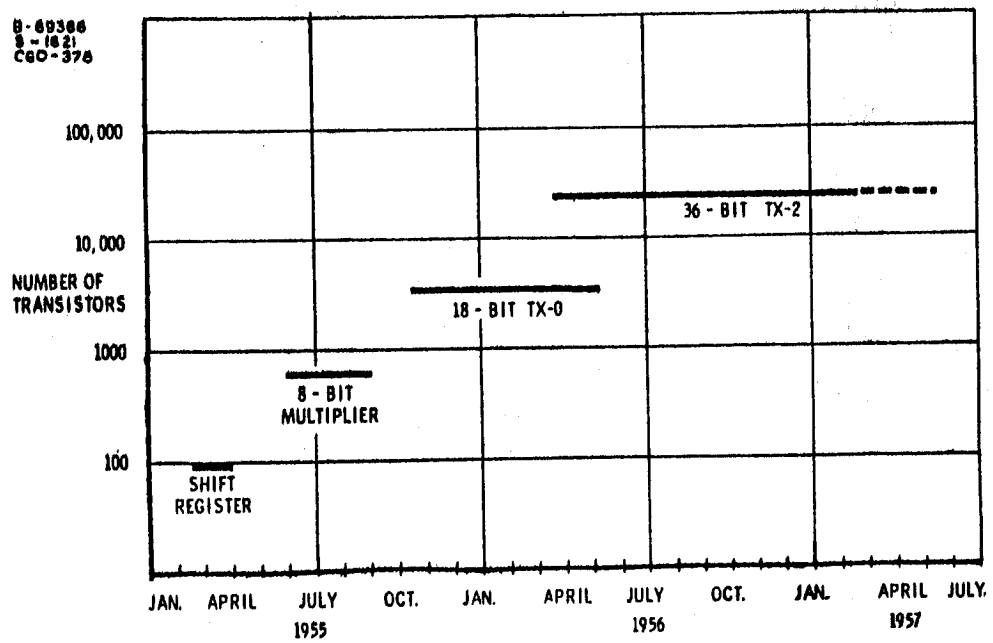

Fig. 2. Steps in the Lincoln TX-2 Development Program

C. Design Objectives

In describing design objectives, it should be pointed out that speed of operation was not the primary consideration to which all other attributes were sacrificed. It would have been

possible, at the expense of a few more logic circuits, to increase the speed of multiplication, division, and shift-type operations. Similarly, the operation of the index register system could have been made more efficient at the cost of an additional small, fast memory. The principal objective was rather that of achieving a balance between the factors of speed, reliability, simplicity, flexibility and general virtue.

A key aspect is that of expandability which, in an experimental computer in an active environment, certainly ranks with the foregoing qualities in importance. The address structure in the TX-2 permits an expansion of the memory by about a factor of 4, partly to allow for new memory developments, such as the transistor-driven 64 x 64 array which was begun following the completion of TX-0. New instructions and pieces of terminal equipment will certainly be added during the course of future operation. Extra space and spare plugs have been artfully distributed about in constructing the computer frame. Finally, modular construction will permit a fairly easy physical expansion when required.

The result of all this activity has been a computer of relatively large capability. In addition to incorporating high-speed transistor circuits and a large magnetic-core memory array, the Lincoln TX-2 has two major and distinguishing design characteristics:

1. The structure of the arithmetic element can be altered under program control. Each instruction specifies a particular form of machine in which to operate, ranging from a full 36-bit computer to four 9-bit computers with many variations. Not only is such a scheme able to make more efficient use of the memory in storing data of various word lengths, but it also can be expected to result in greater over-all machine speed because of the increased parallelism of operation.

   Peak operating rates must then be referred to particular configurations. For addition and multiplication, these peak rates are given in the following table:

PEAK OPERATING SPEEDS OF TX-2

Word Lengths (in bits)	Additions per second	Multiplications per second
36	150,000	80,000
18	300,000	240,000
9	600,000	600,000

2. Instead of one instruction counter, the TX-2 has 32 such counters which are assigned separately to different users of the computer, who then compete for operating time from instruction to instruction. A special part of the machine selects a particular user based partly on a predetermined priority schedule and partly on the current needs of that user. This multiple-sequence operation, in which many essentially independent instruction sequences interrupt and interleave one another, is an extension of the breakpoint operation found in DYSEAC of the National Bureau of Standards.

The value of these features will have to be assessed during the course of future machine operation. The features themselves are discussed in more detail in the following sections of this report.

II. A FUNCTIONAL DESCRIPTION OF THE LINCOLN TX-2 COMPUTER

A. Introduction

TX-2 is a large-scale digital computer, designed and built at the MIT Lincoln Laboratory, which uses new memory and circuit components and some new logical design concepts. The computer will be a research tool in scientific computations, data-handling, and real-time problems. The design of the computer reflects not only the characteristics of the components available, but also the nature of the intended applications. This section describes the functions and organization of the computer that are important from the user's point of view.

B. General Structure of TX-2

TX-2 is a parallel, binary computer with words 36 digits long. The internal memory is random-access and will initially consist of 69,632 registers of parity-checked, magnetic-core memory and about 24 additional toggle-switch and flip-flop registers. About 150,000 instructions can be executed per second. Instructions are indexed, single-address type and a fixed-point, signed-fraction, ONE's - complement number system is used.

Several unusual ideas incorporated in the organization of the system reduce the amount of information unnecessarily manipulated during program sequences. Furthermore, the organization facilitates the execution of several operations simultaneously, thereby increasing the effective speed of the computer.

The principal registers and information paths in the computer are illustrated schematically in Fig. 3. A,B,C,D,E,F,M, and N are the 36-bit flip-flop registers in the machine. M and N are memory buffer registers, each of which has a parity flip-flop and associated circuitry used to check the parity of memory words. P,Q, and X are 18-digit registers; X also has a parity digit which is used to check the parity of words in the X-memory. Control flip-flops are not shown in Fig. 3.

Instructions are full memory words and are placed in the control element during the instruction memory cycle. During the operand memory cycle, an operand is usually transmitted between the memory element and some other element--always through the exchange element. The 36-digit configuration of the memory is not, however, maintained throughout the computer during operation timing. A programmer can, in effect, simultaneously control several independent, with operands of shorter word lengths during the execution of each

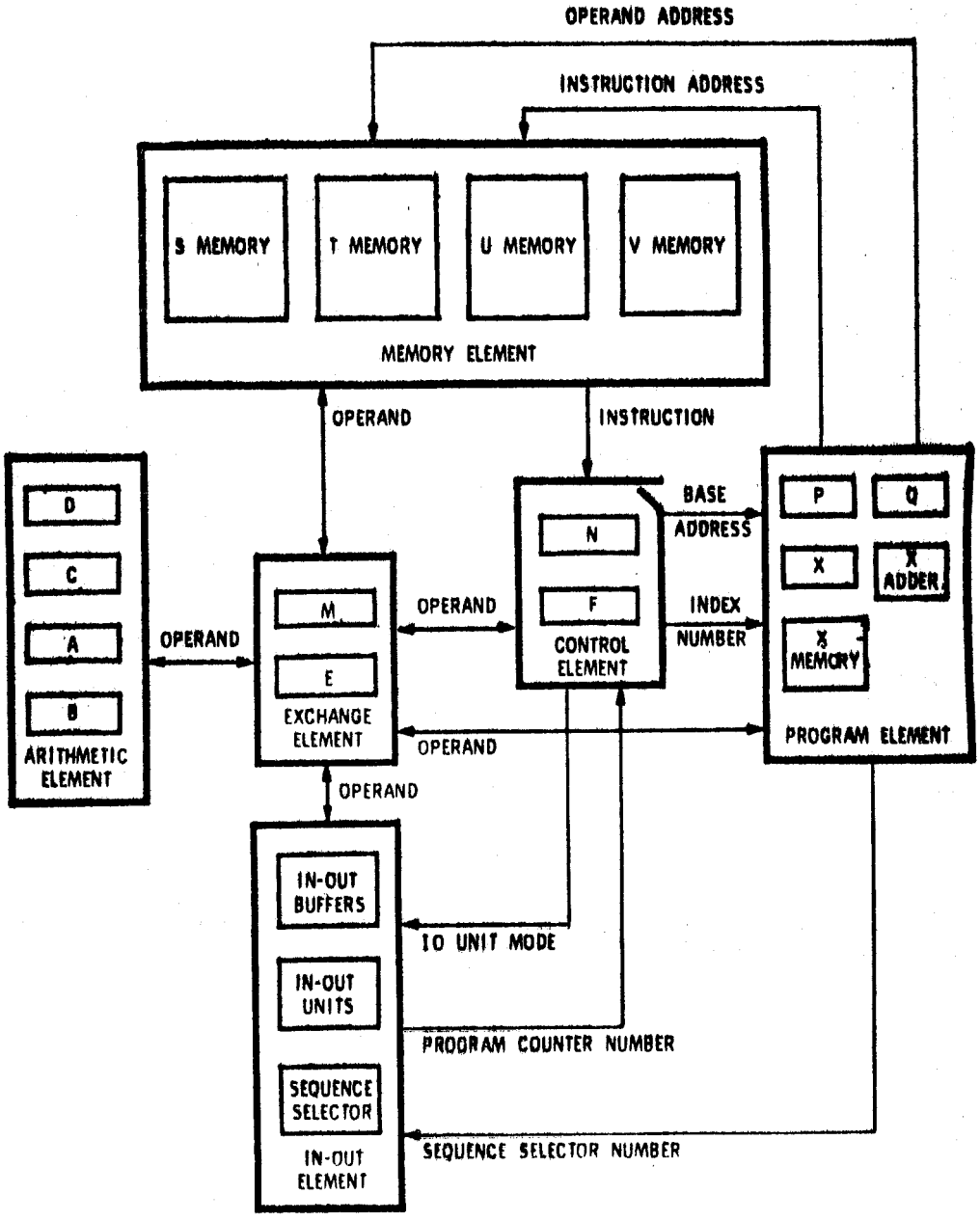

Fig. 3 TX-2 System Schematic

instruction. This flexibility is realized by specifying a particular system configuration with each instruction.

The computer communicates with the outside world through units in the in-out element, several of which can be simultaneously operated. Whenever any piece of data (input or output) is ready to be transferred between an in-out unit and a register in the memory element, signals to the program element from the in-out element automatically call into operation the instruction sequence associated with the in-out unit. This process is referred to as multiple-sequencing and will be described in Section III of this report.

C. Memory Element

The availability of a large, fast, core memory for TX-2 permitted an emphasis on the design of a machine with a completely random-access memory with a design capacity of 262,144 words. The homogeneous aspect of such a large memory system simplifies the programmer's coding problems and permits continued high-speed operation regardless of where the program is located in the internal memory.

The TX-2 memory element (Fig. 4) is divided into four, independently operating memory systems, each containing up to 65,536-digit words. The operating speed of TX-2 is determined by the cycle time for the memories: The 65,536-word S-memory is expected to have a cycle time of between 6.0 and 7.0 μsecs, and the 4,096 T-memory, a cycle time between 5.0 and 6.0 μsecs. Both memories have parity checks.

Although the U-memory is not presently specified, it may contain a 4,096-word core memory in the initial system. The V-memory consists of 8 flip-flop registers in the central machine and 16 toggle-switch registers which contain the program sequence executed whenever the START button on the operator's console is pushed. The contents of the toggle-switch registers can be used as instructions or operands, but cannot be altered by a program. The six 36-bit registers A,B,C,D,E, and F are part of the V-memory, but their contents can be used only as operands during the execution of an instruction. The programmer has, in a limited sense, a two-address instruction machine when he refers to these registers in load and store type instructions. The other two flip-flop registers in the V-memory are a 60-cps clock and a random-number register.

When an instruction calls for the storing of an operand in memory, the operand memory cycle can be extended up to 2.0 μsecs. The extension occurs between the time that the memory register is read and the time that it is rewritten. During this extension, the contents of memory-registers in the central computer are transferred, the parity of the word read from memory is checked, and the parity of the new memory word is computed. Because the extended cycle is less than

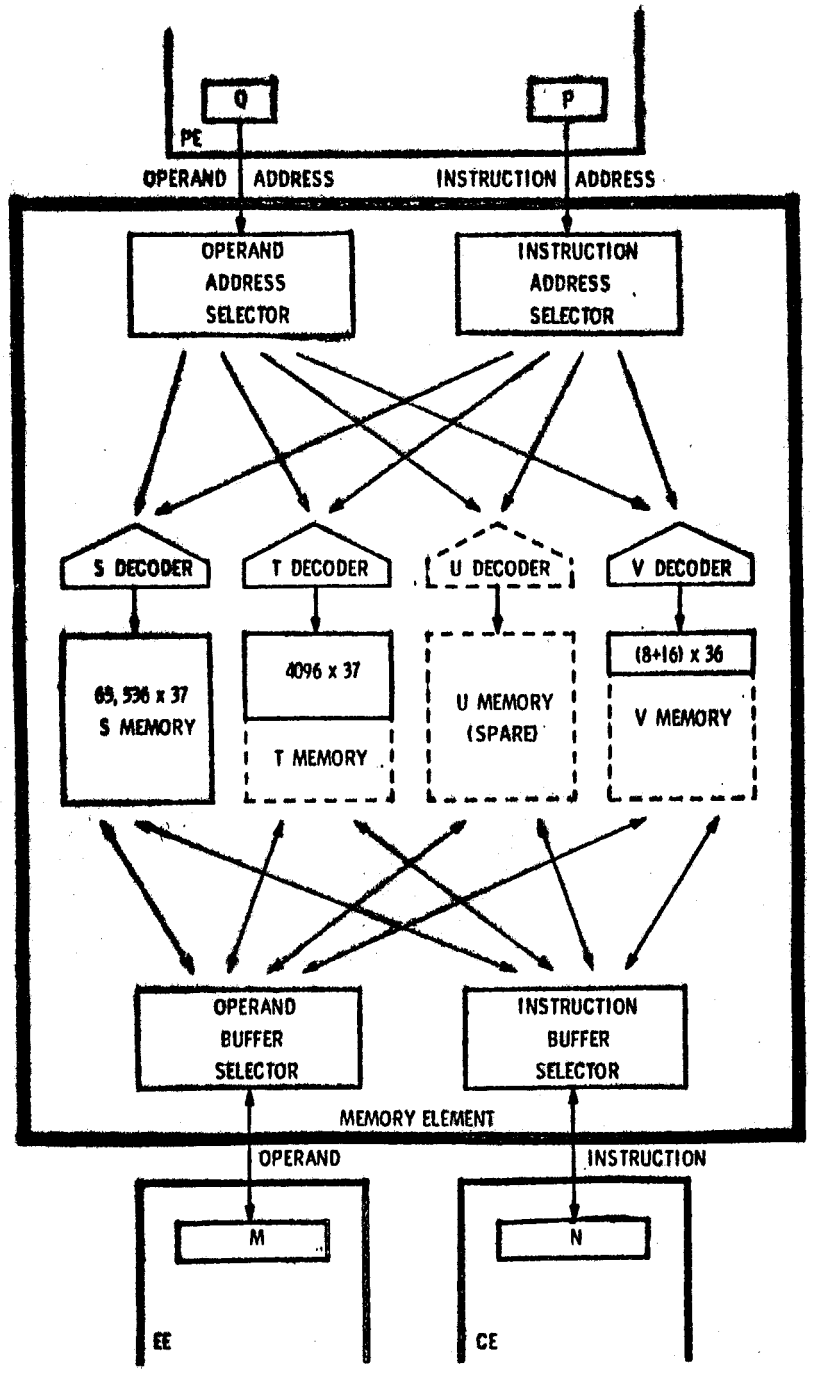

Fig. 4. TX-2 Memory Element - Showing method of using two address registers and two buffer registers to simultaneously operate two out of four memories

the two complete cycles usually used for instructions that modify words, computing efficiency is increased.

The P-register in the program element specifies the location of an instruction in memory and the N-register in the control element holds the instruction after it has been read from memory. The first 2 digits (reading left to right) of P select the memory system from which the instruction word is to be obtained; the remaining 16 digits address the word within the memory. Similarly, the Q-register locates the operand in one of the memory systems, and the operand is placed in the M-register.

D. Control and Indexing

An instruction word read into N has the structure shown in Fig. 5. The first 2 digits of the word specify information to the in-out element, and the 4 cf digits specify the computer configuration. The interpretation of the b and n digits is discussed in Section III. The cf digits are discussed in subsection H, p.17 .

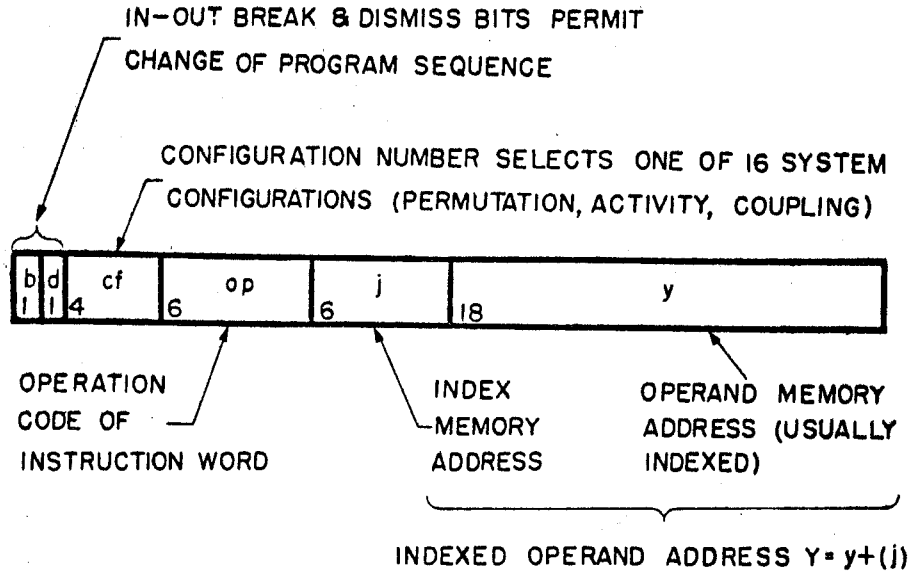

Fig. 5. TX-2 Instruction Word Structure

The operation code for the instruction is specified by the 6 op digits. On simple load and store type instructions, these

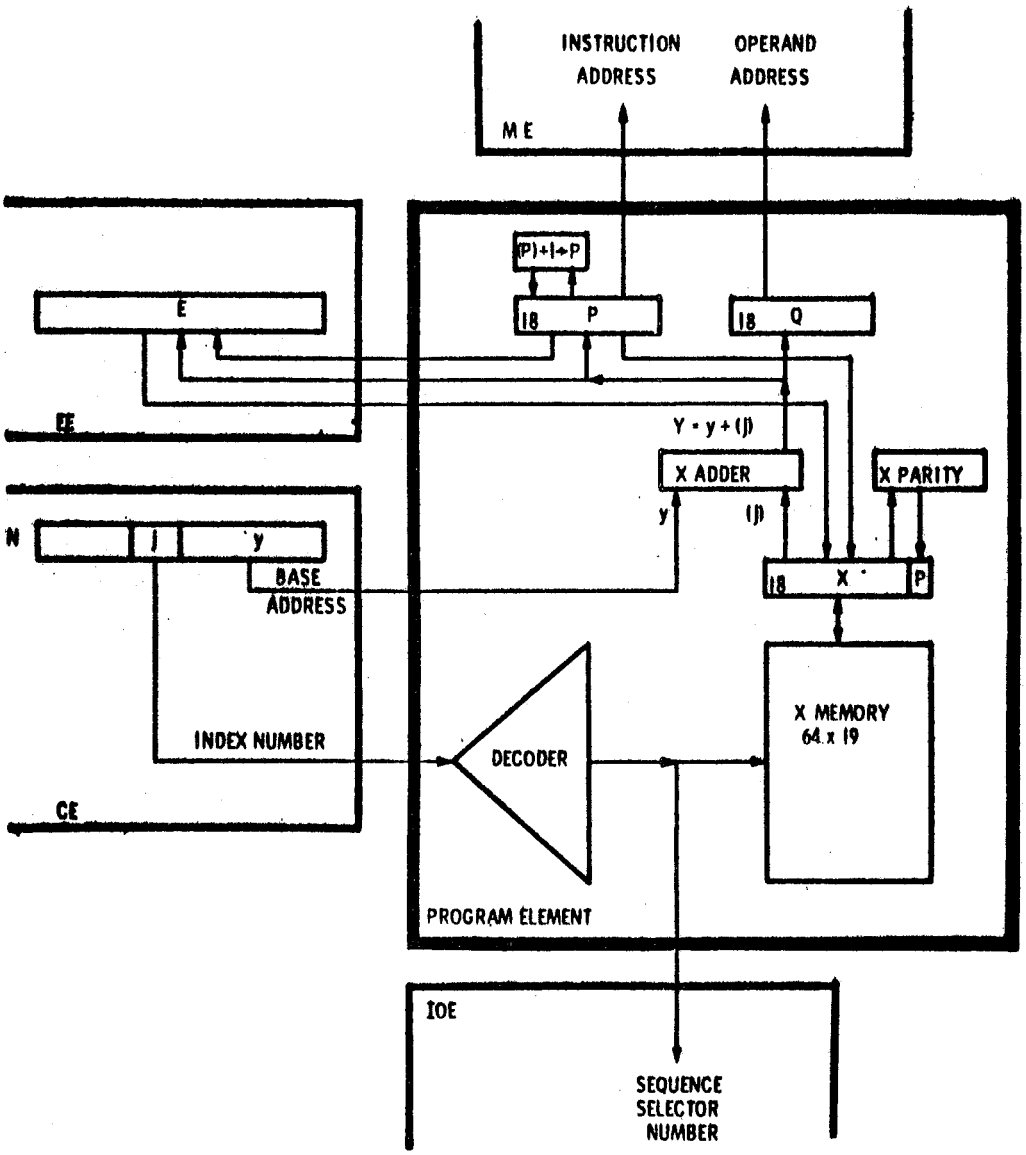

Fig. 6. Program Element Showing paths enabling index adding and storing and loading program counter from the index memory and the exchange element

6 digits are further subdivided into two groups of three. The first group determines the operation and the second specifies the register in the central computer that is being loaded, or whose contents are being stored.

The base address for the operand, formed by the 18 y-digits, is usually modified by the contents of the index register selected by the 6 j-digits. The index-registers form a unique, 64-register, parity-checked, core memory which has an access time of 1.0 μsec. The contents of the specified index register is read into the X-register of the program element via the paths indicated in Fig. 6. The base address and the index are fed into a full adder circuit which produces the sum, $X = y + (j)$, in about 1.0 μsec. The over-all complexity of the program element was reduced by having the adder produce both the sum, Y, and the unmodified base address, y: either of these quantities can be directed to the operand memory address register, Q. Whenever the FIRST ($j = 0$) index register is chosen, the adder produces only the unmodified base address. The effect is the same as having that index register contain ZERO and the programmer can avoid index modification altogether.

The P-register which holds the memory address of instruction words, normally is indexed by one as each instruction is executed, but the output of the index adder is sometimes directed to P when jump instructions are executed. The adder also provides a communication path for index jump instructions from the X-memory to the memory element by way of the exchange element.

E. Arithmetic Element (AE)

Fig. 7 shows the registers and sufficient basic operations in the arithmetic element for adding, multiplying, dividing, shifting, and executing various logic instructions. Operation timing for most of the TX-2 instructions is also performed in the arithmetic element.

The design of the AE reflects the desire to attain high-speed operation for TX-2, even when the AE is carrying out lengthy instructions. The only instructions which take longer than one memory cycle for execution are those which involve shifting. These are, for example, multiplication, division, arithmetic and cyclic shifting, and normalization. Therefore, the AE contains a sufficient number of storage registers to permit these instructions to be carried out in the AE while the remainder of TX-2 is free to execute other instructions.

The four registers in the AE can each communicate with the E-register in the exchange element and thus with the Memory Element. As mentioned earlier, these registers are addressable

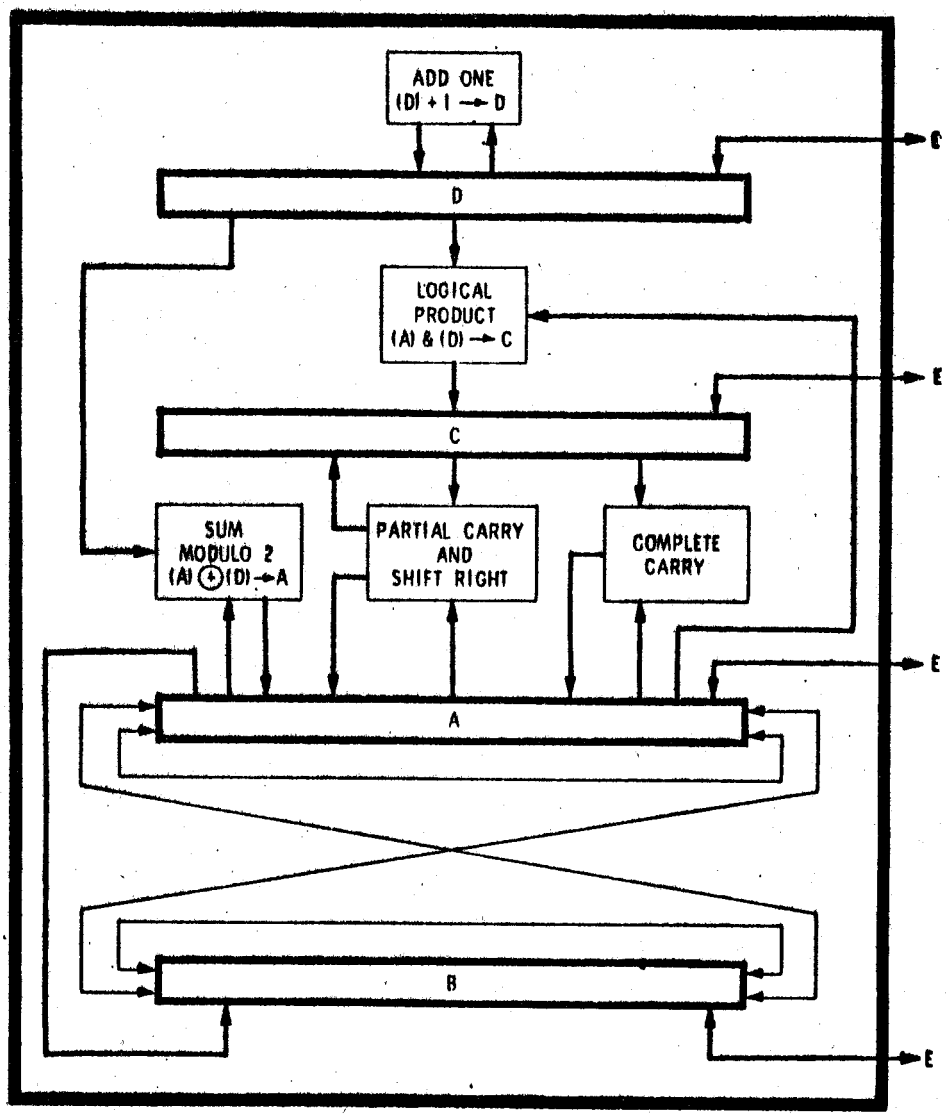

Fig. 7. Circuits and Transfer Paths
(General) Of Any TX-2 Arithmetic Element Forms

as part of the V-memory system. Therefore, programmers have access to the results in any register of an AE computation.

The AE registers, designated by A,B,C, and D, are described below:

The A-Register accumulates the results of all the arithmetic operations, except division, for which it holds the remainder. It holds one of the operands and accumulates the results of the three logic operations (AND, INCLUSIVE OR, EXCLUSIVE OR) which, it should be noted, are bit-wise operations. The information in the A-register can also be shifted (i.e. multiplied by some positive or negative power of two) or cycled (i.e. shifted, without preserving the special significance of the sign bit, as in a closed ring).

The B-Register serves as an extension of A during multiplication, certain shifts and cycles, and, in a sense, during division when the least significant digits of the double-length dividend are stored in B. The resulting quotient then appears in B. Moreover, the information in B can be shifted or cycled independently of A. In multiplication, the multiplier originally in A is transferred via parallel paths directly into B (where the least significant digit then controls the operation).

The C-Register stores the partial carries during arithmetic operations, which is most important during multiplication, as described later. Since these partial carries are actually bit-wise logic products AND, C is also used to accumulate logic products.

The D-Register holds the multiplicands, divisors, addends and one of the operands for the logic operations. It also holds the numbers which control the shifting and cycling of A and B, namely the number of places, up to 62, and the direction, right or left. The ability of D to count is used also to accumulate the results of normalizing A and counting ONES in A.

Each of the AE registers can also be complemented, thus allowing subtractions to be performed.

F. AE Circuits

There are four add-one circuits on D, so that different parts of A and B can be controlled separately and simultaneously. For simplicity, just one add-one circuit is shown in Fig. 7. These add-one circuits use the simultaneous carry principle, permitting one count every 0.4 μsec; each can count up to 127.

The logical-product (AND) circuit of A and D into C and the sum-modulo

2 (EXCLUSIVE OR) circuit of A and D into A when used at the same time are called a partial add. When the complete-carry circuit is activated after a partial add, the result is a full addition of D and A into A. The complete-carry circuit uses the high-speed carry principle and takes about 1.5 μsecs for 36 bits.

The partial-carry and shift-right circuit is also known as "multiply step" and was, we believe, first used on Whirlwind I. As it is used in multiplication, this circuit makes a full addition unnecessary for each ONE in the multiplier. Carries are extended only one stage during each step, except the last, when a complete carry is executed. The complete process takes about 16 μsecs at most for a full, 36-digit multiplication. The process for division, on the other hand, requires a complete addition at each step and consequently takes about 72 μsecs in the worst case.

Two features of the AE control should be mentioned here. A 7-bit step counter, like the add-one circuit on D, controls multiplication and division and limits the shifting in normalizing and the cycling in counting ONE's. A flip-flop, which signifies overflow during addition and division, is also used to remember the sign of the product during multiplication and the sign of the quotient during division. If division causes an overflow, the sign is replaced by the overflow state and the quotient is lost.

Control of the arithmetic element is independent of the rest of the machine. Thus, non-AE instructions can be executed while the AE continues to perform a long shift operation or a division.

G. System Timing

The high speed of TX-2 is attained in part by overlapping the operation of as many components as is logically possible without incorporating large amounts of circuitry. The time-consuming cyclic operations in an indexed, single-address computer are the instruction memory cycle, the index memory cycle, the index addition time, the operand memory cycle, and the operation timing. These cycles occur in the mentioned sequence during the execution of ordinary instructions. Several asynchronous "clocks," which use a 5-megacycle pulse-source, control the cycles. The instruction and operand memory cycles can be overlapped if they take place in different memory systems.

The overlap of these cycle times for a sequence of load-type instructions is illustrated in Fig. 8 (a). Here different

instruction and operand memory systems are assumed to have roughly equal cycle times. If a sequence of store-type instructions is executed which requires extended memory cycles for the operands, then the situation is as shown in Fig. 8 (b). Fig. 8 (c) shows the time used when both the instruction and the operand are in the same memory.

"Peak" operating speed for the computer is attained only when the situation is as shown in Fig. 8 (a); in the situation shown in Figs. 8 (b) and 8 (c) the operating speed could be increased by adding circuitry, but only at considerable cost. It is interesting to note that if the computer is to run at peak speeds, the address of the operand used by the present instruction must be available before the earliest moment at which the next instruction memory cycle could begin. If the total accumulated time, from the beginning of an instruction memory cycle till the time that the address of the operand is known, is greater than the instruction memory cycle time, then the computer cannot run in the ideal manner shown in Fig. 8 (a). This means that the access time of memories and the index-add time must be kept as short as possible.

Fig. 8 (d) shows the timing of events when the in-out element causes a change in program sequence by changing the contents of the P-register. The additional X-memory cycle, which must be performed, produces a timing situation similar to that of the X-memory load and store instructions.

The operation timing for an instruction is executed when the operand is available from memory. Only the arithmetic element step-counter instructions (multiply, divide, shift, etc.) require an operating timing cycle longer than a memory cycle. Since only the arithmetic element is tied up by these instructions, the control element permits any non-arithmetic element instruction to be executed while the AE is busy. Division takes up to 75 μsecs, so the programmer can write as many as 14 non-AE instructions following a divide, all of which can be executed before the division is completed.

H. Configuration Control

The design of a general-purpose computer must necessarily reflect the contradictory demands for both short and long words, floating and fixed-point arithmetic operations, and a multitude of logic and decision instructions. The computer should be able to process information at an optimum rate, in a variety of problems, without the need for intricately coded programs. This ability should be achieved without excessively complex and costly circuitry.

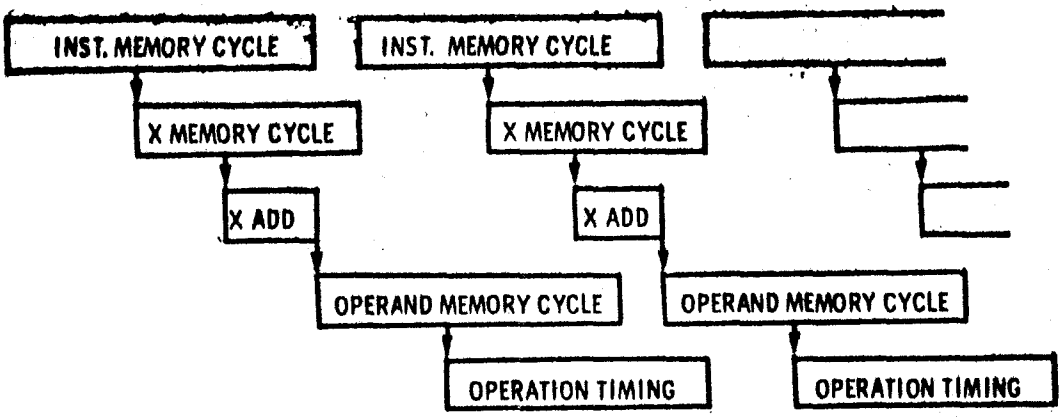

Fig. 8 (a) Consecutive Load Type Instructions
Instructions and Operands in Different Memories

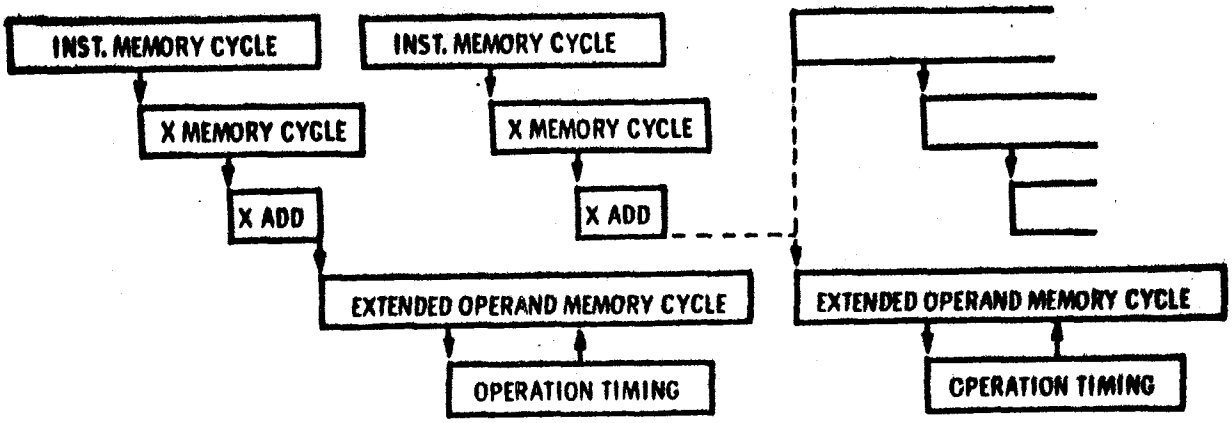

Fig. 8 (b) Consecutive Store Type Instructions

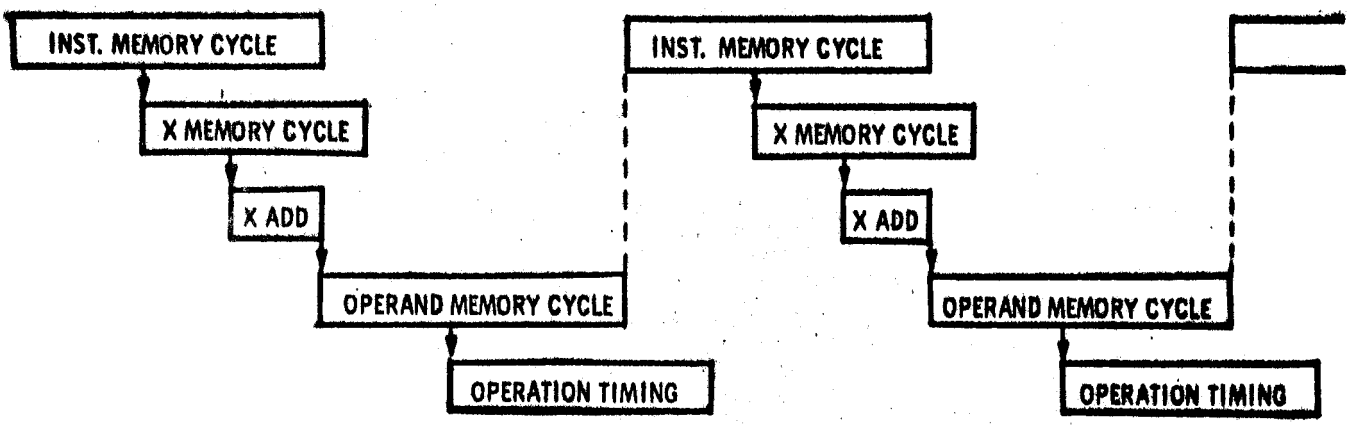

Fig. 8 (c) Instruction and Operand In Same Memory

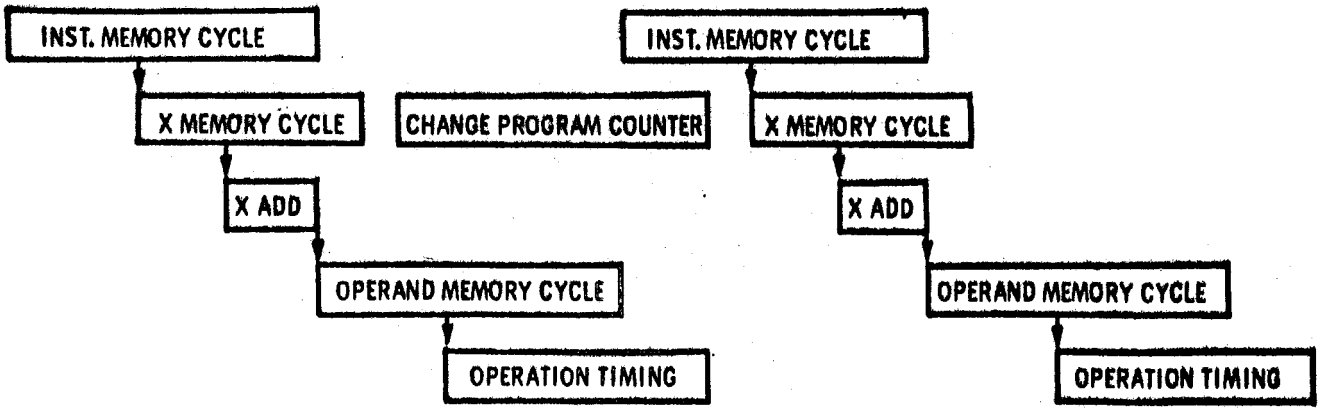

Fig. 8 (d) Change Sequence

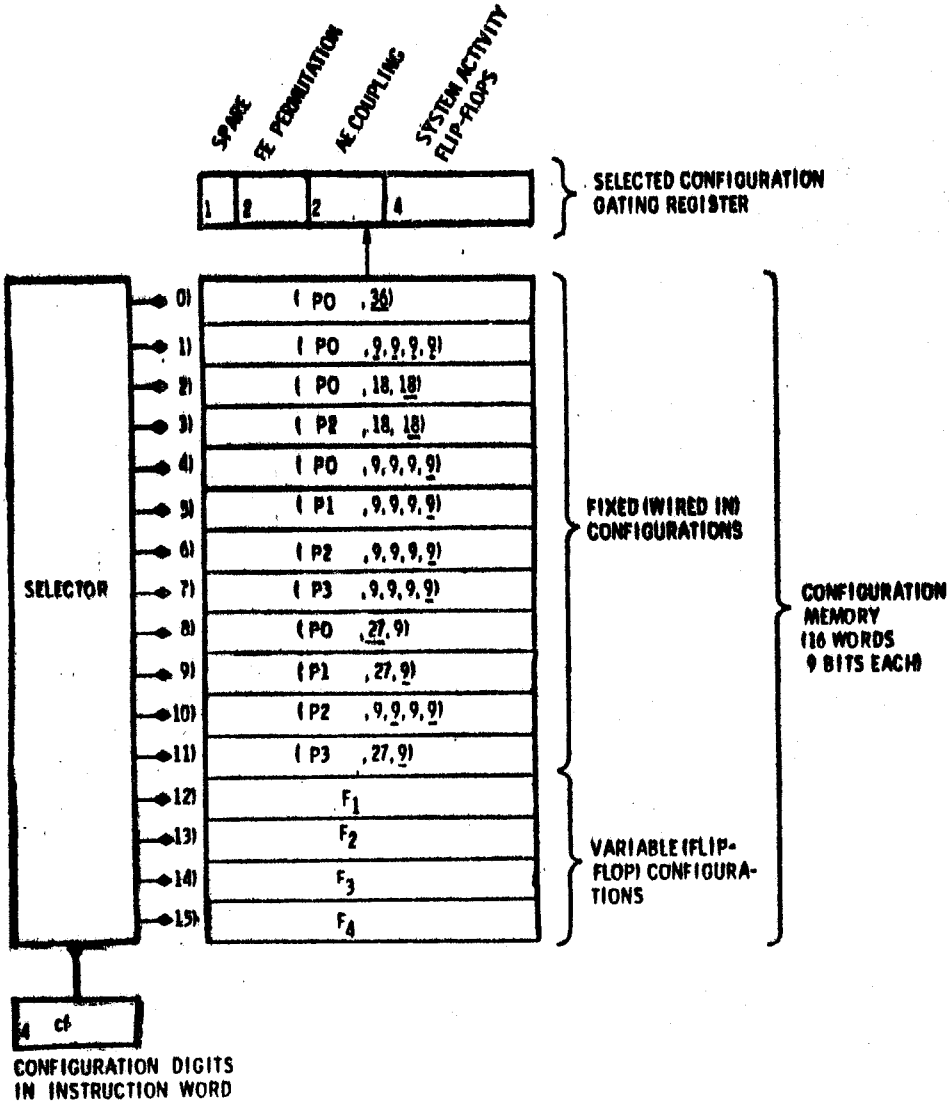

Fig. 9. TX-2 Configuration Selection

The 36-digit word of TX-2 represents a reasonable length for operands in some numerical computations, notably scientific and engineering computations. Though floating-point arithmetic operations are not included in the instruction code, both they and multiple-precision operations can be easily synthesized by means of the existing instructions. The logic instructions in the code facilitate operations on individual digits. Also, a configuration which the programmer specifies with each instruction permits him to perform arithmetic operations on operands that are less than 36 digits long. When operands are less than 36 digits, several can be manipulated simultaneously.

The four cf digits in an instruction word (see Fig. 5) are decoded as shown schematically in Fig. 9, selecting one of 16 configurations. The selected configuration, represented by a 9-digit word, is placed in a flip-flop register whose output levels determine a static configuration for the entire computer while the instruction is executed. The notation used to represent the contents of the first 12 registers will be clarified in the following discussion.

The full 36-digit word length is always maintained for instruction words, but during operation timing, every 36-digit register in the memory, exchange and arithmetic elements is considered as broken into 9-digit quarters, numbered from 1 to 4, right to left as in Fig. 10 (a). While the instruction is being executed, these quarters are recombined on the basis of the configuration.

Information is usually moved about in the machine by parallel transfers between registers (see Fig. 9). The EE permutation digits select one of the four permutations P0, P1, P2, P3, shown in Fig. 10 (b). The chosen permutation effects the corresponding cross-communication paths between the quarters of the E- and M-registers of the exchange element. While operands are transmitted through the EE, the quarters follow the set of paths determined by the selected permutation. The result is that the operand is shifted 9n places to the left as it moves from M to E or 9n places to the right as it moves from E to M, n = 0,1, 2 or 3. Thus, the programmer can have any quarter of the AE communicate with any quarter of the ME.

This ability to communicate is focused more sharply by having the configuration specify a system activity (see Fig. 9). All operation timing events, in a given quarter of the AE and EE and the quarter of the ME connected via the selected permutation path in the EE, are controlled by the activity flip-flop on that quarter. If the activity flip-flop of a given quarter is a ONE, as specified by the configuration, the operation timing

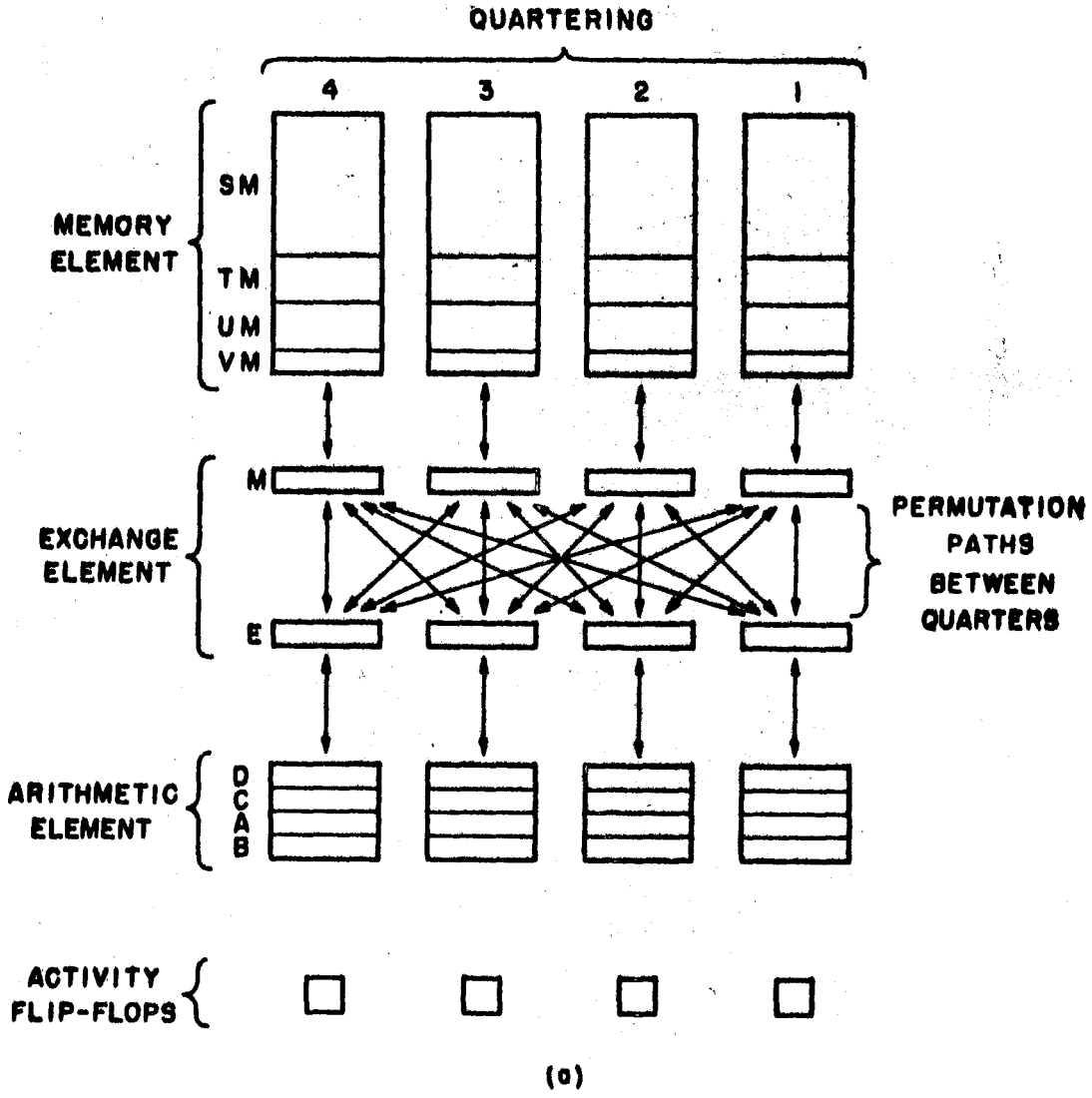

Fig. 10 (a) Quartering Permutation Paths and Activity Flip-Flops Shown

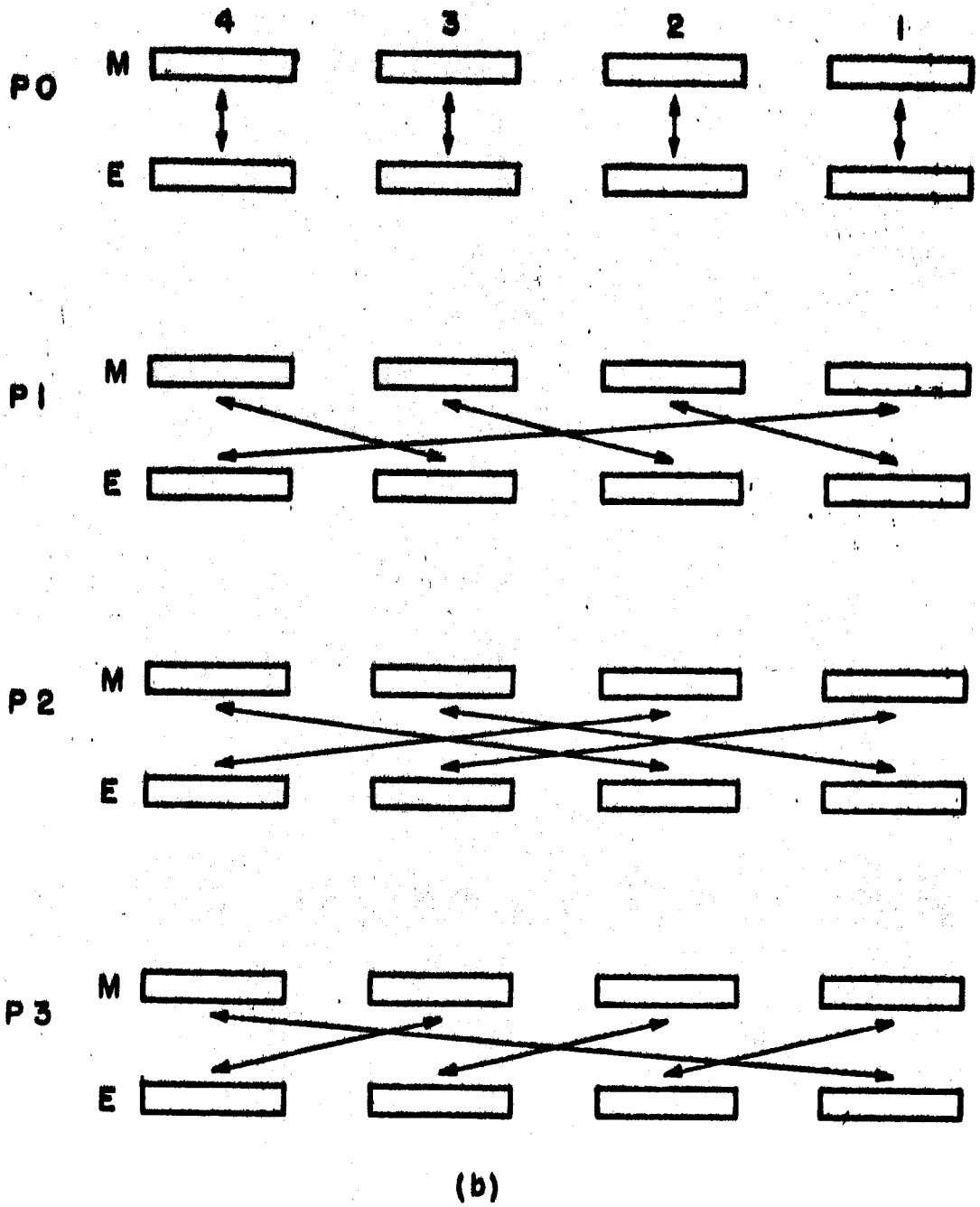

Fig. 10 (b) Paths in Exchange Element

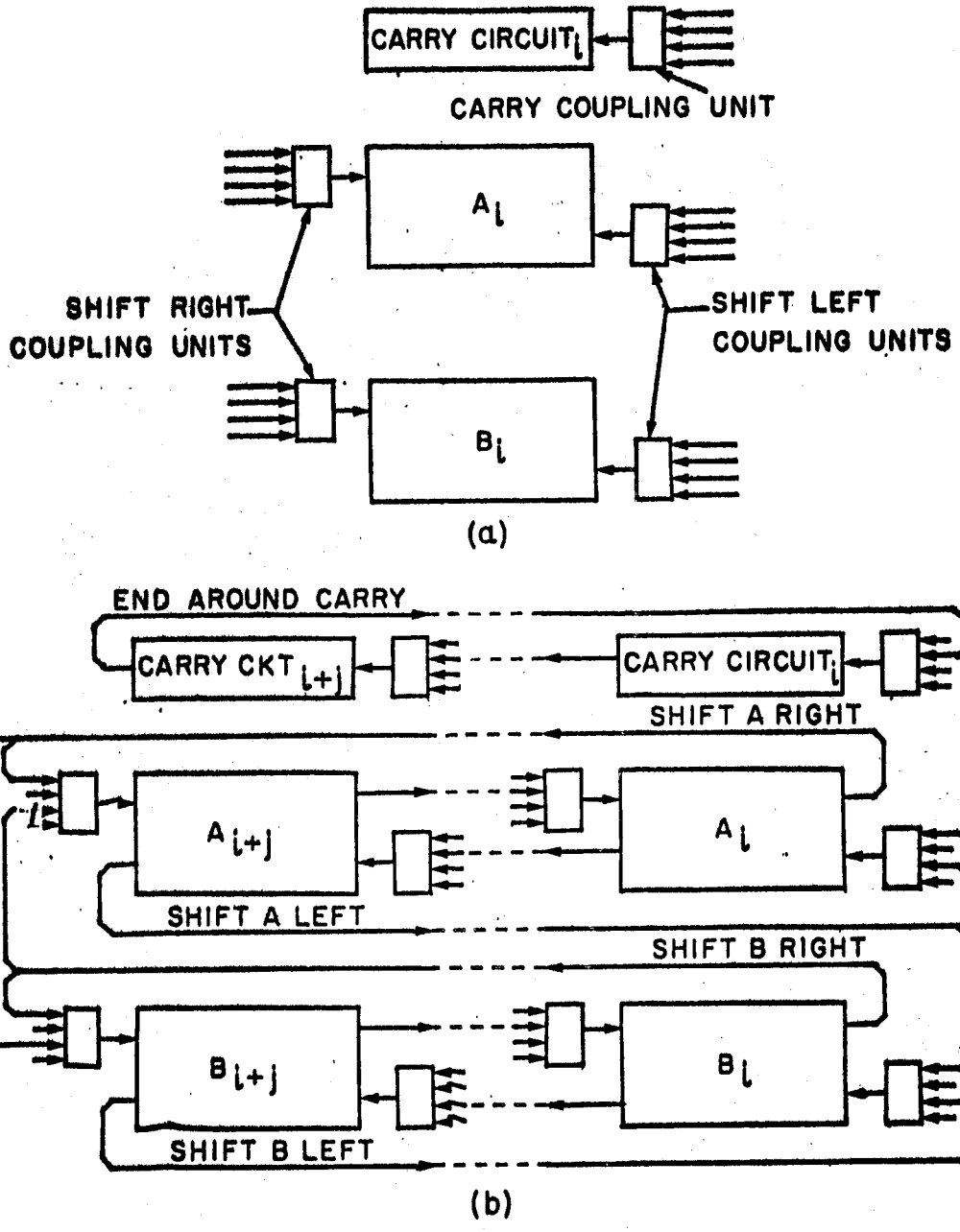

Fig. 11 (a) ith Quarter Coupling Units
(b) Coupling Unit Connections

events of the instruction occur in that quarter. If the activity flip-flop is a ZERO, nothing happens.

During the execution of arithmetic operations, the AE coupling digits (see Fig. 9) further specify the connections of the lateral information paths between quarters in the AE. Information flows laterally only through the shift and the carry circuits, and the connection of these circuits alone determines the word lengths of the numerical quantities manipulated in the AE.

As shown in Fig. 11 (a), every quarter of the AE has coupling units at each end which receive the shift and carry information entering the quarter. The general type of connections between several quarters is shown in Fig. 11 (b). The digit length of operands, during add and shift operations, is determined by the number of quarters coupled together. In TX-2, from 1 to 4 quarters can be coupled together to permit arithmetic operations on 9-, 18-, 27-, or 36-digit operands. The various combinations of coupling-unit connections actually chosen by the AE coupling are symbolized in Fig. 11 (c). Since A-register, B-register, and AB-register shifts are permitted in the arithmetic element, the programmer can obtain 18-, 36-, 54-, or 72-digit shifts. All the possible shift (and cycle) configurations are shown in Fig. 11 (d).

Only those inputs to the coupling units which would yield useful arithmetic element structures are realized by the AE coupling. It should be emphasized that the programmer can realize several arithmetic elements simultaneously. The coupling, (36), gives only one 36-bit arithmetic element, but the coupling, (18,18), gives two complete, independent, 18-bit arithmetic elements which are separately, but simultaneously, controlled by the instruction being executed. Two arithmetic elements are again available with the coupling, (27,9), one 27 bits and the other 9 bits long, and the (9,9,9,9) coupling gives four, 9-bit arithmetic elements. The permutation paths in the exchange element permit each arithmetic element to communicate with any quarter of a memory word and the activity flip-flops can specify just which of the realized arithmetic elements actually will be active and will communicate with the connected part of memory.

In Fig. 12, several examples are given of the different configurations which can be realized in TX-2. The most straightforward has one, 36-digit arithmetic element and communicates directly with memory. The notation, (P0, $\underline{36}$) signifies the permutation (no shift) and the form of the arithmetic element (one 36-digit). The underlining indicates that the whole system is active. Slightly more varied is the (P0, 9,$\underline{9}$,9,$\underline{9}$) configuration which specifies four, 9-digit arithmetic elements communicating directly with memory, but with only two of them active. The (P2, 9,$\underline{9}$,9,$\underline{9}$) configuration

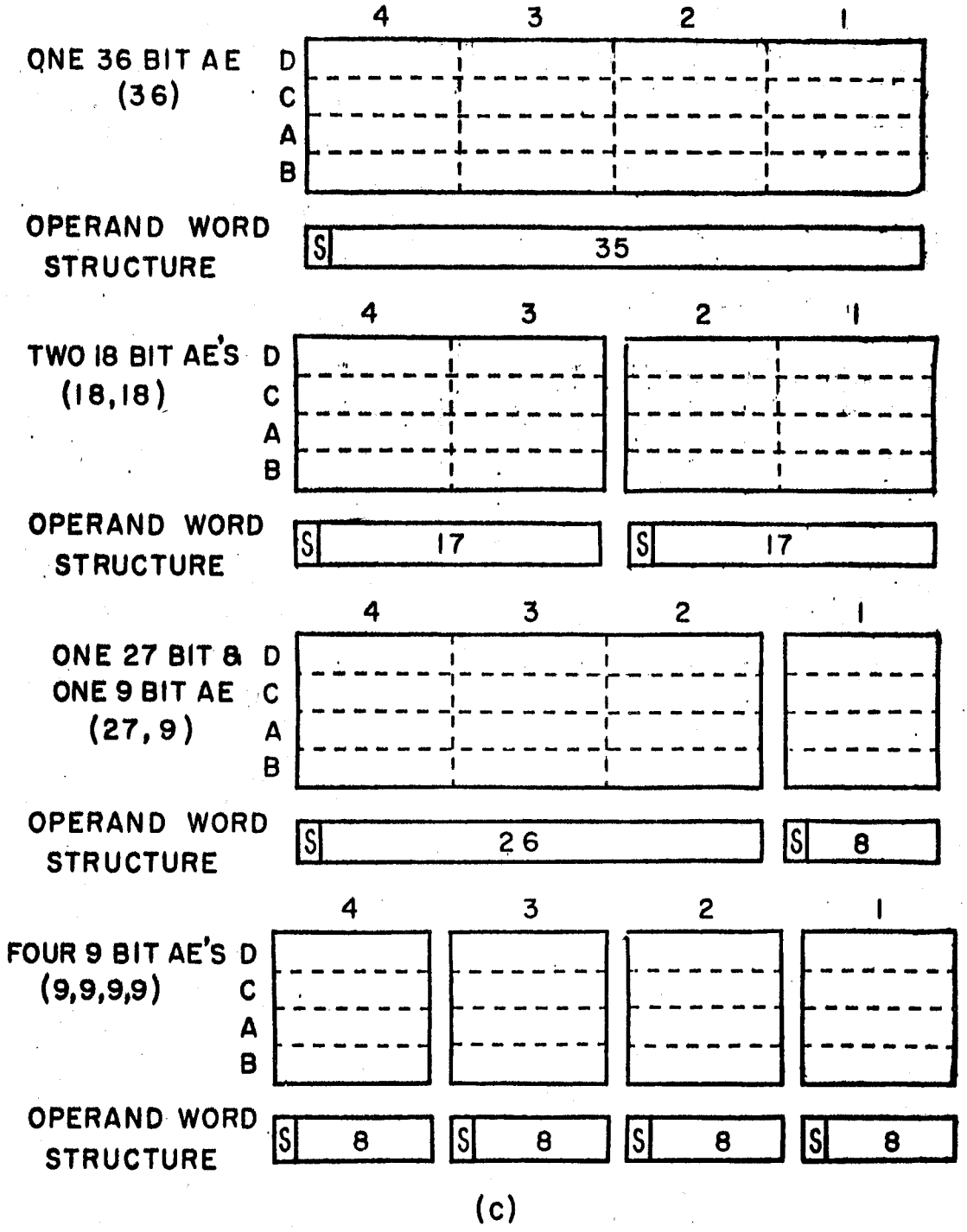

Fig. 11 (c) TX-2 Configuration - Arithmetic Elements
And Operand Word Structures

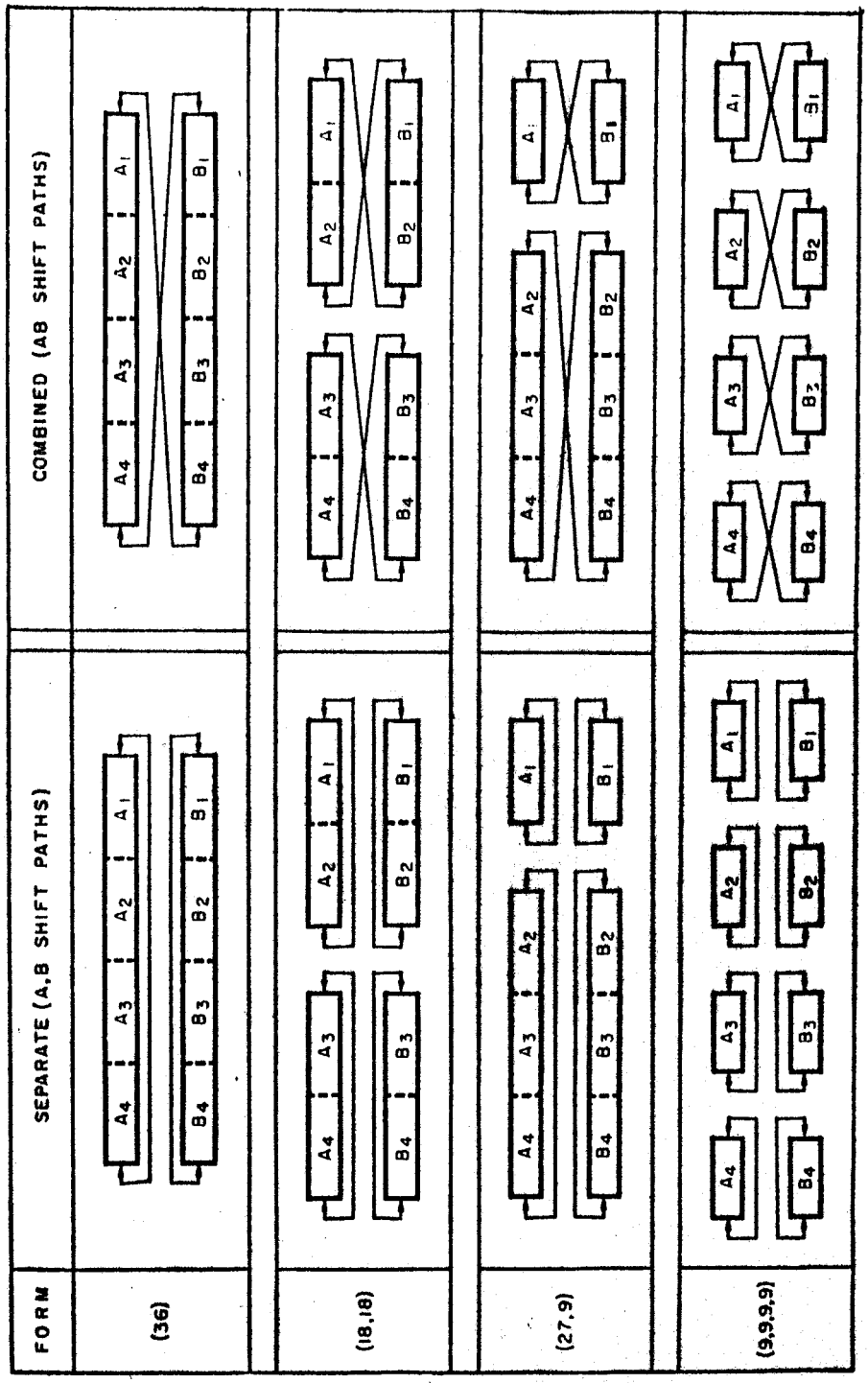

Fig. 11 (d) TX-2 Shift Path Arrangements

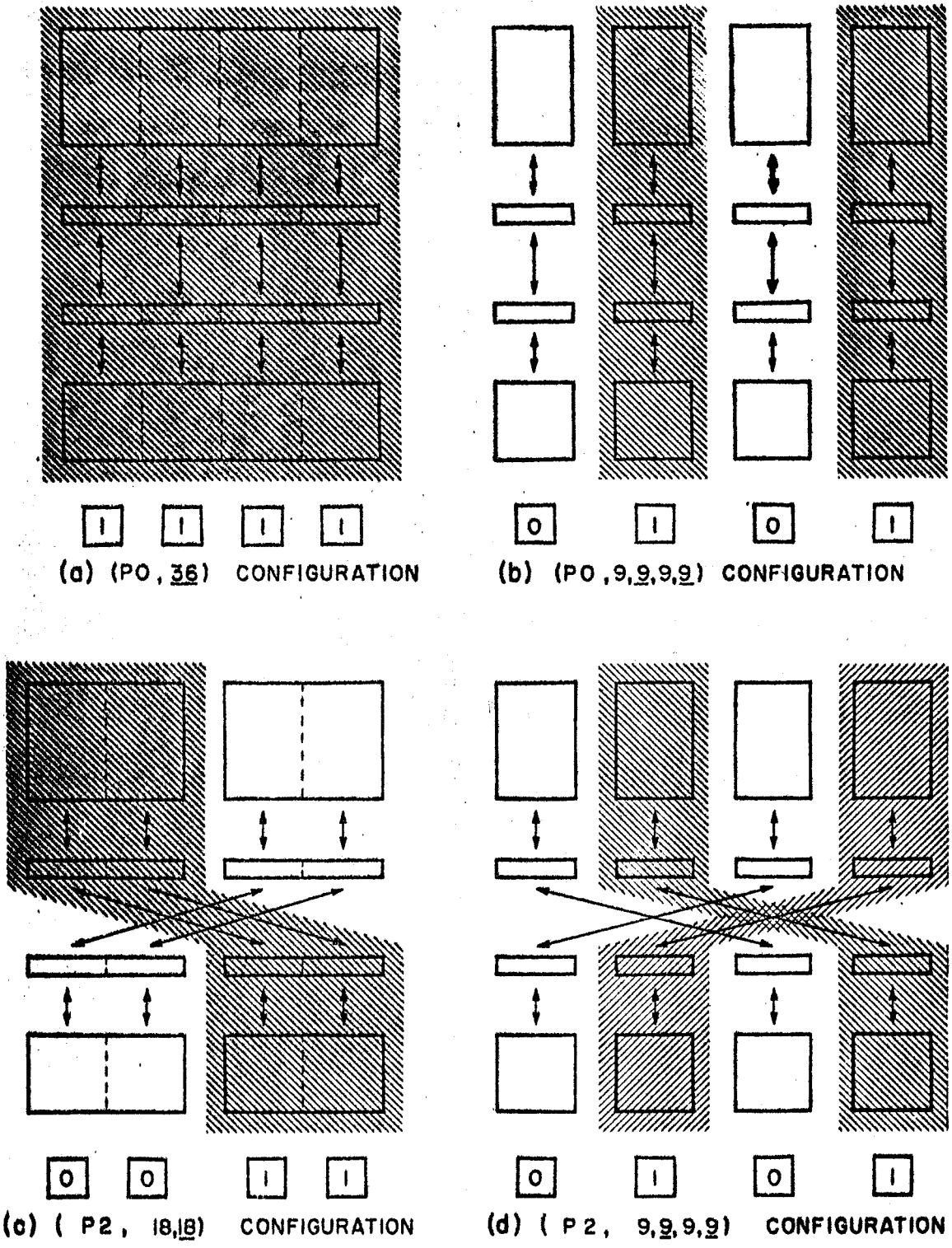

Fig. 12. TX-2 Configurations
Areas Of Activity During Execution Of Instruction Shown Shaded. Effect Of AE Couplings Illustrated By Juxtaposition Of Quarters

has the same arithmetic elements but with the associated memories interchanged. The (P2, 18,18) configuration illustrates an 18-digit arithmetic element which uses the "other" half of memory.

One of the digits in the 9-digit configuration is at the moment unused, but will probably be used to control the extension of the sign of numbers as they pass through the EE on the way from the ME to the AE. The scheme presently under consideration would permit programmers, for example, to add a 9-digit memory operand to an 18-digit arithmetic element. This scheme would permit closer packing of operands in memory and significantly increase the speed of solving some real-time problems, where the sign of short pieces of data must be extended so that higher precision can be maintained during computations. The working details of the scheme have yet to be fixed.

The memory from which the programmer chooses a configuration for use with each instruction was shown in Fig. 9. Twelve of the memory registers are fixed circuitry whose contents cannot be changed without changing the wiring of the computer. The configurations represented by the contents of these registers are assumed to be useful to most programmers. The last four registers in the memory consist of the 36 digits of the F-register. As will be seen, the programmer can quite simply alter the contents of this register and thereby obtain any of the possible configurations. The total number of distinct possible configurations is less than 2^9.

I. Operation Code

Only 51 of the 64 possible operation codes are currently decoded to define instructions. In Table I the effect of each instruction is described. If the selected configuration defines several computers, the operation takes place in all of them simultaneously and independently. The notation used in the definition of the operation is described in Table II.

The instructions are grouped according to type. Load and store instructions transfer an operand between a selected register and memory. The load-complement instructions are variants which load the ONES complement into the selected registers. The Exchange instruction interchanges the contents of A and of the addressed memory register. The insert instruction allows any set of bits in A, as specified by the bits in B, to be stored in memory. The j-bits of the load and store instructions, which refer to the index memory, select the index register involved so that the operand address is not modified.

All of the add and step-counter instructions can also be classed as load type instructions, in so far as the operand memory cycle

is concerned. The multiply instruction forms the full product in the A and B registers. Division is the inverse of multiplication, the double-length divident in A and B being divided by the memory operand. The remainder is left in A and the quotient in B. Normalize instructions shift the contents of A and B left until the magnitude of the number in A is between one-half and one. The number of shifts to do this (the normalizing factor) is subtracted from the memory operand in D. Since more than 18 bits are required to specify all the possible shifts for the (9,9,9,9) configuration, the shift and cycle instructions use a memory operand, rather than the address section of the instruction to specify the number of places to shift. The count ones instruction adds the number of ONE's bits in A to the memory operand in D, thus providing a simple means for determining bit density in areas of storage, since the ONE's count for several words can be accumulated in D.

The two replace-add instructions, using the index memory, facilitate instruction and index modification. Both require two memory-cycle times for execution.

The two in-out read instructions transmit information between the memory and the selected in-out unit. The details of these instructions and the in-out select instruction are given in Section III.

Single bits in memory can be manipulated with the three bit-setting instructions. The bit-sensing instruction facilitates the use of single bits in memory as operands. The variety of available jump instructions simplifies the coding of, logic decision functions. The two index-jump instructions permit indexed program loops to refer successively in either the forward or backward direction to operands in a data block. The unconditional-jump instruction uses the cf digits to specify whether the selected index register will be used to remember the previous contents of P. These contents are always transmitted to the E-register whenever a jump occurs.

Arithmetic overflows can be caused by addition, subtraction, and division instructions. Such overflows as do occur are remembered in overflow flip-flops in the arithmetic element. The overflow condition can be detected by a jump instruction, or by the in-out element in a manner described in Section III. If an overflow is anticipated, however, it can be shifted into the A register by executing a normalize instruction. A normalize usually shifts AB left. However, if an overflow exists, AB is shifted right one place and the overflow placed in the most significant digit position of A to the right of the sign digit. The memory operand is also increased by one in the D register, rather than decreased. This

interpretation of an overflow permits floating-point operations to be programmed very simply in the arithmetic element. The in-out select and operate instructions differ from all the others in the sense that the y-digits are used to specify different operations. In-out select chooses the mode in which an in-out unit will run. The operate instruction will control individual useful commands, e.g., round-off.

J. Instruction Times

The average execution time for instructions depends upon whether one memory or two different, overlapped memory systems are used for instructions and operands. If one memory is used, the execution time is the sum of the instruction and operand cycle times; if two memories are used, the execution time is the longer of the two memory cycle time. It should be remembered that any instruction which involves storing an operand in memory has the normal operand memory cycle time extended by from 1.0 to 2.0 μsecs. Instructions which alter or transfer the contents of index memory registers require approximately two memory cycles, even when instruction and operand memory cycles are overlapped.

Successive step-counter instructions require a time which depends upon the length of the longest active arithmetic element. In the case of multiply, divide, and count ONE's, this time is a function of the length of the operand word only, but the shift, cycle, and normalize times depend upon the number of places actually shifted. Divide requires about 2.0 μsecs per digit and all other step-counter instructions 0.4 μsecs per digit. These shift times become significant only when they exceed the one or two memory cycles already required. In the worst 36-digit case, about 75 μsecs are required for division and 19 μsecs for multiplication. A 72-place shift would take 32 μsecs. These are the times required for these instructions when they are written in sequence. If the operand word is shorter, then these times become proportionally less, down to the minimum memory times required.

K. Summary

The organization of TX-2 permits a programmer, who pays considerable attention to coding details, to receive a worthwhile reward in the form of increased efficiency of operation. The operating speed can be doubled when instruction and operands are stored in different memories. Higher speeds result from sequencing instructions so that non-AE instructions are executed concurrently with AE step-counter instructions. And the ability to choose a configuration with each instruction means not only that some instructions take less time, but also that many of them can be eliminated from a program altogether.

However, this versatility and efficiency is not accompanied by a

disastrous loss in simplicity. The system organization is such that details can be easily ignored by the naive programmer, without the details having even subtly unfavorable effects. If all the digits in an instruction word are ZERO, except for the operation code and the base address, then TX-2 appears as a simple single-address computer with operand words 36-bits long and with a single, uniformly addressed, 70,000-word memory.

If the j-bits are used, then the machine is enlarged to become an indexed, single-address, computer, with operand words 36 bits long, for which the entire instruction code is meaningful. When the b and d bits are used, the programmer can control the manner in which several in-out units, running concurrently, can cause the computer to change program sequences. And by selecting various configurations, the programmer can perform more operations simultaneously with each instruction.

The different facilities for indexing, memory overlap, instruction overlap, multiple-sequencing, and configuration can be ignored or used as the programmer desires. Ignoring them would seem to permit straight-forward coding; using them actually permits much shorter and faster codes for a given function. Each facility is represented by a clear concept of what the facility permits, the only real difficulty being the number of simultaneous actions possible with each instruction. However, higher speeds and greater system capacity are obtained by shorter cycle times, increased bit storage and greater simultaneousness of events. In TX-2, all three aspects are emphasized.

TABLE I

Type	Mnemonic Code	Operation	Name
Load	lda	$(Y) \rightarrow A$	Load into A
	ldb	$(Y) \rightarrow B$	Load into B
	ldc	$(Y) \rightarrow C$	Load into C
	ldd	$(Y) \rightarrow D$	Load into D
	lde	$(Y) \rightarrow E$	Load into E
	ldE	$(Y) \rightarrow E$	Load into E
	ldf	$(Y) \rightarrow F$	Load into F
	lca	$\overline{(Y)} \rightarrow A$	Load complement into A
	lcb	$\overline{(Y)} \rightarrow B$	Load complement into B
	ldx	$(y) \rightarrow j$	Load into index
Store	sta	$(A) \rightarrow Y$	Store A
	stb	$(B) \rightarrow Y$	Store B
	stc	$(C) \rightarrow Y$	Store C
	std	$(D) \rightarrow Y$	Store D
	stf	$(F) \rightarrow Y$	Store F
	exa	$(Y) \rightarrow A$; $(A) \rightarrow Y$	Exchange A
	ins	$(B) \& (A) \vee (B) \& (Y) \rightarrow Y$	Insert digits of A
	stx	$(j) \rightarrow y$	Store index
Add	add	$(A) + (Y) \rightarrow A$	Add
	sub	$(A) + \overline{(Y)} \rightarrow A$	Subtract
	dma	$\lvert(A)\rvert + \lvert\overline{(Y)}\rvert \rightarrow A$	Difference of magnitude
	and	$(A) \& (Y) \rightarrow A$	Logical and
	ori	$(A) \vee (Y) \rightarrow A$	Logical or - inclusive
	ore	$(A) \oplus (Y) \rightarrow A$; $(A) \& (Y) \vee (C) \rightarrow C$	Logical or - exclusive (and accumulate product)
	axm	$(j) + (y) \rightarrow y$	Add index to memory
	amx	$(j) + (y) \rightarrow j$	Add memory to index
Set bit	sbo	$1 \rightarrow Y_j$	Set j-th bit one
	sbz	$0 \rightarrow Y_j$	Set j-th bit zero
	sbc	$(Y_j) \rightarrow Y_j$	Set j-th bit complement

TABLE I (Continued)

Type	Mnemonic Code	Operation	Name
	mul	(A) x (Y) → AB	Multiply
	div	(AB) ÷ (Y) → A (remainder), B (quotient)	Divide
	sha	(A) x $2^{(Y)}$ → A	Shift A
	sab	(AB) x $2^{(Y)}$ → AB	Shift AB together
	shb	(B) x $2^{(Y)}$ → B	Shift B
Step-Count	cya	(A) cyc (Y) → A	Cycle A
	cab	(AB) cyc (Y) → AB	Cycle B together
	cyb	(B) cyc (Y) → B	Cycle B
	nab	(AB) x 2^{nf} → AB; (Y) - nf → D	Normalize AB
	coa	(Y) + no → D	Count ones in A
	rds	(Y) → 10; (10) → Y	Read and shift
In-out	rdn	(Y) → 10; (10) → Y	Read without shift
	jpe	If (E_j) = 1, then y → P	Jump if j-th bit of E is a one
	jpp	If any (A)>0, then Y → P	Jump if the contents of any A is positive
	jpn	If any (A)≤0, then Y → P	Jump if the contents of any A is negative
	jpz	If any (A) = 0, then Y → P	Jump if the contents of any A is zero
	jpo	If any (A) overflowed, then Y → P	Jump if the contents of any A has overflowed
Jump	jxp	If (j)≥0, then (j)-cf→j, y → P	Jump if index positive and decrease index
	jxn	If (j)<0, then (j)+cf→j, y → P	Jump if index negative and increase index
	jpu	If cf = 1,3, then (P)+1→j; If cf = 0,1, then y → P; If cf = 2,3, then Y → P	Jump unconditionally
Misc.	ios		In-out select
	opr		Operate

TABLE II

Notation	Meaning
$\longrightarrow$	goes into
(x)	contents of x
$Y = y + (j)$	indexed memory address
$\lvert(x)\rvert$	magnitude of (x)
$\overline{(x)}$	one's complement of (x)
$\&$	logical and operation
$\vee$	inclusive or operation
$\oplus$	exclusive or operation
$+$	one's complement addition
nf	number of [illegible] to normalize
no	number of ones
Y_j	j-th digit of register Y

III. THE LINCOLN TX-2 INPUT-OUTPUT SYSTEM

A. Introduction

The input-output system of the Lincoln TX-2 computer contains a variety of input-output devices suitable for general research and control applications. The system is designed in such a way that several input-output devices may be operated simultaneously. Since the computer is experimental in nature, and changes in the complement of input-output devices are anticipated, the modular scheme used will facilitate expansion and modification. The experimental nature of the computer also requires that the input-output system provide a maximum of flexibility in operating and programming for its input-output devices.

The input-output devices, currently scheduled for connection to TX-2, include magnetic-tape units for auxiliary storage; photoelectric paper-tape readers for program input; a high-speed printer, cathode-ray-tube displays, and Flexowriters for direct output; analog-to-digital conversion equipment; data links with other computers; and miscellaneous special-purpose equipment. This section will not be concerned with the details of these devices, but will limit itself to a discussion of the logical incorporation of them into the system.

In describing the TX-2 input-output system, occasional reference will be made to certain aspects of the design of other parts of the TX-2 as set forth in Section II.

B. The Multiple-Sequence Program Technique

Of the various organizational schemes which permit the simultaneous operation of many devices, we have chosen the "multiple-sequence program technique" for incorporation in TX-2. A multiple-sequence computer is one that has several program (instruction) counters. If the program sequences associated with these program counters are arranged to time-share the hardware of the central computer, a machine can be obtained which will behave as if it were a number of logically separate computers. We call these logical computers sequences and therefore refer to TX-2 as a multiple-sequence computer. By associating each input-output device with such a sequence, we effectively obtain an input-output computer for each device.

Since the one physical computer in which these sequences operate is capable of performing only one instruction at a time, it is necessary to interleave the sequences if they are to operate simultaneously. This interleaving process can take place aperiodically to suit the needs of and under the control of, whatever individual input-output devices are operating. The number of sequences which can operate simultaneously, and the complexity of the individual sequences, is limited by the peak and average data-handling rate of the central computer hardware.

In a multiple-sequence computer, the main body of the computation can be carried out in any sequence, but if maximum efficiency of input-output operation is to be achieved, the bulk of arithmetic operations must be confined to a few special sequences, called main sequences, which have no associated input-output devices. The input-output sequences may then be kept short, and a large number of them can be executed at once.

C. Multiple-Sequence Operation in TX-2

In TX-2, one-half of the index-register memory has been made available for storing program counters. Thus, a total of 32 sequences may be operated in the machine. (Actually an additional sequence of special characteristics is obtained by using index register number 0 as a program counter. This special sequence will be discussed later). Some of these sequences are associated with input-output devices. Others perform functions, such as interpreting arithmetic overflows that are called into action by conditions arising within the central computer. Finally, there are the main sequences which are intended to carry out the bulk of the arithmetic computations performed by the machine.

A priority scheme is used to determine which sequence will control the computer at a given time. If more than one sequence requires attention at the same time, control of the machine will go to the sequence having the highest priority and instructions addressed by its program counter will be executed.

Table I is a list of the sequences currently planned for inclusion in TX-2. They are listed in approximate order of priority with the highest at the top. Asterisks mark sequences which are not associated with any particular in-out device. A special sequence (number 0) has first priority and will be used to start any of the other sequences at arbitrary addresses. The next two sequences interpret alarms (under program control). These three sequences have the highest priorities, since they must be capable of interrupting the activities of other sequences. The input-output devices follow, with high-speed, free-running units carrying next highest priorities. The main sequences (we anticipate three) are at the bottom of the list. The priority of any sequence may be easily changed, but such changes are not under program control. Priorities are intended to remain fixed under normal operating conditions. The list totals about 25 sequences, leaving eight spaces for future expansion.

Switching between sequences is under the control of both the input-output devices (generalized to include alarms, etc.) and the programmed instructions within the sequence.

Once a sequence is selected and its instructions are controlling

the computer, further switching is under control of the programmed instructions. Program control of sequence switching is maintained through two bits, called the break and dismiss bits, in each instruction. The break bit governs changes to higher-priority sequences. When the break bit permits a change, and some higher-priority sequence requests attention, a change will be made. The dismiss bit indicates that the sequence has completed its operation (for the moment, at least) and that lower-priority sequences may receive attention. The interpretation of the break and dismiss bits will be discussed in more detail.

D. The TX-2 Input-Output Element

The TX-2 input-output element is shown schematically in Fig. 13. It consists of a number of input-output devices, associated buffers, and a sequence selector. Each device has enough control circuitry to permit it to operate in some selected mode once it has been placed in that mode by signals from the central computer. Associated with each device is a buffer storage of appropriate size. This buffer may be large or small, to suit individual data-rate requirements, but the buffers used in TX-2 will generally be the smallest possible. For the most part, buffering for only one line of data from the device (e.g., 6 bits for a paper-tape reader) will be provided. Each input-output device is associated with one stage of the sequence selector. The sequence selector provides the control information necessary for proper interleaving of the program sequences. When it is desired to add a new input-output device to the computer, the three packages, in-out unit, buffer, and sequence-selector stage, must be provided.

As shown in Fig. 13, data is transferred between the input-output element and the central computer by way of the exchange element. Fig. 13 indicates two-way paths between the E-register and all in-out buffers. Actually, most devices are either readers or recorders, but not both, and therefore require one-way paths only. Only the necessary paths are provided; the drawing simply shows the most general case.

Signals from the sequence selector connect the appropriate buffer register to the E-register to transfer data. When a sequence is selected (i.e., its program counter is supplying instruction locations), the associated buffer is connected to the E-Register, and all other buffers are disconnected. A read instruction will effect a transfer of information between the buffer and the E-register. A particular buffer is thus accessible only to read instructions in the sequence associated with the buffer's in-out unit.

Fig. 13 shows paths from the Sequence Selector to a coder which proves an output called the program-counter number. These paths are used in the process of changing sequences to be described in a later section.

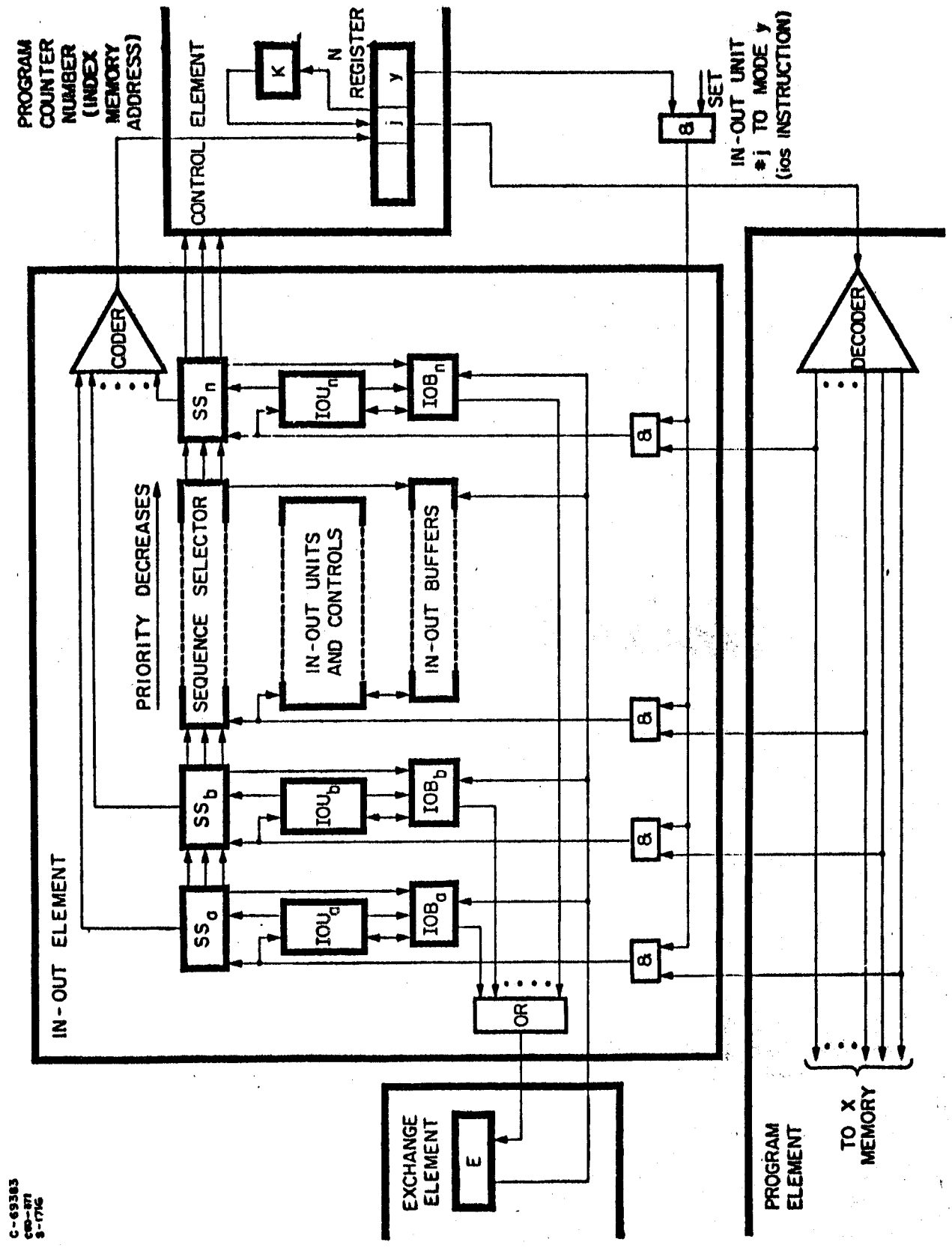

Fig. 14. TX-2 In-Out Element

Fig. 13 also shows paths for mode selection in the in-out element. The use of these paths is described in the next section under 2., ios.

E. Input-Output Instructions

In addition to the break and dismiss bits on all instructions, the programmer has three computer instructions for operating the input-output system. There are two read instructions, rdn and rds, which transfer data between the in-out devices and the central computer memory. The third instruction, ios, selects the mode of operation of the in-out devices.

1. rdn and rds

Both of the read instructions obtain a word from memory. If the in-out device associated with the sequence in which the read instruction occurs is in a reading (input) mode, appropriate bits of the memory word are altered, and the modified word is replaced in memory. If the in-out device is in a recording (output) mode, appropriate bits of the memory word are fed to the selected in-out buffer, and the word is replaced in memory. Thus, the same read instruction suffices for both input and output operations. The distinction between rdn and rds lies in the assembling of full memory words from short buffer words. An rdn instruction will place the 6 bits from a tape reader in the right 6 bits of a 36-bit memory word. The remaining 30 bits will be left unchanged. An rds instruction for the same tape reader will place the 6 bits in a splayed pattern (every sixth bit across the memory word) and will shift the entire word one place to the left before replacing it in memory. Except for the shift, the other 30 bits remain unchanged. A sequence of 6 rds instructions, one for each of 6 tape lines and all referring to the same memory address, will suffice to assemble a full 36-bit word.

The distinction between rdn and rds could be obtained from mode information in the in-out device, but the inclusion of both instructions in the order code allows the programmer to interchange the two types freely to suit his needs. The rdn instruction makes use of the permutation aspect of the TX-2 configuration control and is, therefore, particularly convenient for dealing with alphanumeric Flexowriter characters. Configuration is not applicable to the rds instruction.

2. ios

The ios instruction serves to put a particular in-out device into a desired mode of operation. The j-bits of the instruction word, normally the index register number, in this case specify the unit number of the in-out device. This number is the same as the program

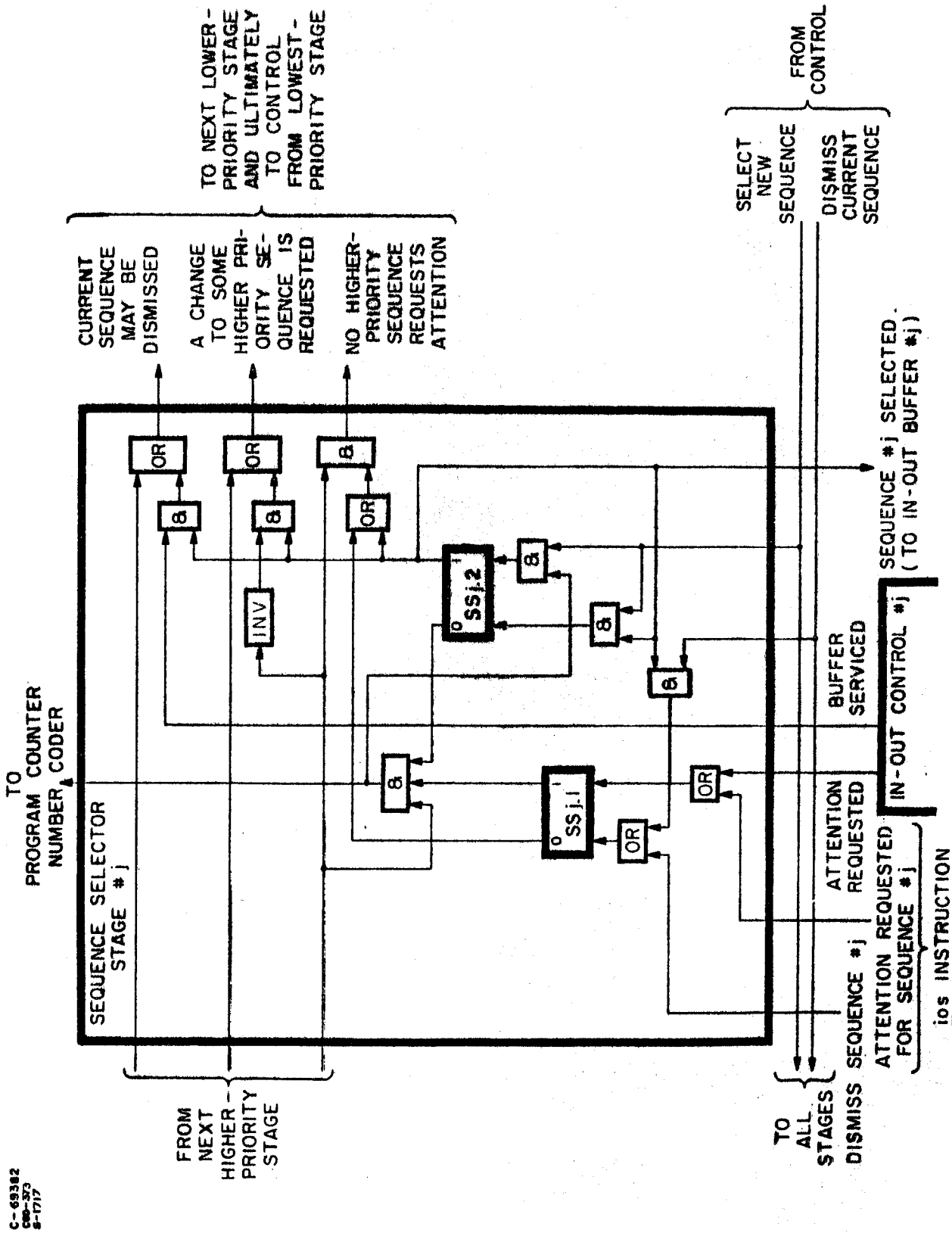

Fig. 13. TX-2 Sequence Selector Stage

counter number for the associated sequence, although the correspondence is not necessary. The y-bits of the instruction word specify the mode of operation in which the unit is to be placed. Two of the y-bits are sent directly to the jth sequence selector stage and serve to control the sequence, regardless of the mode of its associated in-out device. These two bits allow ios instructions to arbitrarily dismiss or request attention for any sequence in the machine. By means of these instructions, one sequence can start or stop all others in the machine. A third y-bit determines whether the mode of the in-out device is to change as a result of the instruction. If it is to change, the remaining 15 bits specify the new mode. An ios instruction occurring in any sequence can thus start or stop any sequence and/or change the mode of its in-out device.

A further property of the ios instruction is that it leaves in the E-register a map of the state of the specified in-out control prior to any changes resulting from the instruction itself; ios instructions may, therefore, be used to sense the state of the in-out system without altering it in any way.

F. Sequence-Changing and Operation of the Sequence-Selector

At some point just before the completion of the instruction memory cycle in TX-2, the Control must decide whether the next instruction would be taken from the current sequence or from some new sequence. The information on which this decision must be based comes from the break and dismiss bits of the instruction word currently in use and from the sequence selector. Fig. 14 is a detailed drawing of one stage of the sequence selector. All stages, except that with the highest-priority are identical. The lowest-priority stage returns the final three control signals to the control element.

Each stage of the sequence selector retains two pieces of information concerning its associated sequence. One flip-flop (ss j.1) remembers whether or not the sequence is selected (i.e., whether or not it is receiving attention). The priority signal (labelled no higher priority sequence requests attention) passes from higher to lower priority stages until it encounters a stage which requests, but is not receiving, attention. Such a stage is said to have priority at the moment, and its output to the program-counter-number coder prepares the number of the new program counter in anticipation of a sequence change.

The process of changing sequences involves storing the program counter for the old sequence and obtaining the counter for the new. Actually, to speed up the over-all process, the new program counter is obtained first, so that it may be used while the old is being stored. Using the paths shown in Fig. 13, the new program counter number is placed in the j-bits of the N-register. The new program counter is then obtained from the X-memory and interchanged with the old program counter contents

which have been in the P-register.* The K-register, which has been holding the old program counter number since the last sequence change, is now interchanged with the j-bits, and the old counter is stored at the proper location in the X-memory. The state of the sequence selector is changed, to conform to the change of sequence, by sending a select new sequence command from Control. This command clears the ss j.2 flip-flop in the old-sequence stage and sets the ss j.2 flip-flop to a ONE in the new-sequence stage.**

F. Interpretation of the Break Bit

The programmer uses the break bit of an instruction word to indicate whether or not change to a higher priority sequence may occur at the completion of the instruction. The fact that a programmer permits a break does not mean that the sequence has completed its current task, but merely that no harm will be done if a change to some higher-priority sequence is made. Breaks should be permitted at every opportunity if a number of in-out devices are operating. The sort of situation in which a break cannot be permitted occurs when the E-register is left containing information which the program requires at a later step. If a change occurred in this case, the contents of the E-register would be destroyed and lost to the program.

When a break is permitted by the current instruction, a sequence change will actually take place only if some higher-priority sequence requests attention. A signal from the sequence selector to the control element provides this information (Fig. 14). When a break type of sequence change is made, the ss' j.1 flip-flop in the sequence selector remains unchanged, and the sequence which was abandoned in favor of one of a higher-priority continues to request attention.

H. Interpretation of the Dismiss Bit

The dismiss bit is used by the programmer to indicate that the sequence presently in use has completed its task. To provide synchronization in the in-out system, dismiss bits must be programmed between attention requests from the in-out devices. In this case, the dismiss in-operation guarantees that the computer will wait for the next signal from the in-out device before proceeding with the associated program sequence.

The dismiss bit is also used to accomplish the halt function in

---

* The P-register is shown in Fig. 6 of Section II., P. 10 .

**The relative timing of the central computer actions during the change process is shown in Fig. 8d., p. 19 .

TX-2. A multiple-sequence computer halts when all sequences have been dismissed and all in-out units turned off. The priority signal from the sequence selector to the control element provides the information as to whether or not any sequence in the machine requests attention. When none request attention, the control stops all activity in the machine as soon as a dismiss bit appears on an instruction in the sequence being used. Activity is resumed in the machine as soon as some in-out device or push button requests attention.

The sequence change which results from a dismiss bit is identical with that resulting from a break except that a dismiss current sequence command accompanies the select new sequence command from Control to the Sequence Selector (Fig. 14).

I. Starting a Multiple-Sequence Computer

In a single-sequence computer the starting process involves resetting the program counter to some arbitrary value and starting the control. In a multiple-sequence computer, the program counter for a particular sequence must be reset and the sequence started. In TX-2 a special sequence (number 0) has the highest priority and is used to facilitate starting. This sequence has the special feature that its program counter always starts at an initial memory location specified by a set of toggle switches. Attention for the sequence is requested by pushing a button on the console. By executing a short program stored in the toggle-switch registers of the V-memory, this sequence can start (or stop) any other sequence in the machine. The starting process for an arbitrary sequence involves resetting its program counter by means of an ldx (load index register) instruction, and starting its sequence with an ios instruction.

J. The Arithmetic Element in Multiple-Sequence Operation

While efficient operation requires that the bulk of arithmetic operations be carried out in a main sequence, the arithmetic element in TX-2 is available to all sequences. Since once a change has been made to a higher-priority sequence, control cannot return to a lower-priority sequence until the higher-priority one has been dismissed, a simple rule allows the arithmetic element to be used in any sequence without confusion. If, whenever a higher-priority sequence requires the arithmetic element, it stores the contents of any registers it will need (A,B,C,D, or F) and reloads them before dismissing, all lower-priority sequences will find the registers as they left them. This storing and loading operation requires time and, therefore, lowers the total handling capacity, but the flexibility obtained may well be worth the loss in capacity.

The step-counter class of arithmetic element instructions is a special problem. These instructions can require many microseconds to complete, and while TX-2 is designed to allow in-out and program element instructions to take place while the arithmetic element is busy, the case can arise in which an arithmetic element instruction (load, store, etc.) appears before the AE is finished with a step-counter class instruction. The machine would normally wait in an inactive stage until the operation is complete, but since there is a chance that some higher-priority sequence may request attention in the interim and have instructions which can be carried out, provision is made to keep trying changes to higher-priority sequences as they request attention. The machine thus waits in an inactive state only when no higher-priority sequences have instructions which can be performed. This provision allows the programmer to ignore the arithmetic element in considerations of peak- and average-peak rate calculations when he desires to operate a maximum number of in-out devices.

K. Conclusions

Multiple-sequence operation of input-output devices, as realized in TX-2, has a number of significant characteristics. Among them are:

1. A number of in-out devices may be operated concurrently with a minimum of buffering storage.

2. Machine time is used efficiently, since no time need be lost waiting for input-output devices to complete their operation. Other machine activity may proceed meanwhile.

3. Each input-output device may be treated separately for programming purposes. Efficiency of operation is obtained automatically when several separately programmed devices are operated simultaneously, although average- and peak-rate limitations must be considered.

4. Maximum flexibility in programming for input-output devices is obtained. The full power of the central machine may be used by each input-output sequence if desired. Routines for each device may be as long or as short as the particular situation requires.

5. The modular organization of the input-output equipment simplifies additions and modifications to the complement of in-out devices

6. The organization of buffering storage allows the amount and kind of such storage to be tailored to the needs of the individual devices and the data-handling requirements to be met by the system.

7. The multiple-sequence program technique appears to be particularly well suited to the operation of a large number of relatively slow input-output devices of varying characteristics, as opposed to a smaller number of high-speed devices.

TABLE I

TX-2 Sequence Assignments in the Order of Their Priority

* Start-Over (special index register number 0 sequence)
* In-out alarms
* Arithmetic alarms (overflows, etc.)
Magnetic Tape units (several sequences)
High-speed printer
Analog-to-digital converter
Photoelectric paper tape readers (several sequences)
Light Pen (photoelectric pick-up device)
Display (several sequences)
MTC (Memory Test Computer)
TX-0
Digital-to-analog converter
Paper tape punch
Flexowriters (several sequences)
* Main sequences (three)

---

* The sequences have no input-output device.

IV. MEMORY UNITS IN THE LINCOLN TX-2

A. Introduction

The three, high-speed memories in TX-2 are random access and use ferrite cores. The largest has 65,536 words, 37 digits long, and operates at a cycle time of 6.5 μsecs; it is referred to as the S-memory. The T-memory is driven entirely by transistors and has a capacity of 4,096 37-digit words with a 5.5-μsec cycle time. The smallest and fastest is the X-memory which has a capacity of 64 19-digit words; its external word-selection and two cores per bit make possible an access time of 0.8 μsec and a cycle time of 4.0 μsec.

B. S-Memory (65,536 Words)

The S-Memory, (Fig. 15) is a coincident-current, magnetic-core unit with a storage capacity of 65,536 37-bit words. The bits in the word are read out in parallel, with a cycle time of 6.5 μsec and an access time of 2.8 μsec. (Cycle time is the time between successive strobe pulses and access time is the minimum delay between setting the address register and strobing). The block diagram (Fig. 16) shows that two, 256-position, magnetic-core switches are used to supply the READ and WRITE current pulses to the X and Y selection lines. The operating characteristics of these switches are such that the contents of the address register are no longer needed after the READ half of the cycle, and the in-

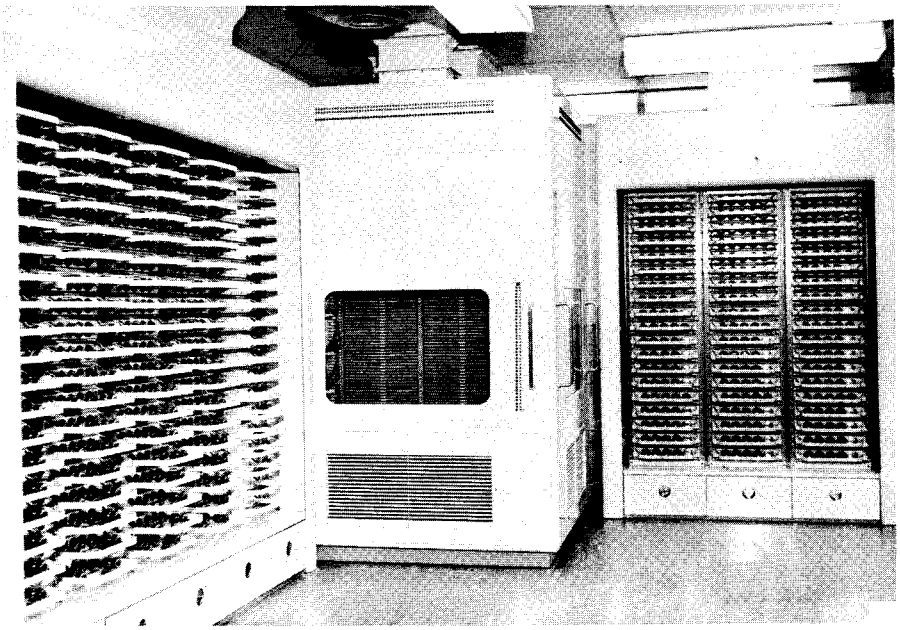

Fig. 15 S-Memory

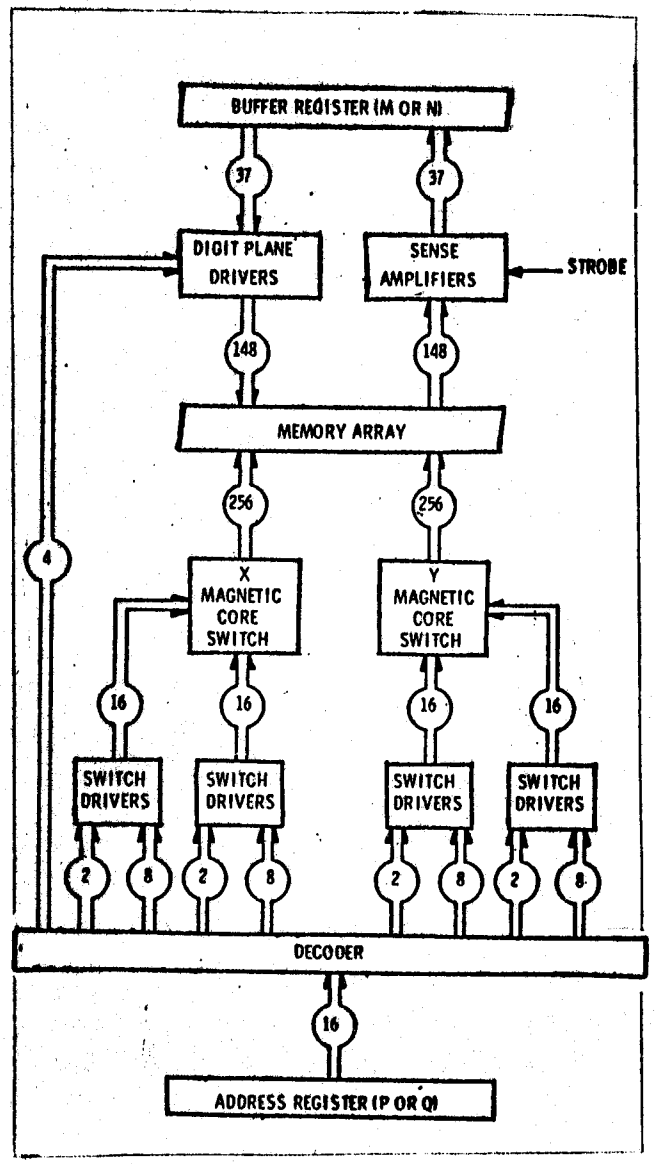

Fig. 16. Block Diagram, S-Memory

terval between READ and WRITE may be extended several μsecs under computer control to permit the other operations to occur. Two coordinates are used to select a register during READ and three coordinates are used for WRITE. In each case, a 2:1 current selection ratio is used. The S-memory with 604 tubes and 1406 transistors, is a 37-digit version of the 19-digit TX-0 memory that has been described in the literature.[1] The basic operation of this type memory has also been described and will not be repeated here.[2]

C. "T" Memory (4096 Words)

The coincident-current, magnetic core, T-memory has 4096 37-bit words (see Fig. 17). The bits of the word are read out in parallel with a cycle time of 5.5 μsec and an access time of 2.4 μsec. A timing diagram is shown in Fig. 18. The computer program can extend, by any amount, the interval between READ and WRITE. Again, A 2:1 current selection ratio is used with two coordinates used to select for READ and three coordinates for WRITE. A total of 1460 transistors and 64 diodes are used (not counting the address and buffer registers and control).

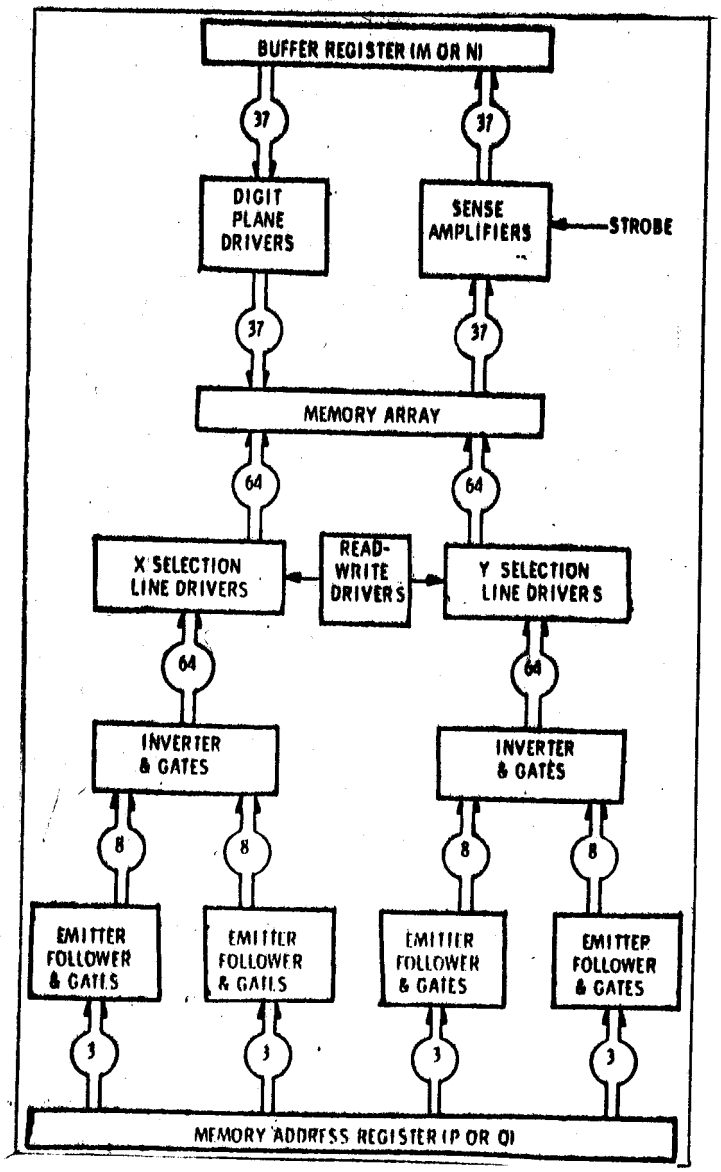

Fig. 17. Block Diagram: T-Memory

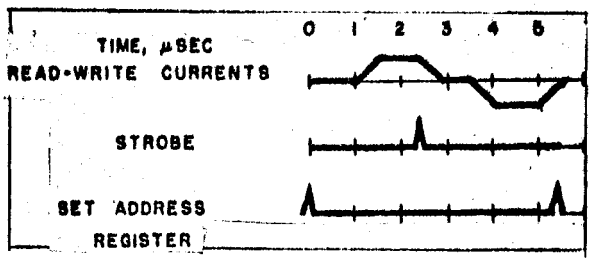

Fig. 18. Timing, T-Memory

1. Mechanical Features

The 64 x 64 x 37 pluggable array (Fig. 19) is contained within a 5-inch cube. (One plane is used for parity checking). The ferrite cores are 47 mils O.D., 27 mils I.D., and 12 mils thick. The material is similar to General Ceramics' S-1 and that is also used in the S-memory.

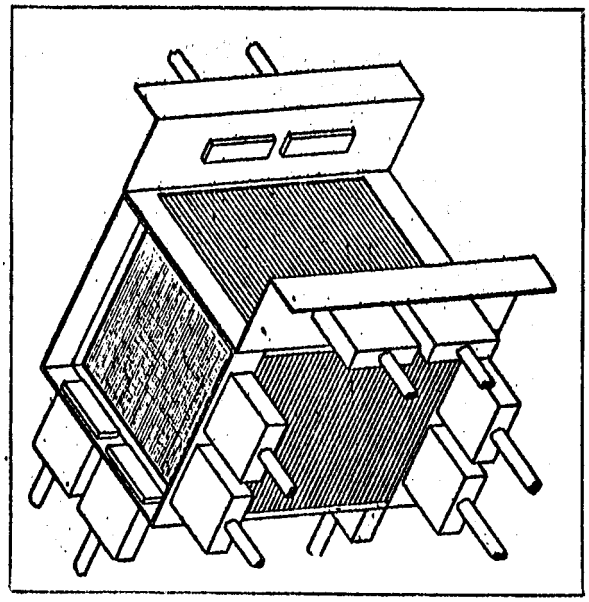

Fig. 19. Pluggable Array, T-Memory

2. Selection Circuits

The logic and circuitry of the memory-address register decoder is shown in Figs. 17 and 20. The input to the emitter follower AND gates is a d-c level of zero or -3 volts. Silicon diodes add a bias shift without the loss that would be associated with

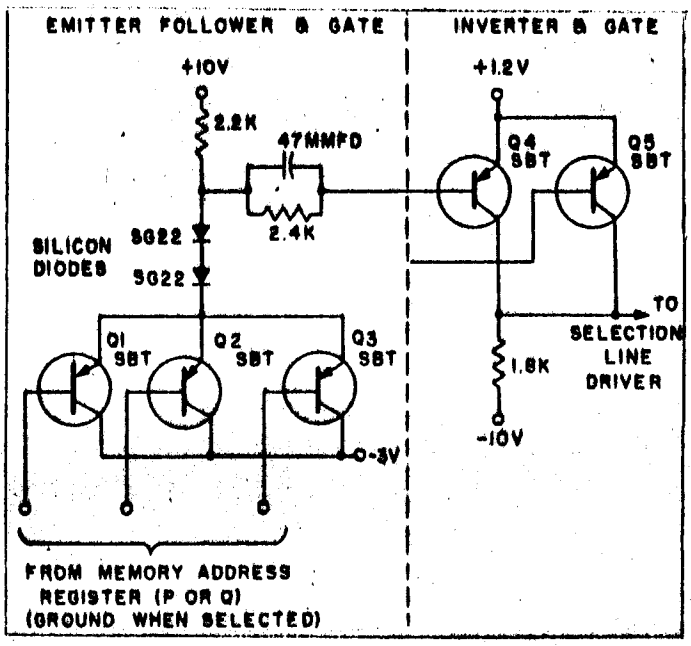

Fig. 20. One Channel, Emitter Follower and Inverter AND Gates, T-Memory

a simple voltage divider. The +1.2-volt supply for the inverter AND gates is a single divider for the whole memory - the load on the divider is constant.

Each inverter AND gate feeds a selection line driver (Fig. 21). Q4 passes the full selection line current (+250 and -250 ma.) and is selected to have a minimum β of 10 for current of either polarity. The transient back voltage of the selection line for this current is 12 volts. The series-connected emitter followers (Q1 and Q2) supply the large amount of current needed to cut off Q3 quickly during selection. The load for Q2 is such that a large surge of current is delivered to Q3 to turn it on quickly when this line is deselected.

Fig. 22 shows the circuit geometry planned for the read-write driver. Currents for READ and WRITE operations are supplied by the 150-volt supplies through R2 and R1 respectively. The currents are switched into the proper selection line drivers by cutting off Q1 or Q2.

3. Digit Circuits

The input of the digit-plane driver shown in Fig. 23 is a standard logic level of 0 or -3 volts. Q2 acts as a switch which connects a voltage source across the digit winding and the parallel R-C

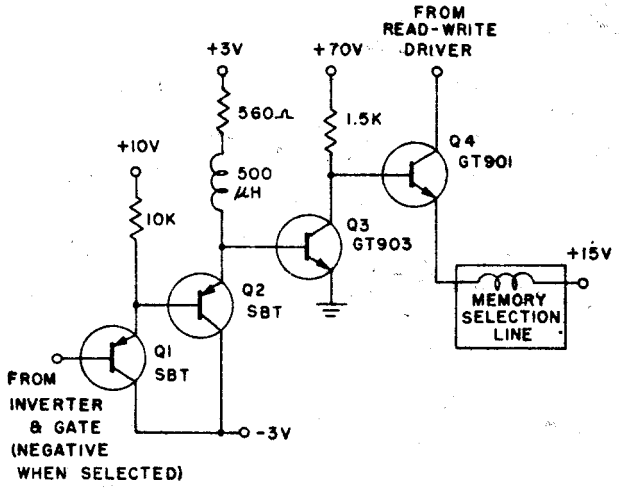

Fig. 21. One Channel, Selection Line Driver, T-Memory

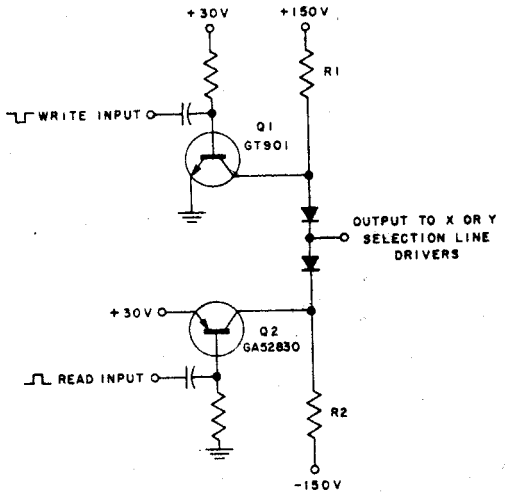

Fig. 22. Read-Write Driver (2 needed), T-Memory

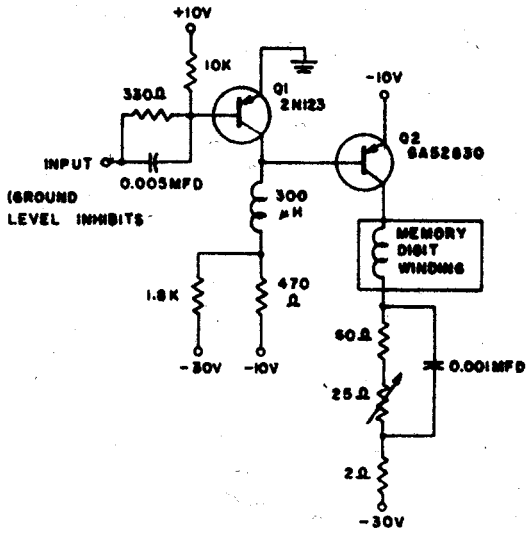

Fig. 23. Digit-Plane Driver, T-Memory

combination. It can be turned on and off very quickly by virtue of the large overdrive of current into its base which is supplied by the combination of Q1 and its collector load. The adjustable resistor is used to set the correct d-c inhibit current which is measured across the 2-ohm resistor; the 0.001-mfd capacitor speeds the current rise time in the digit winding.

In the sense amplifier, shown in Fig. 24, two 160-ohm resistors terminate the sense winding and tie it down to ground. Constant d-c emitter currents are supplied to Q1 and Q2 by the 13K resistors. The voltage divider forms a stable, d-c, collector-to-base voltage and is composed of 1.3K resistors, which form a virtual center tap on the sense-winding by operating in conjunction with the 3.3K resistor. Thus, with both the d-c emitter current and the base-to-collector voltage stabilized, the operating point of Q1 and Q2 is also stabilized. Two 60-μfd electrolytic capacitors in series tie the emitters of Q1 and Q2 together for signal gain. The 6.8K resistors damp the transformer windings. There is no gain for a common-mode input, and a gain of about 22 for a difference-signal input (output measured across half the transformer secondary). The current through the 16K resistor normally flows through Q5, but can be interrupted by an input signal that is large enough to overcome the bias on the 68-ohm resistor. The 5K variable resistance is adjusted so that a 50-mv input signal will be just enough.

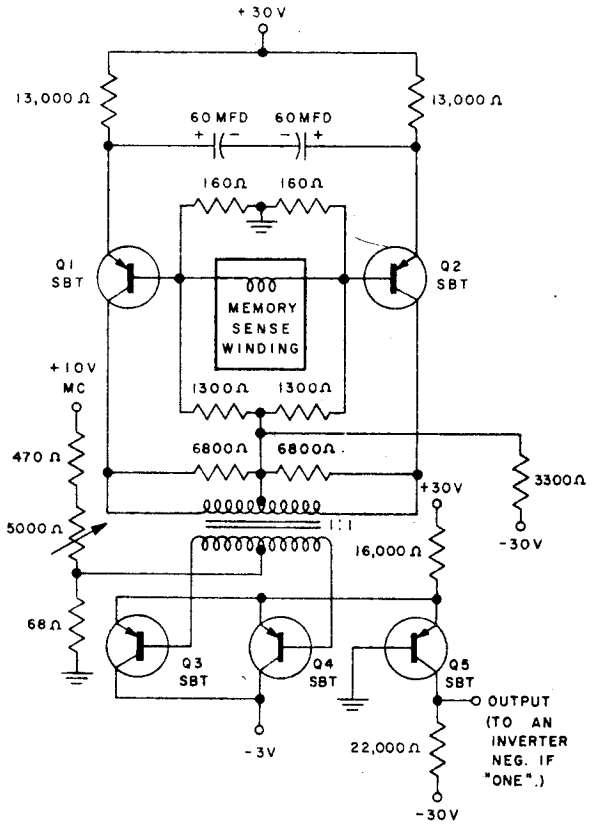

Fig. 24. Sense Amplifier, T-Memory

The normal ONE input signal is 100 mv. All of the +10-volt, marginal-check lines are tied together so that the sense-amplifier clip levels may be remotely checked to determine the memory margins.

D. X-Memory (64 Words)

There are three modes of operation of the X-memory: (1) READ-WRITE, (2) READ, and (3) CLEAR-WRITE. This magnetic-core memory requires external word-selection (two cores per bit) and has a storage capacity of 64 19-bit words. The bits of the word are read out in parallel with a cycle time of 4.0 μsec and an access time of 0.8 μsec. In this memory, cycle time is the time between successive strobe pulses with a repetitive READ-WRITE cycle; access time is, again, the minimum delay between setting the address register and strobing. A total of 434 transistors, 8 diodes, and 1 vacuum tube are used, excluding those for the address and buffer registers and control.

1. Operating Principle

The winding configuration of the single-plane unit is shown in Fig. 25. A word is selected externally by connecting the upper end of a word line (pt. Y, for instance) to a fixed point. The READ driver then puts out a current pulse 4-1/3 times that required to switch a core on a 2:1 basis (Fig. 26). Only one of the two cores (per bit) is switched to the cleared state by this pulse because any previous WRITE operation would have left one core set and one cleared. The switched core generates a pulse in its digit line. This line passes through one of the cores in

the same direction as the word line and through the other core in a direction opposite to that of the word line. Thus, the polarity of the pulse on the digit line during READ, indicates whether a ONE or a ZERO is being read out.

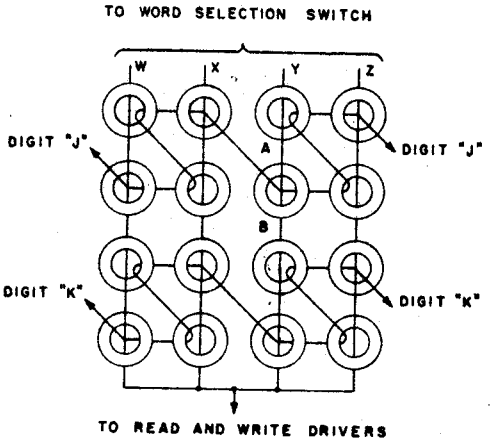

Fig. 25. Winding Configuration, X-Memory

Current always flows in the digit winding. The polarity is controlled by the flip-flop associated with that digit, and the amplitude is 1/3 of the required switch current in a 2:1 system. The digit current is swamped out by the large read current and, therefore, has no effect during READ. During WRITE, a current of 2/3 is sent down the selected word line. The digit current adds to the write current in one core and subtracts from it in the other, so that one core has a current of unity and the other a current of 1/3. Thus, the current ratio used during WRITE is 3:1 with a disturb current of no more than 1/3. Fig. 26 shows the timing and current relationships in cores A and B of Fig. 25.

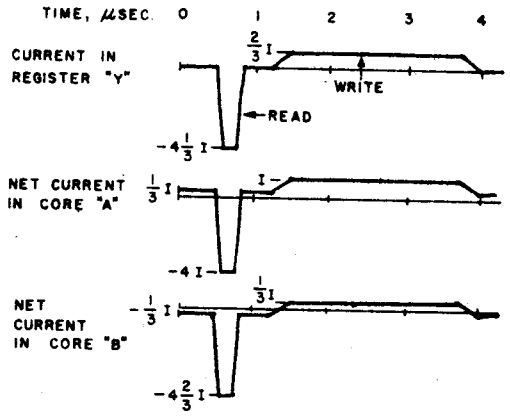

Fig. 26. Timing Diagram, X-Memory

The cores used for the X-memory are 47 mils OD, 27 mils ID, and 12 mils thick. The core material is similar to General Ceramics' type S-3 which differs from S-1 in that the coercive force required for switching is lower and the switching time is longer. We needed the low coercive force so that we could drive the cores with transistors. Access time remained short because, even with transistors, we could overdrive the cores during

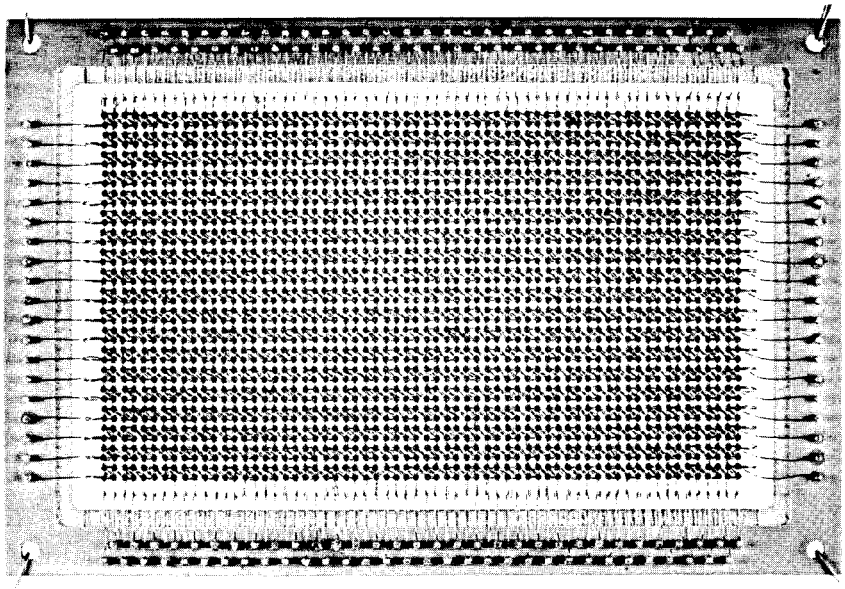

Fig. 27. 64 x 38 Memory Plane, X-Memory

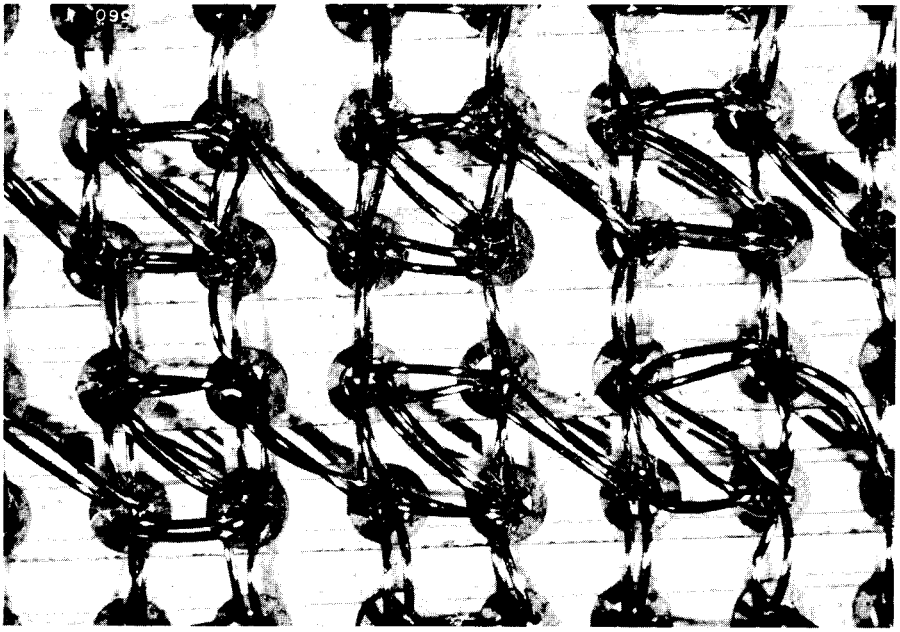

Fig. 28. Section of Memory Plane, Enlarged, X-Memory

READ. Each winding makes sour turns on each core through which it passes. Fig. 27 shows the complete memory plane (4-1/4 x 6-1/4 inches) and Fig. 28 shows part of it enlarged. The cores are mounted on a lucite plate; the wires pass through openings made by the intersection of milled slots on one side of the plate with similar slots on the other side that are milled at right angles to the first. With each winding making four turns per core, the digit current is 8 ma, the write-driver output current is 18 ma, and the read-driver current is 117 ma.

A block diagram of the X-memory is shown in Fig. 29. The method of word selection used is determined partially by the computer's use of the outputs of the j-bits decoder. Either of two write drivers are used and the output of the first level selection determines which one.

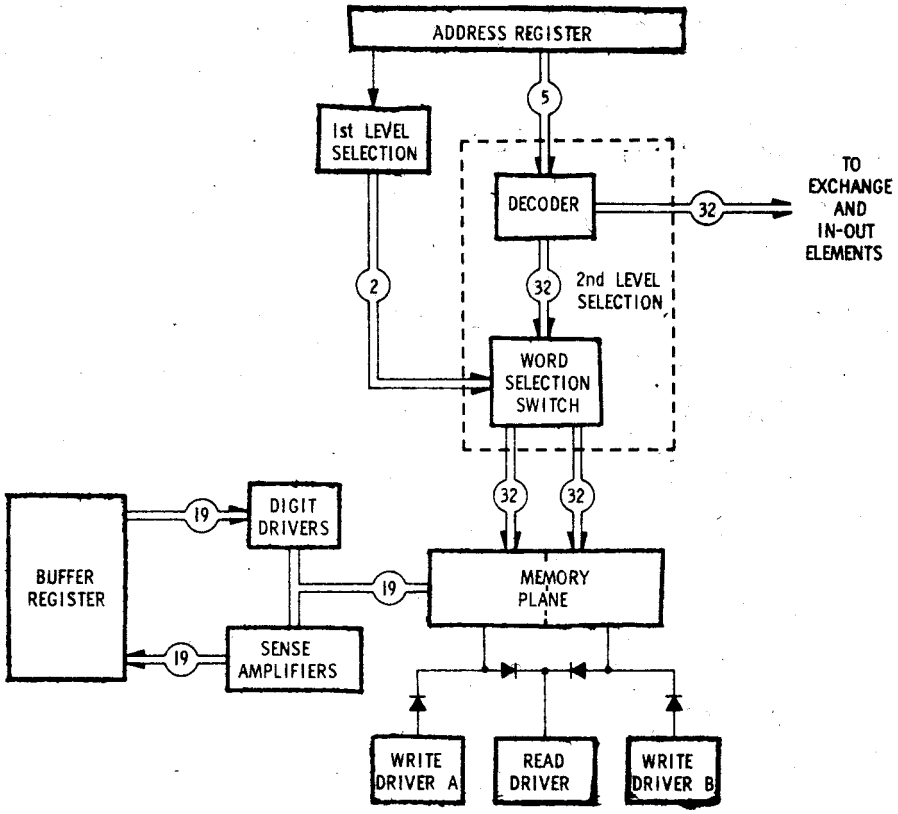

Fig. 29. Block Diagram, X-Memory

2. Selection Circuits

One channel of the selection circuit is shown in Fig. 30. The j-bits decoder uses 5-way emitter follower AND gates which drive parallel inverters (Q6 and Q7). The collector lead of these transistors provides an overdrive of base current into Q8 or Q9 during both selection and de-selection. When neither the read not the write driver is active, the word lines are free to float between 0 and -10 volts. Only one of the first level selection transistors (Q10 or Q11) will be saturated, so base current flows only into either Q8 or Q9. The read driver generates a negative pulse so

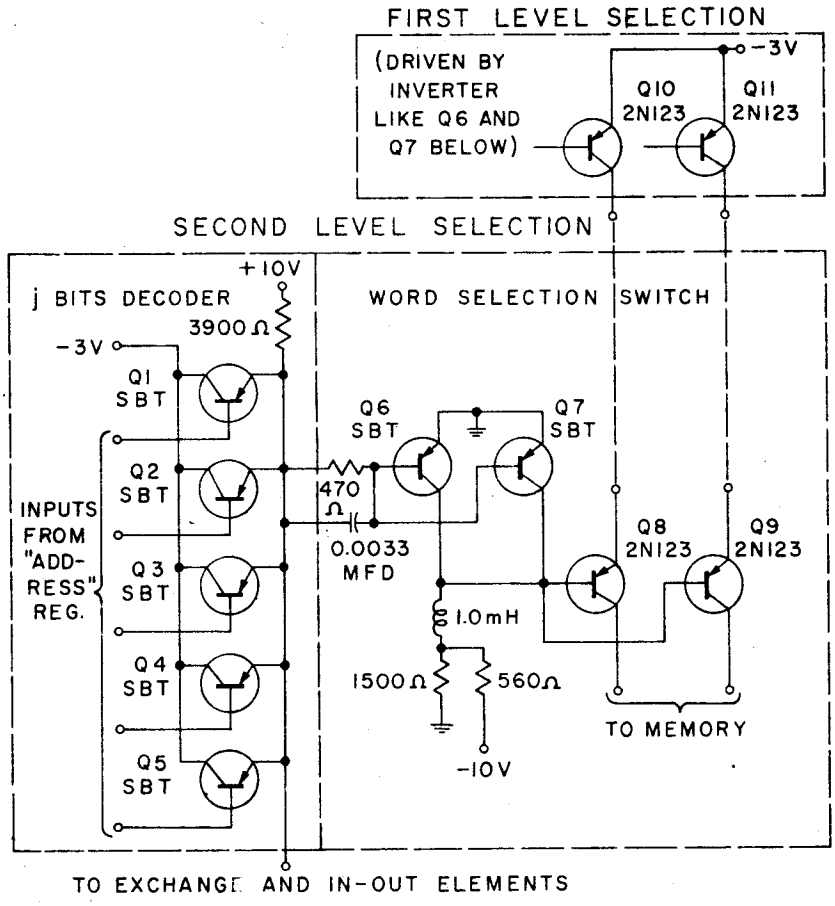

Fig. 30. Register Selection Circuit, X-Memory

that the large read current (117 ma) flows in the normal direction through the 2N123's. The write current flows in the reverse direction, but it is only 18 ma, and doesn't require a very high reverse β . A decoder such as that in the "T" memory would have used fewer transistors, but access time is at a premium here, so the faster circuit was used.

3. Read-Write Drivers

The read driver shown in Fig. 31 consists of three SBT transistors in series (because of the voltage needed) driving a 6197 pentode to saturation. The back voltage presented by the cores to this driver is constant because, as mentioned before, it always switches one of the two cores in each pair. The 5:1 transformer holds the tube load to a low value.

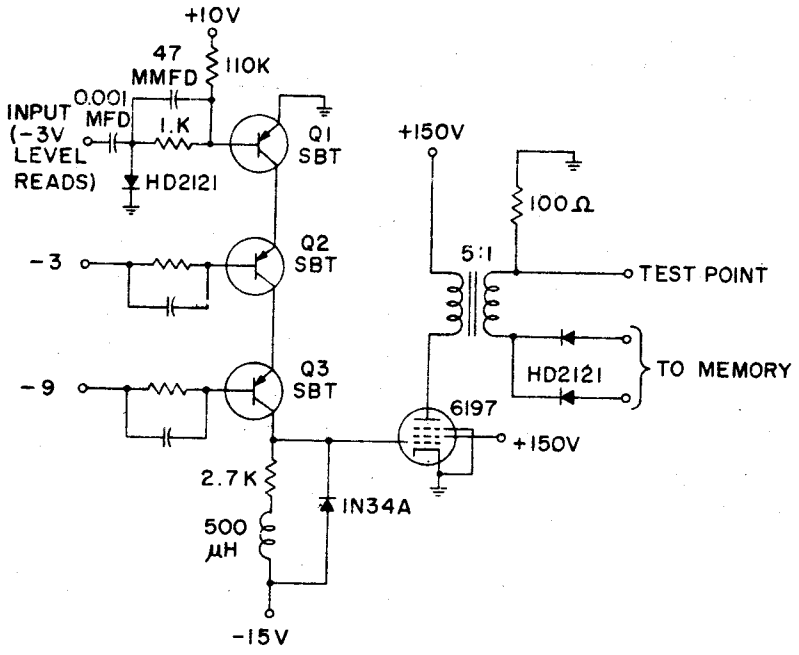

Fig. 31. Read Driver, X-Memory

The write driver (Fig. 32) is very simple - the current in the 1640-ohm resistor is switched into the memory load during WRITE by saturating Q2 which cuts off Q1. Since the selection circuits are returned to -3 volts, the output terminal of this circuit is always below ground.

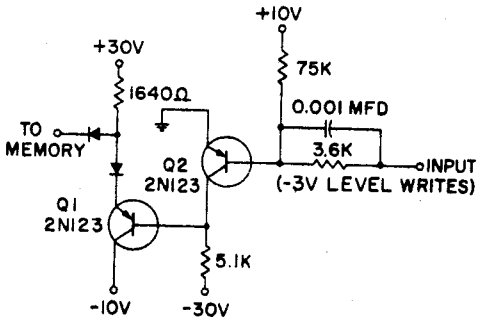

Fig. 32. Write Driver, X-Memory

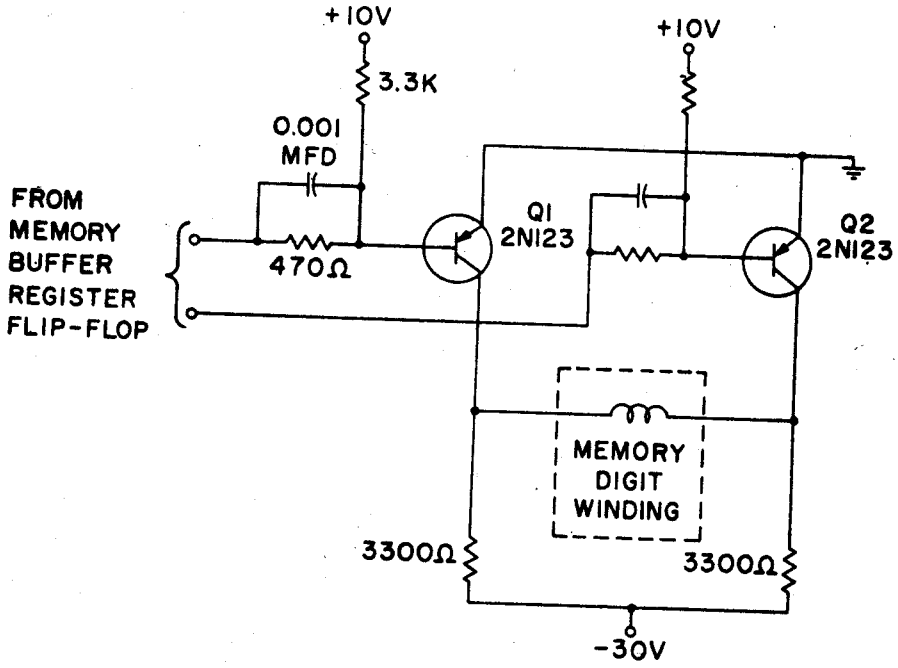

Fig. 33. Digit Driver, X-Memory

4. Digit Circuits

The digit driver (Fig. 33) is connected directly to the corresponding flip-flop in the buffer X-register. One of the two transistors is always saturated so that current always flows in the digit winding and in a direction determined by the flip-flop. The terminals of the digit winding are connected to the input stage (Q1 and Q2) of the sense amplifier shown in Fig. 34 which responds to the voltage difference between the inputs. The open-circuit READ signal on the digit winding is a 0.25-μsec pulse ± 0.5 volt amplitude. The sense amplifier loads the winding to reduce the pulse to about half of this amplitude. A saturation signal is fed to the gates Q3 and Q5 so that the strobe pulse forces the flip-flop to the correct position. If the signal on the free end of the digit winding is positive, the flip-flop is left in the same state; if negative, the flip-flop is complemented.

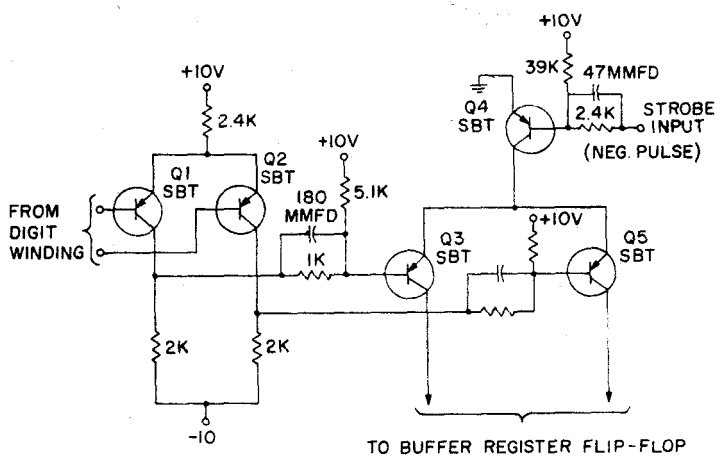

Fig. 34. Sense Amplifier, X-Memory

5. Modes of Operation

The three modes of operation of the X-memory are READ-WRITE, READ, and CLEAR-WRITE. READ-WRITE has been described above. The READ operation, used when the contents of two registers are needed

quickly, performs the necessary function of clearing both cores in each bit before writing. When the computer returns to WRITE in registers that have had a READ cycle only, the CLEAR-WRITE cycle is used. CLEAR-WRITE is the same as READ-WRITE except that the strobe pulse is eliminated. Actually, a WRITE cycle alone would be sufficient but the CLEAR-WRITE cycle was added as an aid to program trouble-shooting, since if a WRITE operation should follow a previous WRITE operation on the same register, some bits would have both cores set. A subsequent READ would clear both cores, their outputs would subtract in the digit winding, and the response of the sense amplifier would be unpredictable.

E. Acknowledgment

The achievements reported above were the result of the imagineering of many people associated with core memory development at Lincoln Laboratory. Major contributions to the system and circuit design were made by S. Bradspies, G.A. Davidson, D.H. Ellis, and J.L. Mitchell. E.A. Guditz was responsible for most of the ideas incorporated in the mechanical design and packaging.

V. TX-2 CIRCUITRY

A. Circuit Configurations

Two basic circuits perform most of the logical operations in the TX-2 computer: a saturated transistor-inverter and a saturated emitter-follower. To the logic designer who works with them, these circuits can be considered as simple switches which are either open or closed.

The schematic diagram of an emitter follower and the symbol used by the logic designers is shown in Fig. 35. With a negative input, the output is "shorted" to the -3 volt supply, as through a switch. When several of these emitter followers are combined in parallel, as in Fig. 36, any one of them will clamp the output to -3V. We have then an OR circuit for negative signals and an

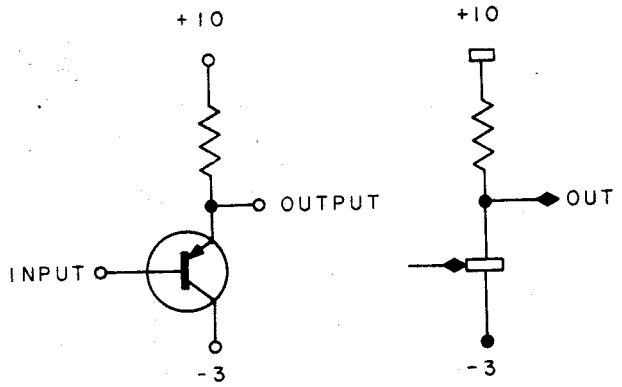

Fig. 35. Emitter Follower

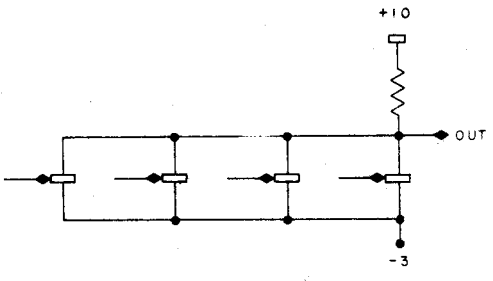

Fig. 36. Parallel Emitter Follower

AND circuit for positive signals. The transistor inverter is shown in Fig. 37 with its logic symbol. Basic AND and OR circuits result from the connection of these simple switches in series or parallel, as in Figs. 38 and 39. More complex networks like the TX-2 carry circuit use these elements arranged in series-parallel as shown in Fig. 40.

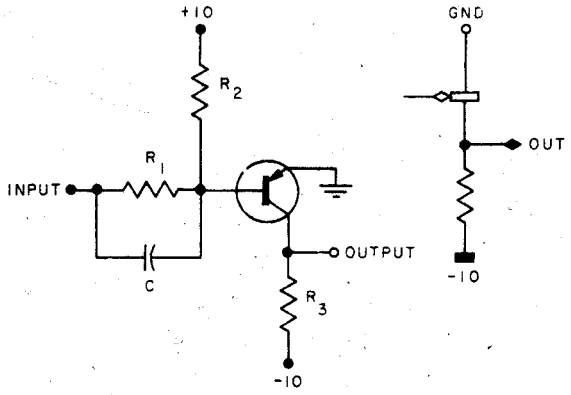

Fig. 37. Inverter

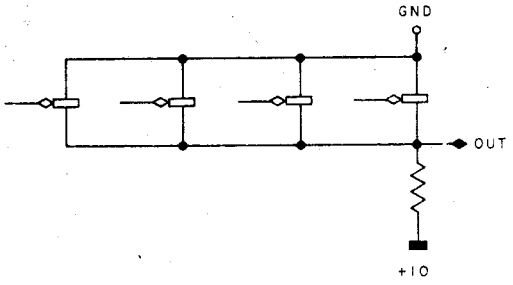

Fig. 38. Parallel Inverters

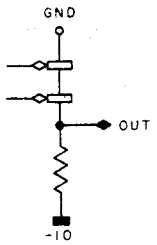

Fig. 39. Series Inverters

In Fig. 37 the resistor, R_1, is chosen so that under the worst combinations of stated component and power supply variations, the drop across the transistor will be less than 200 mv during the "on condition". R_2 biases the transistor base positive during the off condition to provide greater tolerance to noise, I_{co}, and signal variations. Capacitance, C, was selected to remove all of the minority carriers from the base when the transistor is being turned off. The effect of C on a test circuit driven by a fast step is shown in Fig. 41. Note that the delay due to hole storage is only a few millimicroseconds.

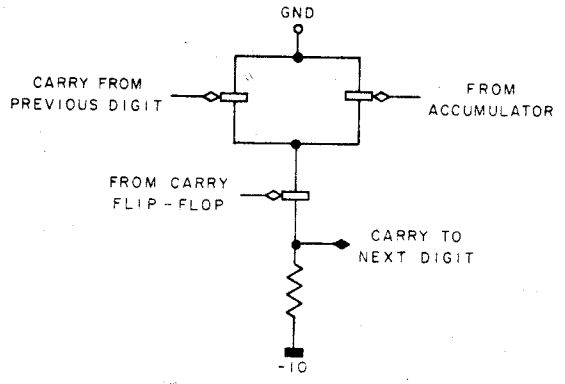

Fig. 40. TX-2 Carry Circuits

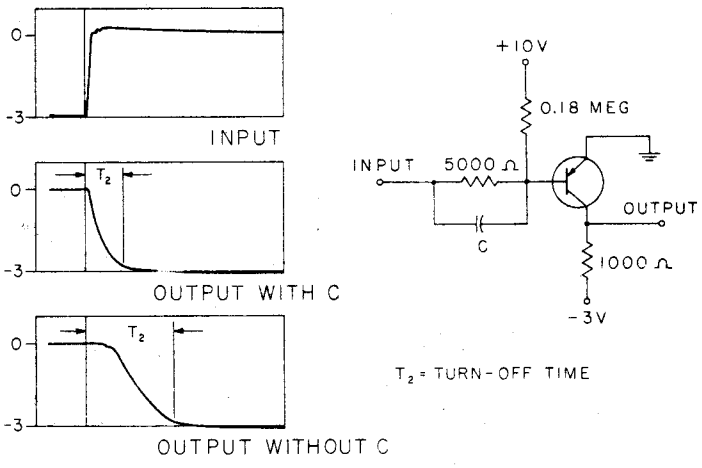

Fig. 41. Turn-Off Time

We run the circuits under saturated conditions to achieve stability and a wide tolerance to parameters without the need for clamp diodes. Unlike vacuum tubes, which always need an appreciable voltage across them for operation, a transistor requires practically no voltage across it. In spite of the delay in turning off saturated transistors, these circuits are faster than most vacuum-tube circuits. Faster circuit speed is not due to the fact that the transistors are faster than vacuum tubes, but because they operate at much lower voltage levels. A vacuum tube takes a signal of several volts to turn it from fully "on" to fully "off"; a transistor takes less than one volt.

B. Flip-Flop

On the basis of previous experience, we decided that the advantages of having one standard flip-flop were worth some complication in TX-2 circuitry. The circuit diagram of the flip-flop package in Fig. 42 is basically an Eccles-Jordan trigger circuit with a three-transistor amplifier on each output. The input amplifiers isolate the pulse input circuits and give high input impedance. The amplifiers give enough delay to allow the flip-flop to be set at the same time that it is being sensed. Fig. 43 shows the waveforms of this flip-flop package when complemented at a 10-megapulse rate. The rise and fall times, about 25 mμsec, are faster than one normally sees in a single inverter that pulls to ground and an emitter follower that pulls to -3 volts. Fig. 44 is a plot of the pulse amplitude necessary to complement the flip-flop at various frequencies. Note the independence of trigger sensitivity to pulse repetition rate. This circuit will operate at a 10-megapulse rate, twice the maximum rate at which it will be used in TX-2.

The TX-2 circuits reproduced most often were designed with a minimum number of components to achieve economies in manufacture

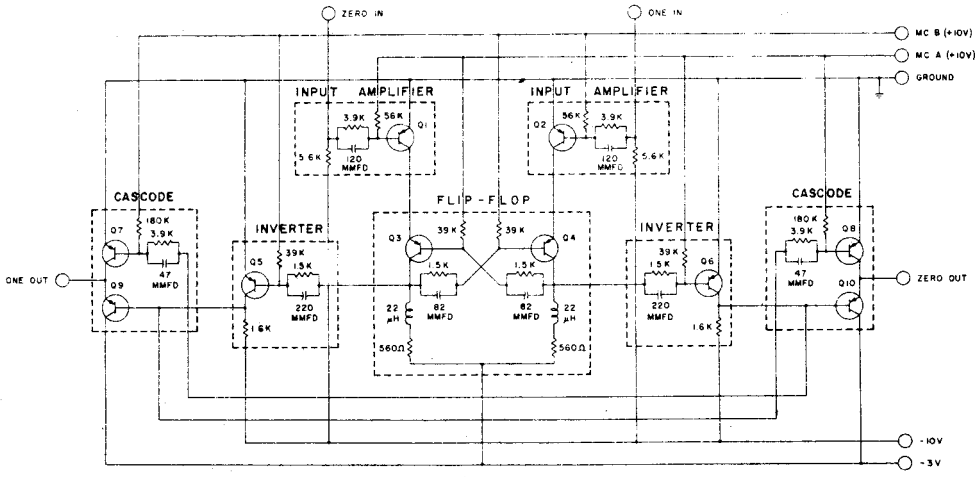

Fig. 42. TX-2 Flip-Flop

and maintenance. The design of less frequently reproduced cir-

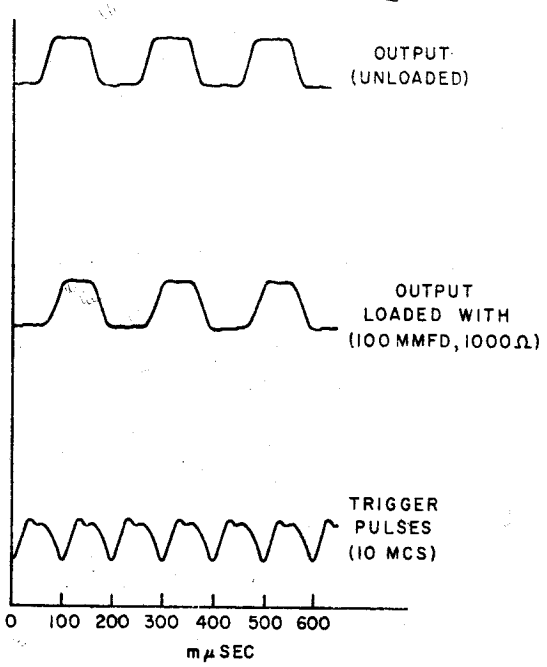

Fig. 43. Flip-Flop Waveforms

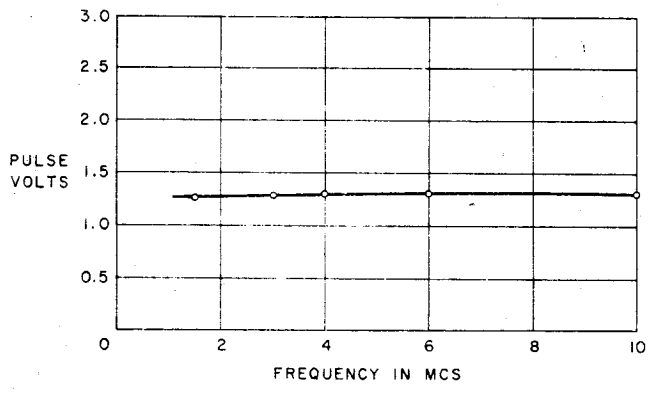

Fig. 44. Trigger Sensitivity

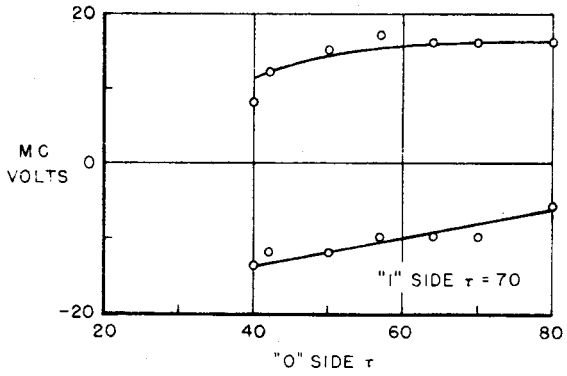

Fig. 45. τ Margins

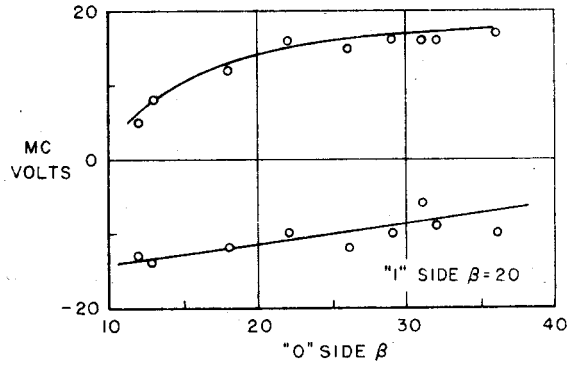

Fig. 46. β Margins

cuits made liberal use of components - even redundancy - to achieve long life and broad tolerance to component variations. The goal was system simplicity and high performance with a lower total number of components than might otherwise be possible. For example, the number of flip-flops in the TX-2 is small compared to the gates which transfer information from one group of flip-flops to another; so the flip-flops were allowed to be relatively complicated. But the TX-2 transfer gates were made very simple; a transfer gate is, in fact, only a single inverter. The emitter is connected to the output of the flip-flop being read and the collector is connected to the input of the flip-flop being set. The output impedance of the flip-flop is so low that, when the output is at the ground level, a pulse on the base of the transfer gate shorts the input of the other flip-flop to ground and sets its condition.

C. Marginal Checking

We planned to incorporate marginal checking in the design of these circuits so that, under a process of regularly scheduled maintenance, deteriorating components could be located before they caused failure in the system. We also found it practical to use this technique, during the design of the circuits, to locate the design center of the various parameters and to indicate the tolerance of circuit performance of these parameters. A further application of marginal checking has been found in other systems during shakedown and initial operation to pin-point noise and other system faults not serious enough to cause failure and, therefore, very difficult to isolate by other means.

The operating condition of the inverters is indicated by varying the +10-volt bias. In the flip-flop schematic in Fig. 42, the inverters were divided into two groups for marginal checking, and the two leads, labeled MCA and MCB, were varied one at a time for the most critical checking of the circuit. The curves in Figs. 43 and 44 show the locus of failure points for various parameters as a function of the marginal checking voltage. Fig. 45 shows the tolerance to tau, a measure of hole storage, and Fig. 46 shows the tolerance to β, the current gain. Operating margins for supply voltages, temperature, and pulse amplitude are shown in Figs. 47 through 50.

Packaging

The number of types of plug-in units was kept small for ease of production and to keep the number of spares to a minimum. The circuits are built on dip-soldered, etched boards and the components are hand-soldered to solid turret lugs. The boards are mounted in steel shells shown in Fig. 51 to keep the boards from flexing. The male and female contacts are machined and gold-plated. The sockets are hand-wired and soldered in panels as in Fig. 52.

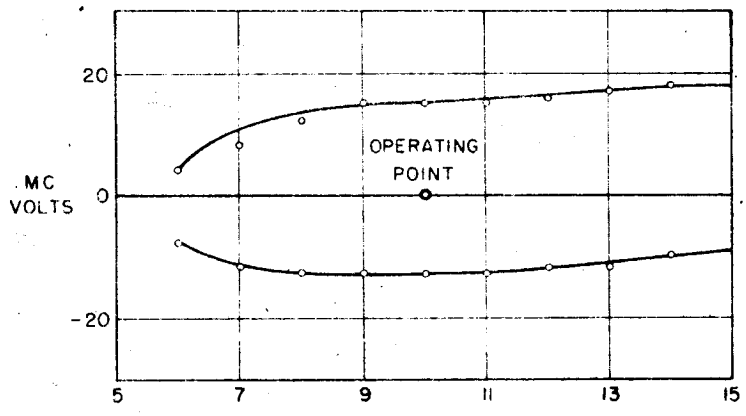

Fig. 47. -10 Volt Supply Margins

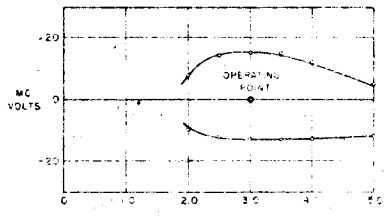

Fig. 48. -3 Volt Supply Margins

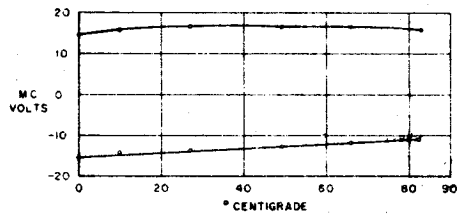

Fig. 49. Temperature Margins

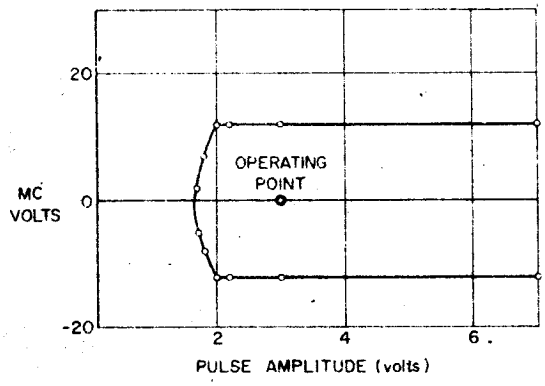

Fig. 50. Pulse Margins

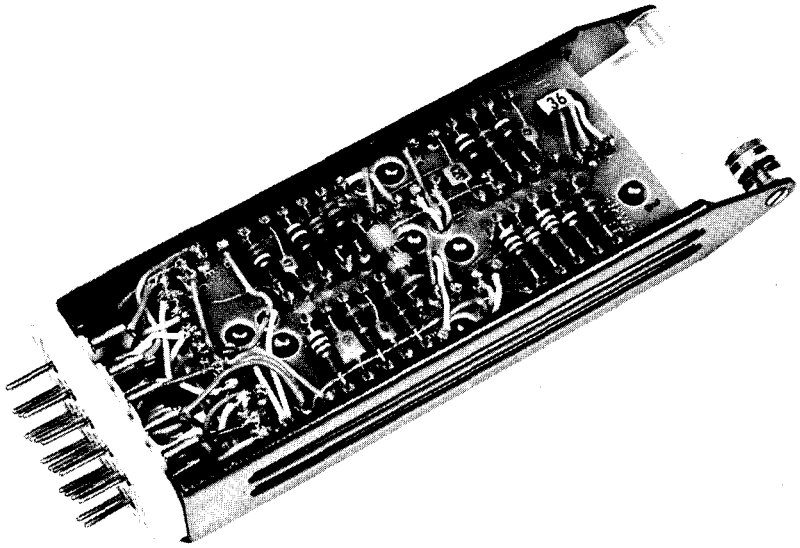

Fig. 51, TX-2 Plug-in Unit

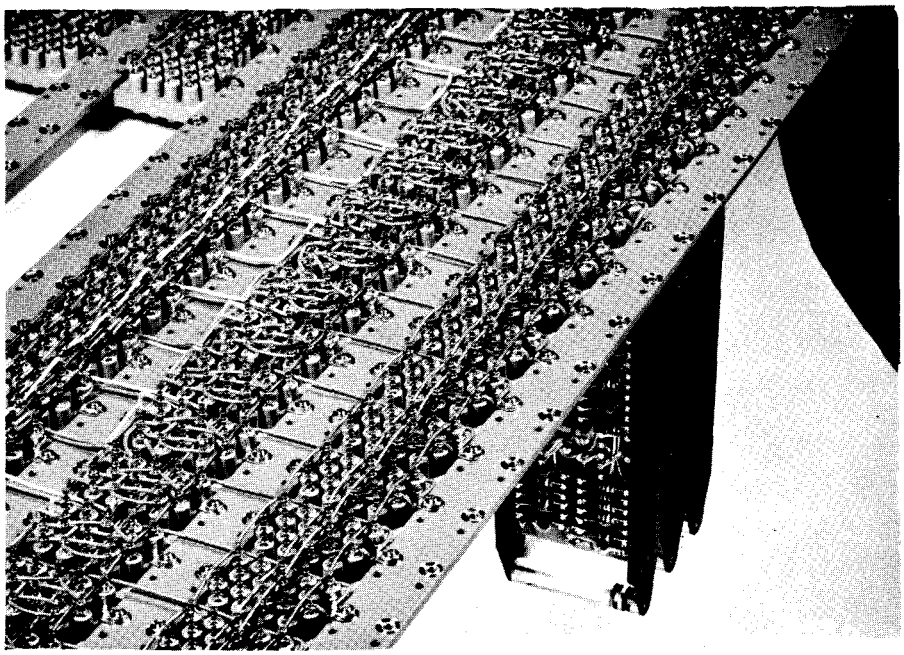

Fig. 52. TX-2 Back Panel

Conclusion

The result of these design considerations is a 5-megapulse control and arithmetic element which will take less than 40 square feet of space and dissipate less than 800 watts of pwoer. The simplicity of the circuits has encouraged a degree of logic sophistication which would not have been chanced before.