THE PHILOSOPHY AND SYSTEM ORGANIZATION OF A SMALL DIGITAL COMPUTER

By

JOHN LEE FIKE, JR.

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1961

Submitted to the Faculty of the Graduate School
of the Oklahoma State University
in partial fulfillment of the requirements
for the degree of
MASTER OF SCIENCE
August, 1962

THE PHILOSOPHY AND SYSTEM ORGANIZATION OF A SMALL DIGITAL COMPUTER

Thesis Approved:

Paul A. McCollum
Thesis Advisor

Dean of the Graduate School

PREFACE

The number and usefulness of digital computers is constantly increasing. Similarly, the demand for engineers with an understanding of not only the applications, but also of the internal structure of digital computers is also increasing.

The availability of a quantity of computer components to the School of Electrical Engineering of the Oklahoma State University suggested that a small computer might be constructed for demonstration and instructional purposes in computer engineering courses. Before such a project could be undertaken, an overall organizational plan was required. Such a plan is presented in this paper.

The various applications of an instructional computer will be examined, and from these a "design philosophy" will be developed. These criteria will then be used in formulating the machine organization. Although many of the components available for construction are circa 1952, the writer has attempted to avoid designing another 1952 computer.

The author expresses sincere thanks to his adviser, Professor Paul A. Mc Collum, for his counsel and guidance. It is deeply appreciated. Also, the help and understanding of Professor William Granet, Acting Director of the Computing Center of the Oklahoma State University, are gratefully acknowledged.

My wife, Gail, and her mother, Mrs. Martha Raper, did all of the typing for this paper. Theirs was a true labor of love, and I thank them for it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION: DESIGN PHILOSOPHY

The design of a digital computer involves many separate levels, from the formulation of the overall purposes of the computer, to the final design and testing of the circuits to accomplish each specific task.

In this chapter, the Oklahoma State Instructional Computer (hereafter called the OSTIC) will be examined in the three areas of construction, operation, and maintenance. From the constraints brought out in this discussion, a "design philosophy" , or set of design criteria, will be formulated for use in the remainder of this paper.

The principal source of components that will be used to construct the OSTIC is the remnants of a digital computer given to the School of Electrical Engineering of the Oklahoma State University, by the Continental Oil Company of Ponca City, Oklahoma. Components available include a magnetic drum with read write heads, motor, and a heavy-duty regulated power supply, together with numerous pluggable flip-flop modules, diode assemblies, and other electronic parts.

A number of important design parameters, such as component maximum operating speeds, voltage levels, and pulse shapes were thus already determined before the present design was conceived. It should also be mentioned that one of the design goals was to minimize the number of purchased components, such as flip-flops, and diodes, since, unlike some other computers constructed by universities, OSTIC will probably not receive a large amount of outside financial support.

As the name implies, this computer is primarily intended for use in computer logic design classes. It may be noted that a great deal

of the instructional value of such a machine lies in the actual construction and testing. It is the intent that the machine will be built, unit by unit, by graduate students under the direction of faculty members. In order to be certain that these efforts are of greatest benefit, an overall plan must be devised (preferably during the early stages of construction) that outlines the purpose, goals, and basic design criteria for the computer. This overall plan must be fairly complete and sufficiently detailed that no important questions (such as, "How does this machine add?") remain unanswered. At the same time, the plan must not attempt to present an overly detailed final design. In the first place, this would be much too time consuming for any one person to create; more important, the individuals doing the actual construction and testing may very likely find preferred ways to accomplish certain operations. Finally, and most important of all, the overall design should be straight-forward, and readily understandable by persons not possessing a large amount of experience in the area of digital computers. Students of this background will not only build the machine, but will later use it to learn how a computer operates.

It is anticipated that this machine will find varied applications in its role as an instructional computer. One of the most likely areas will be in classroom demonstrations. Here, the emphasis will be on how a digital computer functions internally, rather than (as is so often the case) how the computer brings in data and (somehow) turns out answers. Class demonstration requires two things: first, a console so designed that the students can clearly see the contents of the various registers; second, a provision for slow-speed operation so that the students can see each step of the computation as it is performed. Also, it is obvious that the demonstration machine should be sufficiently simple that it may be understood (both internally and externally) and

used by students in computer design courses.

Another possible application for such a machine would be as a test computer for various research and development projects involving pulse or digital circuitry. In other words, the computer might be called upon to serve as a programmable pulse generator. Perhaps the most important consideration here would be reliability. A student who is using the computer as a device to test, for example, a transistorized shift register, needs an extremely reliable machine. If the test set-up does not function properly, he should be quite certain that the failure is in his device, rather than in the computer. The example of a transistorized shift register points up another consideration: the OSTIC should have, in addition to a "standard" console (indicator lights and switches), a large patch panel where all computer pulses, gates, and register outputs are available for circuit use, together with adjustable power supply voltages. A panel of this type would not only be convenient for the application mentioned; it would be invaluable for machine maintenance. A third consideration might be mentioned here: test equipment should be easily set up and versatile in operation. The versatility may be taken care of by a flexible instruction list, but a large amount of set up time would be required if the user (who would probably need only a five- or ten-instruction program) had to enter each instruction into the drum memory, either by operating the console or by punching and then reading-in cards or paper tape. One solution to this problem might be a separate read-only memory of eight, sixteen, or thirty-two words, addressable by the computer in a manner similar to the regular memory and physically provided by a bank of toggle switches on the console.

The existence of a controls laboratory at OSU containing a number of analog computers suggests another possible application

for the OSTIC. This would be as an element in a digital control system, or in the area of real-time systems investigation. Here, the requirement is not only for reliability, but also for a high degree of flexibility in both programming and in data transfer. Programming flexibility would seem to imply the presence of a variety of test and branching codes, perhaps together with some sort of masking or logical operations. The emphasis here would not be on the standard arithmetic operations of addition, subtraction, multiplication, and division, but rather on the controlling functions. The problem of data transfer would imply flexibility in input/output equipment, with perhaps a number of buffered peripheral units.

Since this machine will be a digital computer, the question of computation for problem solving might be raised. It is anticipated that the OSTIC will be used little, if at all, for this type of operation, since computers are already available on campus with much more speed and many programming aids. The problems inherent in writing even a small program for a binary computer with no compiler or assembly routines available make such an application extremely unlikely, to say the least.

Finally comes the all-important question of maintenance. This computer will be maintained by students and technicians who will probably not be overly familiar with computers in general, and who certainly cannot be expected to become intimately acquainted with the operation of each individual circuit. Also, there will be a continuing turnover in the student personnel associated with maintaining the computer. The entire machine must be designed with this in mind; maintenance procedures, especially those concerned with pin-pointing malfunctioning units, should be extremely simple and straight-forward.

Upon taking all of the foregoing considerations together,

one finds that a fairly clear-cut design philosophy emerges. It becomes clear that a simple machine organization is desirable, from the standpoint of the graduate student whose thesis project forms a part of the machine and who needs to understand its place in the overall design; from the standpoint of the school, since simplicity usually implies economy of components; from the standpoint of the student in a computer design course, whose first contact with the "insides" of a computer will probably be through the OSTIC; from the standpoint of the students (and faculty) who use the computer; and from the standpoint of the maintenance personnel, for whom an easily understood machine is usually an easily repaired machine.

The OSTIC should be an extremely reliable computer, both because a large amount of time (and money) will probably not be available to make constant repairs, and because some of the areas of greatest benefit will be those in which the computer is used as a means of testing and monitoring the performance of other equipment.

The final design criterion forces the designer to make so-called "engineering decisions". This is the requirement that the computer be practical; in other words, the OSTIC should be not only usable, but useful. It was noted that the uses for this machine would be primarily in the areas of demonstration, test, and system investigation. These applications necessitate a flexible computer, with perhaps a limited arithmetic command list, but with an extremely flexible list of testing and data-handling commands. Implicit also is a rather large console, with many operating options. In order to justify the effort that has been and will be put into the design and construction of this computer, a machine of maximum usefulness must be the constant goal.

CHAPTER II

THE OVERALL SYSTEM ORGANIZATION OF THE OSTIC

This chapter presents a discussion of the overall system organization developed using the criteria presented in the preceding chapter. The basic tenets set forth in this chapter are the foundations upon which the remainder of the design rests.

Because of the requirements of component economy, simplicity, and reliability, the binary number system was selected for use in the OSTIC. Practically every text on digital computers presents material on the choice of a number system (1, 2, 3, 4) ; it will suffice to state here that this computer would be much more expensive and complicated if built as coded-decimal machine, and further, that the use of a binary computer in digital instructional courses is not a disadvantage to the student, since many computers in the "real world" are natural binary machines.

A second fundamental decision was that the OSTIC should be a parallel machine. "Parallel", as used here, means that the binary digits, or bits, of a given computer word are always transmitted and operated upon simultaneously. It was decided that the bits of any given word would be stored in parallel on the magnetic drum, that they would be read off of the drum in parallel, and that they would be moved about within the machine in parallel. The primary consideration in selecting parallel operation was that the OSTIC was to be used for slow speed or step-by-step demonstration in the classroom; it was felt that a parallel computer would be more readily understood by the students than would a serial machine. Also, a parallel computer can be

organized and built in a more straight-forward manner; if a separate wire is assigned to each bit, the circuit designer has fewer problems. Finally, it should be noted that most authors agree that a serial machine uses fewer components (1, 4); in this case, the economy consideration was compromised somewhat for the sake of overall design simplicity.

The next problem was to select the method of instruction sequencing. Four-address, three-address, two-address, one-plus-one address, and single-address computers have been built[1], and each system has its advantages. The system chosen for the OSTIC was the single-address system, whereby the instruction word specifies the operation to be performed and (normally) the location of the operand. A special counter, called (in the OSTIC) the "Instruction Counter", keeps track of the location of the next instruction. The Instruction Counter is incremented at some time during the execution of each instruction, and it is presumed that the next instruction will always be placed (by the programmer) at the memory location corresponding to the contents of the Instruction Counter. Again, the single-address system was chosen for reasons of economy and simplicity of design; it is not an exaggeration to observe that a three-or four-address machine is a programmer's dream and a designer's nightmare.

Having decided upon parallel binary operation, using a magnetic drum memory for storage of both data and single-address instructions,

---

[1] Some examples of computers using the various types of address structure are:

Three-address	Univac File Computer
Two-address	IBM 305 (RAMAC)
One-plus-one address	IBM 650
Single-address	IBM 704, 709, 7090

the rest of the functional units of the OSTIC may be specified. It was decided to use a double-length, doubly-addressable accumulator, and to refer to the two halves as the "UPPER" (high-order) and "LOWER" (low-order) ACCUMULATORS. A word distributor, or "D-REGISTER" was chosen for temporary storage of operands and to perform miscellaneous tasks. A register to store the current instruction is implicit in a digital computer; in the OSTIC this is called the "INSTRUCTION REGISTER" , and may be further described as a combination of an "OPERATION REGISTER" and an "ADDRESS REGISTER" . The INSTRUCTION COUNTER has been previously mentioned; it would operate in conjunction with an "INCREMENT REGISTER" that would perform operations associated with incrementing the INSTRUCTION COUNTER.

In addition to the units mentioned above, an "AUXILIARY COUNTER" was found to be necessary for certain operations. The possibility exists (although somewhat remote) that indexing registers might be added to the OSTIC;therefore, they should be included as functional units. It should be noted that the input/output equipment (except for the console) is not considered, since its characteristics do not enter into the internal organization.

Logical Diagram number 1 (Figure 1) presents the functional units of the OSTIC , and the manner in which they are tied together for data transmission purposes. The common point is the "DATA CHANNEL" , a group of bit lines that handle all data transmission between the drum and any register, between registers, and between input/output units and the memory. Both data words and instruction words pass over the DATA CHANNEL. The cycle impulses, together with the output of the OPERATION DECODER, ADDRESS DECODER, or address comparison circuits (none of which are shown on Logical Diagram number 1) , cause the output lines of a given unit to be

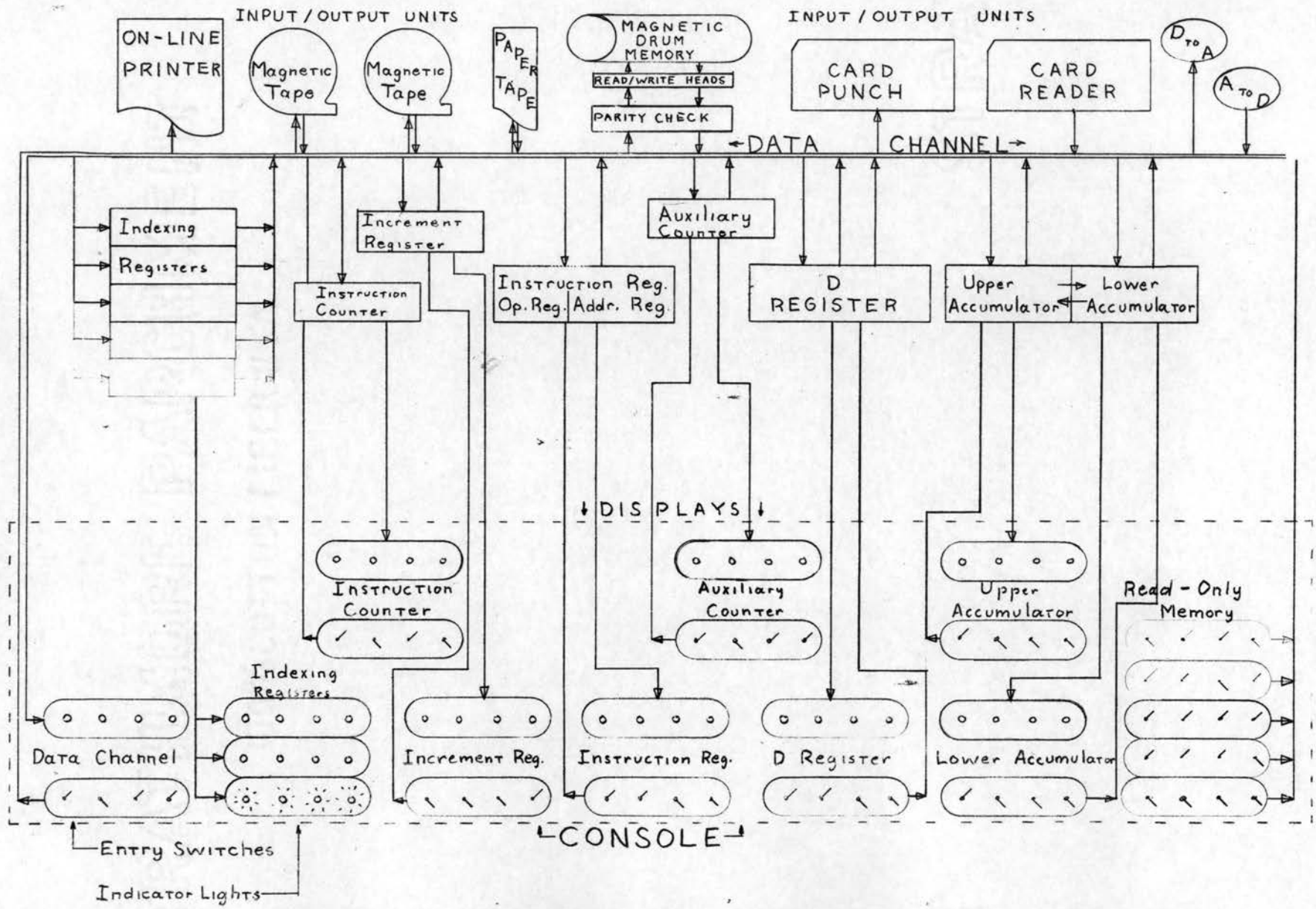


Figure 1. Logical Diagram number 1, Data Channel.

gated onto the DATA CHANNEL. A moment later the input lines of some other unit are gated from the DATA CHANNEL. After the transfer of data or an instruction has occurred, the DATA CHANNEL input gate is removed, and then the DATA CHANNEL output gate is removed. The entire operation of this computer is based upon the data channel concept, since the acquisition and execution of all computer instructions involves basically little more than the gating of the proper units onto and off of the data channel in some predetermined sequence.

Upon examination of Logical Diagram number 1, several other design concepts become apparent. One of the most important is that since all of the operating units share the same DATA CHANNEL, it is not at all difficult to make all of them addressable in the program. This concept is relatively new in computer design (5, 6), and while simplifying the construction of the machine, it vastly increases the flexibility of programming.

It will be noted also that each unit communicates directly with the computer console, rather than indirectly through the DATA CHANNEL. While this approach results in a rather imposing console with many indicator lights and switches, it is felt that enabling the student to read the contents of all units simultaneously would be invaluable for classroom demonstration. The ability to enter data into any register by merely setting the data word in a row of toggle switches and pushing an "enter" button has advantages in ease of operation.

The "READ-ONLY MEMORY, as shown on Logical Diagram number 1, is simply a number of rows of toggle switches mounted in one section of the console. Each row would correspond to one word of memory and would be addressable for read-out in the same manner as the drum memory.

Finally, it should be pointed out that the various types of in-

put output units in Logical Diagram number 1 are shown for illustration only. It is presumed, however, that these units or their buffers will communicate with the DATA CHANNEL.

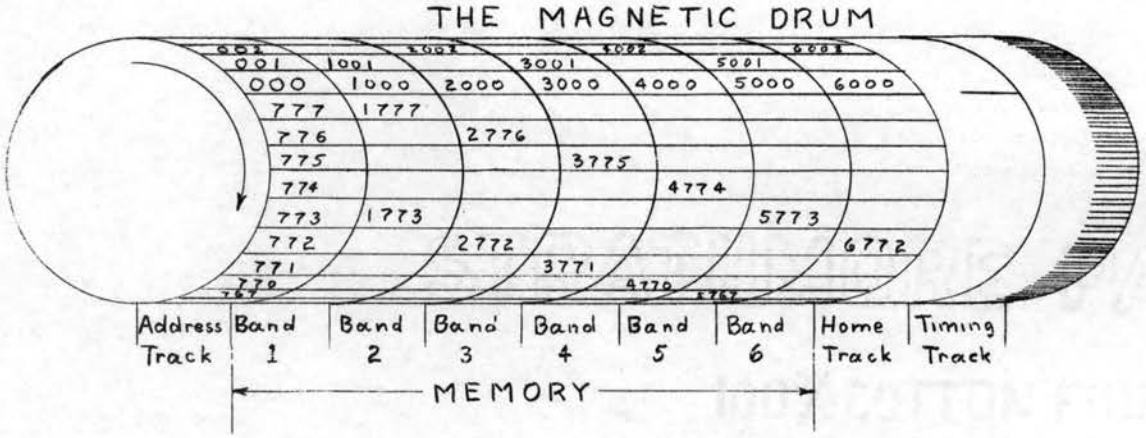
The remainder of this paper will describe the method of controlling the operation of the various units.

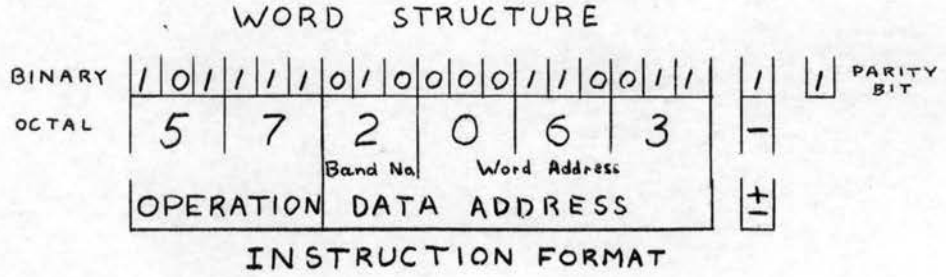
CHAPTER III

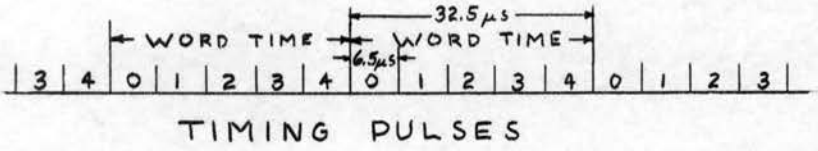
MEMORY AND TIMING

As is true with most digital computers using a rotating magnetic drum for the main memory, timing in the OSTIC is a function of the angular position of the drum. The drum in the OSTIC rotates at a nominal 3450 rpm (one revolution equals approximately 17.4 milliseconds). It is 7 1/2 inches in diameter by 10 inches in length. One-hundred-twenty-eight read/write heads are located around the drum, and a permanently machined timing track providing 2560 timing pulses per drum revolution (one timing pulse every 6.8 microseconds) is located at one end.

Since it was decided to read data onto and off of the drum in parallel, words could be located as desired. Accordingly, the 2560 timing pulses available were divided among 512 word times, providing five timing pulses per word (see Figure 2). One word time in the OSTIC is then equal to approximately 34.0 microseconds. A "home" pulse will be provided on a separate timing track, and "address" tracks will be permanently recorded with the binary word addresses 000 000 000 to 111 111 111, which is equivalent to words 000 to 777_8, or to words 000 to 512_{10} (note that the choice of 512 word times per revolution provides convenient addressing, since $2^9 = 512_{10}$). Word location 000_8 will immediately follow the home pulse in angular position, and location 777_8 will immediately precede it. The five timing pulses for the ith word time will be denoted by T_i0, T_i1, T_i2, T_i3, and T_i4, or more generally (where the particular word time is unimportant) as 0 time, 1 time, 2 time, 3 time, and 4 time. This timing convention will be used throughout the remainder of this paper.







SUGGESTED BAND ADDRESSES

Band number	Use
0	Registers, Read-only memory
1	Band 1
2	Band 2
3	Band 3
4	Band 4
5	Band 5
6	Band 6
7	Input / Output Equipment

Figure 2. Memory, Timing, and Words. 9X7 7-62

The timing pulses, home pulse, and address pulses will be read by read/write heads that have had their write circuits disabled. The pulses will go into a "CLOCK REGISTER" that will provide the various pulses and gates needed by the rest of the machine, as well as providing a constant check on the timing and address pulses to see that they occur in the proper order.

To reduce timing problems, the address pulses of word T_i will be permanently recorded in such a manner that they will be available for checking at T_i 1 time. The bits of word T_i itself will be written onto and read off of the drum at $T_i 3$ time. Chapter 5 relates the use of address pulses in reading from and writing onto the drum.

It was decided to use a word length of 18 bits in the OSTIC, together with a sign bit and a parity bit. The sign bit is transmitted with the word bits, but is not considered an integral part of the word for purposes of shifting and multiplication. The parity bit is used only in conjunction with drum storage; it is generated by the "PARITY REGISTER" on a "drum write" (store) operation, and is checked by the PARITY REGISTER on a "drum read" operation. The parity mode is "even parity", as is used by most parity-check devices; the total number of bits in a word, including sign and parity bit, is always supposed to be an even number.

The use of an 18 bit word for both data and instructions brings up the question of instruction format and addressing. It was stated previously that there were 512 word times during one revolution of the drum. In order to represent 512 separate addresses in binary form, 9 bits must be used ($2^9 = 512$). However, it would be desirable to have more than 512 memory locations; in fact, if a total of twenty bits are recorded on the drum for each word (18 word bits, 1 sign bit, 1 parity bit), a total of 20 read/write heads are all that are necessary to store 512 words, while some 128 read/write heads

are available. A further consideration is that since binary notation is somewhat unwieldly for everyday use, it would be very desirable if programming of the computer could be done in octal (base 8) notation. Octal requires 3 binary bits for each octal digit; therefore, binary addresses must be in multiples of 3 bits if they are to be represented in octal form.

In view of these considerations, it was decided to use 12 bits of the instruction word for the data address, and the remaining six bits for the operation code. This resulted in a 6 octal digit instruction (see Figure 2) the first two digits of which were the operation code, and the last four were the address. Further, the address may be divided into a "band address" for the first octal digit (first three bits) and a "word address" for the last three octal digits (last nine bits). The drum memory thus was planned to have six bands of 512 words each, for a maximum (if all are used) of 3072 words of drum memory.

In addition, it is proposed that OSTIC be provided with either sixteen or thirty-two words (a power of two would present fewer problems in the address circuits) of "read-only" memory in the form of switches on the control console. These memory words will be addressable only on read operations; the binary system in the computer will allow the use of relatively inexpensive single-pole double-throw toggle switches. The availability of a fairly large, quickly altered read-only memory would be invaluable for demonstrations and test programs.

It will be noted that since band numbers 0 and 7 are not used for drum addresses, there are 512 addresses of the form $0XXX_8$ and and 512 addresses of the form $7XXX_8$ also available. It is intended therefore, that the 0XXX addresses be reserved for internal machine use (register addresses, read-only memory, etc.), while the 7XXX addresses be reserved for input/output equipment.

It is intended that band addresses 1 through 6 be reserved for drum addresses, even though not all six bands are used at first.

The use of two octal digits for operation codes results in a possible command list of 64_{10} operation codes, ranging from 00 to 77_8. If the sign of an operation were also taken into consideration, a total of 128_{10} operation codes could be used. This, however, is not recommended. Further discussion of this point will be found in Chapter 10.

To summarize briefly, the drum memory is divided into six bands, numbered from one to six. Each band contains 512 words, numbered from 000 to 777_8. Each word has eighteen bits, one sign bit, and (on the drum only) one parity bit, for a total of twenty bits. The instruction word uses the first two octal digits of the word for the operation code, and the last four for the address. The address is broken down into a one digit band number (or band address) and a three digit word address.

Internally, the machine uses five timing pulses per word time, called (for the ith word) T_i0, T_i1, T_i2, T_i3, and T_i4. Where the particular word is of no importance, 0 time, 1 time, 2 time, 3 time, and 4 time will be used. The word addresses corresponding to the addresses of the words in each band are recorded on special address tracks; these are available at 1 time, while the contents of the word itself are available at 3 time.

CHAPTER IV

CYCLES AND CONTROL

Almost every digital computer must perform, internally or externally, two basic functions. One of these is to acquire, by some means, the next instruction to be executed. The other is to execute that instruction.

No matter how simple or how complex the individual instruction, the computer performs its operations in this manner. It has been said that a digital computer is perhaps the most complex form of sequential machine, since at the start of a program the instructions and data stored internally will (if no input is assumed to occur during the program) determine the state of the machine at any later time until the program is completed.

Thus, the computer steps through a program, finding an instruction, executing it, finding the next instruction, executing it, etc. The time consumed in acquiring the next instruction in the OSTIC will be called the "instruction cycle", and the time consumed in executing that instruction will be called the "data cycle", although data may not always be transferred on all data cycles.

The term "cycle" should be further defined, since a misconception could easily result from its indiscriminate use. As used here, a "cycle" is one or more whole word times, a word time being considered to run from the leading edge of one 0 time pulse to the leading edge of the next 0 time pulse. A cycle may sometimes consist of only one word time, as, for example, the time required to acquire the next instruction from the D-register (see Chapter 5), or it may require many word times, as, for example, in the

case of acquiring a multiplicand from drum storage, then multiplying it by a multiplier in the upper accumulator. However, one rule is steadfast; in the OSTIC, a cycle is always composed of an integral number of whole word times.

The method of execution of the various operations in this computer will be illustrated by Logical Diagrams, using special graphical symbols. The reader is referred to Appendix A for an explanation of the symbols used.

It is planned that the selection of the appropriate cycle (data or instruction) will be performed by a "CYCLE SELECTION FLIP-FLOP" (bistable multivibrator) as shown on Logical Diagram number 2 (Figure 3). When one side of this flip-flop is conducting, the computer will be in the data cycle mode; when the other side is conducting, the machine will be in the instruction cycle mode.

The general operation of cycle selection is also illustrated in Logical Diagram No. 2. Briefly, the CYCLE SELECTION FLIP-FLOP is set to one mode or the other by a "CYCLE TURN-ON" pulse at 0 time. The CYCLE-TURN ON pulse (a very sharp pulse of short duration) is allowed to set the CYCLE SELECTION FLIP-FLOP only when a "CYCLE END" impulse has turned on one of the two "CYCLE END latches" during the previous word time.

A detailed description of the operation will now be presented, using the timing chart on Logical Diagram No. 2. Reading from left to right, it is seen that the CYCLE SELECTION FLIP-FLOP is assumed to be set for data cycles, resulting in the DATA CYCLE MASTER gate being on at the start of the time interval under consideration. A "DATA CYCLE END" impulse is then assumed (upper left corner of diagram, reading downward). Although this

MASTER CYCLE CONTROL

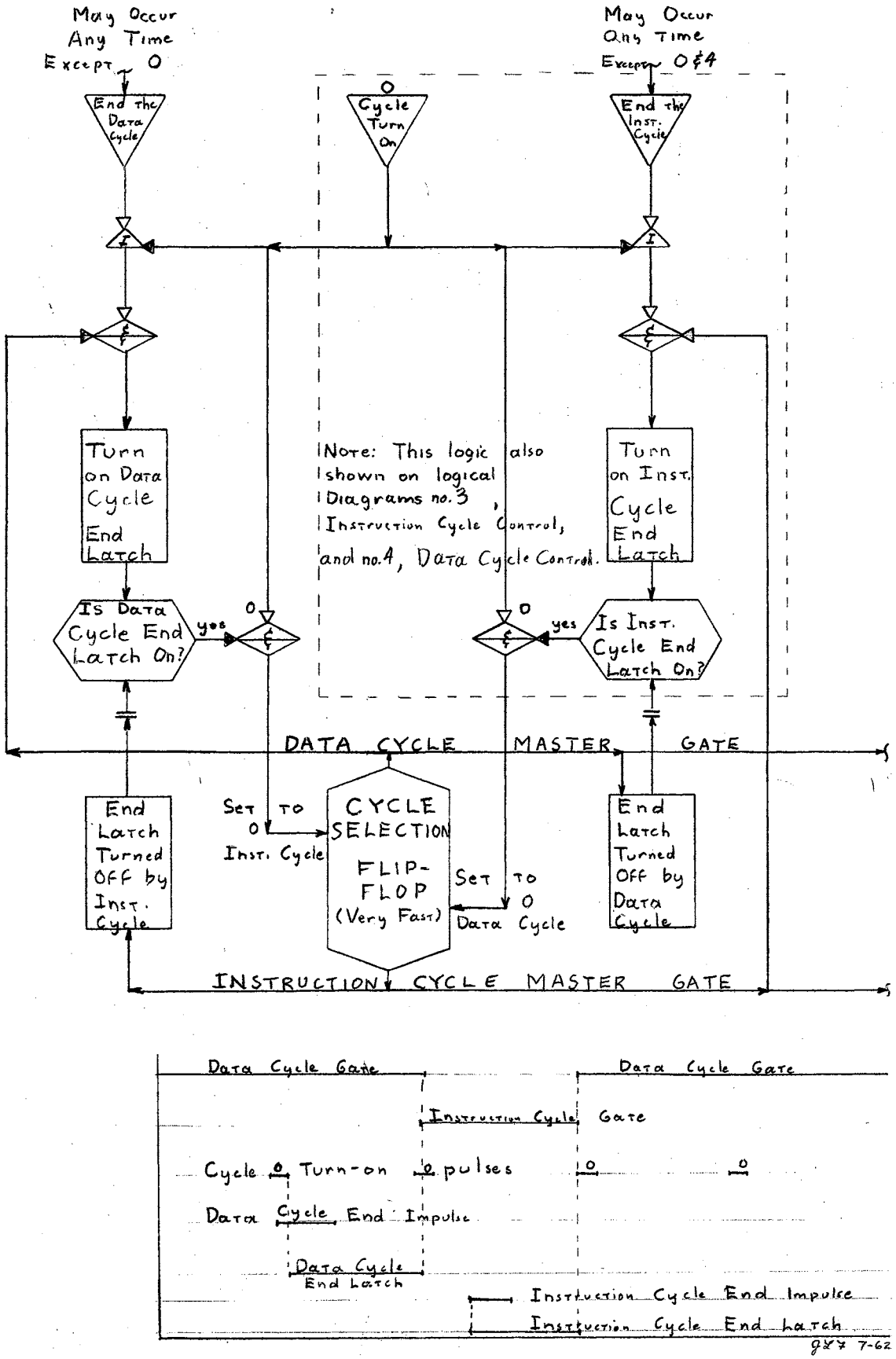


Figure 3. Logical Diagram number 2, Master Cycle Control.

pulse began at 0 time, it was inhibited by the CYCLE TURN-ON pulse. At 0 time, however, the CYCLE TURN-ON pulse is not present, while the DATA CYCLE MASTER gate is on, so the DATA CYCLE END pulse turns on the "DATA CYCLE END latch". This latch is simply a resettable flip-flop.

At 0 time of the next word time, since the DATA CYCLE END latch is on, the CYCLE TURN-ON pulse is able to set the CYCLE SELECTION FLIP FLOP to INSTRUCTION CYCLE. The INSTRUCTION CYCLE MASTER gate, turning on, causes the DATA CYCLE END latch to be reset, or turned off.

Sometime between 0 and 4 time of the instruction cycle , an INSTRUCTION CYCLE END pulse occurs. Since the CYCLE TURN-ON pulse is not present to inhibit it, and the INSTRUCTION CYCLE MASTER gate is present, the INSTRUCTION CYCLE END impulse turns on the INSTRUCTION CYCLE END latch. At 0 time of the following word time, the CYCLE TURN ON pulse is gated by the AND circuit from the INSTRUCTION CYCLE END latch to set the CYCLE SELECTION FLIP-FLOP to data cycle. The DATA CYCLE MASTER gate, turning on, resets the INSTRUCTION CYCLE END latch to the off position.

The salient points are that there exists either a DATA CYCLE MASTER gate or an INSTRUCTION CYCLE MASTER gate at all times; that these gates remain on until set to turn off by CYCLE END impulses; that the turn-off or END impulses may occur at 1, 2, 3, or 4 time for data cycles, or at 1, 2, or 3 time for instruction cycles; and that the MASTER GATES are always on from 0 time of one word time to 0 time of a later word time or, in other words, that the MASTER GATES are always on for integral multiples of word times.

Logical Diagram number 3 (Figure 4), Instruction Cycle Control, will now be examined. It will be noted that the INSTRUC-

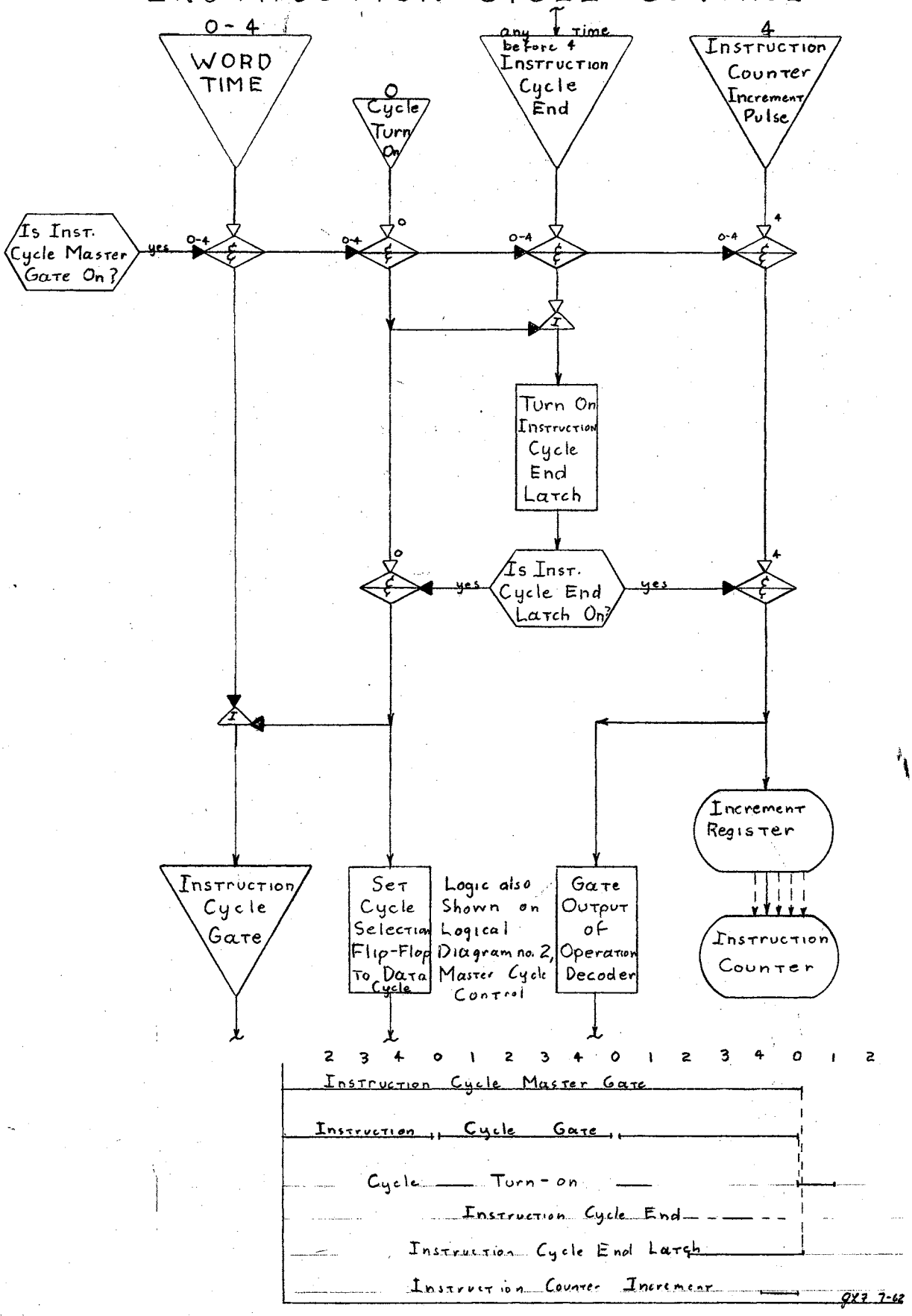


Figure 4. Logical Diagram number 3, Instruction Cycle Control.

TION CYCLE MASTER gate, CYCLE TURN-ON pulse, and INSTRUCTION CYCLE END latch are repeated from the previous diagram.

Referring to the diagram and timing chart, it will be noted that as long as the INSTRUCTION CYCLE MASTER gate is on, the WORD TIME pulse is gated by an AND circuit to provide an INSTRUCTION CYCLE gate for the rest of the machine. In other words, the INSTRUCTION CYCLE MASTER gate does not itself provide instruction cycles to the rest of the computer; rather, it controls another impulse to provide this function. Also note that whenever the INSTRUCTION CYCLE END latch has been turned on by an INSTRUCTION CYCLE END impulse, the WORD TIME pulse is inhibited by the CYCLE TURN-ON pulse. This is to prevent signal race problems caused by a lag in the turning off of the INSTRUCTION CYCLE MASTER gate (7).

The only other item requiring explanation on Logical Diagram No. 3 is the procedure for incrementing the INSTRUCTION COUNTER. It was mentioned previously that the location of the next instruction will always be found in this counter; therefore, after an instruction is acquired on an instruction cycle, the INSTRUCTION COUNTER should be incremented so that it will contain the address of the next instruction. Note that nothing has yet been said concerning the size of this increment. Most, if not all, single address computers are constructed so as to take the next instruction from the location immediately following the last instruction. For example, an instruction might be stored in location 2301, the next instruction in 2302, the next in 2303, and so on. This would imply an increment of one. It is proposed that the OSTIC be equipped to increment the INSTRUCTION COUNTER by 1, 2, 4, 8_{10}, 16_{10}, 32_{10}, 64_{10}, or 128_{10}. Since the OSTIC's instruction counter is a binary

device, this method of variable incrementation is seen to be merely a matter of adding a 1 into the first, second, third, fourth, fifth, sixth, seventh, or eighth position of the twelve-position binary counter. It can be shown that this relatively simple device will greatly increase the operating speed of the computer (see Chapter 11).

The desired increment is placed in the INCREMENT REGISTER some time beforehand (see Chapter 9 for details), and thus the problem here is simply to provide an INCREMENT pulse to the INCREMENT REGISTER at the proper time. This pulse will be gated by the INCREMENT REGISTER into the proper position of the INSTRUCTION COUNTER. Incrementing is accomplished by the INSTRUCTION COUNTER INCREMENT pulse, occurring at 4 time, which is gated by the INSTRUCTION CYCLE MASTER gate. If the INSTRUCTION CYCLE END latch is on, an AND circuit will allow the INCREMENT pulse to enter the INCREMENT REGISTER. It will be noted that the only time the INSTRUCTION COUNTER can be incremented is at 4 time of the last word time of an instruction cycle; this must be so, because during any other part of the instruction cycle the machine will be using the contents of the INSTRUCTION COUNTER in seeking the next instruction. This also explains why INSTRUCTION CYCLE END pulses must occur prior to 4 time.

Logical Diagram No. 4 (Figure 5) illustrates the logic used on data cycles. This logic is necessarily more complex than the instruction cycle logic, because while essentially the only operation performed by the computer on the instruction cycle is to search for and acquire the next instruction, data cycles involve the testing, adding, shifting, and so on, necessary to carry out the instructions. In general, the data cycle is broken up into one, two, or three sub-cycles (each of which may

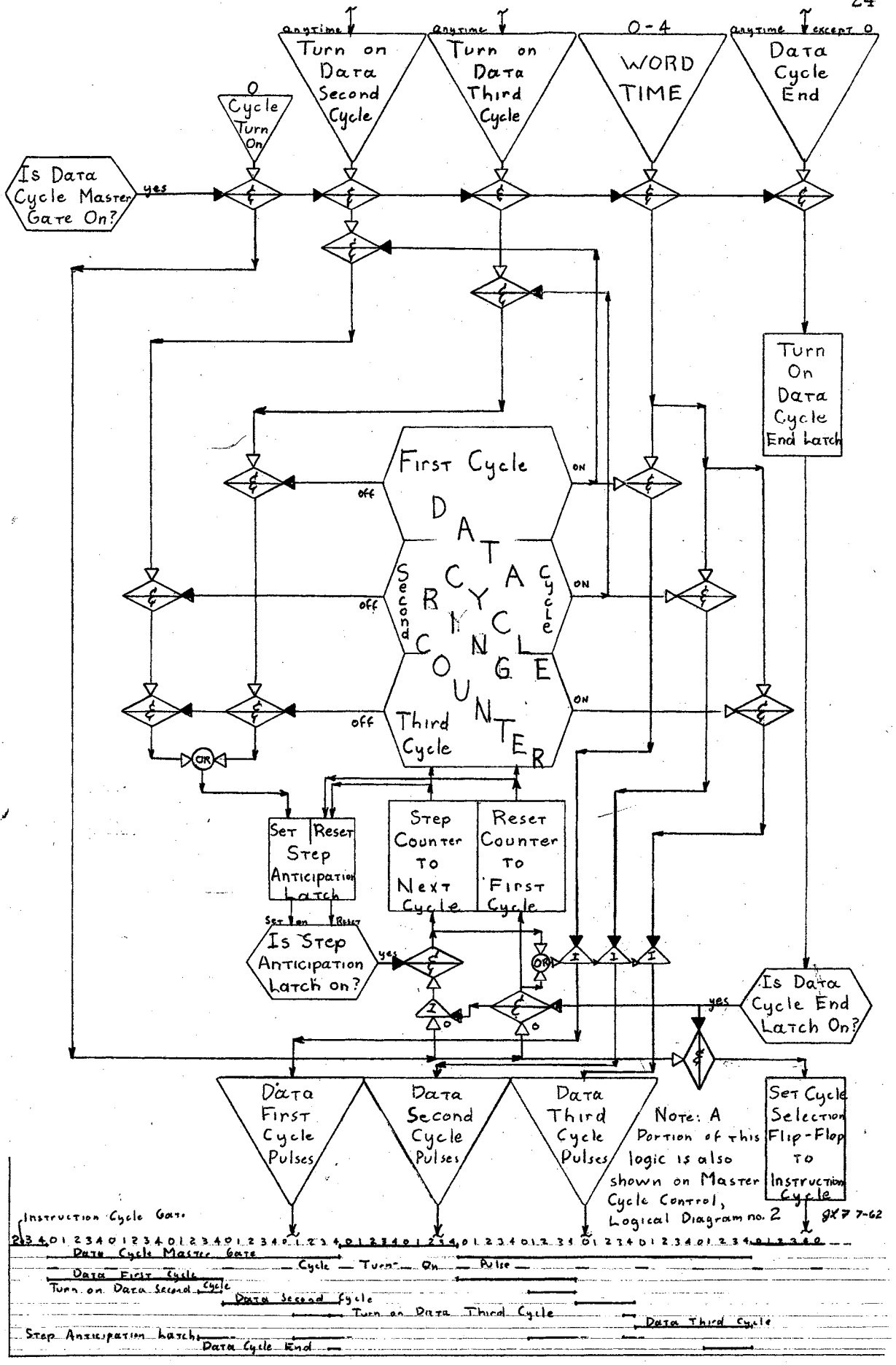


Figure 5. Logical Diagram number 4, Data Cycle Control.

be one or many word times in length). These sub-cycles are called "DATA FIRST CYCLE", "DATA SECOND CYCLE", and "DATA THIRD CYCLE". Whenever the computer goes from instruction cycle operation to data cycle operation, the first word time of the data cycle is always a DATA FIRST CYCLE. If an impulse is received during that word time to turn on a DATA SECOND CYCLE, the next word time will be a DATA SECOND CYCLE; if no such impulse is received, the machine will remain in DATA FIRST CYCLE operation until either a TURN-ON DATA THIRD CYCLE or a DATA CYCLE END impulse is received. If the machine is in DATA THIRD CYCLE operation, the only impulse that will bring about a change is a DATA CYCLE END impulse. The computer can never go from DATA THIRD CYCLE back to DATA FIRST CYCLE or DATA SECOND CYCLE nor can it go from DATA SECOND CYCLE to DATA FIRST CYCLE. In other words, the operation always starts with DATA FIRST CYCLE and proceeds (when so directed by "TURN-ON" impulses) to DATA SECOND CYCLE, and thence to DATA THIRD CYCLE. The data cycle may be ended at the end of any word time by a DATA CYCLE END impulse at any time (other than 0 time) of that word time, regardless of the occurrence of a TURN-ON impulse at any time during that word time. In other words, DATA CYCLE END takes precedence over a TURN-ON impulse.

The operation of the data cycle control will be illustrated using the timing chart and Logical Diagram number 4 (Figure 5). It is assumed that an instruction cycle precedes the data cycle. The instruction cycle ends at 0 time, and the DATA CYCLE MASTER gate turns on. The DATA CYCLE RING COUNTER has been previously reset to first cycle, and the STEP ANTICIPATION latch has been reset. Thus, as soon as the DATA

CYCLE MASTER gate is available, it gates the WORD TIME pulse to the first cycle AND circuit. Since the DATA CYCLE RING COUNTER is set to first cycle, the WORD TIME pulse is available to the rest of the machine as a DATA FIRST CYCLE pulse. The DATA FIRST CYCLE pulses are available for three word times. During the third word time, a TURN-ON DATA SECOND CYCLE impulse is received. Since the DATA CYCLE MASTER gate is on, the first cycle stage of the DATA CYCLE RING COUNTER is on , and both the second and third stages are off, the TURN-ON pulse sets the STEP ANTICIPATION latch. At 0 time of the next word time, the CYCLE TURN-ON pulse is available through an AND gate controlled by the DATA CYCLE MASTER gate. The CYCLE TURN ON pulse tests the setting of STEP ANTICIPATION latch by means of an AND circuit. If the STEP ANTICIPATION latch is set, the CYCLE TURN ON pulse momentarily inhibits the output of the WORD TIME pulses until the DATA CYCLE RING COUNTER can advance, and also resets the STEP ANTICIPATION latch. WORD TIME pulses are then available from the DATA SECOND CYCLE output.

The timing chart shows the DATA SECOND CYCLE pulses as being available for two word times. During the second word time, a TURN-ON DATA THIRD CYCLE impulse is received, which (since first and third cycles are off, and second cycle is on) turns on the STEP ANTICIPATION latch. Later, during the same word time, a DATA CYCLE END impulse is received (this pulse sequence can occur on accumulator add-in operations, among others; see Chapter 6). The DATA CYCLE END impulse turns on the DATA CYCLE END latch. At 0 time of the next word time, the CYCLE TURN-ON pulse is prevented

by the DATA CYCLE END latch from advancing the RING COUNTER. Instead, the CYCLE TURN-ON pulse resets the RING COUNTER to first cycle, resets the STEP ANTICIPATION latch, inhibits the WORD TIME pulse (to prevent signal race problems), and sets the CYCLE SELECTION FLIP-FLOP to data cycle (see, also, Logical Diagram number 2).

Two word times of instruction cycle are then shown on the timing chart. Following that come two word times of DATA FIRST CYCLE pulses, followed by one word time of DATA SECOND CYCLE. During this word time a TURN-ON DATA THIRD CYCLE is received, as before. Since no DATA CYCLE END pulse is received, however, the CYCLE TURN-ON pulse can then step the RING COUNTER to third cycle and reset the STEP ANTICIPATION latch. Two word times of DATA THIRD CYCLE follow, during the second of which a DATA CYCLE END impulse sets the DATA CYCLE END latch. The operation of resetting the various latches and changing to the instruction cycle proceeds as before.

It is important to remember that a data cycle may consist of a DATA FIRST CYCLE, DATA SECOND CYCLE, and DATA THIRD CYCLE. These cycles may consist of one or more word times, and are changed by TURN-ON DATA SECOND CYCLE or TURN-ON DATA THIRD CYCLE impulses. The first cycle taken after the machine goes into data cycle operation is the DATA FIRST CYCLE. If a DATA CYCLE END impulse occurs, data cycle operation ceases at the end of that word time, regardless of any TURN-ON pulses.

CHAPTER V

DATA ACQUISITION AND TRANSMISSION

Having developed the logic of providing control cycles for the computer, attention will now be directed toward the matters of acquiring data and instructions for immediate use, and storing data for future use.

Basically, the computer acquires a word for one of two purposes; either on a data cycle for use in arithmetic operations, in which case the word would be called an "operand", or on an instruction cycle, in which case the word would be used as the next instruction. Various miscellaneous transfers of words or portions of words may be made during the execution of certain instructions, but these will be of no concern in this chapter. The computer may acquire a word from either the magnetic drum, which will be called a "drum read" operation, or from an internal register, which will be called a "non-drum read" operation. Storage of words onto the magnetic drum will be termed a "drum write" operation, and may occur only during a data cycle (storage occurring during input operations is not considered here; see Chapter 10).

Drum Read operation is illustrated on Logical Diagram number 5 (Figure 6), and its associated timing chart. The basic operation is simply that the contents of some drum address is desired. The address is sought, and, when found, the bits in that location are read onto the DATA CHANNEL and thence into the appropriate register.

Read operations during a data cycle will only occur on DATA FIRST CYCLES. The presence of the OPERAND gate

DRUM READ

Approximately 1 time

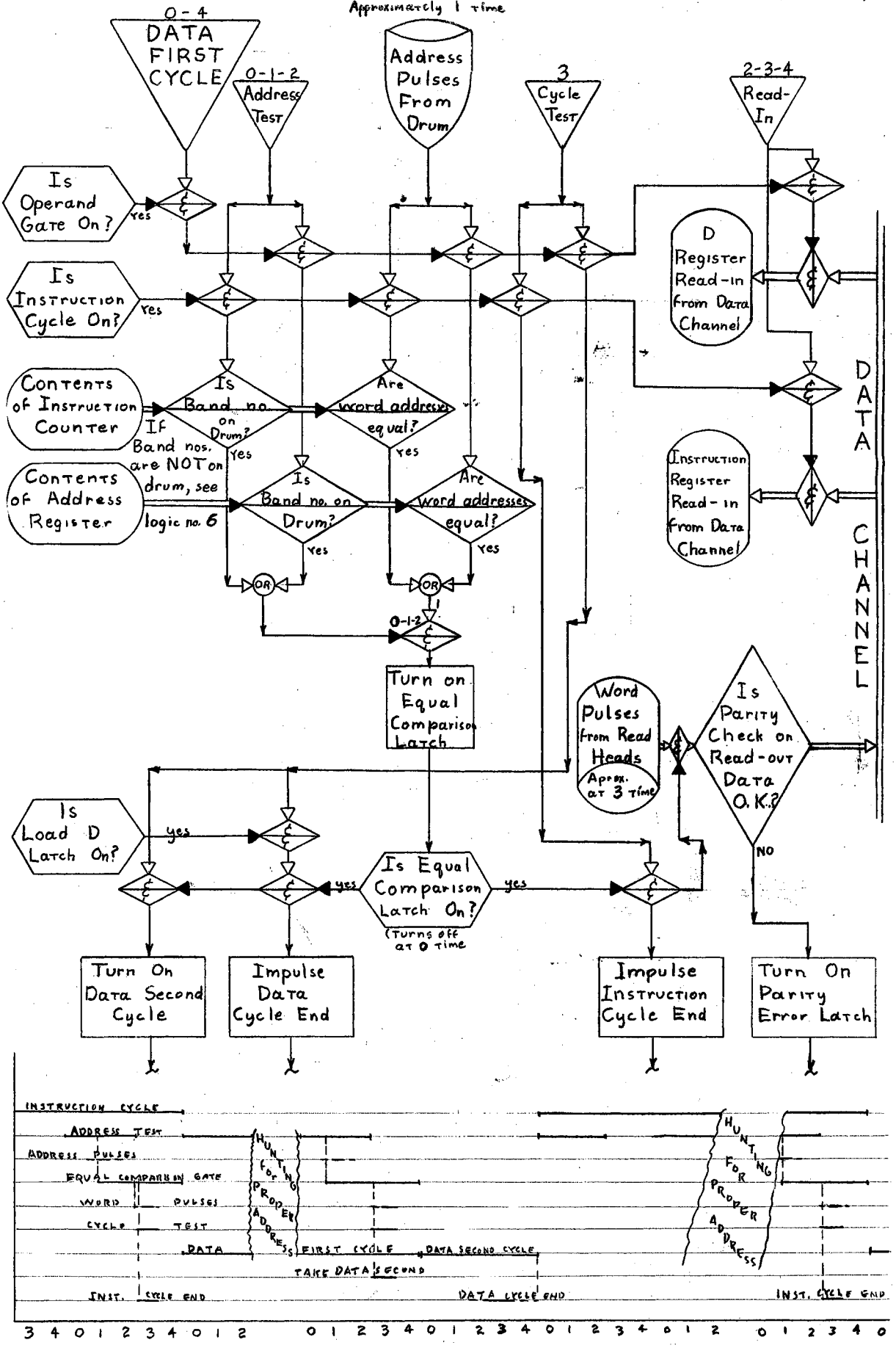


Figure 6. Logical Diagram number 5, Drum Read.

(from the OPERATION DECODER) signals that it is desired to read the contents of some storage location (or register) into the D-REGISTER. Read operations during an instruction cycle will be signaled simply by the presence of an INSTRUCTION CYCLE gate (since the only purpose of an instruction cycle is to acquire the next instruction). The contents of the desired storage location (or register) will be read into the INSTRUCTION REGISTER.

The ADDRESS pulses are available from the read heads at approximately 1 time, and are compared against the contents of the INSTRUCTION COUNTER (on instruction cycles) or against the contents of the ADDRESS REGISTER (on data cycles). If an equal comparison is detected, the contents of the word are read onto the DATA CHANNEL when the bit pulses are available from the read heads at 3 time, and are read in from the DATA CHANNEL into the appropriate register.

Using the timing chart at the bottom of Logical Diagram number 5, it is seen that at the beginning of the interval the INSTRUCTION CYCLE gate is assumed to be on. At 0-1-2 time the ADDRESS TEST pulse is gated by the INSTRUCTION CYCLE gate to interrogate the band number of the address contained in the INSTRUCTION COUNTER. If the band number is in the range 1 through 6, the ADDRESS TEST pulse is allowed to gate the ADDRESS pulses from the word address comparison circuitry. These ADDRESS pulses are read from the address track at approximately 1 time and, if the INSTRUCTION CYCLE gate is on, are compared with the word address portion of the INSTRUCTION COUNTER. If an unequal comparison occurs, no output results, and the comparison is made again with the next set of ADDRESS pulses. If, as in the timing chart, an equal comparison occurs, a 1 time pulse appears at the output

of the word address comparison circuit. This pulse is gated by the ADDRESS TEST pulse to turn on the EQUAL COMPARISON latch. At 2-3-4 time the READ IN pulse, gated by the INSTRUCTION CYCLE gate, gates the INSTRUCTION REGISTER to read in from the DATA CHANNEL. At approximately 3 time the EQUAL COMPARISON latch gates the bit pulses of the desired word into the PARITY REGISTER and thence onto the DATA CHANNEL, from which they are read into the INSTRUCTION REGISTER. The word is always read through the PARITY REGISTER onto the DATA CHANNEL, and if the parity is incorrect, the PARITY ERROR latch is turned on for future use. At 3 time the CYCLE TEST pulse tests the circuit, and since the INSTRUCTION CYCLE gate and the EQUAL COMPARISON latch are on, the CYCLE TEST impulses INSTRUCTION CYCLE END. The EQUAL COMPARISON latch, if on, is turned off at 0 time (not shown).

The next word time after an INSTRUCTION CYCLE END impulse is automatically a DATA FIRST CYCLE (see Chapter 4). If the OPERAND gate is on, the DATA FIRST CYCLE impulse allows the ADDRESS TEST pulse to test the band address of the ADDRESS REGISTER and the ADDRESS pulses from the word address track to be compared with the word address in the ADDRESS REGISTER. As is shown on the timing chart, several word times (512 maximum) may pass before the equal comparison occurs. When it does occur, the ADDRESS TEST pulse provides a gate allowing the equal comparison pulse to turn on the EQUAL COMPARISON latch. At 2-3-4 time the READ-IN pulse is gated by the DATA FIRST CYCLE-OPERAND gate to allow the D-REGISTER to be read in from the DATA CHANNEL, and at 3 time the bit pulses of the desired word are gated by the EQUAL COMPARISON latch to pass through the PARITY REGISTER, onto the

DATA CHANNEL, and into the D-REGISTER. Also at 3 time the CYCLE TEST pulse is gated by the DATA FIRST CYCLE-OPERAND gate to test the circuit. If the LOAD D-REGISTER gate were on, this would imply that the only operation desired was to read the desired word into the D-REGISTER, and as soon as this was accomplished, DATA CYCLE END would be impulsed. It will be noted that TURN-ON DATA SECOND CYCLE is always impulsed at 3 time of the word time during which the operand is acquired, and DATA CYCLE END is also impulsed if LOAD D is on.

Non-drum read, Logical Diagram number 6 (Figure 7), is similar to drum read, except that the desired word is located somewhere other than on the magnetic drum. The operation is exactly the same as far as the testing of the band numbers is concerned. However, the presence of a 0 band number indicates a non-drum location, and the operation proceeds somewhat differently from that point.

Using the timing chart on Logical Diagram number 6, it is found that a data cycle is the first complete cycle shown. At 0 time the DATA FIRST CYCLE pulse is gated by the OPERAND gate and allows the ADDRESS TEST pulse to test the band number of the ADDRESS REGISTER. The NON-DRUM READ latch is turned on at approximately 0 time (if a 7 band number were detected, the STORAGE SELECTION ERROR latch would also be turned on. The NON-DRUM READ latch, together with the DATA FIRST CYCLE-OPERAND pulse, gates the contents of the ADDRESS REGISTER in to the ADDRESS DECODER. The DATA FIRST CYCLE-OPERAND gate also allows the READ-IN pulse to gate the D-REGISTER for read-in from the DATA CHANNEL at 2-3-4 time. Meanwhile, the ADDRESS DECODER has selected the desired register, and at 3 time a WRITE pulse reads the contents

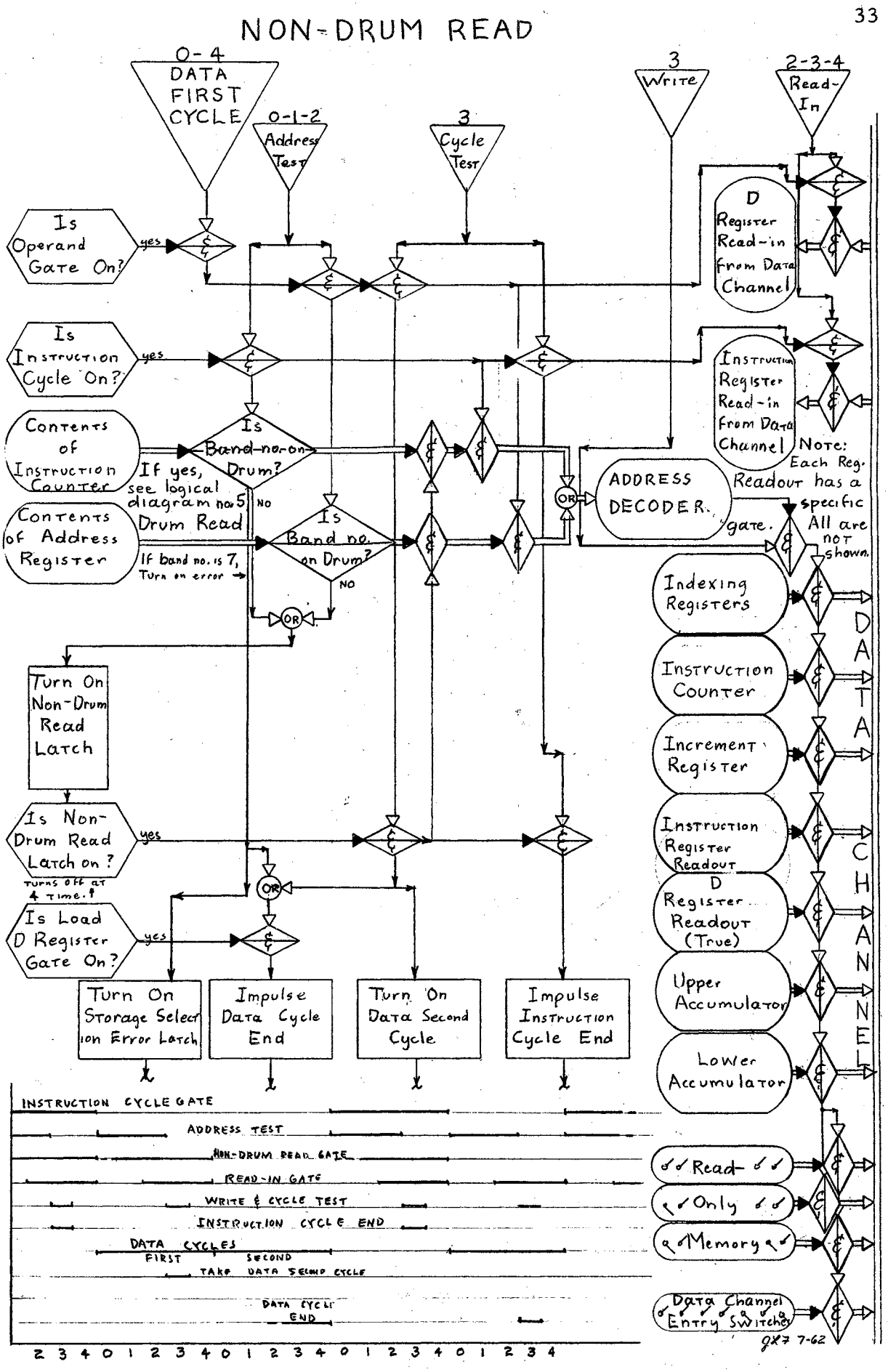


Figure 7. Logical Diagram number 6, Non-Drum Read

of the desired register onto the DATA CHANNEL and thence into the D-REGISTER. Also at 3 time the CYCLE TEST pulse interrogates the circuit through an AND gate from the DATA FIRST CYCLE-OPERAND gate. TURN-ON DATA SECOND CYCLE is always impulsed, and if the LOAD D-REGISTER gate is on, DATA CYCLE END is impulsed. The NON-DRUM READ latch is reset at 4 time. Note that only one word time is needed to acquire a non-drum operand.

If the INSTRUCTION CYCLE gate is on, the ADDRESS TEST pulse interrogates the band number of the INSTRUCTION COUNTER. If the band number is 0, the NON-DRUM READ latch is turned on and the contents of the INSTRUCTION COUNTER gated into the ADDRESS DECODER. The INSTRUCTION CYCLE gate allows the READ-IN pulse to gate the INSTRUCTION REGISTER to read in from the DATA CHANNEL at 2-3-4 time, and the WRITE pulse, together with the ADDRESS DECODER output, reads out the proper register onto the DATA CHANNEL. The CYCLE TEST pulse passes through the INSTRUCTION CYCLE gate and NON-DRUM READ latch gate to impulse INSTRUCTION CYCLE END at 3 time. The NON-DRUM READ latch is reset at 4 time to prevent transient outputs from the ADDRESS DECODER caused by the INSTRUCTION COUNTER being incremented at 4 time. Note that this seemingly innocuous Logical Diagram implies that the contents of any register may be used as either data or as the next instruction. This is a feature that is comparatively rare in computer design. (6).

The last class of operations covered in this chapter will be the "drum write" or "store" operations, Logical Diagram number 7, (Figure 8). In the OSTIC, a store operation will only occur on a data cycle, and is only valid for a drum address. The operation will be illustrated using a "store-upper accumulator"

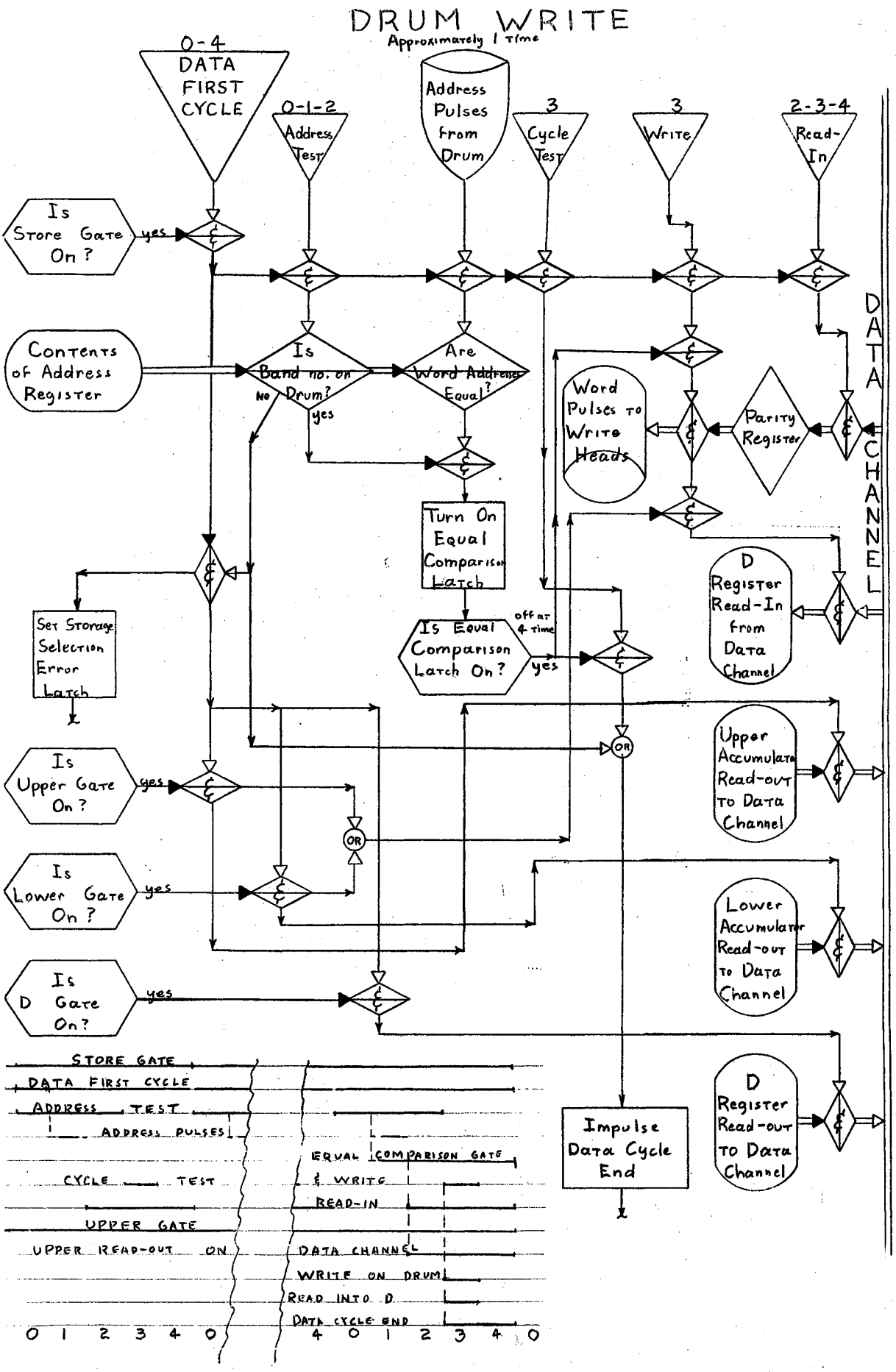


Figure 8. Logical Diagram number 7, Drum Write

operation, in which the contents of the UPPER ACCUMULATOR will be written onto the drum in the location specified by the contents of the ADDRESS REGISTER, and simultaneously read into the D-REGISTER.

The STORE gate allows a DATA FIRST CYCLE pulse to gate the ADDRESS TEST pulse at 0-1-2 time. If the band number of the ADDRESS REGISTER is on the drum, the ADDRESS TEST pulse provides a gate for EQUAL COMPARISON pulses. When the ADDRESS pulses from the address track of the drum are equal to the word address portion of the ADDRESS REGISTER, the EQUAL COMPARISON pulse is gated by the ADDRESS TEST pulse to turn on the EQUAL COMPARISON latch. The DATA FIRST CYCLE-STORE pulse is gated by the UPPER gate to read the contents of the UPPER ACCUMULATOR onto the DATA CHANNEL. At 2-3-4-time the READ-IN pulse is gated by the DATA FIRST CYCLE-STORE to allow the contents of the DATA CHANNEL to be read into the PARITY REGISTER, where the proper parity bit is generated. At 3 time the WRITE pulse is gated by the DATA FIRST CYCLE-STORE and the EQUAL COMPARISON latch to write the PARITY REGISTER output into the desired drum location, and also by the UPPER gate (or LOWER, if on) to read the contents of the DATA CHANNEL into the D-REGISTER. Also at 3 time the CYCLE TEST pulse is gated by DATA FIRST CYCLE-STORE and the EQUAL COMPARISON latch to impulse DATA CYCLE END.

Note that a non-drum band number will cause the ADDRESS TEST pulse to set the STORAGE SELECTION ERROR latch and impulse DATA CYCLE END. Also note that the presence of either an UPPER or LOWER gate will cause the contents of the appropriate register to be stored onto the drum and read into the D-REGISTER simultaneously, but the presence of a D gate

will cause only the contents of the D-REGISTER to be read out and stored.

To summarize, words are read into the D-REGISTER on DATA FIRST CYCLES when the OPERAND gate is on, or into the INSTRUCTION REGISTER when the INSTRUCTION CYCLE gate is on. The contents of either a drum location or a machine register may be read in this manner. Store operations, on the other hand, are valid only for drum addresses. These also take place on DATA FIRST CYCLE. The contents of the UPPER ACCUMULATOR or LOWER ACCUMULATOR may be stored on the drum and also automatically placed in the D-REGISTER. The contents of the D-REGISTER may also be stored on the drum.

CHAPTER VI

ACCUMULATOR OPERATIONS

The heart of a digital computer is the accumulator, for it is there that the majority of the operations that justify the existence of the computer are performed .

It is not within the purview of this paper to deal extensively with the detailed design of the accumulator circuits. The writer's philosophy is that the computer will present various gates and pulses to the accumulator at the proper time, and it is then up to those who design the accumulator to provide the desired results at the proper time.

The method of representation of numbers within the OSTIC will be in "sign and magnitude" binary form. In other words, a plus 27_{10} would be represented as 000000000000011011 with a 0 (plus) sign. A minus 27_{10} would be 000000000000011011 with a 1 (minus) sign. Numbers will always be stored as sign and magnitude, and will usually be used in this form.

The arithmetic operation performed by the accumulator will be that of binary addition of the DATA CHANNEL output to the either the UPPER or LOWER ACCUMULATOR, with end-around carry. In this case, both accumulators will be considered together as a single 36 bit ACCUMULATOR with the sign bit to the right of the low-order position (only one sign bit is used for the entire ACCUMULATOR). On add and subtract operations (but not multiply, shift, or logical operations), the sign bits will be added in a manner similar to the magnitude bits of the words, and a carry (if one occurs) is allowed to propagate from the sign position into the low-order position of the LOWER and

from the high-order position of the UPPER into the sign position. This is called an "end-around carry". For example, in adding $+27_{10}$ in the D-REGISTER to $+55_{10}$ in the LOWER, the entire ACCUMULATOR and D-REGISTER would appear as

```
D-REGISTER                                  000000000000011011 0 (sign)
ACCUMULATOR 000000000000000000000000000110111 0
Correct Answer 000000000000000000000000001010010 0
```

Although the ACCUMULATOR actually consists of two eighteen-bit numbers and a sign bit, to which is added an eighteen-bit number and a sign bit, most of the remaining examples in this paper will show an ACCUMULATOR consisting of a ten-bit word and a sign, to which is added a five-bit word and a sign. The preceding example should serve to illustrate how unwieldly eighteen- and thirty-six-bit examples can become. The use of a shorter word for purposes of explanation does not, of course, alter the manner in which the operations take place. Consider, for example, the case in which the ACCUMULATOR contains a +15, and a +17 is added from the D-REGISTER into the LOWER.

```
D-REGISTER (17)          10001 0
ACCUMULATOR (15) 0000001111 0
            (32) 0000100000 0
```

Subtraction is accomplished in the OSTIC by entering the sign-and-magnitude value of the number to be subtracted (subtrahend) into the D-REGISTER. The 1's complement value of the magnitude portion is then read onto the DATA CHANNEL and added into the ACCUMULATOR. A 1 is also added into the sign position. If the ACCUMULATOR has a minus sign, the magnitude of the number in the ACCUMULATOR must be placed in 1's complement form before the addition operation takes place. If the sign of the result is minus (1), the ACCUMULATOR must

be complemented after addition in order to reflect the correct sign-and-magnitude answer. On subtraction, an eighteen-bit word composed entirely of 1's (1's complement of zero) must be added into the LOWER if the 1's complement of the D-REGISTER is added into the UPPER, and similarly, 1's must be added into the UPPER if the complement of the D-REGISTER is added into the LOWER. If two negative numbers are added, an end-around carry from the high-order position of the UPPER into the sign position must take place and a complement cycle must be taken following the addition. If the end-around carry does not occur, then a number too large for the ACCUMULATOR (overflow) has been generated. If two positive numbers are added, an end-around carry must not take place, and no complement cycle is necessary. If an end-around carry does take place, then an overflow has occurred. If two numbers of opposite sign are added, an overflow is simply not possible, and the presence of an end-around carry indicates that no complement cycle is needed, while the absence of an end-around carry indicates that a complement cycle must follow. Table I will serve to illustrate the above-mentioned rules, using a five-bit UPPER, a five-bit LOWER, and a five-bit D-REGISTER.

This chapter relates the logical operations necessary to accomplish sixteen possible combinations of resetting the ACCUMULATOR prior to the operation, addition of the D-REGISTER contents to the UPPER or to the LOWER, subtraction of the D-REGISTER contents from the UPPER or from the LOWER. Furthermore, each operation may be executed in algebraic fashion or by using the absolute value (magnitude) of the operand. It is necessary, therefore, to examine the combinations of signs and operations which imply the use of true or complement values.

TABLE I

BINARY ADDITION WITH END-AROUND CARRY

11111	0	D-REGISTER at start.
0010111111	0	ACCUMULATOR at start.
0011011110	0	Correct answer.

No end-around carry occurs. No complement cycle needed.

Example 2. Add D to UPPER.

01111	0	D-REGISTER at start.
0010111111	0	ACCUMULATOR at start.
1010011111	0	Correct answer.

Note that if the sign bit of the number added is 0 (plus), nothing need be added to the LOWER when the D-REGISTER is added to the UPPER, and vice-versa.

Example 3. Add D to LOWER.

00001	0	D-REGISTER at start.
1111111111	0	ACCUMULATOR at start.
0000000000	0	Partial Sum.
	1	End-around carry.
0000000000	1	Incorrect answer.

This example illustrates an overflow. Note that, if both numbers are plus, an end-around carry signals an overflow.

Case II. Two negative numbers are added.

Example 1. Add D to LOWER.

11111	1	D-REGISTER at start.
0010111111	1	ACCUMULATOR at start.
00000	1	D-REGISTER complement.
11111		1's added into UPPER.
1101000000	1	ACCUMULATOR complement.
1100100001	0	Partial sum.
	1	End-around carry.
1100100001	1	Complement is necessary.
0011011110	1	Correct answer.

Note that the end-around carry is necessary to correct the sign, and that a complement cycle is needed after addition.

I (Continued)

Example 2. Add D to UPPER

01111	1	D-REGISTER at start
0010111111	1	ACCUMULATOR at start
10000	1	D-REGISTER Complement
11111	1	1's added into LOWER
1101000000	1	ACCUMULATOR Complement
0101100000	0	Partial sum
	1	End-around carry
0101100000	1	Complement is necessary
1010011111	1	Correct answer

Example 3. Add D to LOWER

00001	1	D-REGISTER at start
1111111111	1	ACCUMULATOR at start
11110	1	D-REGISTER Complement
11111		1's added into UPPER
0000000000	1	ACCUMULATOR Complement
1111111111	0	Partial sum
	?	No end-around carry, therefore an overflow has occurred
0000000000	0	Incorrect answer

Case III. Numbers of opposite signs are added.

Example 1. D-REGISTER is positive, ACCUMULATOR negative. Add D to LOWER. Result is negative.

11111	0	D-REGISTER at start
0010111111	1	ACCUMULATOR at start
11111	0	D-REGISTER true-figure.
1101000000	1	ACCUMULATOR Complement.
1101011111	1	Partial sum.
		No end-around carry occurs; therefore, complement result.
0010100000	1	Correct answer.

Example 2. D-REGISTER is positive, ACCUMULATOR negative. Add D to UPPER. Result is positive.

11111	0	D-REGISTER at start.
0010111111	1	ACCUMULATOR at start.
11111	0	D-REGISTER true-figure.
1101000000	1	Partial sum.
	1	End-around carry means no complement.
1100100001	1	Correct answer.

I (Continued)

Example 3. D-REGISTER is positive, ACCUMULATOR negative. Add D to UPPER. Result is negative.

```
      10111   0  D-REGISTER at start.
1111111111    1  ACCUMULATOR at start.
10111         0  D-REGISTER true-figure.
0000000000    1  ACCUMULATOR Complement.
1011100000    1  Partial sum.
                 No end-around carry, therefore complement
                 result.
0100011111    1  Correct answer.
```

Example 4. D-REGISTER is positive, ACCUMULATOR negative. Values are equal. Add D to LOWER. Result is zero.

```
      11011   0  D-REGISTER at start.
0000011011    1  ACCUMULATOR at start.
      11011   0  D-REGISTER true-figure.
1111100100    1  ACCUMULATOR Complement.
1111111111    1  Partial sum.
                 No end-around carry occurs; therefore,
                 complement result.
0000000000    1  Correct answer.
```

Note that it is possible to develop a negative zero.

Example 5. D-REGISTER is negative. ACCUMULATOR is positive. Add D to LOWER. Result is positive.

```
      11111   1  D-REGISTER at start.
0010111111    0  ACCUMULATOR at start.
      00000   1  D-REGISTER Complement.
11111            1's added into UPPER.
0010111111    0  ACCUMULATOR (not complemented).
0010011111    1  Partial sum.
                 End-around carry occurs; therefore, do
                 not complement result.
0010100000    0  Correct answer.
```

I (Continued)

Example 6. D-REGISTER is negative, ACCUMULATOR positive. Add D to UPPER. Result is negative.

```
     11111   1  D-REGISTER at start.
0010111111   0  ACCUMULATOR at start.
00000        1  D-REGISTER complement.
    11111       1's added into LOWER.
0010111111   0  ACCUMULATOR (not complemented).
0011011110   1  Partial sum.
                No end-around carry occurs; therefore,
                complement result.
1100100001   1  Correct answer.
```

Example 7. D-REGISTER is negative, ACCUMULATOR positive. Add D to UPPER. Result is positive.

```
     10111   1  D-REGISTER at start.
1111111111   0  ACCUMULATOR at start.
01000        1  D-REGISTER complement.
     11111      1's added into LOWER.
1111111111   0  ACCUMULATOR (not complemented).
0100011110   1  Partial sum.
               1 End-around carry occurs. Do not comple-
                 ment result.
0100011111   0  Correct answer.
```

Example 8. D-REGISTER is negative, ACCUMULATOR positive. Values are equal. Add D to LOWER. Result is zero.

```
     11011   1  D-REGISTER at start.
0000011011   0  ACCUMULATOR at start.
     00100   1  D-REGISTER complement.
11111           1's added into UPPER.
0000011011   0  ACCUMULATOR (not complemented).
1111111111   1  Partial sum.
                No end-around carry occurs. Complement
                result.
0000000000   1  A correct answer of a negative zero is
                developed.
```

I (Continued)

Example 9. D-REGISTER is negative, ACCUMULATOR positive. Values are equal. Add D to UPPER. Result is zero.

10101	1	D-REGISTER at start.
0101000000	0	ACCUMULATOR at start.
01010	1	D-REGISTER complement.
11111		1's added into LOWER.
1010100000	0	ACCUMULATOR (not complemented).
1111111111	1	Partial sum.
		No end-around carry occurs. Complement result.
0000000000	1	Correct answer.

Example 10. Add a positive number in the D-REGISTER to a negative zero in the ACCUMULATOR. Result should be same as D-REGISTER. Use add to LOWER.

10101	0	D-REGISTER at start.
0000000000	1	ACCUMULATOR at start.
10101	0	D-REGISTER true-figure.
1111111111	1	ACCUMULATOR complement.
0000010100	1	Partial sum.
	1	End-around carry occurred; therefore, do not complement result.
0000010101	0	Correct answer.

Example 11. Add a negative number in the D-REGISTER to a negative zero in the ACCUMULATOR. Result should be same as D-REGISTER. Use add to LOWER.

10101	1	D-REGISTER at start.
0000000000	1	ACCUMULATOR at start.
01010	1	D-REGISTER complement.
11111		1's added to LOWER.
1111111111	1	ACCUMULATOR complement.
1111101010	0	Partial sum.
	1	End-around carry.
1111101010	1	Complement is necessary.
0000010101	1	Correct answer.

Note that Examples 10 and 11 illustrate that addition of a positive or a negative number to a negative zero will result in the proper sum.

If the operation is an "Add Magnitude" (or "Reset Add Magnitude") operation, the contents of the D-REGISTER will be given a plus sign and the true value will be added into either the UPPER, or into the LOWER. If the operation is "Add" (or "Reset Add"), and the D-REGISTER is plus, the contents of the D-REGISTER will be given a plus sign and the true value will be added into either the UPPER, or into the LOWER. If the operation is "Add" (or "Reset Add"), and the D-REGISTER is minus, the contents of the D-REGISTER will be given a minus sign and the complement value will be added into either the UPPER, or into the LOWER.

If the operation is a "Subtact Magnitude" (or "Reset Subtract Magnitude") operation, the contents of the D-REGISTER will be given a minus sign and the complement value will be added into either the UPPER, or into the LOWER. If the operation is "Subtract" (or "Reset Subtract"), and the D-REGISTER is plus, the contents of the D-REGISTER will be given a minus sign and the complement value will be added into either the UPPER, or into the LOWER. If the operation is "Subtract" (or "Reset Subtract") and the D-REGISTER is minus the contents of the D-REGISTER will be given a plus sign and the true value will be added into either the UPPER, or into the LOWER.

At this point, it will be helpful to tabulate in Table II the rules established for ACCUMULATOR addition operations (the OSTIC's ACCUMULATOR, it will be remembered, only adds).

Logical Diagram number 8 (Figure 9A and 9-B) illustrates the embodiment of the rules in computer logic. No timing chart accompanies the diagram; it is felt that the above listed rules, together with several detailed examples, will better enable the reader to understand the operation.

TABLE II

RULES FOR ACCUMULATOR COMPLEMENT AND RESET

		Accumulator Sign	
		Plus	Minus
Reset gate on?	Yes	Reset entire Accumulator to plus zero before addition.	Reset entire Accumulator to plus zero before addition.
	No	Do nothing to Accumulator or to sign bit.	Take 1's complement of entire Accumulator. Sign bit remains one.

TABLE III

RULES FOR ACCUMULATOR OPERATION CODE AND SIGN

	Operation	
	Add	Subtract
Magnitude gate is on.	Read out true value of D onto Data Channel.	Read out complement value of D onto Data Channel.
Magnitude not on, D +	Read out true value of D onto Data Channel.	Read out complement value of D onto Data Channel.
Magnitude not on, D -	Read out complement value of D onto Data Channel.	Read out true value of D onto Data Channel.

TABLE IV

RULES FOR ACCUMULATOR ADD-IN

	Upper Gate On	Lower Gate On
True value read out of D-Register	Add contents of Data Channel into Upper. Add in 0 sign bit.	Add contents of Data Channel into Lower. Add in a 0 sign bit.
Complement value read out of D	Add contents of Data Channel into Upper. Add 1's into Lower. Add in a 1 sign bit.	Add contents of Data Channel into Lower. Add 1's into Upper. Add in a 1 sign bit.

TABLE V

RULES FOR ACCUMULATOR END-AROUND CARRY AND OVERFLOW

	True figure add-in is on. Accumulator is plus before addition.	Complement add-in is on. Accumulator is minus before addition.	True figure add-in is on, and Accumulator is -, or complement add-in is on, and Accumulator is +.
End-around carry.	Set Overflow latch. Impulse Data Cycle End.	Turn on Data Third Cycle.	Impulse Data Cycle End.
No end-around carry	Impulse Data Cycle End.	Turn on Data Third Cycle. Set Overflow latch.	Turn on Data Third Cycle.

ADDITION, SUBTRACTION, & COMPLEMENT

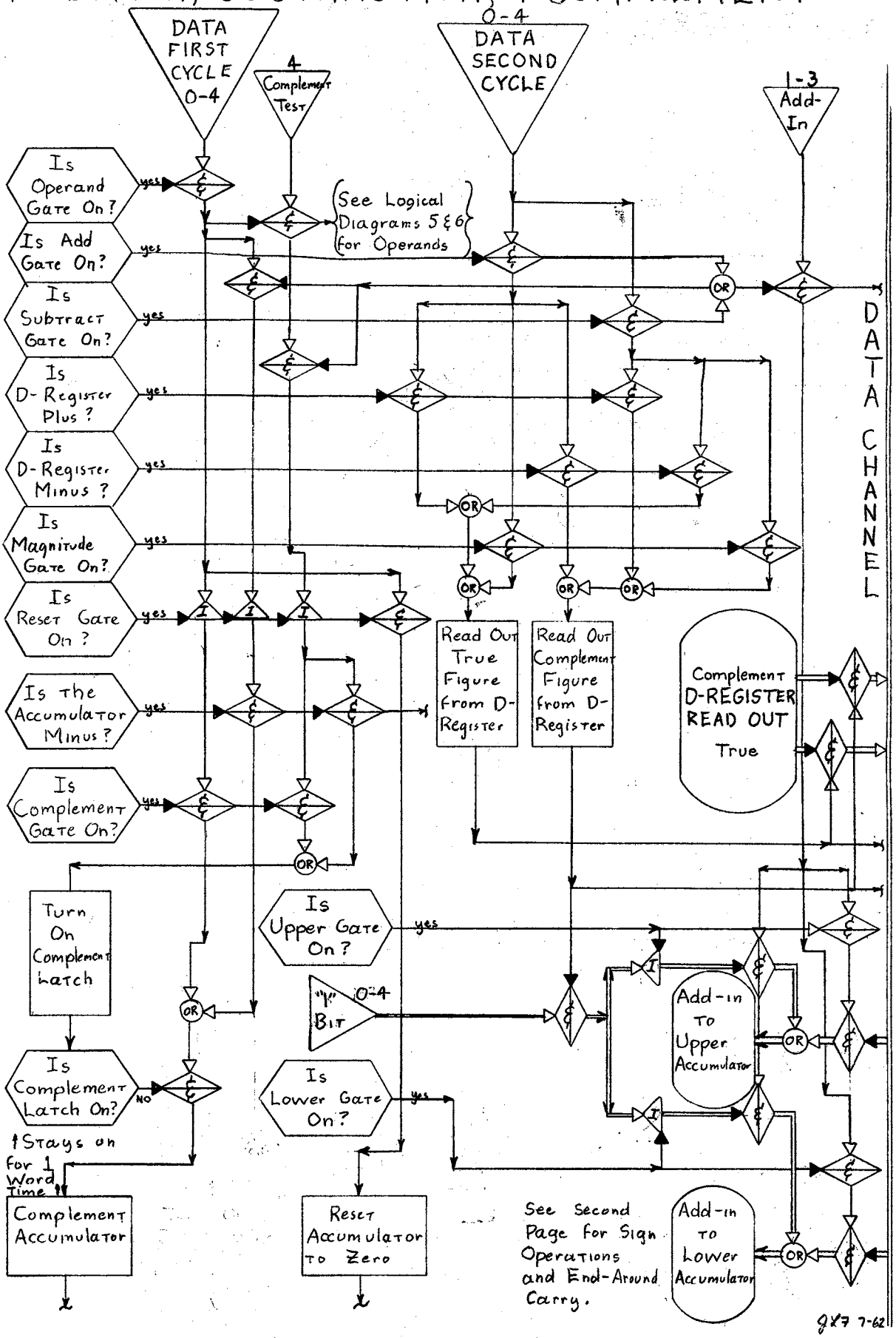


Figure 9-A. Logical Diagram number 8-
Addition, Subtraction, and Complement.

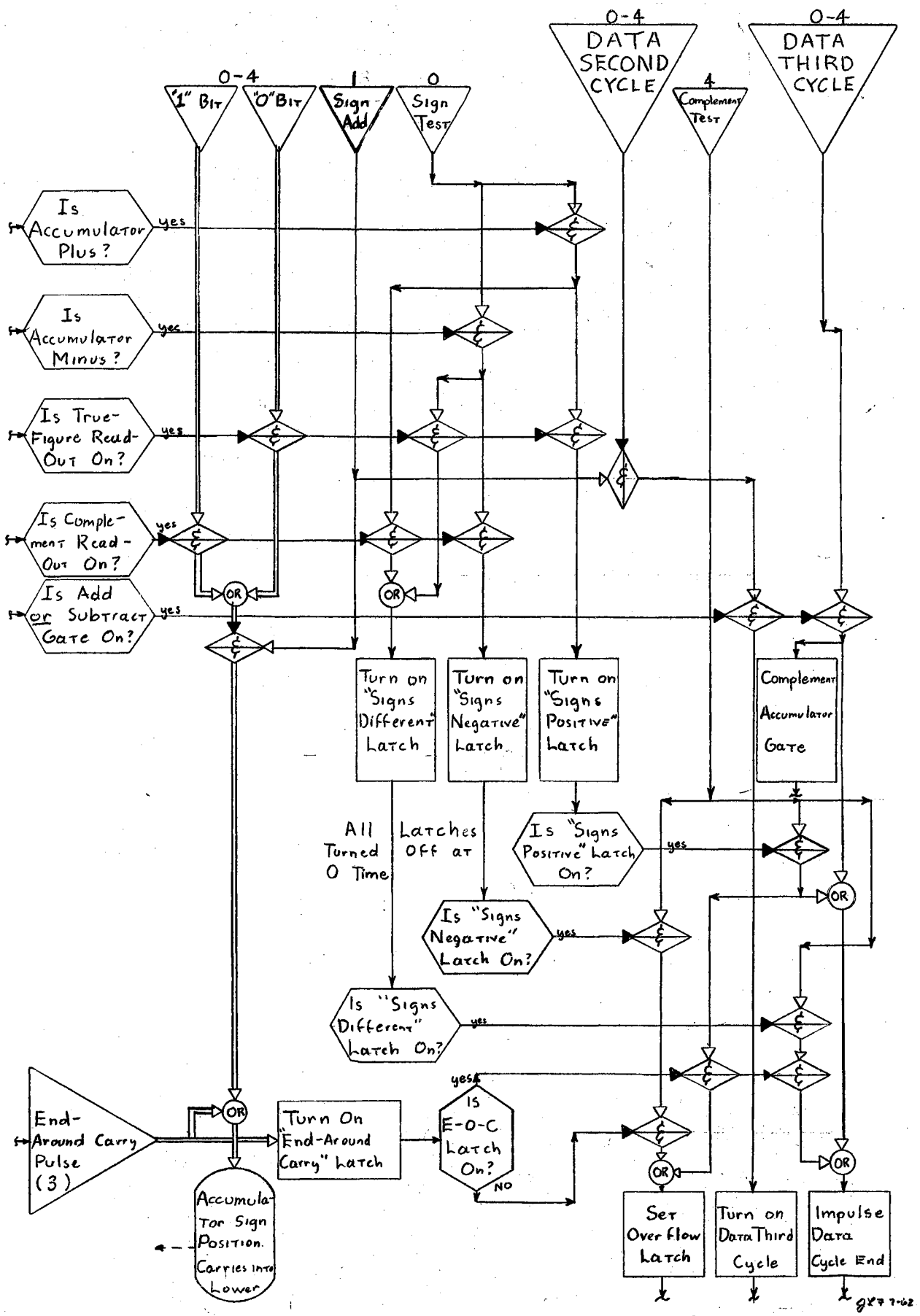


Figure 9-B. Logical Diagram number 8-

Addition, Subtraction, and Complement.

The first example considered will be an operation, where both values are plus, and no overflow results. The presence of an OPERAND gate and DATA FIRST CYCLE causes the computer to find the desired operand, as described in Chapter 5. The RESET gate is not on, and the ACCUMULATOR sign is plus, so neither reset nor complement takes place. When the operand is found, it is placed in the D-REGISTER and TURN-ON DATA SECOND CYCLE occurs. The DATA SECOND CYCLE pulse tests the signs of the registers, and, since both the D-REGISTER and ACCUMULATOR are plus, gates the D-REGISTER to read out its true figure onto the DATA CHANNEL. At 1-3 time the ADD-IN pulse adds the contents of the DATA CHANNEL and a "0" (plus) sign into the LOWER ACCUMULATOR and the SIGN POSITION, respectively (second sheet). At 0 time of the same word time, a SIGN TEST pulse had set up the end-around carry test circuits, and since a TRUE FIGURE READ OUT gate and an ACCUMULATOR PLUS gate were on, turned on the SIGNS POSITIVE latch. No end-around carry occurs, so the COMPLEMENT TEST pulse at 4 time impulses DATA CYCLE END.

The second example will be a "Reset Subtract Magnitude Upper" operation, using a positive operand. In this case, as before, the operand is acquired and placed in the D-REGISTER on DATA FIRST CYCLE. Meanwhile, on the first word time of DATA FIRST CYCLE, the ACCUMULATOR is reset to plus zero. When the operand is found and DATA SECOND CYCLE is turned on, the SUBTRACT, D-REGISTER PLUS, and MAGNITUDE gates are all on, so the D-REGISTER complement read-out is gated onto the DATA CHANNEL. Since the COMPLEMENT READ-OUT and UPPER gates are on, a word of 18 1's is added into the LOWER by the use of inhibit gates. The actual add-in

to both ACCUMULATORS is accomplished by the ADD-IN pulse. Since COMPLEMENT ADD-IN is on, a "1" is added into the sign position at 1 time by the SIGN ADD pulse (the ADD-IN pulse is not used, because of possible conflicts with an end-around carry). Also, the SIGN TEST pulse at 0 time turns on the "SIGNS DIFFERENT" latch. An end-around carry will not occur when any number is added to zero; therefore, the COMPLEMENT TEST pulse finds the END-AROUND CARRY latch off at 4 time and, since TURN-ON DATA THIRD CYCLE has been previously impulsed by the SIGN ADD pulse, a DATA THIRD CYCLE pulse complements the ACCUMULATOR and impulses DATA CYCLE END.

The last example used will be an "Add Lower" operation, where both the ACCUMULATOR and the operand are minus beforehand. It is assumed that an overflow will occur.

The DATA FIRST CYCLE pulse finds the RESET GATE off, the ADD gate on, the ACCUMULATOR MINUS gate on, and therefore complements the contents of the accumulator. At 4 time the DATA FIRST CYCLE-OPERAND gate and ACCUMULATOR MINUS gate allow the COMPLEMENT TEST pulse to turn on the COMPLEMENT latch. This latch will remain on until 4 time of the next word time, when it is turned on again by the COMPLEMENT TEST. The purpose of the COMPLEMENT latch is to prevent continuous re-complementing of the ACCUMULATOR.

The DATA SECOND CYCLE pulse finds the ADD GATE and the D-REGISTER MINUS gates on, and therefore provides a COMPLEMENT READ-OUT gate to the D-REGISTER. The COMPLEMENT READ-OUT and the LOWER gate cause 1's to be added into the UPPER, while the contents of the DATA CHANNEL is being added to the LOWER by the ADD-IN pulse. A "1" bit is added into the SIGN POSITION, and the SIGN TEST

pulse turns on the "SIGNS NEGATIVE" latch. It was assumed that an overflow occurred; for both signs negative, an over-flow occurs when an end-around carry does not occur. At 4 time, the COMPLEMENT TEST pulse finds the SIGNS NEGATIVE latch on and the END AROUND CARRY latch off, and therefore turns on the OVERFLOW latch. The complement cycle is still taken, however.

Note that DATA THIRD CYCLE is always turned on, and that DATA CYCLE END is then impulsed if necessary. This takes advantage of the fact that DATA CYCLE END overrides any TURN ON pulse (see Chapter 4). Also note that provision is made for a separate COMPLEMENT operation; see Chapter 7 for further discussion.

Multiplication in the OSTIC is accomplished by successive operations of shifting, testing, and addition. The multiplier is placed in the UPPER prior to the start of the multiply operation. At the beginning of the multiply operation, the multiplicand is acquired and placed in the D-REGISTER. The LOWER is reset to zero, and the entire ACCUMULATOR is then shifted left one position, the high-order position of the UPPER being shifted into a special "Test Position" just off the ACCUMULATOR. The TEST POSITION is then checked; if it contains a "1", the contents of the D-REGISTER (true value) is added into the LOWER, and the entire ACCUMULATOR is then again shifted left one position. If the test position did not contain a "1" when checked, only the left shift is performed. Following the left shift, the test position is again checked for a "1", and the add-and-shift or shift-only operation is repeated. This testing process is repeated eighteen times, because the word is eighteen bits long. The sign bits are neither added nor shifted; the sign is set to "0" (plus) if both the ACCUMULATOR

and D-REGISTER have the same sign, and to "1" (minus) otherwise.

Two examples of multiplication, using five-bit words, are presented in Table VI.

TABLE VI

BINARY MULTIPLICATION

Example 1. Multiply the number 01101 0 (+) by 11011 1 (-). Registers at beginning of Data Second Cycle:

	01101	0	D-REGISTER
	1101100000	1	ACCUMULATOR
?			Test position
1	1011000000	1	Shift left 1
1	1011001101	1	Add D to LOWER
1	0110011010	1	Shift left 1
1	0110100111	1	Add D to LOWER
0	1101001110	1	Shift left 1
0	1101001110	1	Do not add (test position contains a 0)
1	1010011100	1	Shift left 1
1	1010101001	1	Add D to LOWER
1	0101010010	1	Shift left 1
1	0101011111	1	Add D to LOWER
?	0101011111	1	Sign of answer is 1 (-)

Note that five-bit numbers were multiplied, and a total of five shifts were made. To check, the two numbers will be multiplied by the usual method:

$$
\begin{array}{r}
01101 \\
\times\ 11011 \\
\hline
01101 \\
01101 \\
00000 \\
01101 \\
01101 \\
\hline
0101011111
\end{array}
$$

The multiplicand is 13_{10}, and the multiplier is 27_{10}. Therefore, the product should be 321_{10}, which it is.

VI (Continued)

Example 2. Multiply 11111 1 (-) by 11111 1 (-)
Registers at beginning of Data Second Cycle.

	11111	1	D-REGISTER
	1111100000	0	ACCUMULATOR
?			Test position
1	1111000000	1	Shift left 1
1	1111011111	1	Add to LOWER
1	1110111110	1	Shift left 1
1	1111011101	1	Add to LOWER
1	1110111010	1	Shift left 1
1	1111011001	1	Add D to LOWER
1	1110110010	1	Shift left 1
1	1111010001	1	Add D to LOWER
1	1110100010	1	Shift left 1
1	1111000001	1	Add D to LOWER
?	1111000001	0	Sign of answer is 0

Note that this is the largest possible product of two five-bit numbers, and illustrates that a product longer than ten bits can never be developed. Similarly, the largest product of two eighteen-bit numbers is thirty-six bits long.

Checking:

```
    11111
  x 11111
  -------
    11111
   11111
  11111
 11111
11111
---------
  1011101
1011101
---------
111010001
11111
---------
1111000001
```

Logical diagram number 9 (Figure 10) presents the "Multiply" operation. The DATA FIRST CYCLE pulse resets the LOWER to zeros, sets the five-position AUXILIARY COUNTER to 01110 (14_{10}), and sets the SHIFT OR ADD latch to shift. DATA SECOND CYCLE then interrogates the SHIFT OR ADD latch for the next thirty-six word times, as is shown on the timing chart (only the add cycles are numbered; eighteen of these must occur). During the first word time DATA SECOND CYCLE finds the SHIFT OR ADD latch in the shift setting, so the entire ACCUMULATOR is shifted one position to the left, which brings the high-order bit of the UPPER into the TEST POSITION. At 4 time the MISCELLANEOUS RESET pulse sets the SHIFT OR ADD latch to add, and during the next word time a "1" bit is added into the AUXILIARY COUNTER by the ADD-IN pulse. If the TEST POSITION contains a "1", the contents of the D-REGISTER (which is read onto the DATA CHANNEL on all DATA SECOND CYCLES) is added into the LOWER. The MISCELLANEOUS RESET pulse turns the SHIFT OR ADD latch back to shift at 4 time.

The operation will continue until an overflow pulse from the AUXILIARY COUNTER impulses DATA CYCLE END. The overflow pulse is the carry pulse out of the high-order position of the counter. Since the counter has five positions, the overflow pulse will occur when a "1" is added to a 31_{10}, or in binary form,

```
  11111   counter
      1   add 1
  -----
  00000   result
1         overflow
```

Note that if 01110 is placed into the counter to begin with, then 10010 (18_{10}) added in will cause an overflow. Note also that the logic is so designed that the last addition to the LOWER (if signaled by a "1" in the TEST POSITION) occurs simultaneously

MULTIPLICATION

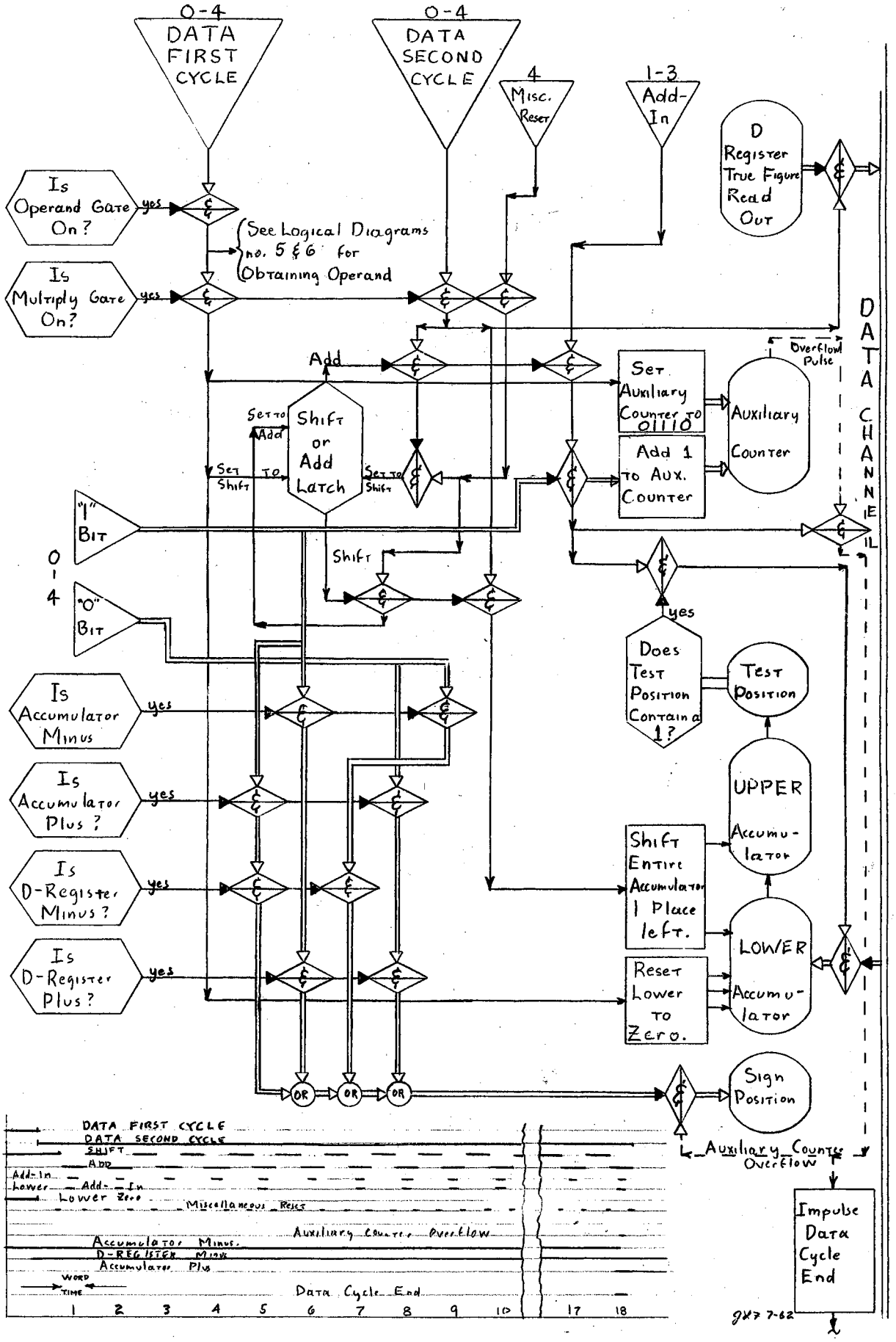


Figure 10. Logical Diagram number 9, Multiplication.

with the addition to the AUXILIARY COUNTER, which causes the overflow.

Since the sign of the product is determined from the sign of the D-REGISTER and the sign of the ACCUMULATOR, setting the sign, should be the last matter taken care of, for otherwise an oscillation of sign could occur from one word time to the next. The sign is therefore set up immediately when the multiplication begins, but is not placed in the ACCUMULATOR sign position until the overflow, which signals the end of the operation, has occurred.

The timing chart illustrates the multiplication of some number in the D-REGISTER by 1100101111.....01.

Logical Diagram number 10 (Figure 11) illustrates the process of shifting the entire ACCUMULATOR one or more positions to either the left or right. There is a great deal of similarity between this process and the process of multiplication, since the AUXILIARY COUNTER is incremented each time a shift operation takes place, and the presence of an overflow from the AUXILIARY COUNTER signals that the shifting is complete. One difference will be noticed, however. Since no addition takes place, the ACCUMULATOR is shifted at the same time that the counter is incremented, rather than on the following word time.

The timing chart on Logical Diagram number 10 shows five different shift operations. Reading from left to right, the first sequence is a nine-position shift, followed by shifts of five, seven, one, and zero positions. The nine-position shift will be described in detail.

The DATA FIRST CYCLE pulse finds the SHIFT gate on, and so gates the 1's complement value of the ADDRESS REGISTER to be read out onto the DATA CHANNEL. At 3 time of the same

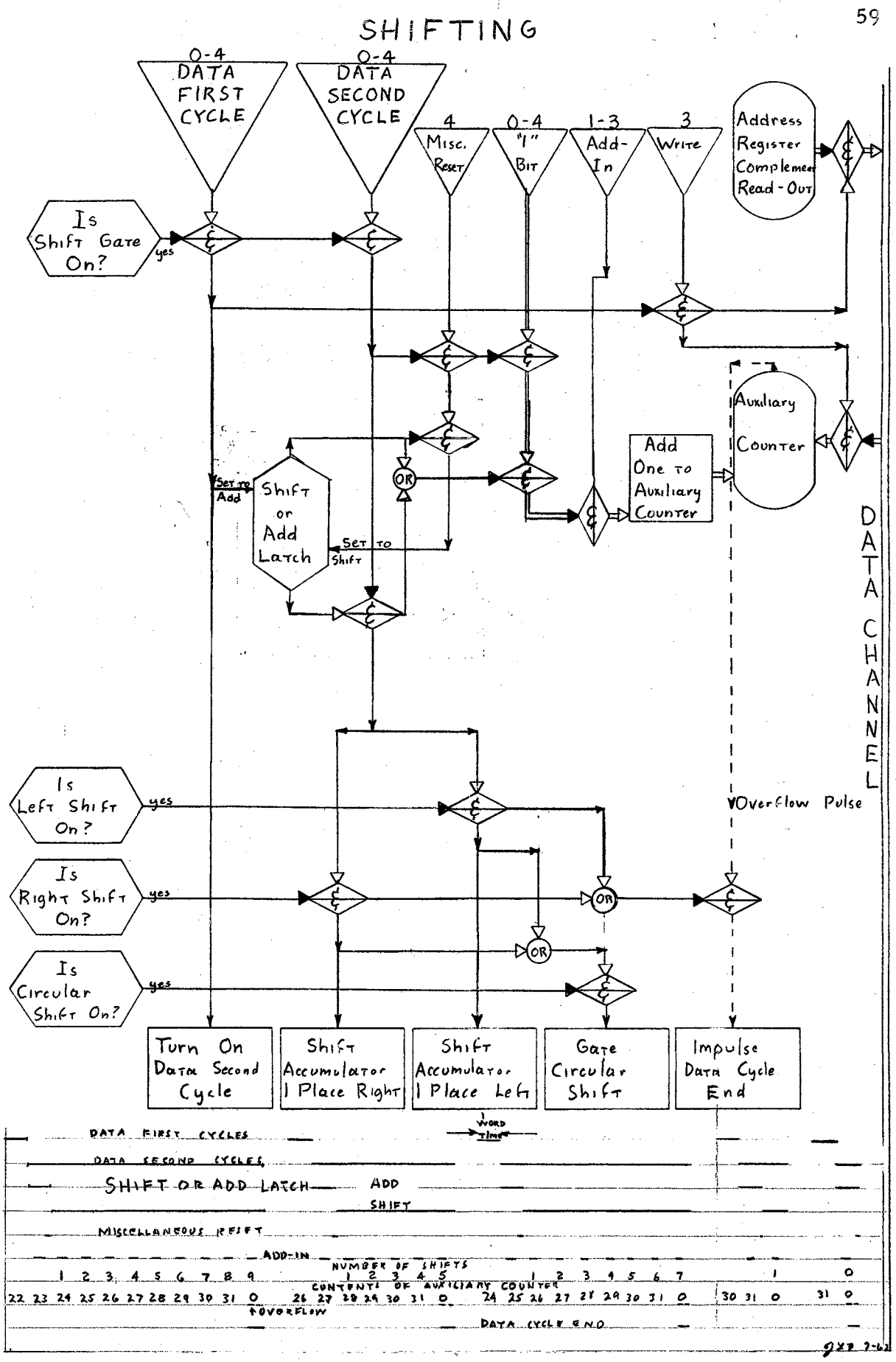


Figure 11. Logical Diagram number 10, Shifting.

word time, a WRITE pulse copies the contents of the DATA CHANNEL into the AUXILIARY COUNTER. The DATA FIRST CYCLE pulse also sets the SHIFT OR ADD latch to add, and impulses TURN-ON DATA SECOND CYCLE.

The DATA SECOND CYCLE impulse finds the SHIFT OR ADD latch set to add, and so, together with the ADD-IN pulse, adds a "1" bit into the AUXILIARY COUNTER. At 4 time the MISCELLANEOUS RESET pulse sets the SHIFT OR ADD latch to shift. The next word time finds the DATA SECOND CYCLE pulse shifting the ACCUMULATOR either left or right, and circular or non-circular, as determined by the various control gates (a circular shift occurs when the bits shifted out of one end of the accumulator reappear, in order, at the other end). Note that the SHIFT OR ADD latch is not set back to add, but remains in the shift setting. Also note that addition of "1's" to the AUXILIARY COUNTER takes place on both add and shift settings. When the AUXILIARY COUNTER overflows, DATA CYCLE END is impulsed, and the operation is complete.

In examining the theory behind this operation, it is found that the largest number that may be stored in a five-position binary counter is 31_{10}, or 11111. An overflow occurs when 1 is added to 11111 already in the counter. Note that the 1's complement of any five-bit binary number is also the number 31's complement. For example, the 1's complement of 01001 (9_{10}) is 10110 (22_{10}). Therefore, if the 1's complement of the desired number of shifts is entered into the AUXILIARY COUNTER prior to the shift operation, and if 1 is added to the counter for each shift of one position, then the counter will contain thirty-one when the desired number of shifts has occurred. However, the counter overflows when 1 is added to thirty-one, making thirty-two. Therefore, a single addition

is made prior to any shifting, so that the addition occurring simultaneously with the last shift will cause the counter to overflow, and end the operation.

It will be noted that a data address of zero (00000) on a shift instruction will correctly result in a zero-place shift. Also, the greatest number of shifts that may take place for one instruction is thirty-one (11111).

CHAPTER VII

LOGICAL OPERATIONS

Because of the special-purpose and instructional applications of this computer, a repertoire of logical operations would be quite useful. Although actually carried out in the ACCUMULATOR, the nature of these operations is somewhat different from the operations covered in Chapter 6. Accordingly, this chapter is devoted solely to logical operations.

Logical operations, as considered here, are those operations involving two binary numbers, where the value of each position of the result (1 or 0) is dependent only upon the values in the corresponding position of the original numbers. All logical operations may be represented by a "truth table", where the values of the two input numbers, (which will be called, for want of a better name, the "A-operand" and the "B-operand") are shown along the top and left side, and the values of the result are shown at the intersection of the appropriate rows and columns.

Example 1.

Truth Table-OR addition.

B-operand \ A-operand	0	1
0	0	1
1	1	1

Sample Operation

A-operand	0011
B-operand	0101
Result	0111

Since there are two possible values for the A-operand, two possible values for the B-operand, and four possible values for the result, there are 16 possible logical operations. These are shown in Table VII.

TABLE VII

BINARY LOGICAL OPERATIONS

A / B	0 1 / 0 0 1 / 1 1 0	0011 / 0101 / 0110
Logical Ring addition Exclusive OR (no carry) $A\bar{B} + \bar{A}B$		
A / B	0 1 / 0 1 0 / 1 1 1	0011 / 0101 / 0111
OR; A + B Inclusive OR		
A / B	0 1 / 0 0 0 / 1 0 1	0011 / 0101 / 0001
AND (Logical Multiply) AB		
A / B	0 1 / 0 0 1 / 1 0 0	0011 / 0101 / 0010
$A\bar{B} = \bar{A} + B$		
A / B	0 1 / 0 1 1 / 1 0 1	0011 / 0101 / 1011
$A + \bar{B}$		
A / B	0 1 / 0 0 1 / 1 0 1	0011 / 0101 / 0011
TRIVIAL		

A / B	0 1 / 0 1 0 / 1 0 1	0011 / 0101 / 1001
Logical Compare $AB + \bar{A}\bar{B}$		
A / B	0 1 / 0 1 0 / 1 0 0	0011 / 0101 / 1000
Not OR (NOR) $\overline{AB} = \overline{A + B}$		
A / B	0 1 / 0 1 1 / 1 1 0	0011 / 0101 / 1110
Not AND (NAND) $\overline{AB} = \bar{A} + \bar{B}$		
A / B	0 1 / 0 1 0 / 1 1 1	0011 / 0101 / 1101
$\bar{A} + B$		
A	0 1 / 0 0 0 / 1 1 0	0011 / 0101 / 0100
$\bar{A}B = A + \bar{B}$		
A / B	0 1 / 0 1 0 / 1 1 0	0011 / 0101 / 1100
TRIVIAL		

VII (Continued)

	A	
	0 1	0011
B	0 0 0	0101
	1 1 1	0101

TRIVIAL B

	A	
	0 1	0011
B	0 1 1	0101
	1 0 0	1010

TRIVIAL $\overline{B}$

	A	
	0 1	0011
B	0 1 1	0101
	1 1 1	1111

TRIVIAL 1

	A	
	0 1	0011
B	0 0 0	0101
	1 0 0	0000

TRIVIAL 0

It appears that there are a maximum of 10 logical operations that one might conceivably wish to perform, and six "don't cares". (8). Consider the process of "ring addition", with a provision for complementing either the A-operand, the B-operand, or both, and complementing or not complementing the result, as shown in Table VIII.

TABLE VIII

LOGICAL RING ADDITION

B-operand	A-operand A 0011	A-operand $\overline{A}$ 1100	
B	0110	1001	True Answer
0101	1001	0110	Complemented Answer
$\overline{B}$	0110	1001	Complemented Answer
1010	1001	0110	True Answer

TRUTH TABLE

Logical Ring Add

	A	
	0 1	
B	0 0 1	
	1 1 0	

Apparently, only two results are available. The result, taking the true values of both the addend and augend, is the ring sum, while the complement of the ring sum is the logical comparison. This might have been deduced from Figure 5, since the truth table for ring addition is the same as the complement of the truth table for logical compare. Note also that there are two ones and two zeros in both truth tables.

Consider now the OR operation. Again, either the addend, augend, or both may be complemented, and the result may or may not be complemented, as shown in Table IX.

TABLE IX

LOGICAL OR ADDITION

TRUTH TABLE
OR Addition

B \ A	0	1
0	0	1
1	1	1

	A-OPERAND		
	A	$\bar{A}$	
B-OPERAND	0011	1100	
B	0111	1101	True Answer
0101	1000	0010	Complemented Answer
$\bar{B}$ 1010	0100	0001	Complemented Answer
	1011	1110	True Answer

Note that this time eight different answers were obtained. In fact, upon examining Table VIII, it is found that all sixteen of the possible logical operations may be obtained by the use of various combinations of complement, ring addition, OR addition, and (for all zero or all ones) resetting the ACCUMULATOR to zeros.

Logical Diagram number 11 (Figure 12) shows the method of executing the "Logical Ring Add" and "Logical OR" instructions.

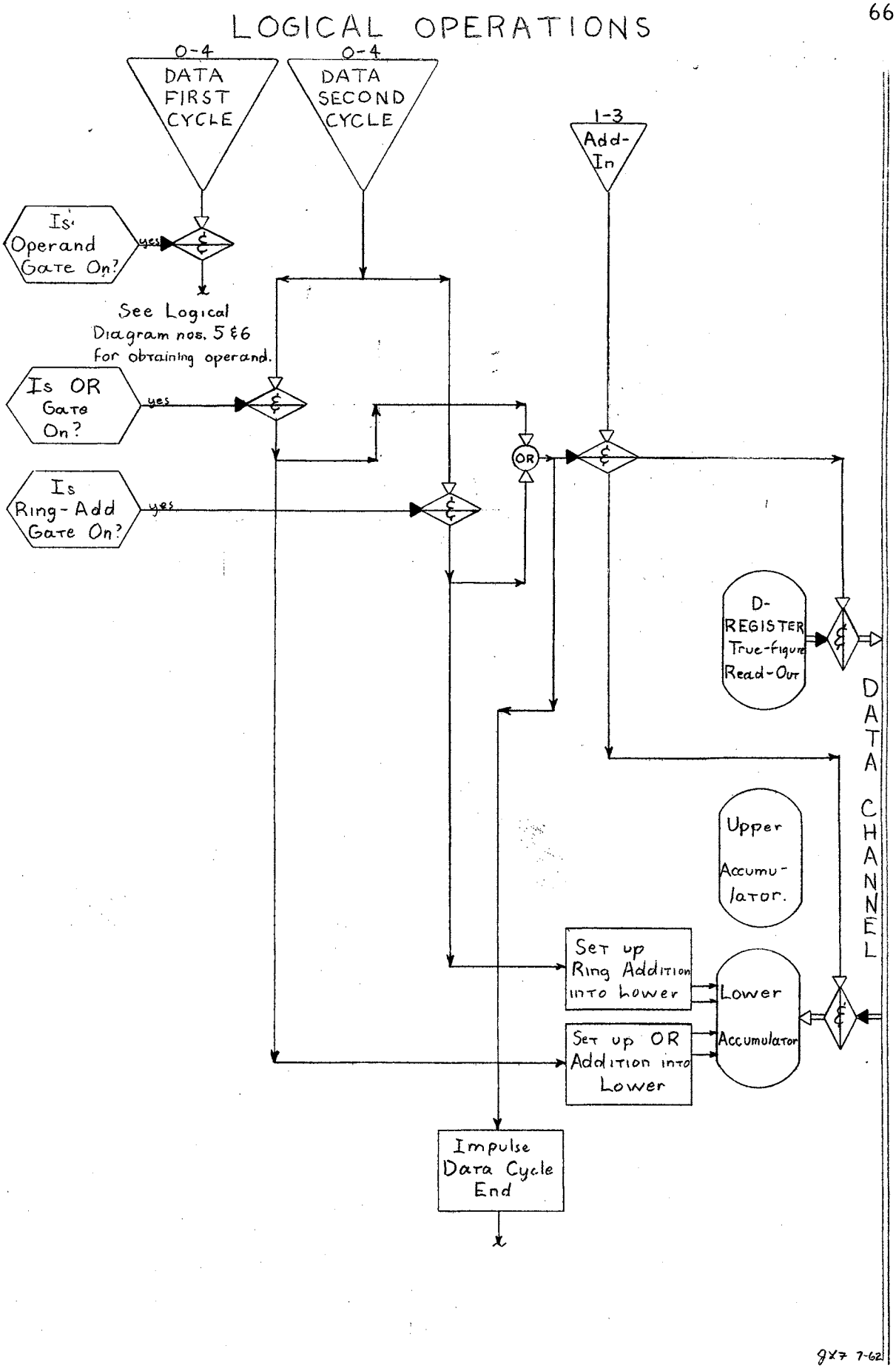


Figure 12. Logical Diagram number 11, Logical Operations.

The B-operand is assumed to have already been placed in the LOWER, and the location of the A-operand is designated by the data address of the instruction. It is acquired and placed in the D-REGISTER on DATA FIRST CYCLE. Following this, the DATA SECOND CYCLE pulse, together with the presence of either the OR gate or the RING-ADD gate, reads out the true value of the D-REGISTER onto the DATA CHANNEL at 0-4 time, and also impulses DATA CYCLE END. The DATA SECOND CYCLE pulse is also gated to set up the LOWER ACCUMULATOR to perform the proper operation, since it is intended that these special operations will only be performed on the contents of the LOWER and the location specified by the data address of instruction. At 1-3 time the ADD-IN pulse gates the contents of the DATA CHANNEL into the LOWER, and the operation is performed.

The "Complement" operation is shown on Logical Diagram number 8 (Figure 9-A). It is intended that a "Load D-Register" operation be carried out in conjunction with the complement operation; therefore, Logical Diagram numbers 5 and 6 are also involved. The complement would occur on the first word time of DATA FIRST CYCLE; the operation would be complete when the desired operand was found and placed in the D-REGISTER. It is felt that, since complement will be used primarily with the "Logical Ring Add" and "Logical OR" operations, it would save time to use the same instruction to bring one of the desired operands into the D-REGISTER. See Chapter 11 for a further discussion of this topic.

CHAPTER VIII

TESTING AND BRANCHING OPERATIONS

One of the fundamental concepts which makes the stored-program digital computer a powerful tool is the concept of internal testing and program branching. In the OSTIC, a branch, or "jump" operation, will be defined as any operation where the address of the next instruction is taken from the data address of the previous instruction, rather than from the address normally generated by incrementing the INSTRUCTION COUNTER. In general, there are a number of different types of branch codes. The simplest is the "Unconditional Jump", where the computer "jumps" to the instruction located in the address corresponding to the data address of the jump instruction. Another type of jump code is a "Test and Jump", where the computer tests some given condition, and then jumps only if the condition is true. For example, a "Jump Accumulator Minus" instruction would cause the computer to test the sign of the ACCUMULATOR. If the sign were minus, the computer would jump to the address specified; if the ACCUMULATOR were plus, however, the jump would not take place, and the next instruction would be taken from the address in the INSTRUCTION COUNTER. The most complex type of jump is the "Copy Jump", where the machine first "copies" the contents of the INSTRUCTION COUNTER into the D-REGISTER, then jumps to the location specified by the data address of the jump instruction. This type of instruction is almost a necessity for subroutine linkage on a single-address computer.

Logical Diagram number 12 (Figure 13) illustrates the

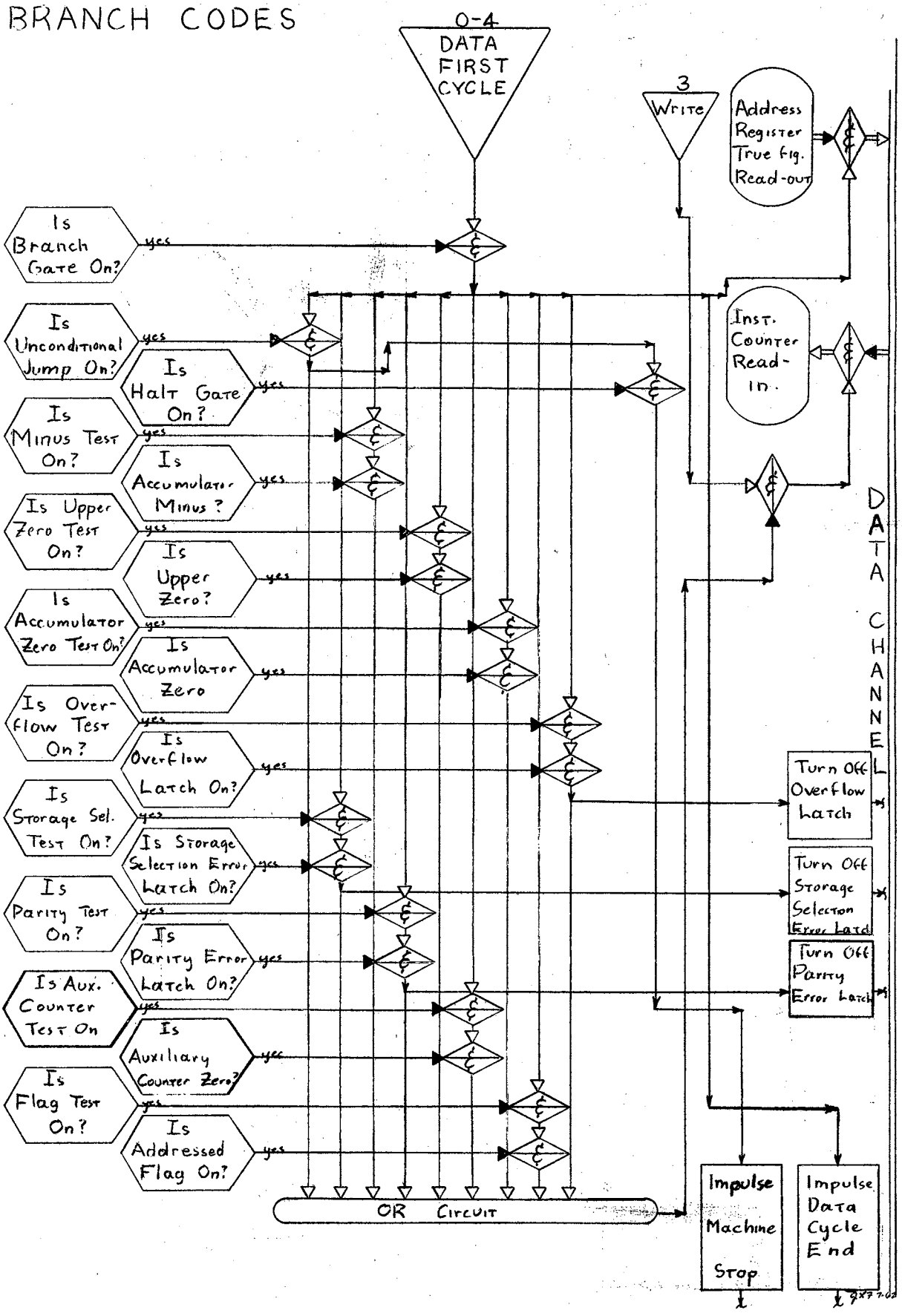


Figure 13. Logical Diagram number 12, Branch Codes.

"Unconditional Jump", together with a number of "Test and Jump" codes. The operation is quite straight-forward. At 0-4 time of the first data-cycle word time, a DATA FIRST CYCLE pulse finds the BRANCH gate on, and is thus enabled to test the various jump gates. For example, if the UNCONDITIONAL JUMP gate is on, the DATA FIRST CYCLE pulse finds a path to the OR circuit. Meanwhile, the DATA FIRST CYCLE pulse has gated the ADDRESS REGISTER contents onto the DATA CHANNEL, and at 3 time the WRITE pulse, gated by the DATA FIRST CYCLE pulse which traveled through the UNCONDITIONAL JUMP gate, reads the contents of the DATA CHANNEL into the INSTRUCTION COUNTER. The DATA FIRST CYCLE impulse also impulses DATA CYCLE END.

As another example, consider the operation shown on Logical Diagram number 12 for a "Jump on Overflow" instruction. The BRANCH gate is on, and the OVERFLOW TEST gate is on, so the DATA FIRST CYCLE pulse tests the OVERFLOW latch. If the OVERFLOW latch is not on, nothing happens, and since DATA CYCLE END is always impulsed, the program continues in regular sequence. However, if the OVERFLOW latch is on, the contents of the DATA CHANNEL are read into the INSTRUCTION COUNTER, and the OVERFLOW latch is turned off. Note that the ADDRESS REGISTER is always read out onto the DATA CHANNEL at 0-4 time if the BRANCH gate is on, but the DATA CHANNEL is read into the INSTRUCTION COUNTER at 3 time only if a jump is desired. This design reduces signal race problems that would occur if both the read-out and read-in were conditional upon the jump being desired.

Logical Diagram number 13 (Figure 14) illustrates the "Copy-Jump" operation. Here, the DATA FIRST CYCLE pulse through the BRANCH gate and the COPY JUMP gate reads out the INSTRUC-

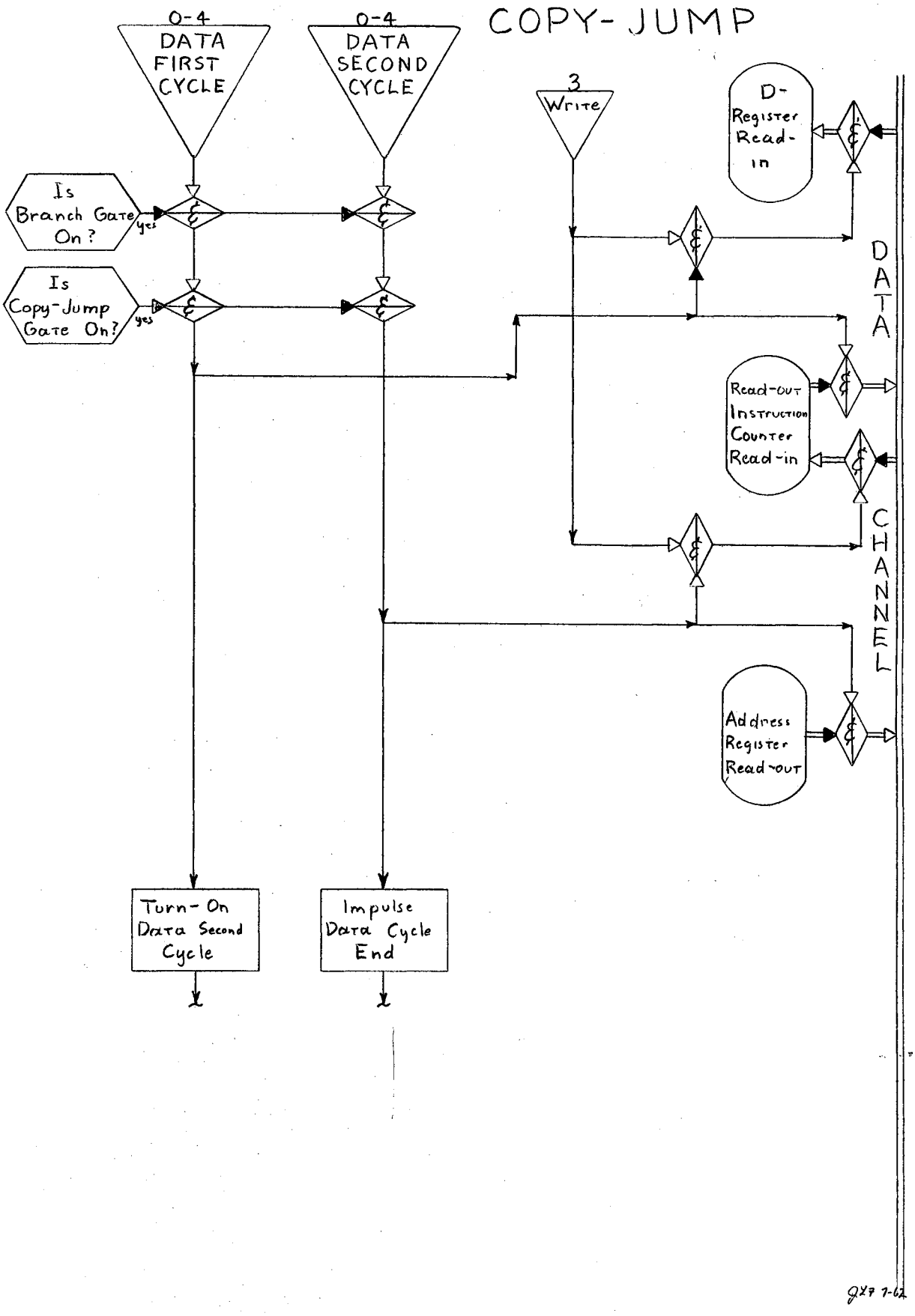


Figure 14. Logical Diagram number 13, Copy-Jump.

TION COUNTER onto the DATA CHANNEL.

It will be remembered that the contents of any internal machine register may be used for either data or instructions, that is, the address of any register may be given either as a data address in an operation, or may be entered into the INSTRUCTION COUNTER (either by means of some type of "jump" operation, or by normal incrementation of the INSTRUCTION COUNTER).

Consider at this point waht would happen if a "Copy-Jump" instruction were given with some of the various registers as the data address. For example, if the INSTRUCTION COUNTER was given is the address, then the contents of the INSTRUCTION COUNTER would simply be placed in the D-REGISTER, and the program would continue in normal sequence. Again, the D-REGISTER itself could be given as the address of the jump, in which case the contents of the INSTRUCTION COUNTER would be entered into the D-REGISTER, following which the address of the D-REGISTER itself, which, since it was the data address of the instruction, would be contained in the ADDRESS REGISTER, would be transferred into the INSTRUCTION COUNTER. Following the transfer, the computer would go into instruction cycle operation, during which (it will be remembered from Chapters 4 and 5) the contents of the memory location specified by the address in the INSTRUCTION COUNTER will be obtained and placed in the INSTRUCTION REGISTER to serve as the next instruction. However, since the D-REGISTER'S address was given, the contents of the D-REGISTER will become the next instruction.

As was stated, DATA FIRST CYCLE read out the INSTRUC-

TION COUNTER at 0-4 time, and also allows the WRITE pulse to read the contents of the DATA CHANNEL into the INSTRUCTION COUNTER at 3 time. The DATA FIRST CYCLE pulse also impulses TURN-ON DATA SECOND CYCLE. The DATA SECOND CYCLE pulse then reads out the ADDRESS REGISTER onto the DATA CHANNEL at 0-4 time and allows the WRITE pulse to read the contents of the DATA CHANNEL into the INSTRUCTION COUNTER at 3 time. The DATA SECOND CYCLE pulse also impulses DATA CYCLE END.

Logical Diagram number 14 (Figure 15) illustrates the "Flag Branching" operations. The flag operations are simply a three-step branching sequence that provide a much greater degree of flexibility to computer operations than is possible with the standard branching operation. Essentially, the operation is simply this. Any one of 512 different on-off conditions (such as, "Is tape drive number 2 ready?", or "Is the divide-overflow latch on?") may be tested by an instruction. If the designated condition is true, then any one of eight program "flags", or resettable latches, may be turned on, or "set". These latches will remain on until turned off by a specific turn-off or "reset" command. Any of the flags may be interrogated at any time by a branch command, in the form of "Jump if Flag X is on". The set command is "Set Flag X if Switch YYY is on", and the reset command is "Reset Flag X". The octal instruction word format is as follows:

07 XYYY.....Set Flag X if Switch YYY is On.

06 X.....Reset Flag X.

1X ZZZZ.....Jump to location ZZZZ if Flag X is On.

It is thus seen that eight separate program flags may be used, with 512 possible conditions, or switches, being tested. Of course, the choice of the digits 0, 1, 6, and 7 in the operation codes is arbitrary.

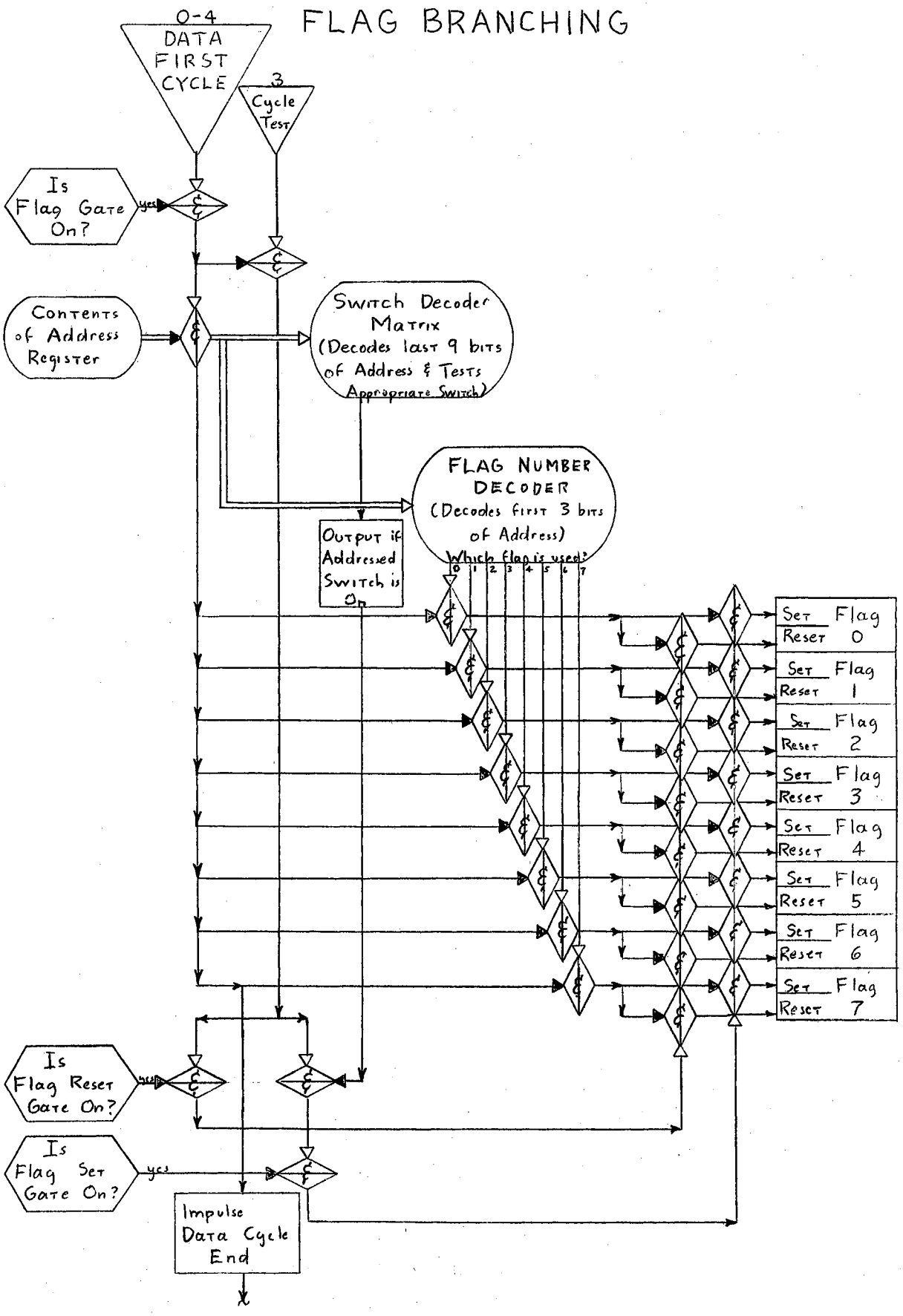


Figure 15. Logical Diagram number 14, Flag Branching.

The operation of testing a program flag and branching if the flag is on is shown on Logical Diagram number 12. The operation is executed in the same manner as any other test and jump operation; if the TEST FLAG X gate is on and the FLAG X latch is on, the contents of the ADDRESS REGISTER is transferred to the INSTRUCTION COUNTER. Otherwise, operation proceeds in normal sequence.

The operation of setting and resetting a program flag is shown on Logical Diagram number 14 (Figure 15). If the FLAG gate is on, the contents the nine low-order binary positions of the ADDRESS REGISTER is gated into a SWITCH DECODER matrix, which accomplishes the testing of the addressed switch to see if it is off or on. The next three binary positions of the ADDRESS REGISTER go into the FLAG NUMBER DECODER, whose function is to determine which of the flags (flag 0 through flag 7) is addressed by that instruction. The FLAG NUMBER DECODER then gates the DATA FIRST CYCLE pulse into the set-reset circuitry of the proper flag. At 3 time a CYCLE TEST pulse then may set or reset the FLAG latch in question. If the FLAG RESET gate is on, the CYCLE TEST pulse is allowed to gate the reset line of the appropriate FLAG latch, and the DATA FIRST CYCLE pulse resets the latch. If a Test Switch YYY" operation is called for, the output of the SWITCH DECODER matrix will occur prior to 3 time, provided the addressed switch is on. This output, together with the FLAG SET gate, allows the CYCLE TEST pulse to gate the DATA FIRST CYCLE to set the appropriate FLAG latch. The DATA FIRST CYCLE pulse also impulses DATA CYCLE END.

CHAPTER IX

MISCELLANEOUS OPERATIONS

A number of miscellaneous operations are required to round-out the instruction repertoire of any digital computer. This chapter will cover these, and in so doing, will complete (except for input/output) the list of operations presented in this paper.

Operations of a more-or less miscellaneous character that have been covered previously (and will not, therefore, be discussed again) include "Load D-Register" and "Store D-Register" (Chapter 5), Shifting (Chapter 6), and "Complement" (Chapter 7).

A fundamental operation in any computer is the "Continue", or "No Operation" instruction, often referred to as "no-op". This is simply an operation that does nothing but continue the program to the next instruction. Because of the fact that all registers in the OSTIC will be addressable on both data and instruction cycles, it is proposed that the OSTIC have no separate no-op instruction. Instead, the "Copy-Jump" may serve as a no-op if 00_8 is made equivalent to the "Copy-Jump" operation code. Then, if a no-op were desired, a "Copy-Jump" could be made to the D-REGISTER. This would simply result in the contents of the INSTRUCTION COUNTER being placed in the D-REGISTER, and then transferred into the INSTRUCTION REGISTER. Since the INSTRUCTION COUNTER is a twelve-position counter, the INSTRUCTION REGISTER would appear as $00XXXX_8$, where $XXXX_8$ is the octal address of the next instruction to be executed. The 00 would cause a second copy jump to be executed, this time to XXXX.

It is further proposed that the "Halt", or stop code , on this

computer be a jump code. As illustrated in Logical Diagram number 12, a "Halt-Jump" code would place the address of the next instruction in the INSTRUCTION COUNTER and impulse MACHINE STOP (see Chapter 10). When the machine is started again, the first instruction executed will be that to which the transfer had been made.

Three other miscellaneous operations are shown on Logical Diagram number 15 (Figure 16). The first of these is the "Set Increment" instruction. Here, the DATA FIRST CYCLE pulse, gated by the SET INCREMENT gate, reads out the ADDRESS REGISTER onto the DATA CHANNEL. At 3 time the WRITE pulse reads the contents of the DATA CHANNEL into the INCREMENT REGISTER. DATA CYCLE END is impulsed on the first word time by DATA FIRST CYCLE for all operations shown.

The other two operations deal with the AUXILIARY COUNTER. The presence of an AUXILIARY COUNTER in the computer suggests that some sort of operation be designed to utilize its counting ability independent of such operations as multiplication and shifting. The "Set Auxiliary Counter" operation simply transfers the contents of the ADDRESS REGISTER (which contains the data address of the "Set Auxiliary Counter" instruction) into the AUXILIARY COUNTER. This is accomplished in one word time by a DATA FIRST CYCLE pulse which, if the AUXILIARY COUNTER gate is on, reads out the ADDRESS REGISTER onto the DATA CHANNEL and allows the WRITE pulse to read the contents of the DATA CHANNEL into the AUXILIARY COUNTER. The other operation is the "Increment Auxiliary Counter" operation. Here, a "1" is simply added into the low-order position of the AUXILIARY COUNTER. This operation is accomplished in one word time by the DATA FIRST CYCLE pulse which, when gated by the INCREMENT AUXILIARY COUNTER gate from the OPERATION DECODER, allows a "1"

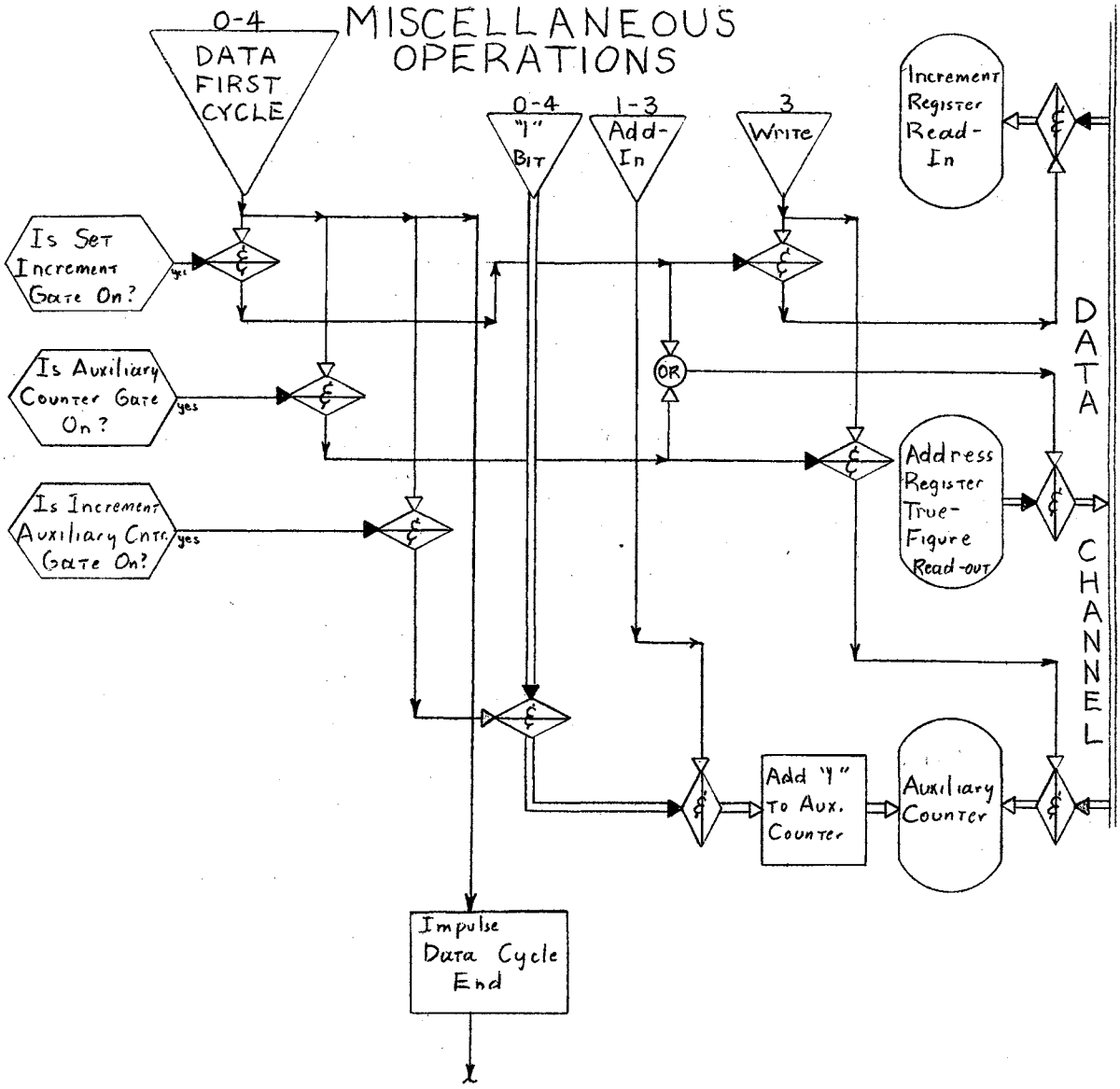


Figure 16. Logical Diagram number 15, Miscellaneous Operations.

BIT pulse to be added into the AUXILIARY COUNTER by the ADD IN pulse at 1-3 time. Chapter 11 covers the use of the AUXILIARY COUNTER in programming.

CHAPTER X

INPUT/OUTPUT AND CONSOLE OPERATIONS

This paper will devote little space to the question of input/output(I/O) for the OSTIC, simply because at the time of writing no input/output equipment is available for use with the machine. It is anticipated, however, that some type of input/output device will be obtained when needed.

Faced with not knowing even the type of input/output equipment (much less its specifications), the system designer can only speculate and make recommendations. It is intended, however, that definite "space" be left in the operation repertoire for a variety of input/output codes. To this end, it is recommended that all eight 7X8 instructions be reserved for I/O operations. Further, it is recommended that all $7XXX_8$(data) addresses be reserved for the same purpose. By thus anticipating the need well in advance, perhaps the problem of having a needed I/O instruction and no place to put it (in the command list) will never arise.

As far as the logical organization of I/O operations is concerned, this is, at best, generalization and recommendation. Therefore, no Logical Diagram for I/O is presented. What would probably be done is that an INPUT or an OUTPUT gate would be turned on, and then the contents of the OPERATION and ADDRESS DECODERS would be made available to the I/O equipment for use in determining the operation to follow. It is ordinarily preferable that blocks consisting of several words be transferred on single I/O operations. In this case, the length of the block would probably be determined by the characteristics of the I/O unit or its buffer. A "drum read" or "drum write" operation might be used here, with the proper number of words

being transferred off of or onto the drum, starting with the location specified by the data address of the I/O instruction. Consideration might also be given to the construction of a small (16 or 32 words) high-speed memory unit, constructed using either magnetic cores or, perhaps, various types of memory devices (to compare the characteristics of each). This memory could serve as a buffer for various I/O equipment.

Unlike the question of input/output, the matter of the computer console can be discussed in more concrete terms. Here, the questions of ease of demonstration and simplicity of operation arise. Before considering the console proper, however, it might be well to discuss some of the operating features needed. First, both the demonstrator and operator will require a means of displaying the contents of the various registers, and methods of changing them if necessary. This was previously discussed, and it was proposed that separate display lights and entry switches be provided for each register. A read-only memory was also deemed desirable; this would consist simply of toggle switches (no display lights are needed) in groups of 19, each group corresponding to one computer word. For slow-speed operation and maintenance, lights indicating the setting of the CYCLE SELECTION FLIP-FLOP (data or instruction cycle) and the setting of the DATA CYCLE RING COUNTER would be useful. Similarily, an indication of the status (set or reset) of the eight program flags would be handy for purposes of demonstration, and, of course, the operator should know if any of the error latches (overflow, storage selection, parity, timing, etc.) were set.

Logical Diagram number 16 (Figure 17) illustrates the various settings and uses of an "Operation Switch" on the console, in conjunction with an " OPERATION" latch that would determine whether the machine was running, or stopped. If the OPERATION latch is set to "stop", then no pulses are available. As is shown, a number

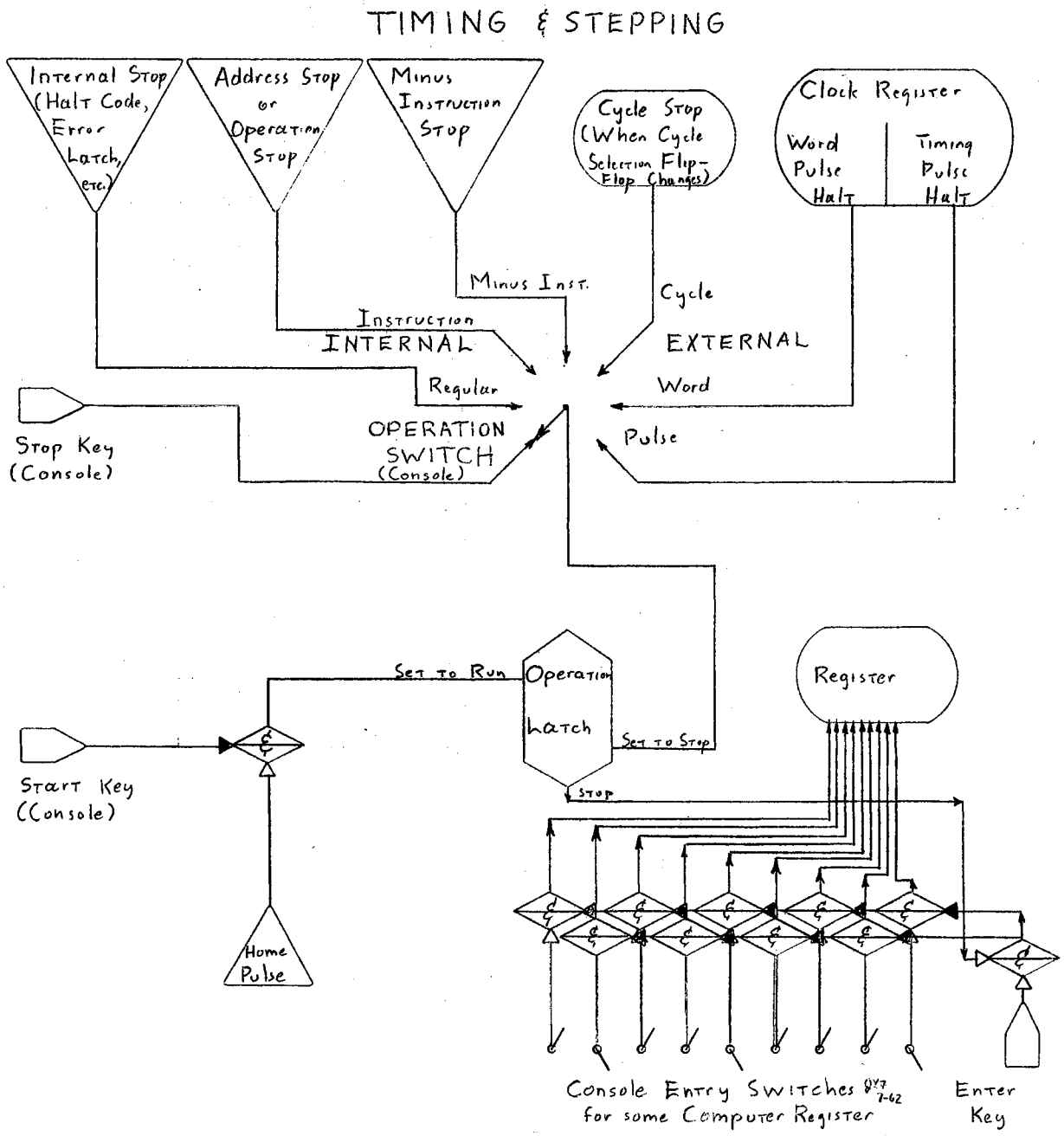


Figure 17. Logical Diagram number 16, Timing and Stepping.

may be entered into a register from the console only when the OPERATION latch is set to stop. The desired number is simply set into the console entry switches corresponding to the desired register, and the "Enter" key for that register depressed. In order to start the machine, the "Start" key on the console is then depressed, allowing the "HOME" pulse to set the OPERATION latch to run. The function of the OPERATION switch is merely to determine in what manner the computer may be stopped. On "Internal" operation, the machine may be set to stop only when the console "Stop" key is depressed, or when an "internal stop" occurs. An internal stop may be a halt code, timing error, storage selection error, or (if desired by the operator) an overflow, or parity error. In addition, a stop may be desired when the contents of either the ADDRESS REGISTER or the OPERATION REGISTER is equal to some predetermined number entered into the console entry switches of the INSTRUCTION REGISTER. The former is termed an "address stop", and the latter an "operation stop". Also, a stop may be desired when instruction having a minus sign enters the INSTRUCTION REGISTER. This possibility will be discussed more fully in Chapter 11.

On "External" operation, the computer may be set to stop at the end of every data and instruction cycle, at the end of each word time, or for each timing pulse. These modes of operation would be used for demonstration and maintenance. Depression of the start key would cause the computer to operate until the next cycle, word, or timing pulse occurred, at which time the machine would again stop automatically. Note that drum operations (read and write) should not be allowed on word time or timing pulse operation, since to do so would present rather complicated problems in circuit design. Drum operations on the "Cycle" setting would be permissable. It should be pointed out that all error latches, as well as the Stop key will, of course, be effective for external as well as internal operation.

Logical Diagram number 16, being for purposes of explanation, does not specifically indicate this.

It is now possible to discuss the computer console itself (Figure 18). This sketch presents a suggested organization plan for the console, while conveying some idea of the overall appearance. Note that all 19 indicator lights and entry switches are shown for only one register. Of course, all 19 would be provided for all registers.

Note that the "Operation " switch (lower right side) has been discussed previously. The "Minus Instruction" switch, which governs the action taken for minus instructions, is covered in Chapter 11. The various operating buttons might be discussed. "Start" and "Stop" were mentioned previously; the "Reset" button would stop the machine, reset all error latches (but not the program flags) and set all registers to plus zero. The "Load" button would be used for program loading. Its operation is covered in the following chapter.

A word should be said concerning the console appearance. Some computer consoles have a forbidding appearance; others are attractive, even to the point of appearing, perhaps, to be less complicated than they actually are. The OSTIC's console, if built as suggested, would contain 152 indicator lights and 152 toggle switches for the registers alone, plus 19 additional toggles for each word of read-only memory. Therefore, some thought should be given to the layout and ultimate appearance of the console prior to the start of construction. A rather bright indicating lamp will be needed if the machine is to be demonstrated effectively before groups of more than a few students. Special miniature toggle switches are available, and at least one computer (Computer Control Company's "Digital Data Processor") uses them effectively to present a neat-appearing console. Finally,

CONSOLE

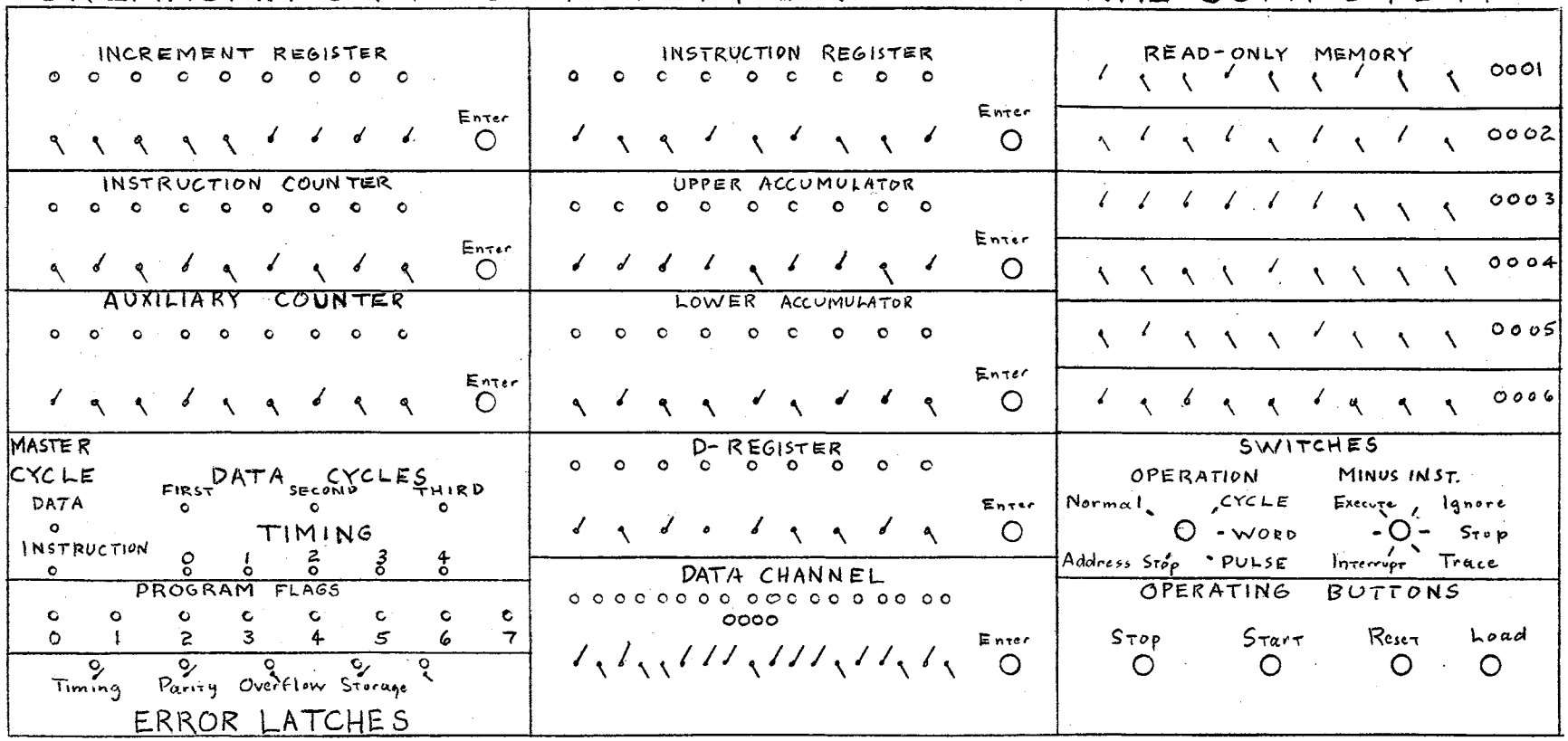


Figure 18. Console.

all important switches and buttons should be accessable to the seated operator, and all indicators should be easily visible.

CHAPTER XI

PROGRAMMING AND OPERATION

Every computer must possess a method for loading the initial program into memory. Most computers have some sort of built-in logic to provide for "bootstrapping", that is, to enable a program to literally read itself into the machine, with only the console switch settings being used. In other words, a method of program loading that does not presuppose anything in the machine memory is highly desirable.

It is suggested that bootstrap program loading be provided for by a "Load" button on the console (Figure 18). It is further suggested that, while the "Console Entry" toggle switches for the various registers not be made addressable in the program, the "Data Channel Entry" switches should be addressable with a data address of 0000_8. Also, the Read-Only memory (which is, of course, addressable) would have addresses of 0001_8, 0002_8, 0003_8, 0020_8 (for 16_{10} words). The intended action would be as follows: Depression of the console "Load" key would set the INCREMENT REGISTER to an increment of 1, set the INSTRUCTION REGISTER to 00 0000_8+, set the CYCLE SELECTION FLIP-FLOP to data cycle and the DATA CYCLE RING COUNTER to data first cycle. No other register would be changed. The desired loading routine would be set in the "Data Channel Entry" switches, and in as many of the "Read-Only Memory" switches as needed. Depression of the "Start" key would then cause 1), a "Copy-Jump" to 0000_8 (with the previous contents of the INSTRUCTION COUNTER placed in the D-REGISTER), 2), the first instruction

of the loading routine to be taken from the "Data Channel Entry" switches. Note that the INSTRUCTION COUNTER automatically takes the next instruction from location 0001_8, and continues to take the remainder of the loading routine from the Read-Only memory in a similar manner. The last instruction entered into the Read-Only memory would, of course, be a "Jump Unconditional" instruction.

The matter of minus instructions should also be discussed. As was mentioned previously, it is proposed that the sign of the instruction word should not enter into the code structure of the operation. Therefore, the sign is available for other use. The writer suggests, in keeping with the goal of flexibility, that a number of options be available to the programmer and operator in relation to minus instructions. Figure 18 shows the console, with the "Minus Instruction" switch at the lower right. The "normal" setting is that of "Execute". For this setting, minus instructions are "executed" in the same manner as plus instructions. The minus sign presents no influence.

The "Ignore" setting means simply that minus instructions are "ignored" by the computer. When the instruction read into the INSTRUCTION REGISTER is found to be minus, the INSTRUCTION COUNTER is simply incremented in normal fashion and the next instruction sought immediately. Thus no data cycle is taken for a minus instruction on the "Ignore" setting.

The "Stop" setting is quite simple. As soon as the minus instruction is read into the INSTRUCTION REGISTER, program execution is halted (see Chapter 10). Upon depression of console "Start" key, the minus instruction is executed in normal fashion, and the program then proceeds in regular order.

The "Trace"setting implies a more sophisticated operation. In computer terminology, a "trace" is a list of the contents of all

important machine registers at various points during the execution of a given program. The trace is used for program analysis and correction, or "debugging". It is proposed that the execution of a minus instruction for the "Trace" setting would cause the contents of all machine registers to be transferred to a special output buffer at the end of the operation, following which they could be punched or printed out for future use. When the program was corrected, the switch could be set to "Execute", and the minus instructions would not be traced. (9).

The last setting to be discussed is the "Interrupt" setting. In normal computer operation, communication with input/output units is completely under control of the program. If an input unit contains information which should be transmitted to the computer, the transmission cannot take place until the input unit is referenced by some sort of "transmit" or "read" instruction. The concept of "program interrupt" is simply that certain input units may be allowed to interrupt the normal sequence of computer operation and transmit data into the computer at times other than during normal input operations. After the input unit is through transmitting data, control is returned to the main program. (9, 10). The "Interrupt" setting is intended to allow these interrupt operations from certain specified input/output units following the execution of any minus instruction, regardless of whether the instruction is an I/O operation. It is anticipated that such a facility would be quite useful in certain areas of real-time control system investigation.

Appendix B gives the 48 operation codes that the writer feels would constitute a reasonable and useful command list for the OSTIC. Although a maximum of 64 commands could be incorporated, it is felt that a machine such as this should certainly have a provision for adding new operation codes at a later time.

Chapter 7 described the use of the logical operations "Ring Add", "OR Add", and "Complement-Load". These operations may be combined to provide all 16 of the logical operations listed in Table VII. For example, if B is in the LOWER, A is in drum location 1510, and the desired operation is $\bar{A}B$, then the following sequence of instructions would provide $\bar{A}B$ in the LOWER:

Operation	Data Address
Complement-Load	1510
OR Add	D-REGISTER

if $\overline{AB}$ is desired:

Reset Add Lower	LOWER
Add Upper	1510
Complement-Load	D-REGISTER
OR Add	UPPER.

Finally, if all 1's are desired in the LOWER, the sequence

Reset Add Lower	UPPER
Reset Add Lower	UPPER
Complement-Load	D-REGISTER

would require a maximum of 10 word times (340 microseconds) if the program was in the Read-Only memory.

It will be noted that a "divide " operation is not included in the command list. This omission was intentional; it was felt that the extra cost of providing built-in division was not justified on this machine. However, a divide routine may be easily programmed, using the AUXILIARY COUNTER to tally the shifts. If the dividend (which is plus) is in the ACCUMULATOR, and the divisor (also plus) in the D-REGISTER, the following routine would prove effective.

Inst. Address	Operation	Data Address
0001_8	Set Auxiliary Counter	01110_2
0002	Subtract Upper	D
0003	Jump Minus	0005_8
0004	Halt-Jump	(Error)
0005	Store D	1513
0006	Add Upper	D
0007	Increment Counter	
0010	Jump Auxiliary Zero	(Operation over)
0011	Shift Left	0001_8
0012	Subtract Upper	1513
0013	Jump Minus	0006
0014	Add Lower	0016
0015	Jump Unconditional	0012
0016	00_8	0001_8

This routine, if placed in the Read-Only memory, would perform division in almost the same manner as would a built-in divide instruction.

Freilich, (11), presents comparative data on various present-day digital computers available for process control applications. One method of comparison used is cost; another is the time required to obtain data from the "bulk memory" of the machine, which would correspond to the drum in the OSTIC, and from the "working memory", which would correspond to either a core-storage unit or to the Read-Only memory. The average access time for the bulk memory in the OSTIC is 256 word times, or 8.9 milliseconds. This is lower than 10 of the 24 computers listed, and higher than 14 (10 of the 14 had times of 8.3 milliseconds). The Read-Only memory in the OSTIC has an access time of one word time, or 34 microseconds. This is somewhat greater than any machine listed, although six machines

had access times of 20 microseconds or more. Twenty-one of the machines listed were binary computers (one was octal), 19 used single-address instructions, 14 had bulk memory capacities of 4096 words or less, seven were parallel machines, and 13 used words of 24 bits or less (one used 11 bits). It might also be mentioned that one machine cost $40,000, another cost $389,600, and the average price was in the neighborhood of $110,000. It may be concluded that the OSTIC, as presented here, compares favorably with commercially available machines for applications involving control systems research and experimentation.

Another interesting comparison might be made between the OSTIC and the IBM 650. The drum in the 650 rotates at 12,500 rpm and has 50 words per band. Therefore, the average access time is 25 word times, one word time being equal to approximately 96 microseconds. For various reasons, the fastest 650 addition speed is 125,000 additions per minute. For the OSTIC, the fastest possible addition would require three word times (one to acquire the instruction, one to acquire the operand, and one to add). Thus, the OSTIC can perform 512/3 additions per drum revolution, or approximately 589,000 additions per minute. The 650 worst case is 50 word times to find the operand, and 50 to find the instruction, or one addition every other drum revolution, resulting in 6250 additions per minute. The OSTIC worst case is 1725 additions per minute.

In closing this chapter, the writer strongly recommends that, as soon as the OSTIC is operating, an assembly program be devised. It is felt that only in this way can the full potential (especially with respect to operating speed) of the OSTIC be utilized.

CHAPTER XII

CONSTRUCTION AND MAINTENANCE

This chapter will present some of the writer's ideas and suggestions regarding construction and maintenance of the Oklahoma State Instructional Computer. It is the writer's philosophy that maintenance must be a primary consideration at all stages in the design and construction of any digital computer.

Back-panel wiring should be cabled, not point-to-point, and wire splices should never be allowed within a cable. A color-coding plan should be adopted early, at least for standard voltages and internal pulses, and should be strictly adhered to. Whenever any modifications, additions, or repairs (no matter how minor) are made to the computer, they should be immediately recorded in a logbook that is kept with the machine. Whenever an unlisted modification is encountered, it should be traced and recorded promptly.

It is further suggested that, insofar as possible, the computer be built in a modular fashion. Since most flip-flops will be used in groups of 18, perhaps each group could be mounted, with indicating neon lamps, on a small panel which would plug into the main chassis. In that way, whole "registers" could be moved around for trouble shooting. In this regard, a few extra units of all types should be kept on hand for such use.

Documentation(or the lack of it)has been the downfall of a number of computer projects such as this. Machine records must be kept up-to-date. In addition, it is suggested that each individual who has the responsibility for designing and building portions of this machine be required to submit, as part of his

project, a list of trouble shooting , testing, and repair procedures for the unit be constructed.

Marginal voltage tests are very valuable procedures in computer maintenance. As soon as power supplies are ready, work should be started toward toward perfecting such procedures.

Finally, a library of the various test and demonstration programs should be maintained.

CHAPTER XIII

SUMMARY

The philosophy and general system logic design was developed for a small magnetic-drum digital computer. The design was continually influenced by the proposed future applications of the machine. That is, the machine is primarily to be used in connection with the teaching of digital systems design, and for demonstrations of digital circuits. In this light, the machine system logic was to be straightforward, and the concepts of control and computation were to be easily understood. In addition, it was to be economical to maintain, and the command list was was to allow considerable flexibility in the application of the machine.

Perhaps the greatest contribution to a straightforward system concept is that of using a central data channel. This allows the control logic for the various operations to consist primarily of transferring data words between the various registers and the data channel lines.

The selection of binary operation and single-address instructions were "natural" selections for a machine of this type. The use of parallel information transmission contributes much toward simplifying the control logic. Parallel operation compromises the economy criteria to some extent; however, it was felt that the requirement of overall simplicity in machine organization was the more important factor in this particular decision.

The representation of numbers within the computer in sign-and-magnitude form is consistent with the use of the machine for

demonstration purposes to students in computer engineering courses, as is the facility for performing the various binary logical operations.

Provision for a very extensive list of branching operations through the use of the program flag concept will prove valuable for experimental applications in control systems work. Flexibility is also provided through the availability of a number of operating options for minus instructions. Operating speed may be increased considerably through the variable incrementation of the instruction counter.

In conclusion, it is felt that the computer design presented herein compares favorably with commercially available computers of similar memory capacity. The flexibility inherent in the design of the OSTIC computer should allow a wide variety of useful and worthwhile applications with the School of Electrical Engineering of the Oklahoma State University.

BIBLIOGRAPHY

(1). Ledley, Robert Steven. Digital Computer and Control System Engineering. New York: Mc Graw-Hill Book Company, 1960.

(2) Flores, Ivan. Computer Logic: The Functional Design of Digital Computers. Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1960.

(3). Phister, Montgomery, Jr. Logical Design of Digital Computers. New York: John Wiley and Sons, Inc., 1958.

(4). Richards, R. K. Arithmetic Operations in Digital Computers. Princeton: D. Van Nostrand Company, Inc., 1955.

(5). Buchholz, Werner. Planning a Computer System: Project Stretch. New York: McGraw-Hill Book Company, 1962.

(6). Beckman, F. S., F. P. Brooks, Jr., and W. J. Lawless, Jr. "Developments in the Logical Organization of Computer Arithmetic and Control Units". The Proceedings of the IRE, XLIX (January, 1961), 53-66.

(7). Kaiser, C. Joseph. " Elimination of Signal Race Conditions in Digital Design". Solid State Design, III (January, 1962), 29-34.

(8). Brooks, F. P., Jr., C. A. Blaauw, and W. Buchholz. "Processing Data in Bits and Pieces". IRE Transactions on Electronic Computers, VIII (June, 1959), 118-24.

(9). Brooks, F. P., Jr. "The Execute Operations - A Fourth Mode of Instruction Sequencing". Communications of the Association for Computing Machinery, III (March, 1960), 168-69.

(10). Turner, L. R., and J. H. Rawlings. "Realization of Randomly Timed Computer Input and Output by Means of an Interrupt Feature". IRE Transactions on Electronic Computer, VII (June, 1958), 141-49.

(11). Freilich, Arthur. "1962 Computer Control Survey". Electronic Industries, XXI (June, 1962), 15-18.

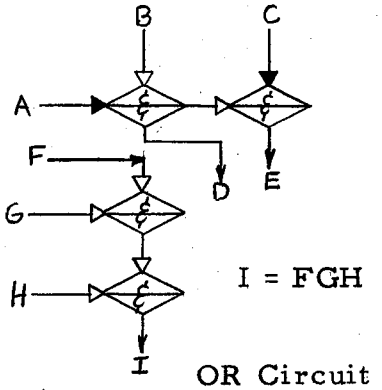
(12). Gorn, Saul, P. Z. Ingerman, and J. B. Crozier. "On the Construction of Micro-Flowcharts". Communications of the Association for Computing Machinery, II (October, 1959), 27-32.

APPENDIX A

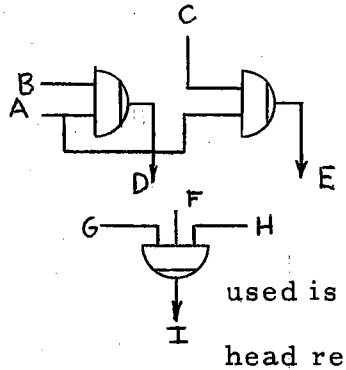
GRAPHICAL SYMBOLS

Listed below are the graphical symbols used in the Logical Diagrams accompanying this paper. An explanation accompanies each.

AND Circuit



I = FGH

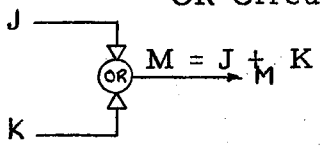


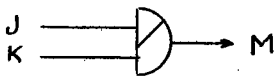
D = AB
E = AC

See Gorn (12), for further discussion. Note that A passes through the connector.

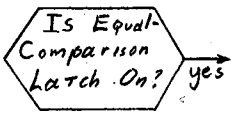
The arrowhead convention used is that the darkened arrowhead represents the signal or pulse of the longer duration.

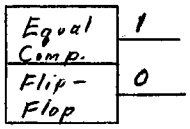
OR Circuit





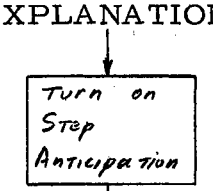
TEST. Is the signal present (yes or no) ?



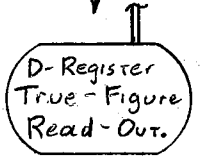


No operation of the logic dependent upon the presence of the signal is valid if the signal is not present, or not on.

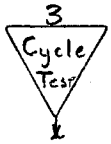
EXPLANATION



A rectangular symbol conveys information as to the action to be taken.



REGISTER: An oval symbol always represents an internal machine register, or its contents.



PULSE: A triangular symbol always represents a pulse. The timing of the pulse is shown at the top.

APPENDIX B

COMMAND LIST

The following operations constitute the command list suggested for incorporation into the OSTIC. The name of each operation is listed, together with the number of word-times required for execution[1], Operation Decoder output gates required, pertinent Logical Diagram numbers, and a brief description of the operation.[2]

---

[1] The abbreviations used in connection with the word-times required are as follows:

Op - Operand required. If located on the drum, from one to 512 word-times are required for acquisition, with an average of 256 required. If located in the read-only memory or an internal register, one word-time required for acquisition.

C - Complement cycle, using one additional word-time, will be required if negative answer is developed.

N - Number of 1's in word (excluding sign bit).

S - Number of positions desired on a shift.

[2] The abbreviations used in connection with the operation descriptions are as follows:

XXXX refers to the contents of a register or a drum location.

ABCD refers to the data address of the instruction in octal form. A refers to the first octal digit, B to the second, C to the third, and D to the fourth.

Operation	Timing	Decoder Gates	Logics	Description
ARITHMETIC CODES:				
Add Upper	Op + 1 + C	Operand, Add, Upper	5, 6, 8	Add XXXX to Upper.
Add Lower	Op + 1 + C	Operand, Add, Lower	5, 6, 8	Add XXXX to Lower.
Reset Add Lower	Op + 1 + C	Operand, Add, Lower, Reset	5, 6, 8	Reset entire Accumulator to plus zero, then add XXXX to Lower.
Add Magnitude Lower	Op + 1 + C	Operand, Add, Lower, Magnitude	5, 6, 8	Add |XXXX| to Lower
Subtract Upper	Op + 1 + C	Operand, Subtract, Upper	5, 6, 8	Subtract XXXX from Upper
Subtract Lower	Op + 1 + C	Operand, Subtract, Lower	5, 6, 8	Subtract XXXX from Lower.
Reset Subtract Lower	Op + 1 + C	Operand, Subtract, Lower, Reset	5, 6, 8	Reset entire Accumulator to plus zero, then subtract XXXX from Lower.
Subtract Magnitude Lower	Op + 1 + C	Operand, Subtract Lower, Magnitude	5, 6, 8	Subtract |XXXX| from Lower.
Reset Multiply	Op + 1 + N + 18	Operand, Multiply	5, 6, 9	Reset Lower Accumulator to zero. Multiply XXXX by Upper.

Operation	Timing	Decoder Gates	Logics	Description
LOAD AND STORE CODES:				
Load D	Op	Operand, Load D	5, 6	Place XXXX in D.
Store D	Op	Store, D	7	Place D in XXXX (drum only)
Store Upper	Op	Store, Upper	7	Place Upper in D and in XXXX (drum only)
Store Lower	Op	Store, Lower	7	Place Lower in D and in XXXX (drum only)
LOGICAL CODES:				
Complement Load	Op	Operand, Complement	5, 6, 8	Place XXXX in D; take 1's complement of entire Accumulator (except sign)
Ring Add	Op + 1	Operand, Ring Add	5, 6, 11	
OR Add	Op + 1	Operand, OR		
SHIFT CODES:				
Shift Left	1 + S	Shift, Left, Shift	10	Shift entire Accumulator CD places left, 32_{10} maximum.
Shift Right Circular	1 + S	Shift Right Circular	10	Shift entire Accumulator CD places right (32_{10} maximum). Bits shifted off right end enter left end.

Operation	Timing	Decoder Gates	Logics	Description
BRANCH CODES:				
Jump Unconditional	1	Branch, Unconditioned Jump	12	Set Instruction Counter to ABCD.
Jump Minus	1	Branch, Minus Test	12	If Accumulator sign is minus, set Instruction Counter to ABCD.
Jump Zero	1	Branch, Accumulator Zero Test	12	If entire Accumulator is zero (plus or minus) set Instruction Counter to ABCD.
Jump on Overflow	1	Branch, Overflow Test	12	If Overflow Latch is on, turn to off and set Instruction Counter to ABCD.
Jump on Parity Error	1	Branch, Parity Test	12	If Parity Error latch is on, turn off and set Instruction Counter to ABCD.
Jump on Zero Auxiliary	1	Branch, Auxiliary Test	12	If Auxiliary Counter is zero, set Instruction Counter to ABCD.
Jump Flag 0 (or 1-7) 8 codes total	1	Branch Flag 0 (or 1-7) Test	12	If Program Flag 0 (or 1-7) is on, set Instruction Counter to ABCD.
Copy Jump	2	Branch, Copy-Jump	13	Copy contents of Instruction Counter into D, then set Instruction Counter to ABCD.

Operation	Timing	Decoder Gates	Logics	Description
PROGRAM FLAG CODES:				
Set Flag 0 (or 1-7)	1	Flag, Flag Set	14	Turn on Program Flag A if condition BCD is true.
Reset Flag 0 (or 1-7)	1	Flag, Flag Reset	14	Turn off Program Flag A
MISCELLANEOUS CODES:				
Set Increment	1	Set Increment	15	Set Increment Register to BCD.
Set Counter	1	Auxiliary Counter	15	Set Auxiliary Counter to CD 31_{10} maximum).
Increment Counter	1	Increment Auxiliary Counter	15	Add 1 to Auxiliary Counter.
Halt-Jump	1	Branch, Unconditioned Jump, Halt	12	Set Instruction Counter to ABCD, then stop machine.
INPUT/OUTPUT CODES:				
Eight codes, 70 through 77				

APPENDIX C

LIST OF PULSES

All machine pulses appearing in the Logical Diagrams are listed by name. Timings, diagram numbers, and a brief description of the function of each pulse accompany the listing.

Name	Timing	Logics	Purpose
Add-in pulse	1 - 3	8, 9, 10, 11, 15	Add into various registers.
Address pulses	1	5, 7	Address of words available.
Address Test	0 -1 - 2	5, 6, 7	Test band number.
Complement Test	4	8	Test for Complement cycle.
Cycle Test	3	5, 6, 7, 13, 14	Determine next cycle. Flag operations.
Cycle Turn-On	0	2, 3, 4	Change cycle settings.
Data First Cycle	0 - 4	All except numbers 1, 2, 16	First operation on data cycle.
Data Second Cycle	0 -4	4, 8, 9, 10, 11, 12, 13	Second operation on data cycle.
Data Third Cycle	0 - 4	4, 8	Third opera tion on data cycle.
Instruction Counter Increment	4	3	As name implies.
Miscellaneous Reset	4	9, 10	Reset various latches.
"1" Bit	0 -4	8, 9, 10, 15	Adding "1's".
Read-in	2 - 3 - 4	5, 6, 7	Data transmission.
Sign Add	1	8	Add-in sign bit
Sign Test	0	8	Set up end-around carry logic.
Word Pulses	3	5, 7	Word bits from read heads.
Word Time	0 - 4	3, 4	Provide cycle gates.
Write	3	6, 7, 10, 12, 13, 15	Write onto drum and into Registers.
"0" Bit	0 - 4	8, 9	Adding "0's"

APPENDIX D

LIST OF LATCHES

All machine latches appearing in the Logical Diagrams are listed by name. Time turned, time turned off, diagram numbers, and a brief description of the function of each latch accompany the listing.

Name	Time Turned On	Time Turned Off	Logics	Use
Complement	4	4	8	Prevent re-complement.
Cycle Selection Flip-Flop	0	0	2, 3, 4	Set machine in either data or instruction cycle mode.
Data Cycle End	any except 0	0	all except 1, 3, 16	Set up change to instruction cycle.
Data Cycle Ring Counter	0	0	4, 5, 6, 8, 10, 13	Set up data first, second, third cycle.
End-Around Carry	prior to 4	0	8	Indicate that an end-around carry has occurred.
Equal Comparison	1	0	5, 7	Set up drum read or write
Program Flags 0 - 7	3	3	14	Various purposes (available to programmer).
Instruction Cycle End	any except 0 and 4	0	2, 3, 5, 6	Set up change to data cycle
Non-Drum Read	0 - 4	4	6	Set up acquisition of non-drum words.
Operation	0 - 4	0 - 4	12, 16	Sets machine to stop or run.
Overflow	4	0	8, 12	Indicates Accumulator overflow on addition.
Parity Error	3		5, 12	Word read from drum has incorrect parity.
Shift-or-Add	0, 4	0, 4	9, 10	Operations involving shifting.
Signs Different	0	0	8	End-around carry and complement.

Name	Time Turned On	Time Turned Off	Logics	Use
Signs Negative	0	0	8	End-around carry and overflow.
Signs Positive	0	0	8	End-around carry and overflow.
Step Anticipation	any except 0	0	4	Set up data cycle change.
Storage Selection Error	0 - 1	0	6, 7, 12	Indicate incorrect data address.

VITA

John Lee Fike, Jr.

Candidate for the Degree of

Master of Science

Thesis: THE PHILOSOPHY AND SYSTEM ORGANIZATION OF A SMALL DIGITAL COMPUTER

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Muskogee, Oklahoma, December 1, 1937, the son of John L. and La Rue Huggans Fike.

Education: Attended elementary schools in Muskogee; graduated from Muskogee Central High School in May, 1955. Attended the Oklahoma State University and one semester of evening classes at the University of Tulsa; received the Bachelor of Science Degree from the Oklahoma State University in May, 1961, with a major in Electrical Engineering; completed requirements for the Master of Science degree in August, 1962.

Professional Experience: Employed by International Business Machines Corporation from March, 1957, until August, 1958, as a Customer Engineer; employed during the summer of 1959 by Shell Oil Company, as a Programmer; employed during the summer of 1960, by the Radio Corporation of America; employed by the School of Electrical Engineering of the Oklahoma State University from September, 1961, until May, 1962, as a Graduate Assistant; employed during all other periods since September, 1958, as a Student or Graduate Assistant by the Oklahoma State University Computing Center.

Professional Organizations:

Eta Kappa Nu; Sigma Tau; Oklahoma Society of Professional Engineers and National Society of Professional Engineers (Junior Member); The Institute of Radio Engineers and the Association for Computing Machinery (Student Member)