THE SYMBOLIC ADDRESS INPUT CONVERTER

Thesis for the Degree of M. S.

MICHIGAN STATE UNIVERSITY

Bruce Herbert Barnes

1957

THE SYMBOLIC ADDRESS INPUT CONVERTER

By

BRUCE HERBERT BARNES

AN ABSTRACT

Submitted to the College of Science and Arts
Michigan State University of Agriculture and
Applied Science in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

Department of Mathematics

1957

Approved: ______________________

ABSTRACT

The SAIC is a symbolic address conversion input routine. It has the following modes of operation:

1. The SAIC will interpret the pseudo coded routine and store it in memory.

2. The pseudo coded routine will be interpreted and punched on tape.

The first mode of operation is limited to routines which will, together with the SAIC, fit into the 1024 words of electrostatic storage. The second option is used when the routine will not fit into electrostatic storage. It is also useful if the routine is to be used repeatedly.

The SAIC accepts instructions in the following format:

operation code	address	connective
O_1 O_2	A N	C

The operation code is the standard Mistic sexadecimal operation code. A is a regional designator. The alphabetic characters may be used as regional designators. If the address of the instruction does not pertain to an electrostatic memory location or if a pure address is desired, A may be left vacant. N is a decimal number composed of one or more digits. If N is a negative number, it is preceded by a minus sign. When a regional designator is used, the value of N is added to the assigned value of A.

The connective C is either a carriage return (CR) or a period. The carriage return is used if the next piece of information is of the same kind as the last accepted. The period is used when the next piece of information on tape is a directive.

The SAIC uses alphabetical operation codes as directives. These directives are orders punched on tape which direct the input operation. The directives are as follows:

1. Dictionary. The dictionary is a set of memory locations in which the values of the regional designators are stored.

2. Store Orders. This directive will store the succeeding information on tape to be stored as order pairs beginning at the specified memory location N.

3. Store Fractions. This directive causes information following it on tape to be stored as fractions beginning at the specified address N.

4. Store Integers. This directive operates in the same way as the store fraction directive except that integers are stored.

5. Jump. The jump directive informs the SAIC that the next piece of information on tape is an instruction and causes control to be transferred to that instruction. This directive is usually used to transfer control from the SAIC to the routine just assembled in memory.

6. Jump to DOI. This directive transfers control to the DOI. It is ordinarily used to input DOI subroutines.

7. Punch. When one desires the routine to be converted and punched on an output tape in machine language, the punch directive must appear as the first item on tape.

8. Store. This directive is used to cause information to be stored rather than converted and punched. The store instruction is needed only if the routine has already been set to the punch mode of operation.

ACKNOWLEDGMENTS

The author wishes to express his gratitude to Professor Gerard P. Weeg for his understanding and guidance throughout the writing of this paper.

THE SYMBOLIC ADDRESS INPUT CONVERTER

By

BRUCE HERBERT BARNES

A THESIS

Submitted to the College of Science and Arts
Michigan State University of Agriculture and
Applied Science in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

Department of Mathematics

1957

12/[illegible]/57
g2818

I. INTRODUCTION

Michigan State University has recently completed the construction of the Mistic, a stored program digital computer, patterned after the Illiac of the University of Illinois. It operates on binary numbers, directed by orders stored two to a word in the memory of the computer. The memory is capable of holding 1024 forty binary digit (bit) words, roughly the equivalent of 12 decimal digits. An order pair has the following format: (1)

8 bits	2 bits	10 bits	8 bits	2 bits	10 bits
operation code	waste	address	operation code	waste	address

The eight bits used for the instruction allow for the use of a two sexadecimal digit operation code. Since there are 1024 memory locations, ten binary bits are needed to assign a numerical address to each. The remaining two bits are not used. Thus, each instruction has an eight bit operation code telling the computer which operation to perform and a ten bit address telling on which number to perform the operation. Some operations do not need an address on which to operate. In these cases the address is used to amplify the operation code, giving information such as how many times the operation is to be performed.

Instructions may be input into the machine in the form described above. However, this method possesses several disadvantages. It is, therefore, advantageous to use an input routine which will accept instructions and numbers in a form more amenable to programming and which will convert them into machine form. There have been a number of such routines written, the most notable being the Decimal Order Input, which is described later in this paper. Although the Decimal Order Input has reduced programming difficulties, it possesses several difficulties. The present paper presents an input routine which attempts to remove certain of these difficulties. It is called the Symbolic Address Input Converter (SAIC).

II. THE SYMBOLIC ADDRESS INPUT CONVERTER

A. Description of the SAIC

The SAIC is a symbolic address conversion input routine. It has the following modes of operation:

1. The SAIC will interpret the pseudo coded routine and store it in memory.

2. The pseudo coded routine will be interpreted and punched on tape.

A combination of the two modes of operation may be employed if desired.

The first mode of operation is limited to routines which will, together with the SAIC, fit into the 1024 words of electrostatic storage. The second option is used when the routine will not fit into electrostatic storage. It is also useful if the routine, though short enough to fit in memory, is to be used repeatedly.

The SAIC accepts instructions in the following format:

operation code	address	connective
$O_1 \; O_2$	A N	C

The operation code is the standard Mistic sexadecimal operation code. A is a regional designator. The alphabetic characters A through Z may be used as regional designators. If the address of the instruction does not pertain to an electrostatic memory location or if a pure address is desired, A may be left vacant. N is a decimal number composed of one, two, three, or more decimal digits. If N is a negative number, it is preceded by a minus sign. When a regional designator is used, the value of N is added to the assigned value of A.

The connective C is either a carriage return (CR) or a period. The carriage return informs the SAIC that the instruction is complete and that the next piece of information on tape is of the same kind as the last word accepted. The carriage return is not printed on the typed copy but causes the carriage of the page printer to return to the left margin. Thus, as the tape is run through the page printer, the instructions are listed in a column. If a period is used, the next piece of information on tape is a directive. The period is always followed by a carriage return.

One of the outstanding features of the SAIC is its use of alphabetical characters as directives. These directives are orders punched on tape which direct the input operation. The directives are as follows:

1. Dictionary. The dictionary is a set of memory locations in which the values of the regional designators are stored. This directive is used by punching DY on tape followed by a CR. After this is the list of regional designators, each one followed by its numerical value and a CR. The last value is followed by a period and a CR. To give an illustration, if the letters A, B, P, Q, and T are to be used as regional designators and assigned the following values:

A = 200
B = 145
P = 5
Q = 26
T = 37,

then the input tape would appear as

DYCR
A200CR
B145CR
P5CR
Q26CR
T37CR

2. Store Orders. The symbol for this directive is SO. It appears on tape as SONCR. In this case N always contains a regional designator. This directive will store the succeeding information on tape to be stored as order pairs beginning at the specified memory location N. The first order after this directive is always a left hand order. There need not be an even number of instructions between successive store order directives. For example, if it is desired to store a routine at A-7, the following would appear on tape:

SOA-7CR
26A3CR
L4B-2CR
S50CR

3. Store Fractions. This directive appears on tape as SFNCR followed by the list of fractions. This directive causes information following it on tape to be stored as fractions beginning at the specified address N. As in the preceding case, N always contains a regional designator.

4. Store Integers. This directive appears on tape as SINCR followed by the list of integers. This directive operates in the same fashion as the store fraction directive except that integers are stored. As before, the address contains a regional designator.

5. Jump. This will appear on tape as JP followed by the instruction to be obeyed. The jump directive informs the SAIC that the next piece of information on tape is an instruction and causes control to be transferred

to that instruction. This directive is usually used to transfer control from the SAIC to the routine just assembled in memory. This directive appears on tape, for example, as JP24NCR.

6. Jump to DOI.(2) The symbol for this directive is JD followed by a CR. This directive transfers control to the DOI. It is ordinarily used to input DOI subroutines. This directive is needed in order to take full advantage of the computer library.

7. Punch. When one desires the routine to be converted and punched on an output tape in machine language, the punch directive must appear as the first item on tape. The symbol for punch is PUCR.

8. Store. This directive is used to cause information to be stored rather than converted and punched. The store instruction is needed only if the routine has already been set to the punch mode of operation. The symbol for store is STCR.

B. Restrictions in Using the SAIC

The routine is punched on tape in such a way that a printed teletype copy will appear in the same form that the program appears on the programmer's written copy. For this reason, all letter shifts and number shifts must be punched. If they are neglected, it is an error; and the SAIC will either make an improper entry, or in some cases, will stop and punch ER on tape, indicating an error. This characteristic of the SAIC is a help to the programmer in checking the routine. By comparing the printed copy with the written copy, one can note many errors such as missed letters or number shifts.

Some of the places where a letter or number shift must be punched are before and after the regional designators and before all directives. The operation code is always punched in number shift; therefore, K and S will appear on tape as + and - respectively.

If the programmer makes a mistake in punching the tape, he will retype a space (all five holes punched) and continue with the routine.

There are three types of addresses: pure addresses, symbolic addresses with N positive, and symbolic addresses with N negative. A few reasonable restrictions must be placed on the addresses. Since all addresses are positive, there is no need for negative addresses. All pure addresses, therefore, must be positive and lie between zero and 1023. If N is negative, its numerical value must not exceed the value of the regional designator. If N is positive, N plus the value of the regional designator should normally not exceed 1023. For example, if A has the assigned value 200, then N should have the range $-200 \leq N \leq 823$. If the address is greater than 1023, it will be interpreted modulo 1024. In all cases N must be at least one digit. If the pure address zero is desired, 0 is written. If the address is the value of the regional designator, A0 is used.

As was previously noted, two types of constants may be input, integers and fractions. Since the Mistic is a fractional machine, integers will not fit in memory without scaling. The scaling for integers is 2^{-39}. Any integer n will be input as $n \cdot 2^{-39}$. All integers will be followed by a CR except the last, which is followed by a period and a CR. If the integer is negative, it will be preceded by a minus sign; if positive, no sign is used.

When writing fractions for machine input, no decimal point is used. Terminal zeros may be included but are not needed. However, all zeros preceding the first non-zero digit must be written. As with integers, all fractions are terminated by a CR or a period and CR. When inputting negative fractions, the same notation is used as with integers, a minus sign preceding the fraction.

C. Advantages of the SAIC

The Computer Laboratory at Michigan State University, like many university installations but unlike most industrial installations, does not employ a staff of professional programmers. Each computer user will be responsible for his own programs. It was primarily for such individuals, not the professional programmer, that the SAIC was written. Since the SAIC is easier to use than any previous input routine, the time required to learn to program should be lessened. There should also be a savings in time after one has learned to program. The time saved in coding will not be so appreciable as the time that may be saved in checking (finding and correcting errors). There are two reasons for this: (1) The format is natural and easy to use. (2) The printed copy is easier to read and to examine for errors.

The most generally used input routine until now has been the DOI. The DOI, however, has four major defects:

1. Its program form is not natural.

2. Modification of address is difficult and in many cases impossible.

3. The input of constants is so difficult that the use of a constant input routine is almost essential.

4. Additions to a program are not easy to make.

The DOI instructions have the following format: X_1X_2 N C. The X_1X_2 is the standard Mistic code. The N is the address and C is a connective. The connectives used for address modification are F, L, and S. F is used if the address is pure. When L is used, the address where the first instruction is stored is added to the value of N. The symbol S is followed by a sexadecimal number m, m = 3, 4,..., L. When this symbol is used, N is added to the value stored at memory location m. This method will accomplish most address modifications desired. However, it is difficult to use, especially for an inexperienced programmer.

The S symbol can be used almost exactly as the Dictionary. The format is one of the major objections. It is difficult to learn how to obtain the maximum flexibility from the DOI. Not only training but considerable practice and experience are needed to learn the tricks and methods used for address modification with the DOI. This is not the case with the SAIC. The use of the symbolic address makes address modification straightforward and simple to accomplish.

Aside from the unnatural form of the DOI, there are many address modifications that cannot be accomplished with the DOI. Often after one has nearly completed his routine, he notices that it is necessary to add instructions to the beginning. Since N is added to the address of the first instruction, all addresses using the L connective must be changed accordingly. This not only involves more time in completing the finished program, but it is a common source of errors.

It is frequently desirable to store, in memory, an instruction that is not an operating part of the routine, but which is used to reset an

instruction that has been changed in the performance of the routine. This is usually done with an S_m connective. In this case the L connective may not be used. The reason for this is that the address of the first instruction stored in memory will not be the same. With the use of the dictionary, this difficulty has been eliminated, since the symbolic addresses are dependent only on the dictionary which is constant throughout the entire routine.

When using the DOI, constants appear on tape as order pairs. To input a positive integer N, $0 < N \leq 2^{39}-1$, the left hand order would appear as 00 F and the right hand order as 00 NF. The address in an instruction in the DOI format is always positive; therefore, it is necessary to devise a method by which negative integers N may be input into the computer. This is accomplished by inputting an order pair which is the sum of LL4095FLL4096F and 00F00NF. This will input the two's complement of the integer, the machine representation of a negative number.

This, obviously, is not as easy as writing the integer and following it with a carriage return, which is the form used by the SAIC. With the SAIC, all integers are input in their standard form. It is not necessary to use an operation code or to find the complement of a number to input its negative, nor to write an integer as an order pair.

The input of fractions is further complicated by a problem of scaling. Fractions are input into the computer by use of the J connective. When fraction is followed by J, the address is multiplied by $2^{39} \times 10^{-12}$. In order that all fractions have the same scaling (2^0), it is imperative that all fractions have a scaling of 10^{12}. Thus, when inputting fractions with the DOI, one not only has to concern himself with operation codes and calculating the complement of numbers, but one is also bothered with a problem of scaling. These problems are not encountered when inputting fractions with the SAIC.

For most computer installations, the Decimal Order Input might be a satisfactory input routine. Wherever such is the case, the SAIC will be an improvement. The Symbolic Address Input Converter, with its natural format, straightforward methods of address modification, and its simple form of constant input, should lend itself to ease of programming.

III. THE SAIC PROGRAM

The Symbolic Address Input Converter is composed of seven major subroutines: the directive determinater, the dictionary, store order routine, store integer and fraction routine, jump, punch, and store.

The directive determinater does primarily just what its name implies. It inputs the alphabetical directives, determines which directive it is, and transfers control to the proper subroutine.

When control is transferred to the directive determinater, it will first input one four bit character. This will be the P of punch, the Y of dictionary, the J of the two jump directives, or S of the store directives. The values of the four possible characters are 0, 6, 11, and 13 for P, Y, S, and J respectively. Thus, if the negative of the character is brought into the accumulator, P will be the only one whose negative is greater than or equal to zero. A sign check can be used here to determine P. The next step is to add six. This will change the accumulator to zero, if the character was Y. Again a sign test will determine Y.

The next step is to input one five bit character. All five hole inputs must be checked for a space which is used to cover an error in typing. At this point in the routine, D (a five hole character), P (0), T (5), I (8), O (9), or F (14) will be in the accumulator. When a five hole character is input, the fifth binary bit is placed in the sign position; therefore, we may use a sign test to determine D. If the character is not D, its negative is sent to the accumulator. Again a sign test will determine P. The next step is to add six to the accumulator. This will make T positive. If the

character is not T, then -2, -3, or -8 will be in the accumulator. Now two may be added to check for I and then 1 may be added to determine O. If the accumulator is still negative, then the character input was F. When a character is determined, control is transferred to the proper subroutine.

The basic subroutine is the dictionary. It is in this that the symbolic addresses are assembled. The dictionary subroutine also has the function of storing the value of the regional designator.

Before the symbolic addresses can be assembled, the value of the regional designator must be stored in memory. This is accomplished by inputting the regional designator and checking to determine if it is a five hole character. The reason for determining whether or not the character has a fifth hole is that the regional designators are stored two to a memory location in order to save space. The addresses will be assembled in the right hand address location. If the character has a fifth hole, preparation will have to be made to shift the addresses to the left hand address position. The last four bits of the character are added to the address of the instruction which sends to memory the value of the regional designator.

Many times during the routine, letter shifts or number shifts will be needed on the tape so that the printed copy will appear in the correct form. Some of these are of special significance to the SAIC; others will be checked to see if they are present. If they are not present, ER is punched on tape and the machine will stop. Such a check is performed here.

At this point the routine is prepared to assemble the address. As each character is input, it is stored in memory. The address storage, where the partial addresses are stored, is multiplied by ten and the new digit is added. The new partial address is then stored in the address storage. The partial address is multiplied by ten by shifting left two places. This multiplies it by four. Adding the partial address to this will put five times the partial address in the accumulator. A left shift of one bit will multiply it by two, leaving ten times the partial address in the accumulator. This process is continued until a carriage return or period is input. When a carriage return is input, the memory location where the regional designator is to be stored is added to the address. This memory location has previously been set to zero. The address is now sent to tis proper place in memory. If the address is the value of a five bit character, it is shifted to the left hand address position before the memory location is added. The dictionary routine will now prepare to input the next regional designator. When a period is encountered, the address is stored and control is transferred to the directive determinater.

The symbolic addresses are assembled in almost the same way as the regional values are stored. However, the memory locations where the regional designators are stored are no longer zero, but contain the values of the regional designators. When the two are added, the correct address will be in the accumulator. Instead of sending the address back to memory, control is transferred to the subroutine which requires this address.

Pure addresses are assembled in the same way except that control is transferred before the regional designators are added. Negative addresses are handled by sending the negative of the address to the accumulator before the regional designators are added.

The store order routine has the function of storing the order pair in memory. Before it can do this, it must pick up the address where the instructions are to be stored and send it to the store instruction. To do this, it first prepares the dictionary to jump back to the store order routine. Then control is transferred to the dictionary, the symbolic address is assembled, and control is transferred back to the store order routine. The store order routine now transfers this address to the store instruction. If the regional designator of the symbolic address is a five hole character, the address is shifted to the right hand address before it is transferred to the store instruction.

The order pairs are assembled in the order storage before they are transferred into their proper place in memory. The first step in assembling the orders is to bring the order storage to the accumulator. If the instruction to be input is a left hand instruction, the order storage is zero. Otherwise, the order storage contains the left hand instructions in the right hand position.

The operation code is now input, using a four bit input and the accumulator is shifted left twelve bits. This will put the operation code in its proper position; and if a left hand instruction was in the order storage, it will be shifted to the left hand position.

The next step is to assemble the address and transfer it to the order storage. This is done in the same way that the address was sent to the store instruction.

The address of the instructions is not necessarily symbolic. The pure addresses are distinguished from the symbolic addresses by the letter shift. Since the operation code is in number shift, a letter shift must precede all symbolic addresses. If the letter shift is not present, the address is assumed to be pure. With the pure address the dictionary is set to transfer control back to the store order routine before the regional designator is added.

At this point it must be determined whether the instruction is a left or right hand instruction. A left-right switch is used for this purpose. The left-right switch is originally set to the left. It is changed to right and back to left as the instructions are assembled.

If a left hand instruction has just been input, it is left in the right hand side of the order storage. The left-right switch is set to right and control is transferred to input the next instruction. After a right hand instruction is assembled, the order storage containing the order pair is sent to its proper place in memory. One is then to be added to the address of the store instruction. The left-right switch is set to the left and control is transferred to input the next instruction.

When a period is encountered, indicating that the last instruction has been input, it must be determined whether the last instruction was a left or right hand instruction. This is done by the use of the left-right switch. If the left-right switch is set for left, the last instruction of a right hand instruction and all the order pairs were entered into memory. This is not the case if the last instruction was a left hand instruction. In this case the left hand instruction must be shifted to the left hand position in the order pair and transferred to its proper place in memory.

The purpose of the integer-fraction routine is to assemble integers and fractions and store them in their proper place in memory. The address

where the first constant is to be stored is assembled in a similar way as is the store order routine and is sent to the store instruction in this routine.

The integers are input in the following way: The first character is input and checked to see if it is a minus sign. If it is a minus sign, the routine is set to store its negative. Otherwise, it is stored in integer storage. The next character is then brought in and checked to see if it is a carriage return or period. If it is not a carriage return or period, it is stored and the integer storage is multiplied by ten and the new character is added. This procedure is continued until a carriage return or period is input.

When a carriage return is input, the integer storage is sent to the accumulator. The integer is then sent to memory. One is added to the store instruction and control is transferred back to the beginning to input the next constant. If a period follows the integer, the same procedure is followed as above except that control is transferred to the directive determinater.

The assembly of fractions follows closely that of integers. The major difference is that when the constant or its negative is sent to the accumulator, it is divided by a power of 10×2^{-39}. The power of ten is equal to the number of decimal digits in the constant. This changes it to the appropriate scaling (2^0). This is accomplished by adding one to the address of the divide instruction after each digit is input. The powers of ten are stored in ascending order beginning at the original address of the divide instruction. Thus, a fraction containing n digits is divided by $10^n \times 2^{-39}$, leaving a scaling of 2^0.

The jump routine is the shortest routine. Its purpose is to transfer control to an instruction. It does this by preparing the store order routine to transfer control to the right hand order of the order storage after the order is input and assembled. After making this change, control is transferred to the store order routine.

The purpose of the punch routine is to set the routines previously described to the punch mode of operation. When the SAIC is in punch mode, the routine is punched on tape in a format to be input by a special input routine. The following format is used for the input tape: N1 000 00 $X_1X_2X_3$. When this appears on tape, the succeeding information is stored in memory beginning at address $X_1X_2X_3$. This address is a sexadecimal number. The order format is O_1O_2 $X_1X_2X_3$. The O_1O_2 is the standard Mistic operation code and $X_1X_2X_3$ is the sexadecimal address. Constants appear on tape in their standard sexadecimal form. To transfer control to a specific instruction, N2 000 followed by the instruction is punched on tape.

In order to produce this tape, it is necessary to make five major changes in the SAIC. These are as follows:

1. When the address where information is to be stored is assembled, it is not sent to the store instruction. Instead, it is sent to a memory location containing N1 as the left hand operation code. The contents of this memory location are then punched on tape.

2. The store instruction in the store order routine is set to punch the order pair on tape.

3. The store instruction in the integer-fraction routine is set to punch the constants on tape.

4. The jump routine is changed in such a way that it will set the store order routine to send the instruction to a memory location having N2 as its left hand operation code. This order pair is then punched on tape.

5. The Decimal Order Input must be set to punch the information in the same format as previously described.

The store routine has the function of setting the SAIC back to the store mode of operation. It does this by replacing the instructions that were changed by the punch routine.

In order to obtain the desired results in as few memory locations as possible, it was necessary to require that each subroutine perform as many distinct functions as possible. For this reason, many instructions in the SAIC are changed several times. As one is reading through the SAIC and trying to understand exactly how it operates, one must keep a close record as to what instruction is presently stored in each memory location that is changed. If one will do this and keep continually in mind the rough outline of how the SAIC accomplishes its various functions, he should be able to follow the SAIC with a minimum amount of difficulty.

IV. APPENDIX

LOCATION	ORDER	NOTES
762	L5 936F	
	40 895F	
763	L5 946F	Stored order pairs
	40 895F	
764	L5 783F	
	40 899F	
765	L5 763F	
	40 879F	Reset store order routine
766	L5 784F	
	40 898F	
767	L5 762F	
	40 907F	
768	L5 782F	Reset integer-fraction routine
	40 924F	
769	L5 781F	
	40 925F	
770	L5 780F	
	40 827F	Reset jump routine
771	L5 779F	
	40 999F	

LOCATION	ORDER	NOTES
772	L5 778F	
	40 1018F	Reset D.O.I.
773	L5 777F	
	40 1019F	
774	L5 776F	
	40 1001F	
775	26 831F	
	26 831F	Jump to directive determinater
776	40 001F	
	22 000F	
777	42 1016F	
	00 039F	
778	42 1001F	
	L4 1016F	
779	S5 000F	
	26 1014F	
780	L5 931F	
	40 895F	
781	F5 924F	Stored order pairs
	40 924F	
782	S5 000F	
	40 000F	
783	F5 898F	
	40 898F	

LOCATION	ORDER	NOTES
784	22 888F	
	40 000F	
785	L5 786F	
	40 895F	
786	42 827F	
	22 798F	
787	L5 785F	
	40 907F	
788	L5 826F	Prepare to punch store address on
	40 879F	tape
789	L5 824F	Set store order routine to punch
	40 898F	
790	L5 823F	
	40 899F	
791	L5 822F	Set integer-fraction routine to
	40 924F	punch
792	L5 821F	
	40 925F	
793	L5 820F	Set jump routine to punch N2 000
	40 828F	$0_1 0_2\ X_1 X_2 X_3$
794	L5 818F	
	40 999F	
795	L5 817F	
	40 1018F	Set D.O.I. to punch

LOCATION	ORDER	NOTES
796	L5 815F	
	40 1019F	
797	L5 814F	
	40 1001F	
798	26 831F	Jump to directive determinater
	L5 827F	N1 000 $0_1 0_2$ $X_1 X_2 X_3$
799	82 40F	Punch on tape
	26 910F	Jump to integer-fraction routine
800	L5 827F	Add N1 000
	82 40F	Punch on tape
801	26 894F	Jump to store order routine
	50 813F	00•••011•••1
802	J0 950F	Instruction to Q
	S5 00F	Q to A
803	L4 812F	Add N2 $0_1 0_2 0_3$
	82 40F	Punch on tape
804	26 831F	Jump to directive determinater
	S5 000F	Address to A
805	40 002F	Address to 002
	L4 827F	Add N1 $0_1 0_2 0_3$ to A
806	82 40F	Punch on tape
	26 1020F	Jump to D.O.I.
807	40 001F	Instructions to 001
	50 813F	

LOCATION	ORDER	NOTES
808	J0 001F	Instruction to Q
	S5 000F	Instruction to A
809	82 40F	Punch on tape
	26 831F	Jump to directive determinater
810	L5 000F	Order pair to A
	82 40F	Punch on tape
811	26 1020F	
	26 1020F	Jump to D.O.I.
812	N2 000F	
	00 000F	
813	00 000F	
	11 4095F	
814	22 807F	
	22 001F	
815	36 810F	
	00 039F	
816	1L 3054F	
	J3 001F	
817	42 1001F	
	L0 816F	Stored order pairs
818	22 804F	
	22 804F	
819	42 950F	
	22 801F	

LOCATION	ORDER	NOTES
820	L5 819F	
	40 895F	
821	26 926F	
	26 926F	
822	S5 00F	
	82 40F	
823	26 900F	
	26 900F	
824	22 888F	
	82 40F	
825	40 827F	
	26 800F	Stored order pairs
826	L5 825F	
	40 895F	
827	N1 000F	
	00 000F	
828	L5 931F	
	40 895F	Set store order routine for jump
829	L5 945F	Set dictionary for store order
	40 867F	jump
830	22 888F	Jump to store order routine
	22 888F	
831	81 4F	Input one 4 hole character
	40 962F	Store character

LOCATION	ORDER	NOTES
832	L1 962F	Minus character to A
	36 787F	If P, jump to punch
833	L4 970F	Add 6
	32 843F	If positive, the character is Y
834	91 4F	Input one 5 hole character
	32 836F	Is it negative (D or space)?
835	L0 969F]	Is it a space?
	36 834]	
836	26 999F	If D, jump to D.O.I.
	40 962F	Store character
837	L1 962F	Minus character to A
	36 828F	If P, jump to jump routine
838	L4 970F	Add 6
	36 764F	Is it a T?
839	L4 968F	Add 2
	32 903F	Is it an I?
840	L4 968F	Add 2
	36 878F	Is it an O?
841	26 905F	Is it an F?
	L5 953F	Set 864 for negative address
842	40 864F	
	26 861F	Jump to input address
843	L5 967F	Set first period jump
	40 863F	

LOCATION	ORDER	NOTES
844	L5 966F	Set second period jump
	40 868F	
845	L5 965F	Set store order jump
	40 867F	
846	91 4F	Input one character
	36 929F	Is it a five hole character?
847	L0 969F	
	36 846F	Is it a space?
848	L5 963F	
	40 864F	Set for positive address
849	41 960F	
	L5 959F	
850	40 866F	Set dictionary storage
	42 867F	
851	L5 958F	Set five hole jump
	40 865F	
852	91 4F	Input one 5 hole character
	32 857F	Does it have a fifth hole?
853	L0 969F]	
	36 852F]	Is it a space?
854	L4 964F	Restore last four bits of character
	40 962F	Store character
855	L5 956F]	
	40 865F]	Set five hole jumps

LOCATION	ORDER	NOTES
856	L5 955F	
	40 894F	
857	L5 962F	Character to A
	L4 866F	
858	40 866F	Add character to dictionary instruction
	42 867F	Send address to store instruction
859	91 4F	Input one character (N.S.)
	36 929F	Does it have a fifth hole?
860	L0 969F	
	36 859F	Is it a space?
861	91 4F	Input one character
	32 868F	Does it have a fifth hole?
862	L0 969F	
	36 861F	Is it a space?
863	L4 954F	Is it a period? If not, it is a
	36 874F	carriage return
864	L5 960F	Address to A
	26 865F	For use with a non-symbolic address
865	22 866F	Five hole jump
	J0 957F	Set Q to zero
866	00 20F	Left shift 20 bits
	L4 983F	Add dictionary
867	22 867F	Store order jump
	40 983F	Store in dictionary

LOCATION	ORDER	NOTES
868	26 846F	Jump to input next regional designators
	L0 961F	Is it a minus sign?
869	32 841F	
	L4 961F	Restore character
870	40 962F	Store character
	L5 960F	Partial address to A
871	00 2F	
	L4 960F	Multiply by ten
872	00 1F	
	L4 962F	Add new character
873	40 960F	Store in address storage
	26 861F	Jump to input next character
874	L5 952F	
	40 868F	Set period jump
875	26 864F	Jump to address assembly
	L5 951F	
876	40 888F	Set period jump
	26 864F	Jump to address assembly
877	26 878F	
	26 878F	
878	L5 948F	
	40 888F	Reset period jump
879	L5 946F	
	40 895F	Set to send address to store instruction

LOCATION	ORDER	NOTES
880	L5 945F	
	40 867F	Set to store order jump
881	26 848F	Jump to assemble address
	40 962F	Store instruction
882	L1 896F	Left-right switch to A
	L4 944F	Add left switch
883	36 831F	If positive, jump to directive determinater
	L5 962F	Instruction to A
884	00 20F	Left shift 20 bits
	22 898F	Jump to store in memory
885	L5 943F	
	40 895F	Set to form instruction
886	L5 942F	
	40 863F	Set period jump
887	L5 944F	
	40 896F	Set left-right switch
888	41 950F	Set word storage to zero
	L5 947F	
889	40 894F	Set five hole jump
	L5 950F	Word storage to A
890	80 8F	Input eight bits (operation code)
	00 12F	Left shift twelve bits
891	40 950F	Store in word storage
	91 4F	Input one character

LOCATION	ORDER	NOTES
892	32 901F	Does it have a fifth hole?
	L0 969F]	Is it a space?
893	32 891F]	
	26 848F	Jump to address assembly
894	26 895F	Five hole jump
	10 20F	Right shift 20 bits
895	42 950F	Send address to word storage
	L5 950F	Word storage to A
896	22 896F	Left-right switch
	40 950F	Instruction to word storage
897	L5 941F]	Set left-right switch to right
	40 896F]	
898	22 888F	Jump to input next instruction
	40 (000)F	Store in memory
899	F5 898F	Add one to store address
	40 898F	
900	L5 944F]	Set left-right switch to left
	40 896F]	
901	26 888F	Jump to input next instruction
	40 960F	Store character
902	L5 940F]	
	40 864F]	Set for pure address
903	26 861F	Jump to address assembly
	L5 939F]	

LOCATION	ORDER	NOTES
904	40 923F	Set integer-fraction switch to integer
	26 906F	Set to input routine
905	L5 938F	
	40 923F	Set integer-fraction switch to fraction
906	L5 937F	
	40 927F	Set integer jump
907	L5 936F	Set to send address to store instruction
	40 895F	
908	26 880F	Jump to pick up store address
	L5 935F	
909	40 927F	Set period jump
	22 922F	Jump to store last integer
910	L5 934F	
	40 922F	Set for positive integer
911	81 4F	Input one character
	L0 961F	
912	32 927F	Is it a minus sign?
	L4 961F	Restore character
913	40 933F	Send to integer storage
	91 4F	
914	36 917F	Does it have a fifth hole?
	L0 969F	
915	32 913F	Is it a space?
	L4 954F	

LOCATION	ORDER	NOTES
916	32 908F	Is it a period or carriage return? If period, jump
	22 922F	Jump to store integer
917	40 962F	Store character
	L5 933F	Integer storage to A
918	J0 957F	Set Q to zero
	00 2F	
919	L4 933F	Multiply by ten
	00 1F	
920	44 962F	Add new character
	40 933F	Store in integer storage
921	F5 923F	
	40 923F	Add one to divide instruction
922	22 913F	Jump to input next character
	L5 933F	Integer to A
923	22 923F	Integer-fraction switch
	66 971F	Divide by a power of ten
924	S5 F	Fraction to A
	40 (000)F	Store in memory
925	F5 924F	
	40 924F	Add one to store instruction
926	L5 939F	
	42 923F	Reset divide instruction
927	26 910F	Jump to pick up next integer
	L5 932F	

LOCATION	ORDER	NOTES
928	40 922F	Set for negative integer
	22 911F	Jump to input next character
929	92 259F	Punch letter shift
	92 194F	Punch E
930	92 258F	Punch R
	OF F	Stop
931	42 950F	
	22 950F	
932	22 913F	
	L1 933F	
933	00 F	
	00 F	Integer storage
934	22 913F	
	L5 933F	
935	26 831F	
	L5 932F	
936	42 924F	
	26 910F	
937	26 910F	
	L5 932F	
938	22 923F	
	66 971F	
939	22 924F	
	66 971F	

LOCATION	ORDER	NOTES
940	L5 960F	
	26 895F	
941	22 898	
	40 950F	
942	L4 954F	
	32 875F	
943	42 950F	
	L5 950F	
944	22 896F	
	40 950F	
945	26 894F	
	40 983F	
946	42 898F	
	26 885F	
947	26 895F	
	10 20F	
948	41 950F	
	L5 947F	
949	82 20F	
	26 885F	
950	00 F	
	00 F	Word storage
951	41 950F	
	22 881F	

LOCATION	ORDER	NOTES
952	26 831F	
	L0 961F	
953	L1 960F	
	26 865F	
954	00 000F	
	00 005F	
955	22 894F	
	10 20F	
956	22 865F	
	J0 957F	
957	00 000F	
	00 000F	
958	22 866F	
	J0 957F	
959	00 20F	
	L4 983F	
960	00 F	
	00 F	Address storage
961	00 000F	
	00 011F	
962	00 F	
	00 F	Storage
963	L5 960F	
	26 865F	

LOCATION	ORDER	NOTES
964	00 000F	
	00 015F	
965	22 867F	
	40 983F	
966	26 846F	
	L0 961F	
967	L4 954F	
	36 874F	
968	00 000F	
	00 002F	
969	80 000F	
	00 015F	
970	00 000F	
	00 006F	
971 - 982		Powers of ten
983 - 998		Dictionary
999 - 1023		D.O.I.

V. REFERENCES

1. Gill, S., et al. Illiac Programming A Guide to the Preparation of Problems for Solution by the University of Illinois Digital Computer. Digital Computer Laboratory, Graduate College, University of Illinois, Urbana, Illinois, April 1, 1955.

2. Wheeler, D. J. University of Illinois Digital Computer Library Routine X1-18, Decimal Order Input. Urbana, Illinois, December 2, 1953. (Mimeographed.)