B. Campbell

PROJECT WHIRLWIND

TRAINING PROGRAM MATERIAL

DIGITAL COMPUTER LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Copy

Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: SHORT GUIDE TO CODING AND WHIRLWIND I OPERATION CODE

To: Group 61 and Applications Group

From: Philip R. Bagley

Date: September 2, 1952; Revised November 28, 1952

Abstract: This note contains an up-to-date version of the Short Guide to Coding and the Whirlwind I Operation Code.

FOREWORD

The following definitions have been adopted and consistently adhered to:

sequence: the numberical or other arrangement of a set of words stored or performed
instruction: a 16-digit binary word used to control the computer
operation: the 5 digits of an instruction which go to the operation control switch
command: a control pulse from the control matrix
process: an automatic manipulation initiated by a command
modulo: (abbreviated "mod") A number p modulo q is defined as the numerator of the fractional remainder when p is divided by q

Ex. 1: 60 mod 32. $\frac{60}{32} = 1 + \frac{28}{32}$, hence 60 mod 32 = 28

Ex. 2: 1.37 mod 1. $\frac{1.37}{1} = 1 + \frac{.37}{1}$, hence 1.37 mod 1 = .37

SECTION 1. SHORT GUIDE TO CODING

COMPUTER PROGRAMS

Program. A program is a sequence of actions by which a computer handles a problem. The process of determining the sequence of actions is known as programming.

Flow diagrams. A flow diagram is a series of statements of what the computer has to do at various stages in a program. Lines of flow indicate how the computer passes from one stage of the program to another.

Coded program. Programs and flow diagrams are somewhat idependent of computer characteristics, but instructions for a computer must be expressed in terms of a code. A set of instructions that will enable a computer to execute a program is called a coded program, and the process of preparing a coded program is known as coding. Individual coded instructions call for specific operations such as multiply, add, shift, etc.

COMPUTER COMPONENTS

Registers and words. A register has 16 digit positions, each able to store a one or a zero. A word is a set of 16 digits that may be stored in a register. A word can represent an instruction or a number.

Arithmetic element. Arithmetic operations take place in the arithmetic element, whose main components are three flip-flop registers, the A-Register, the Accumulator, and the B-Register (AR, AC, and BR). The 16 digit positions of AR starting from the left are denoted by AR 0, AR 1 ... AR 15. The digit positions of AC and BR are denoted in a similar fashion. Words enter AC through AR; BR is an extension of AC to the right.

Storage. The term "register" by itself refers to the main electrostatic storage, which consists of 1024 registers, each of which is identified by an address. These addresses are 11-digit binary numbers from 32 to 1055. The computer identifies a register by its address. Electrostatic storage may at some future date be expanded to 2048 registers, numbered 0 through 2047.

Input-output. All information entering or leaving the computer is temporarily stored in the input-output register (IOR). The computer regulates the flow of information between the internal storage and IOR, and also calls for any necessary manipulation of external units.

Control element. The control element controls the sequence of computer operations and their execution. The control element takes the instructions one at a time from storage, where the instructions are stored as individual words.

Inter-connections. The four main elements of the computer (storage, control, arithmetic, and input-output) are connected by a parallel communications system, known as the bus.

REPRESENTATION OF INSTRUCTIONS

Operation section. When a word is used to represent an instruction the first (left-hand) 5 digits, or operation section, specify a particular operation in accordance with the operation code.

Address section. The remaining 11 digits, or address section, are interpreted as a number with the binary point at the right-hand end. For the majority of instructions this number is the address of the register whose contents will be used in the operation. In the instructions slh, slr, srh, srr, clc, and clh, the number specifies the extent of a shift, and also an additional variant, such as roundoff; in rs, rd, and rc, the address section is not used.

Example. The instruction ca x has the effect of clearing AC (making all the digits zero) and then copying into AC the word that is in the register whose address is x. If q is a quantity in some register, the operation needed to copy q in AC is not ca q but ca x, where x is the address of the register that contains q.

REPRESENTATION OF NUMBERS

Single-word representations. When a word is used to represent a number the first digit indicates the sign and the remaining 15 are numerical digits. For a positive number the sign digit is zero, and the 15 numerical digits with a binary point at their left specify the magnitude of the number. The negative -y of a positive number y is represented by complementing all the digits, including the sign digit, that would represent y. (The complement is formed by replacing every zero by a one and every one by a zero.) In this way a word can represent any multiple of 2^{-15} from $-1 + 2^{-15}$ to $1 - 2^{-15}$. Neither +1 nor -1 can be represented by a single word. Zero has two representations, either 16 zeros or 16 ones, which are called +0 and -0 respectively.

Overflow--increase of range and precision. With single-word representation the range is limited to numbers between $-1 + 2^{-15}$ and $1 - 2^{-15}$. Programs must be so planned that arithmentic operations will not cause an overflow beyond this range. The range may be extended by using a scale factor, which must be a 30-digit number. Overflow will stop the computer in an arithmentic check alarm except where special provision has been made to accommodate the overflow (see sa operation).

COMPUTER PROCEDURE

Sequence of operations. After the execution of an instruction the program counter in the control element holds the address of the register from which the next instruction is to be taken. Control calls for this instruction and carries out the specified operation. If the operation is not sp or cp the address in the program counter then increases by one so that the next instruction is taken from the next consecutive register. The sp and cp instructions permit a change in this sequential procedure.

Tranfers. A transfer of a digit from one digit position to another affects only the latter digit position, whose previous content is lost.

Zero. All sums and differences resulting in zero are represented as negative zero (1.111 111 111 111 111) except in the two cases: (+0) + (+0) and (+0) - (-0). The sign of a zero resulting from multiplication, division, or shifting is in accordance with the usual sign convention.

Manipulation of instructions. Words representing instructions may be handled in the arithmetic element as numbers.

Procedure in the arithmetic element. The execution of an addition includes the process of adding in carries; this process treats all 16 digits as if they were numerical digits, a carry from AC 0 being added into AC 15. (This compensation is necessary because of the method of representing negative numbers.) A subtraction is executed by adding the complement. Multiplication, division, scale factoring, shifting (by not cycling) and roundoff are all executed with positive numbers, complementing being performed before and after the process when necessary. For roundoff the digit in BR 0 is effectively added into AC 15.

BR. The final binary value of digits which pass from AC to BR or vice versa as a result of operations which multiply, divide, scale factor, or shift (but not cycle) is determined by the sign digit assigned to AC at the end of the operation. If the sign is negative the digits were in effect complemented as they crossed the digit-boundary between AC and BR. If the sign is positive no complementing occurred. The net effect is that a number in BR is treated as a positive magnitude, the sign of the number being indicated by the sign digit of AC. Therefore, if a number is to be recalled from BR for further operations, it is necessary to compensate for any change in the sign digit of AC which may have occurred after the number was placed in BR.. No complementing of any sort occurs in the execution of the cycle instructions, during which AC and BR may be considered a closed ring of 32 digit positions.

NOTATION FOR CODING

Addresses. A coded program requires certain registers to be used for specified purposes. The addresses of these registers must be chosen before the program can be run on the computer, but for study purposes this final choice is unnecessary, and the addresses can be indicated by a system of symbols or index numbers.

Writing a coded program. Reigsters from which control obtains instructions may be called action registers, and should be listed separately from registers containing other information, which may be called data registers. A coded program is written out in two columns: the first contains the index number of each action or data registers, and the second column indicates the word that is initially stored in that register. In many cases part or all of a word may be immaterial because the contents of the register in question will be changed during the course of the program. This state of affairs is indicated by two dashes, for example, ca--.

Conventional notation. In order to make a program more readily understandable to others and more easily remembered by the author himself, it is desirable to write short descriptions of the functions served by certain key instructions and groups of instructions. It is also desirable to indicate breaks and confluences in the "flow" of the program and to indicate instructions which are altered or otherwise abnormally used during the program. Some generally accepted symbols for this purpose are exemplified and described below:

	120	td 124	
start--->	121	ca 161	initial entry (i.e., start of program)
	122	td 132	

```
139-->       123  ca 181        re-entry point, showing origin of re-entry

             124  su(182)       address altered by program, initial value shown

             125  sr 16

             126  cp 128        conditional short break in consecutivity
                                            (note other form below)

             127  ad 140

             128  ad 133

             129  ts            address indicated by arrow (e.g. address = 130 in
                                 this case), used primarily at early stages of
                                 writing

             130 (ca217/cs217)  word altered by program, alternative values
                                                  shown

             131  sp 78         no break in consecutivity, despite sp operation,
                                 where a closed subroutine is called in

(122,167)    132  ts (-)        address altered by program, initial value
                                 immaterial, locations of altering instructions
                                 shown, alternative values not shown

             133 |ca 217        semi-pseudo instruction, serves both as instruction
                                 and number

             134  sp 95         short break in consecutivity, used especially
             135  p3             where a closed subroutine with program
             136  ex 114         parameters is called for

             137  cp 141        conditional break in consecutivity (note short
                                 form above)

             138  ts 114

             139  sp 123        break in consecutivity (note short form above)

             140 ||rs 0         pseudo-instruction, serves only as a number,
                                 not as instruction

137, 171->   141  ts 171        entry point, showing origins of entry
```

<u>The abbreviations RC, CR</u>. Abbreviations used in referring to the register that contains a certain word or the word in a certain register are

RC . . . = (Address of) register containing . . .

CR . . . = Contents of register (whose address is) . . .

The symbol si x. When an address forms part of an instruction it is represented by the last 11 digits of a word whose first 5 digits specify an operation. An address that is not part of an instruction is represented by the last 11 digits of a word whose first 5 digits are zero, which is equivalent to specifying the operation si. Thus the word for an unattached address x may be written si x. It may also be written as $+x$ or as $+x \times 2^{-15}$.

SECTION 2. WHIRLWIND I OPERATION CODE

NOTES ON THE OPERATION CODE

Introduction. The Whirlwind I Operation Code has been rewritten to bring it up-to-date, and to incorporate all notes, wherever possible, with the specific operations to which they apply, regardless of the undue repetition. Included under each operation are the average time of execution, the function, the contents (if altered) of AC, BR, AR, IOR, SAM, and register x after the operation, and possible alarms.

Abbreviations. The abbreviations used are the following:

AC	=	Accumulator	IOR	=	In-Out Register
AR	=	A-Register	ES	=	Electrostatic Storage
BR	=	B-Register	x	=	address of a storage register
SAM	=	Special-Add Memory	n	=	a positive integer

Contents of various registers. The contents of AC, BR, AR, IOR, SAM, and the register whose address is x are undisturbed unless the contrary is stated.

Alarms. Arithmetic check, divide error, and check register alarms due to programming cannot be caused except as specifically noted. M-1623, "Programming for In-Out Units" discusses in-out alarms.

Execution times. The times given are average times for the execution of single instructions which are stored in ES and which refer to addresses in ES. Further details are given in M-1623 and in E-~~4~~40.

In-Out Operations. Operations which call for the transmission of information to and from various units of terminal equipment termed "in-out operations," are described briefly in the Operation Code. Details of the actual application of these operations (si, bi, rd, bo, and rc) appear in M-1623.

Three-letter operations. The three-letter operations slh, slr, srh, srr, clc, and clh utilize part of the address section of the instruction (namely, digit 6) to specify the operation. If an address is inserted in one of these instructions by a ta or td operation, care must be taken to maintain the presence or absence of digit 6 in the address of the modified instruction. The two-letter designations, sl, sr, cl, are ambiguous and cannot be used in programs, but they may be used in general descriptions and comments.

Operation	Function	Number	Binary	Time	
si pqr	select in-out unit/stop	#0	00000	45 microsec	si

Stop any in-out unit that may be running. Select a particular in-out unit and start it operating in a specified mode, designated by the digits p q r; or, stop the computer. si 0 will stop the computer; si 1 will stop the computer only if the "Conditional Stop" switch is ON. An in-out alarm may subsequently occur if the computer is not ready to receive information transmitted to it from the selected in-out unit. A transfer check alarm may result from the use of an illegal si address. For further details, see M-1623, "Programming For In-Out Units."

rs x	reset	#1	00001	30 microsec	rs

Reset any flip-flop storage registers connected to the "reset on rs" circuit.

bi x	block transfer in	#2	00010	(see M-1623)	bi

AVAILABLE ABOUT JAN. 1953

Transfer a block of n words or characters from an in-out unit to ES, where register x is the initial address of the block in ES, and $\pm$ n times 2^{-15} is contained in AC. The computer is stopped while the transfer is taking place. After a block transfer, AC contains the address which is one greater than the ES address at which the last word was placed; AR contains the initial address of the block in ES. Fur further details, see M-1623, "Programming For In-Out Units."

rd x	read	#3	00011	30 microsec	rd

Transfer word from IOR to AC, then clear IOR. (Wait, if necessary, for information to arrive in IOR from an in-out unit.) Contents of AR is identical to contents of AC. The address section of the instruction has no significance. For further details, see M-1623.

bo x	block transfer out	#4	00100	(see M-1623)	bo

AVAILABLE ABOUT JAN. 1953

Transfer block of n words from ES to an in-out unit, where x is the initial address of the block in ES, and $\pm$ n times 2^{-15} is contained in AC. The computer is stopped while the transfer is taking place. After the block transfer, AC contains the address which is one greater than the ES address from which the last word was taken and stored; AR contains the initial address of the block in ES. For further details, see M-1623, "Programming For In-Out Units."

Operation	Function	Number	Binary	Time	
rc x	record	#5	00101	30 microsec	rc

Transfer contents of AC via IOR to an in-out unit. IOR is cleared only after an rc used as a display instruction. The address section of the instruction has no significance. For further details, see M-1623, "Programming For In-Out Units."

ts x	transfer to storage	#8	01000	86 microsec	ts

Transfer contents of AC to register x. The original contents of x is destroyed.

td x	transfer digits	#9	01001	86 microsec	td

Transfer last 11 digits of AC to last 11 digit positions of register x. The original contents of the last 11 digit positions of register x is destroyed.

ta x	transfer address	#10	01010	86 microsec	ta

Transfer last 11 digits of AR to last 11 digit positions of register x. The original contents of the last 11 digit positions of register x is destroyed. The ta operation normally follows an sp or sf operation.

ck x	check	#11	01011	48 microsec	ck

Compare contents of AC with contents of register x. If contents of AC is identical to contents of register x, proceed to next instruction; otherwise stop the computer and give a "check-register alarm." (+0 is not identical to -0).

ex x	exchange	#13	01101	86 microsec	ex

Exchange contents of AC with contents of register x (original contents of AC in register x, original contents of register x in AC and AR). ex 0 will clear AC without clearing BR.

cp x	conditional program	#14	01110	30 microsec	cp

If number in AC is negative, proceed as in sp. If number in AC is positive, proceed to next instruction, but clear the AR.

sp x	subprogram	#15	01111	30 microsec	sp

Take next instruction from register x. If the sp instruction was at address y, store y + 1 in last 11 digit positions of AR. All of the original contents of AR is lost.

ca x	clear and add	#16	10000	48 microsec	ca

Clear AC and BR, then obtain contents of SAM (+1, 0, or -1) times 2^{-15} and add contents of register x, storing result in AC. The contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

Operation	Function	Number	Binary	Time	
cs x	clear and subtract	#17	10001	48 microsec	cs

Clear AC and BR, then obtain contents of SAM (+1, 0, or -1) times 2^{-15} and subtract contents of register x, storing result in AC. The contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

ad x	add	#18	10010	48 microsec	ad

Add the contents of register x to contents of AC, storing result in AC. The contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

su x	subtract	#19	10011	48 microsec	su

Subtract contents of register x from contents of AC, storing result in AC. The contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

cm x	clear and add magnitude	#20	10100	48 microsec	cm

Clear AC and BR, then obtain contents of SAM (+1, 0, -1) times 2^{-15} and add magnitude of contents of register x, storing result in AC. The magnitude of the contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

sa x	special add	#21	10101	48 microsec	sa

Add contents of register x to contents of AC, storing result in AC and retaining in SAM any overflow (including sign) for use with next ca, cs, or cm instruction. Between sa and the next ca, cs, or cm, for which the sa is a preparation, the use of any instruction which clears SAM will result in the loss of the overflow, with no other effect on the normal function of the intervening operation. (In addition to ca, cs, and cm, the following operations clear SAM: ad, su, sa, ao, dm, mr, mh, dv, sl, sr, and sf.) If the overflow resulting from the sa is to be disregarded, care must be taken to destroy it before the next ca, cs, or cm instruction. The contents of register x appears in AR. SAM is cleared before, but not after, the addition is performed.

ao x	add one	#22	10110	86 microsec	ao

Add the number 1 times 2^{-15} to contents of register x, storing the result in AC and in register x. The original contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

Operation	Function	Number	Binary	Time	
dm x	difference magnitudes	#23	10111	48 microsec	dm

Subtract the magnitude of contents of register x from the magnitude of contents of AC, leaving result in AC. The magnitude of contents of register x appears in AR. SAM is cleared.

mr x	multiply and roundoff	#24	11000	65 microsec	mr

Multiply contents of AC by contents of register x. Roundoff result to 15 significant binary digits and store it in AC. Clear BR. The magnitude of contents of register x appears in AR. SAM is cleared.

mh x	multiply and hold	#25	11001	65 microsec.	mh

Multiply contents of AC by contents of register x. **Retain the** full product in AC and the first 15 digit positions of BR, the last digit position of BR being cleared. The magnitude of contents of register x appears in AR. SAM is cleared.

dv x	divide	#26	11010	120 microsec	dv

Divide contents of AC by contents of register x, leaving 16 binary digits of the quotient in BR and ± 0 in AC according to the sign of the quotient. The instruction slr 15 following the dv operation will roundoff the quotient to 15 binary digits and store it in AC. Let u and v be the numbers in AC and register x respectively when the instruction dv x is performed. If $|u| < |v|$, the correct quotient is obtained and no overflow can arise. If $|u| > |v|$, the quotient exceeds unity and a divide-error alarm will result. If $u = v \neq 0$, the dv instruction leaves 16 ones in BR; roundoff in a subsequent slr 15 will cause overflow and give an arithmetic check alarm. If $u = v = 0$, a zero quotient of the appropriate sign is obtained. The magnitude of contents of register x appears in AR. SAM is cleared.

slr n	shift left and roundoff	#27	11011	41 microsec	sl

Shift contents of AC and BR (except sign digit) to the left n places. The integer n is treated modulo 32; digits shifted left out of AC 1 are lost. (Shifting left n places is equivalent to multiplying by 2^n, with the result reduced modulo 1.) Roundoff the result to 15 binary digits and store it in AC. Clear BR. Negative numbers are complemented before and after the shift, hence ones appear in the digit places made vacant by the shift of negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction slr n must be a zero to distinguish slr n from slh n described below. The instruction slr 0 simply causes roundoff and clears BR. SAM is cleared. Roundoff may cause overflow, with a consequent arithmetic check alarm.

Operation	Function	Number	Binary	Time	
slh n	shift left and hold	#27	11011	41 microsec	slh

Shift contents of AC and BR (except sign digit) to the left n places. The integer n is treated modulo 32; digits shifted left out of AC 1 are lost. (Shifting left n places is equivalent to multiplying by 2^n, with the result reduced modulo 1.) Do not roundoff nor clear BR. Negative numbers are complemented before and after the shift, hence ones appear in the digit places made vacant by the shift of a negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction slh n must be a one to distinguish slh n from slr n described above. SAM is cleared.

srr n	shift right and roundoff	#28	11100	41 microsec	srr

Shift contents of AC and BR (except sign digit) to the right n places. The integer n is treated modulo 32; digits shifted right out of BR 15 are lost. (Shifting right n places is equivalent to multiplying by 2^{-n}.) Roundoff the result to 15 binary digits and store it in AC. Clear BR. Negative numbers are complemented before and after the shift, hence ones appear in the digit places made vacant by the shift of a negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction srr n must be a zero to distinguish srr n from srh n described below. The instruction srr 0 simply causes roundoff and clears BR. SAM is cleared. Roundoff (in a srr 0) may cause overflow, with a consequent arithemetic check alarm.

srh n	shift right and hold	#28	11100	41 microsec	srh

Shift contents of AC and BR (except sign digit) to the right n places. The integer n is treated modulo 32; digits shifted right out of BR 15 are lost. (Shifting right n places is equivalent to multiplying by 2^{-n}.) Do not roundoff the result nor clear BR. Negative numbers are complemented before and after the shift, hence ones appear in the digit places made vacant by the shift of a negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction srh n must be a one to distinguish srh n from srr n described above. SAM is cleared.

sf x	scale factor	#29	11101	97 microsec	sf

Multiply the contents of AC and BR by 2 sufficiently often to make the positive magnitude of the product equal to or greater than 1/2. Leave the final product in AC and BR. Store the number of multiplications in last 11 digit places of AR and register x, the first 5 digits being undisturbed. If all the digits in BR are zero and AC contains $\pm$ 0, the instruction sf x leaves AC and BR undisturbed and stores the number 33 times 2^{-15} in the last 11 digit positions of AR and register x. Negative numbers are complemented before and after the multiplication (by shifting), hence ones appear in the digit places made vacant by the shift. SAM is cleared.

Operation	Function	Number	Binary	Time	
clc n	cycle left and clear (BR)	#30	11110	41 microsec	clc

Shift the full contents of AC and BR (including sign digit) to the left n places. The integer n is treated modulo 32; digits shifted left out of AC 0 are carried around into BR 15 so that no digits are lost. Clear BR. No roundoff. With the clc operation there is no complementing of AC either before or after the shift; the actual numerical digits in AC and BR are cycled to the left. The digit finally shifted into the sign digit position determines whether the result is to be considered a positive or negative quantity. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction clc n must be a zero to distinguish clc n from clh n described below. The instruction clc 0 simply clears BR without affecting AC.

clh n	cycle left and hold	#30	11110	41 microsec	clh

Shift the full contents of AC and BR (including sign digit) to the left n places. The integer n is treated modulo 32; digits shifted left out of AC 0 are carried around into BR 15 so that no digits are lost. With the clh operation there is no complementing of AC either before or after the shift; the actual numerical digits in AC and BR are cycled to the left. The digit finally shifted into the sign digit position determines whether the result is to be considered a positive or negative quantity. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction clh n must be a one to distinguish clh n from clc n described above. The instruction clh 0 does nothing.

ALPHABETIC LIST OF OPERATIONS

This is an alphabetic list of Whirlwind operations, including operations and designations which have become obsolete since 1950.

Operation	Number	Remarks	Operation	Number	Remarks
ad	18		qm	0	now dm
ao	22		qp	31	obsolete
bi	2		qr	30	obsolete
bo	4		qs	12	obsolete
ca	16		rc	5	
ck	11		rd	3	
cl	2	now clc	ri	0	obsolete
cl*	2	now clh	rs	1	
clc	30		sa	21	
clh	30		sf	29	
cm	20		si	0	
cp	14		sl	27	now slr
cs	17		sl*	27	now slh
dm	23		slh	27	
dv	26		slr	27	
ex	13		sp	15	
mh	25		sr	28	now srr
mr	24		sr*	28	now srh
qd	7	obsolete	srh	28	
qe	13	now ex	srr	28	
qf	23	obsolete	su	19	
qh	6	obsolete	ta	10	
ql	2	now clc	td	9	
ql*	2	now clh	ts	8	

Signed P R Bagley
P.R. Bagley

Approved C. R. Wieser
C.R. Wieser

cc: C.W. Adams

PRB/sc/jmm/jmc/mrs

PROJECT WHIRLWIND

Report R-90-1

THE BINARY SYSTEM OF NUMBERS

Submitted to the
OFFICE OF NAVAL RESEARCH
Under Contract N5ori60
Project NR-048-097

Report by
Margaret Florencourt Mann
DIGITAL COMPUTER LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge 39, Massachusetts

Project DIC 6345

January 15, 1946
(revised: February 29, 1952)

Report R-90-1

ABSTRACT

The representation of decimal numbers in the binary system and the processes of binary arithmetic are explained.

THE BINARY SYSTEM OF NUMBERS

I. REPRESENTATION OF NUMBERS

The decimal system takes its name from the fact that it is based on ten digits (0, 1, ... 9) and all numbers are composed of those 10 digits. The binary system, analogously, takes its name from the fact that it is based on 2 digits, (0, 1), and all numbers in the binary system are made up of those 2 digits. The decimal system has a base of 10; the binary system has a base of 2.

Decimal System	Equivalence	Binary System	Equivalence
1	1×10^0	1	$1\times2^0 = 1$
2	2×10^0	10	$1\times2^1 + 0\times2^0 = 2$
3	3×10^0	11	$1\times2^1 + 1\times2^0 = 3$
4	4×10^0	100	$1\times2^2 + 0\times2^1 + 0\times2^0 = 4$
5	5×10^0	101	$1\times2^2 + 0\times2^1 + 1\times2^0 = 5$
6	6×10^0	110	$1\times2^2 + 1\times2^1 + 0\times2^0 = 6$
7	7×10^0	111	$1\times2^2 + 1\times2^1 + 1\times2^0 = 7$
8	8×10^0	1000	$1\times2^3 + 0\times2^2 + 0\times2^1 + 0\times2^0 = 8$
9	9×10^0	1001	$1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = 9$
10	$1\times10^1 + 0\times10^0$	1010	$1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = 10$
11	$1\times10^1 + 1\times10^0$	1011	$1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 11$
12	$1\times10^1 + 2\times10^0$	1100	$1\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0 = 12$
13	$1\times10^1 + 3\times10^0$	1101	$1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = 13$
14	$1\times10^1 + 4\times10^0$	1110	$1\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 = 14$
15	$1\times10^1 + 5\times10^0$	1111	$1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 = 15$
16	$1\times10^1 + 6\times10^0$	10000	$1\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1 + 0\times2^0 = 16$
17	$1\times10^1 + 7\times10^0$	10001	$1\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = 17$
20	$2\times10^1 + 0\times10^0$	10100	$1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0 = 20$

Decimal numbers, since they have a base of 10, may be broken up into powers of 10:

e.g. $305.798 = 3\times10^2 + 0\times10^1 + 5\times10^0 + 7\times10^{-1} + 9\times10^{-2} + 8\times10^{-3}$

In the same way, binary numbers, since they have a base of 2, may be broken up into powers of 2:

e.g. $101.011 = 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}$

It can be seen from this arrangement of the powers of the bases, that the decimal places (units, tens, hundreds, tenths, hundredths, thousandths, etc.) have a definite relation to the powers of the base 10 in the decimal system. They are numbered off consecutively from left to right, from $+\infty$ to $-\infty$, these numbers corresponding exactly with the powers of the base 10; the decimal point is placed between the units (0) place and the tenths (-1) place.

The binary places (units, twos, fours, eights, sixteens, halves, fourths, eighths, etc.) are also numbered exactly according to the powers of the base 2; the binary point is placed between the units (0) place and the halves (-1) place. Therefore, the place and point arrangement is the same in both decimal and binary systems.

e. g.

Decimal Place and Binary Place No. / No.	+ ∞ . . . 3, 2, 1, 0. -1, -2, -3, ... - ∞
305.798	3 0 5. 7 9 8
101.011	1 0 1. 0 1 1

II. CONVERSION OF DECIMAL NUMBERS TO BINARY NUMBERS

In order to convert a number from the decimal system to the binary system, the number must be changed from powers of 10 to powers of 2; therefore, the powers of 2 are taken out of the decimal number. They may be taken out as follows in a brute-force manner, or more simply as shown later in an algorism. Follow the work sheet at the end of the examples.

a. Integers.

Example 1 Given: 18

To Find: Binary Equivalent (First Method)

Method: Extraction of Powers of 2.

Take out of the decimal number, 18, the highest power of 2, = 16 = 2^4. A 1 is now placed in binary place No. 4, corresponding to the power of 2 found. Subtracting 16 from 18 leaves 2. The highest power of 2 in 2 is 2^1 = 2; therefore, put a 1 in binary place No. 1. Subtracting 2 from 2 leaves 0, so the conversion is completed. Since 4 and 1 were the only powers of 2 in the given number, 1's appear only in binary places 4 and 1. The coefficients of the non-appearing powers must have been zero, so zeros are entered under all the other binary place numbers.

Example 2 Given: 730

To Find: Binary Equivalent (First Method)

Method: Extraction of Powers of 2.

The highest power of 2 in 730 is 2^9 = 512, so a 1 goes in No. 9
730 - 512 = 218

The highest power of 2 in 218 is 2^7 = 128, " " " " " No. 7
218 - 128 = 90

The highest power of 2 in 90 is 2^6 = 64, " " " " " No. 6
90 - 64 = 26

The highest power of 2 in 26 is 2^4 = 16, " " " " " No. 4
26 - 16 = 10

The highest power of 2 in 10 is 2^3 = 8, " " " " " No. 3
10 - 8 = 2

The highest power of 2 in 2 is 2^1 = 2, " " " " " No. 1

No other powers of 2 appear so their coefficients must be zero.

Place No. / No.	10	9	8	7	6	5	4	3	2	1	0.	-1	-2	-3	-4	-5	-6	-7	-8
18	0	0	0	0	0	0	1	0	0	1	0.	0	0	0	0	0	0	0	0
730	0	1	0	1	1	0	1	1	0	1	0.	0	0	0	0	0	0	0	0
0.147	0	0	0	0	0	0	0	0	0	0	0.	0	0	1	0	0	1	0	1

A simpler method of conversion of decimal <u>integers</u> to binary integers is shown in the following algorism. Powers of 2 are taken out of the decimal number by successive divisions by the base 2. The remainders after successive divisions of the number (and its quotients resulting from successive divisions by 2) indicate the coefficients of the powers of the base.

Using the same examples:

Given: Decimal Number 18

To Find: Binary Equivalent (Simpler Method)

Method: Algorism

If there is a 0's power of 2 (2^0) contained in the number, its presence will be indicated by a remainder after the first division of the given number by 2. If there is a 1's power of 2 (2^1) contained in the number, its presence will be indicated by a remainder after the first division of the resulting quotient by 2. If there is a 2^2 contained in the number its presence will be indicated by a remainder after the next division of the resultant quotient by 2, etc. That is, the coefficients of the powers of 2 are the remainders after successive divisions.

2) 18	dividend	
2) 9	quotient	0 remainder = coefficient 0×2^0
2) 4	quotient	1 remainder = coefficient 1×2^1
2) 2	quotient	0 remainder = coefficient 0×2^2
2) 1	quotient	0 remainder = coefficient 0×2^3
0	quotient	1 remainder = coefficient 1×2^4

This same algorism may be applied to the conversion of the decimal number 730 to a binary number:

2) 730	
2) 365	0×2^0
182	1×2^1
91	0×2^2
45	1×2^3
22	1×2^4
11	0×2^5
5	1×2^6
2	1×2^7
1	0×2^8
0	1×2^9

1011011010. = 730.

b. <u>Fractions</u>

Given: .147

To Find: Binary Equivalent

Method: Extraction of Powers of 2

The highest power of 2 in .147 is 2^{-3} = .125 so a 1 goes in No.-3
.147 - .125 = .022

The next power of 2 in order is 2^{-4} = .0625, but this power of 2 is not contained in .022, so coefficient of the (-4) place = 0.

2^{-5} = .03125; not in .022, so 0 in No. (-5).

2^{-6} = .015625; is in .022, so 1 in No. (-6).
.022 - .015625 = .006375

2^{-7} = .0078125; not in .006375, so 0 in No. (-7), etc.

That method of converting a decimal to the binary system always works, but it is laborious and offers many chances for mistakes in the division and subtraction of such long numbers.

There is another method based on the same principle of taking out powers of 2, which, however, is much simpler. Given a decimal: -- by the former method, if it is larger than 2^{-1} (=.5), a 1 goes in the -1 place; if the number (or the remainder after subtraction of .5) is greater than or equal to .25 ($=2^{-2}$), a 1 goes in the -2 place. However, it is the same thing to say if twice the given decimal is larger than 2 x 2^{-1} (=1), a 1 is put in the -1 place; if 4 times the given decimal is larger than 4 x 2^{-2} (=1), a 1 is put in the -2 place. If 8 times the given decimal is larger than 8 x 2^{-3} (=1), a 1 is put in the -3 place. This is the same thing as doubling the number (or its remainder after a power of 2 is taken out) at each step and comparing it with 1. If the result becomes greater than 1, a 1 is taken out, and the doubling process starts again on the remainder.

Given: .147

To Find: Binary Equivalent

Method: Algorism

The former method started out by asking:

is .147 $\geq$ .5?	(If so, a 1 goes in No. -1; if not, a 0 goes in -1.)
is .147 $\geq$ .25?	(" " " " " " No. -2; " " " " " " -2.)
is .147 $\geq$ .125?	(" " " " " " No. -3; " " " " " " -3.)

This method starts out by asking:

is $2(.147) \geq 1$? (If so a 1 goes in No. -1; if not, a 0 goes in -1.)
is $2 \times 2(.147) \geq 1$? (" " " " " " No. -2; " " " " " " -2.)
is $2 \times 2 \times 2(.147) \geq 1$? (" " " " " " No. -3; " " " " " " -3.)

$2(.147) = .294 \not\geq 1$, therefore, 0 in No. -1
$2 \times 2(.147) = .588 \not\geq 1$, " 0 " No. -2
$2 \times 2 \times 2(.147) = 1.176 > 1$, " 1 " No. -3

$(1.176 - 1.000 = .176)$

$2(.176) = .352 \not\geq 1$, therefore, 0 in No. -4
$2 \times 2(.176) = .704 \not\geq 1$, " 0 " No. -5
$2 \times 2 \times 2(.176) = 1.408 > 1$, " 1 " No. -6

$(1.408 - 1.000 = .408)$

$2(.408) = .816 \not\geq 1$, therefore, 0 in No. -7
$2 \times 2(.408) = 1.632 > 1$, " 1 " No. -8, etc.

This result checks with that shown in detail above. It can also be shown that if a decimal repeats itself in the decimal system, it also repeats itself in the binary system. This method is shown more compactly below:

0.147	
0.294	.0
0.588	0
1.176	1
0.352	0
0.704	0
1.408	1
0.816	0
1.632	1
1.264	1
0.528	0
1.056	1
0.112	0

III. CONVERSION OF BINARY NUMBERS TO THE DECIMAL SYSTEM

Given a number in the binary system, it is always a simple matter to convert it to the decimal system. The converted number is simply the sum of the powers of 2 whose presence in the given number is indicated by 1's in the corresponding binary places.

Binary Place	5	4	3	2	1	0.	-1	-2	-3	-4	Decimal Equivalent
	1	0	1	1	0	1.	0	1	0	1	$1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-2} + 1 \times 2^{-4} =$
											$32 + 8 + 4 + 1 + .25 + .0625 =$
											45.3125

The coefficients of the other powers of 2 are zero, so they do not contribute to the converted number.

IV. ARITHMETIC

a. Addition

Since 1 is the largest digit in the binary system, it is evident that any sum larger than 1 must be represented with the aid of carryovers. Therefore, no matter how many 1's are added up in one column, the result under that column must be a 0 or a 1; the rest of the sum is carried over in its binary notation and set up at the head of the adjacent columns to the left as carryover figures. Thus, if a sum of 1's in a column adds up to 6 (which is 110 in the binary notation) a 0 is put at the bottom of the column and the two 1's are put at the head of adjacent columns to the left as carryovers. This is the same as adding the 1's in binary fashion at each step of the columnar addition.

(1 + 1 = 10; 10 + 1 = 11; 11 + 1 = 100; 100 + 1 = 101; 101 + 1 = 110 = 6)

e.g.	1	1	10	2	11	3	111	7	111011	59
	1	1	1	1	1	1	101	5	110111	55
	10	2	11	3	100	4	1100	12	1110010	114

(The small numbers above the examples are carry-over figures put in for ease in following the addition procedure.)

					or, more easily:				
e.g.	111	7	111	7	111	7	111	7	Addends
	011	3	110	6	011	3	110	6	
	101	5	011	3	1010	10	1101	13	
	001	1	111	7	101	5	011	3	
	100	4	010	2	1111	15	10000	16	
	111	7	100	4	001	1	111	7	
	11011	27	11101	29	10000	16	10111	23	
					100	4	010	2	
					10100	20	11001	25	
					111	7	100	4	
					11011	27	11101	29	Sum

b. Subtraction

Subtraction is based on the following rules:

0 from 1 always gives 1, and 1 from 0 always gives 1, but the latter requires "borrowing" from the first column to the left. A 1 in the first column to the left is reduced to 0 by borrowing; a 0 in the first column to the left is reduced to 1, causing the digit in the second column over to be reduced, etc.

10	2	100	4	1000	8	1110	14	1001010	74	Minuend
-1	-1	-1	-1	-1	-1	-1	-1	-0111101	-61	Subtrahend
1	1	011	3	0111	7	1101	13	0001101	13	Remainder

The small numbers show how the digits are changed by borrowing. See also Subtraction under "Complements".

c. Multiplication

Multiplication in the binary system is done exactly as in the decimal system and is based on the multiplication table 0 x 1 = 0, 1 x 1 = 1, 0 x 0 = 0.

```
      101011     or      101011      43 = 2^5 + 2^3 + 2^1 + 2^0     Multiplicand
      101110             101110      46 = 2^5 + 2^3 + 2^2 + 2^1     Multiplier
      000000            1010110     258
     101011            101011      172
    101011            1000000010   1978                             Product
   101011             101011
  000000             1001011010
 101011             1010110
 11110111010        11110111010    1978 = 2^10 + 2^9 + 2^8 + 2^7 + 2^5 + 2^4 + 2^3 + 2^1
```

d. Division

Division in the binary system is carried out exactly as in the decimal system.

```
 1) 1 = 1          1) 0 = 0
                                                   Check on Quotient
         0111.1100100001011001             7.782608
 10111) 10110011.0000000000000000     23) 179.000000    2^2 + 2^1 + 2^0 = 7.
         10111                             161                   2^-1  =  .5
         0101011                            180                  2^-2  =  .25
          10111                             161                  2^-5  =  .031250
          0101001                            190                 2^-10 =  .000976
           10111                             184                 2^-12 =  .000244
           0100100                            60                 2^-13 =  .000122
            10111                             46                 2^-16 =  .000015
            0011010                           140                          7.782607
             10111                            138
             00011000                          200
                10111                          184
                0000100000                      16
                    10111
                    00100100
                      10111
                      0011010
                        10111
                          11000
                          10111
                           0001
```

It should be noted that in order to get the decimal equivalent of the binary quotient to equal the decimal quotient to 5 decimal places, the binary division had to be carried to 16 binary places.

e. Complements

The ordinary complement of a number in the decimal system is obtained by subtracting the number from the next higher power of 10: e.g. complement of 18 = 100 - 18 = 82. The ordinary complement of a number in the binary system is obtained by subtracting the number from the next higher power of 2: e.g. complement of 5 = 2^3 - 5 = 8 - 5 = 3. It can be shown that the ordinary complement of a power of 2 is that power of 2 itself. See Example 3.

Another kind of complement of a number is obtained by subtracting the number from any higher power of 2. Notice its use under "Complements, (Subtraction Using Complements).

	(1)	(2)	(3)
Given:	100101	101010	100000

To Find: Binary Complements

Method: Subtract from next higher power of 2.

(1)	1000000	64	(2)	1000000	64	(3)	1000000	64
	-100101	-37		-101010	-42		-100000	-32
	0011011	27		0010110	22		0100000	32

Another method for finding the ordinary complement of a number in the binary system is to interchange all 0's and 1's and add 1.

	(1)	(2)	(3)
Given:	100101	101010	100000

To Find: Binary Complements

Method: Interchange 0's and 1's and add 1.

(1)	No.	100101	number
		011010	interchange 0's and 1's
		1	add 1
		011011	complement
(2)	No.	101010	
		010101	"
		1	
		010110	
(3)	No.	100000	
		011111	"
		1	
		100000	

These results check with those above.

f. Subtraction Using Complements

Instead of subtracting one number from another, it is possible to take a complement of the subtrahend and add that complement to the minuend, provided the power of 2 which was added to the subtrahend in order to get a complement is subtracted from the answer. Practically, subtracting out the added power of 2 means dropping the 1 in the last binary place on the left, if the power of 2 used in getting the complement is greater than that contained in either number. If not, then the power of 2 must be subtracted out by the usual subtraction method.

Regular Subtraction

1011100		Number		-10011	
-001101	Number	Complement	=	01101	(from 2^6)
1001001	Answer	"	=	1101101	(from 2^8)
		"	=	1111101101	(from 2^{11})

Subtraction by Addition of Complements

1011100		1011100		1011100	
01101	Complement	1101101		1111101101	
1101001	Sum	11001001		10001001001	
-100000	2^6	-10000000	2^8	-10000000000	2^{11}
1001001	Answer	01001001		00001001001	

Notice that in the two examples on the right, dropping the last 1 on the left in the sum gives the same result as subtracting out the power of 2 added to get a complement, because the power of 2 added was greater than that contained in either number.

Signed: M F Mann
Margaret Florencourt Mann

Approved: JWF
J. W. Forrester

MFM:has:cm

Engineering Note E-479

Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: BASIC CONVERSION PROGRAM, SEPTEMBER, 1952

To: 6345 and 6889 Engineers

From: M. Rotenberg

Date: September 4, 1952

Abstract: This note supercedes part 2 of M-1590

Two basic conversion programs will be available by September 15, 1952, one for direct input from standard tape (Direct Basic Conversion) and one for conversion to 5-56 tape (5-56 Basic Conversion). The vocabulary for both of these is identical and is given in this note. No provision is made for the use of floating addresses or multi-register length numbers. The comprhensive conversion program, which will be available soon after September 15, 1952, will treat tapes prepared for basic conversion correctly.

Introduction

The vocabulary of the new basic conversion program provides essentially the same facilities provided by the program previously in use but many details have been changed.

The comprehensive conversion scheme will allow (in addition to the basic facilities described below) a more general number system, a larger number and variety of preset parameters, interpretive instruction codes, automatic selection of interpretive, output, and mistake diagnosis routines, and floating addresses.

Headings

The tape commences with the tape number and author's name typed out:

TAPE 5432-1 T. Brown

or

PARAMETER 1234-5 T. Brown

Each title is followed by one carriage return and then the word FEEDOUT followed by another carriage return. Three inches of blank tape are then fed out.

Both conversion programs print the heading as written, preceded and followed by two carriage returns. In the D. B. program printing of the heading can be surpressed by resetting FF 6 to a negative value by TP 3. The 5-56 Basic punches the tape number readably on tape and provides suitable feed-out before and after.

When octal addresses are used, the heading is written

TAPE 5432-1 T. Brown ≥ OCTAL
with the same feed-out procedure as before.

Otherwise, decimal addresses are assumed.

Address Assignments

At the start of the program and at any point thereafter an address followed by a vertical bar indicates the location into which the next word is to be stored. In the absence of any further indication words will be stored consecutively. In the absence of even an initial indication words will be stored consecutively starting in register 32.

Relative Addresses

With subroutines and block assembly procedures, blocks of instructions may be written with addresses relative to the start of the block. Relative addresses are always decimal and are always followed by an r sign followed by either a comma or vertical bar. The start of a block is indicated by 0r, and other relative addresses such as 17r, may be written if desired and will simply be ignored by the conversion program. An address assignment may be made by writing 35r| which stores the next word in register 35r regardless of consecutivety.

Relative addresses within an instruction simply end with a r sign.

Instructions

Two lower case letters followed by as many digits as are necessary for an address comprise an instruction. The two letters may be any of the following: si, rs, (bi), rd, (bo), rc, -, -, ts, td, ta, ck, -, ex, cp, sp, ca, cs, ad, su, cm, sa, ao, (dm), mr, mh, dv, * , * , sf, - , - . These letter pairs will be assigned code values from 0 through 31 respectively. The function letters in parentheses will be converted properly even though they are not part of the present order code.

Operation cl will be changed from 2 to 30 in the conversion program on the same day it is changed in WWI.

Numbers

Decimal fractions are written as +. or -. followed by exactly 4 digits.

Decimal integers with an implicit factor of 2^{-15}, are written as + or - followed by as many digits as necessary, no decimal point.

Octal numbers are written as 0. or 1. following by exactly 5 octal digits. The 1. is the start of a negative octal number, the remaining digits being the sevens complement of the absolute magnitude.

Preset Parameters

The only preset parameter available will be the personal parameter pp. As

The former sl, sl, sr, sr*, cl cl*, operations are now to be written as letter triples slr, slh, srr, srh, clc, clh.

many of these as desired may be used. Values are assigned to preset parameters anywhere in the program by writing pp 6 = followed by any word. If no word follows, then pp 6 = 0. Parameters are added or subtracted into words by writing + or - pp 6, e.g., ca 7 - pp 6.

Temporary Storage

The zero-th register of a temporary storage block is assigned by writing anywhere in the program t = followed by an address. To refer to a temporary storage register, the third, for example, one writes ca3t.

Duplicated Words

When several consecutive registers are to be set to contain the same word initially the notation DITTO THRU preceded by the word and followed by an address may be used. For example:

+.5000 DITTO THRU 100

will put +.5000 in the next available register and in all registers through and including 100.

50| +.5000 DITTO THRU 100

will put +.5000 in registers 50 through 100 inclusive.

50| +.5000 DITTO THRU 50r

will give the same results as in the previous example.

The word FEEDOUT, and a suitable amount of blank tape must follow any of the DITTO examples above.

End of Program

The end of the program is indicated by the words START AT followed by the address of the register which contains the first instruction to be obeyed in the program proper. This is followed by a carriage return and the word FEEDOUT followed by another carriage return and blank tape. In the case of the D. B. program, the computer will perform a conditional stop before control is transferred to the indicated register.

Feed-out

Prepared tapes will have feedout after every five lines of printing for use in the Comprehensive Conversion System described later. Feedout may, of course, occur in any quantity anywhere, but if it is to be of use it must be greater than 1.5 inches and must be preceded by the word FEEDOUT and a single carriage return.

Illegal Characters

To aid in the detection of mistakes, the Basic Conversion programs will stop wherever a foreign or illegal character occurs on tape. This includes all binary combinations not contained in the Flexowriter code except 000000 (with 7th hole) which is disregarded. It also includes letters g, j, n, q, w, y, and z and back space.

Disregarded Characters

For the convenience of the typist, nullify (111111), space, color shift and the foreign character 000000 (with 7the hole) are completely ignored at all times by the Basic Conversion Programs. Carriage return, tab and comma are ignored between words (i.e., two carriage returns or a tab followed immediately by a carriage return are all right), but may not appear within a word for obvious reasons.

Synonyms

For convenience, the characters ℓ and 1, and tab and carriage return (⤶) are treated intentionally as synonyms.

Signed M. Rotenberg
M. Rotenberg

Approved C. W. Adams
C. W. Adams sk

Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: WWI CONTROL SWITCHES AND PUSHBUTTONS FOR NORMAL OPERATION OF THE COMPUTER

To: Group 61 and Applications Group

From: J. H. Hughes

Date: December 16, 1952

Abstract: This memorandum presents a table of the controls used in normal operation of the WWI Computer, their physical locations, and what they do. It also tells you how to use these controls for some common procedures.

INTRODUCTION

The toggle switches and pushbuttons listed in the attached table are all you normally need for controlling the computer, except for data handling for the Air Defense Project. All other switches in Test Control are for maintenance and trouble shooting; they should be left alone when you are running a program.

HOW TO USE THE CONTROLS FOR COMMON PROCEDURES

1.0 How to Read in a Program from Flexowriter Tape

1.1 5-5-6 Tape

Put the beginning end of the tape in the Photoelectric Tape Reader, taking care that it is going from right to left and that the solid row of seventh holes is on the side near you.

Press the READ IN button and the tape will be read in.

If you turn on the STOP ON si-1 switch the computer will stop at the end of read-in and you can take care of any special resets that the program may need. Otherwise the program will start up as soon as read-in is finished.

1.2 Standard Tape

Before you can read in standard tape you must read in the Direct Conversion Program tape, T-2046. You do this by following the procedure in 1.1 above, with the STOP ON si-1 switch turned on. Then put the Standard

Tape that you want to read in into the PETR, put the Program Counter Reset Switches to 2037 (octal) and press the START OVER button. The Standard Tape will be read in.

Note that the Direct Conversion Program is stored in ES registers 1251 through 2037 (octal). This means that you cannot do a direct read-in of a program on standard tape if it uses these registers. Instead you must have the tape converted to 5-5-6 form in the usual way.

2.0 What to do if You Get an Alarm

2.1 Parity Alarm. Call a technician.

2.2 During Read-In by Direct Conversion Program

2.21 Program Alarm. Usually means that there is not enough room in storage for both your program and the conversion program. Have the tape converted to 5-5-6 form. sometime

2.22 Conversion program stops with PC holding 00001: means that the conversion program has found an error in the tape. Send the tape back to tape room for fixing. If program stops with PC holding 1613 this is normal end of read-in.

2.3 During Read-In of 5-5-6 Tape

2.31 Program Alarm. Usually means that the tape is in the PETR crooked or upside down or backwards. Try again. not always true

2.32 Check Alarm, Program Counter holding 00007. Means that the Sum Check has found an error in the tape. Send it back to the tape room for fixing. If PC holds any other number the alarm is probably the result of a transient error. Try read-in again.

2.4 During the Program

2.41 Overflow, Divide Error or Program Alarms mean that there is something wrong with the program. See Section 3.0 on trouble-shooting programs.

2.42 Parity check alarms, unprogrammed check alarms or inactivity alarms are probably due to computer malfunction. Call a technician.

3.0 How to Trouble Shoot a Program

3.1 General

When you have trouble with a program you must decide whether to use the computer to help you find out what is wrong or simply to record the contents of the most significant registers (PC, PR, AC, etc., according to the kind of alarm) and try and work out what is wrong at your desk. If there is no great rush to get the program fixed and running by a certain date, then it is probably better to use your time rather than the computer's to find out where the program goes awry and why. If, on the other hand, the problem is a rush job, then the computer can give valuable help in the quick detection of program trouble, and you may be justified in using computer time for this purpose.

3.2 Post Mortem Subroutines

The Applications Group has devised a number of subroutines which may be used to print out parts of the program in various ways. One of the most straightforward of these is the "Storage Print Out" subroutine, which prints out the contents of ES so that you can see what has happened in the program up to the time when an alarm happened.

3.3 Selected Pulse

It is possible to run the program through in sections, stopping the computer every time some particular order is given. You do this by throwing the STOP ON SELECTED PULSE switch on. The Time Pulse is selected by the TP Selector in TC 5-3. Do not use TP 5. The order (or two orders) is selected by the plug leads located out back at the Operation Matrix.

3.4 Order by Order

It is possible, but rarely desirable, to run the program through order by order simply by reading in with the STOP ON si-1 switch on and then pressing the ORDER BY ORDER button once for each order. This is the least efficient way of using computer time to trouble shoot your program.

4.0 Manual Insertion

It is possible to change manually the contents of a register of ES by using the following procedure.

In	Flip-Flop	Register		2	Reset	Switches	put	the	"word" you want to insert.
"	"	"	"	3	"	"	"	"	instruction "ca 2".
"	"	"	"	4	"	"	"	"	instruction "ts x" where x is the address of the ES register to be changed.
"	"	"	"	5	"	"	"	"si 0".	

In PC reset switches put 00003.

Press the START OVER button, and the job is done.

SIGNED J. H. Hughes
J. H. Hughes

APPROVED N L Daggett
N. L. Daggett

JHH/cp
Table Attached

CONTROLS USED IN NORMAL OPERATION OF WWI

TYPE OF SWITCH	NAME	LOCATION	WHAT IT DOES
P.B.	ERASE ES	Console & TC5-1	Charges whole surface of every storage tube to the zero state. Do not press unless computer is stopped.
P.B.	READ IN	Console & TC3-5	Stops computer if running. Resets Program Counter to the beginning of the read-in program (which is in Toggle Switch Storage). Restarts computer, which then reads in the tape with the Photoelectric Tape Reader. Flexoreader may be used by putting a special read-in program in Toggle Switch Storage. At the end of read-in the program starts, or the computer stops if the STOP ON si-1 switch is on. TP3 FF Reset is suppressed from time READ-IN button is pressed until read-in is complete.
P.B.	CLEAR ALARM	Console & TC3-5	Clears alarm indication any time and that is all.
P.B.	START OVER	Console & TC3-5	Clears most flip-flops in the computer, resets Flip-Flop Storage, resets Control Switch to ck, resets PC to number held in Program Counter Reset Switches, restarts the computer.
P.B.	START (OVER) AT 40	Console & TC3-5	Same as START OVER except resets PC to 40.
P.B.	STOP (Formerly CHANGE TO P.B.)	Console & TC	Stops the computer from any state. (Do not confuse with "stop clock" which may be called for by various parts of the computer to allow them to complete their job before the computer's main cycle of 8 time pulses continues.)
P.B.	START CLOCK	Console & TC3-5	Restarts computer if hung up in "stop clock". (This might be caused by an "illegal" in-out order.)
P.B.	RESTART	Console & TC3-5	Restarts the computer from the P.B. mode. (Will not restart from "stop clock".)

TYPE OF SWITCH	NAME	LOCATION	WHAT IT DOES
P.B.	ORDER BY ORDER	Console & TC3-8	If computer is stopped, restarts it. Lets computer run to next TP5, and stops.
P.B.	EXAMINE	Console & TC3-8	Starts over and runs to TP5. This enables you to inspect in the PR the contents of the register whose address is in the PC Reset Switches. (Contents are displayed in PR.)
T.S.	STOP ON <u>si-1</u>	Console	If switch is on (up) lets <u>si-1</u> stop the computer.
T.S.	STOP ON SEL. PULSE	Console	If switch is on (up) lets selected Time Pulse of Selected Order stop the computer.
T.S.	DISPLAY SELECTORS	With each display scope	Each switch permits appropriate display to appear on scope.
T.S.'s	FFS RESET switches	With each set of FFS indicators in TC-2	Specify number to which FFS registers will be reset if FFS reset called for.
T.S.'s	PC RESET switches	TC3-8	Specify the number to which Program Counter will be reset on START OVER or EXAMINE.
P.B.	RESET ALL FFS	FF reset panel	If computer is stopped, resets all digits of all Flip-Flop Storage registers to numbers specified by Flip-Flop Reset Switches and D-C insertion plugs.
P.B.	SELECTIVE FFS RESET & RE-START (PB)	FF reset panel TC3-7	Resets those of the FFS registers selected by the switches next to it, and restarts computer (if stopped).
P.B.	SELECTIVE FFS RESET IN MANUAL (PBM)	FFS reset panel	If computer is stopped, resets those of the FFS registers selected by the switches next to it.
T.S.	FFS RESET BY TP3	FFS reset panel	Causes every TP3 to reset those of the FFS registers selected by the switches next to it, except during read-in.

TYPE OF SWITCH	NAME	LOCATION	WHAT IT DOES
T.S.	FFS RESET BY PCEC	FFS reset panel	Causes the PC End Carry to reset those of the FFS registers selected by the switches next to it.
T.S.	FFS RESET BY rs	FFS reset panel	Causes TP3 of every rs order to reset those of the FFS registers selected by the switches next to it.

INPUT PROGRAM, OCTOBER, 1952

DECIMAL

	0	$+0x2^{-15}$	
6--→	1	$+1x2^{-15}$	Conditional stop
(19)	2	Flip Flop Reg #2	(sp y)
(12,10)	3	Flip Flop Reg #3	(word counter)
(16, 8)	4	Flip Flop Reg #4	(accumulated sum-mod-one)
(17)	5	Flip Flop Reg #5	(final sum-mod-one)
20--→(15,9)	6	Flip Flop Reg #6	(ts x or ck 5 or sp 1 or sp 21)
	7	sa 4	add new word to sum-mod-one
	8	ts 4	store new sum
	9	ao 6	increase ts x instruction by one
	10	ao 3	increase word counter by one
	11	cp 18	if word counter is negative, read in next word
23, 27--→	12	ts 3	reset word counter
	13	rd 13	read initial word from tape
	14	cp 12	if word is negative, reset counter
	15	ts 6	if word is positive, place it in 6
	16	ex 4	reset sum-mod-one in case this is WORD block
	17	ts 5	store the previously accumulated sum-mod-one in 5 in case this is CK block
11--→	18	rd 18	read word from tape
	19	ts 2	place it in 2 in case this is SP block
	20	sp 6	perform next the instruction in 6
556 Input--→	21	si 139	select photoelectric reader word input
	22	cao	prepare to reset word counter
	23	sp 12	
Magnetic Input--→	24	si 66	select magnetic tape reader forward
	25	rd 25	read first word of block
	26	cp 12	if negative, reset counter and proceed to read in
	27	sp 24	if positive, select input again to skip to next block
	28	immaterial	these words are not now needed and are
	29	immaterial	assigned no
	30	immaterial	value at present
	31	CLOCK	

Contents assigned to registers 24 through 30 will be changed when the auxiliary drum is installed.

DL-615

INTERPRETED ORDER CODE

WWI Order	Interpreted Order	Name	Decimal	Binary
si			0	00000
rs	itsc	cycle transfer to storage	1	00001
bi	iexc	cycle exchange	2	00010
rd	icac	cycle clear and add	3	00011
bo	icsc	cycle clear and subtract	4	00100
rc	iadc	cycle add	5	00101
	isuc	cycle subtract	6	00110
	imrc	cycle multiply	7	00111
ts	idvc	cycle divide	8	01000
td	ispc	cycle subprogram	9	01001
ta			10	01010
ck	icr	cycle reset	11	01011
	ict	cycle count	12	01100
ex	iat	add and transfer	13	01101
cp	iti	transfer index	14	01110
sp	sp	subprogram	15	01111
ca	ici	cycle increase	16	10000
cs	icd	cycle decrease	17	10001
ad	icx	cycle exchange	18	10010
su	ita	transfer address	19	10011
cm	icp	conditional subprogram	20	10100
sa	its	transfer to storage	21	10101
ao	iex	exchange	22	10110
dm	ica	clear and add	23	10111
mr	ics	clear and subtract	24	11000
mh	iad	add	25	11001
dv	isu	subtract	26	11010
slr	imr	multiply and round-off	27	11011
srr	idv	divide	28	11100
sf	isp	interpreted transfer control	29	11101
clc			30	11110

Jan. 27, 1953

DESIGNATIONS OF IN-OUT EQUIPMENT
(si addresses for all units listed)

Unit	0	1	2	3	Mode
Display scopes	600 octal 384 decimal	601 octal 385 decimal	602 octal 386 decimal		Customary use: Each scope has switches 0,1 & 2 on it. When all are o for each scope, display will appear on all scope
Printers		(Rm. 222) 215 octal 141 decimal	(Computer Rm.) 225 octal 149 decimal	(Not in use) 235 octal 157 decimal	print one character on an rc
Punch	204 octal 132 decimal				punches one character with 7th digit suppressed
	205 octal 133 decimal				punches one character with 7th digit punched
	206 octal 134 decimal				punches three characters with 7th digit suppressed
	207 octal 135 decimal				punches three characters with 7th digit punched
Photoelectric tape reader	211 octal 137 decimal				line-by-line (reads 1 line for each rd)
	213 octal 139 decimal				word-by-line, or automatic 5 - 56 (assembles 3 lines into 1 word for each rd) Reader runs free until dismissed by si 600
Flexo mechanical tape reader	200 octal 128 decimal				line-by-line (reads 1 line for each si + rd)
	202 octal 130 decimal				word-by-line, or automatic 5 - 56 (assembles 3 lines into 1 word for each si + rd, stopping automatically after word is assembled)
Camera	500 octal 320 decimal				index camera
Magnetic tape				(Assoc. with delayed-output equipment)	
	100 octal 64 decimal	110 octal 72 decimal	120 octal 80 decimal	130 octal 88 decimal	re-record, forward
	101 octal 65 decimal	111 octal 73 decimal	121 octal 81 decimal	131 octal 89 decimal	re-record, reverse
	102 octal 66 decimal	112 octal 74 decimal	122 octal 82 decimal	132 octal 90 decimal	read, forward
	103 octal 67 decimal	113 octal 75 decimal	123 octal 83 decimal	133 octal 91 decimal	read, reverse
	104 octal 68 decimal	114 octal 76 decimal	124 octal 84 decimal	134 octal 92 decimal	stop mode forward
	105 octal 69 decimal	115 octal 77 decimal	125 octal 85 decimal	135 octal 93 decimal	stop mode reverse
	106 octal 70 decimal	116 octal 78 decimal	126 octal 86 decimal	136 octal 94 decimal	record, forward
	107 octal 71 decimal	117 octal 79 decimal	127 octal 87 decimal	137 octal 95 decimal	record, reverse

Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: PROGRAMMING FOR IN-OUT UNITS

To: Group 61 and Applications Group

From: Philip R. Bagley

Date: September 17, 1952; revised December, 5, 1952

Abstract: This memo contains the details of programming for the various modes of operation of each in-out unit. The necessary instructions and selection addresses are given, together with the timing and possible alarms involved. This memo supersedes M-1514, M-1551, M-1623, and M-1623 Supplement #1.

Table of Contents:

IN-OUT NOTES

In-out operations. The group of computer operations termed "in-out operations" is composed of the operations involved with the transmission of words into and out of the computer: namely, si, bi, rd, bo, and rc. The block-transfer operations bi and bo will not be available until January, 1953.

si instructions. The action of the si instruction is to select a particular in-out unit and prepare it to start operating in a specified mode designated by the octal address digits p q r. p designates the class of equipment (such as magnetic tape units) and q and r together designate the number of the unit and the mode of operation. An si normally precedes one or more of the other in-out instructions involving bi, rd, bo, and rc, except in the case of the camera, which is operated by a single si alone, or in the case in which si is used to stop the computer or an in-out unit. Any instructions other than in-out instructions may intervene between si and its associated bi, rd, bo, or rc, without affecting the in-out process. Following an si instruction which specifies a read mode, the computer must not execute another si until the process initiated by the earlier si has ended. Since this is not insured by electronic means, it must be insured by programming at least one rd instruction after every si which selects a read mode.

Assigned si addresses. All the si addresses which have been assigned functions are listed under the equipment to which they apply. A complete list of assigned si addresses in numerical order is given at the end of this section. Unassigned si addresses may not be used indiscriminately. At present, the use of addresses containing 1's in the binary digit positions 5, 6, or 10 of an instruction will cause the computer to stop in a transfer check alarm. Certain other unassigned addresses are "illegal": that is, they may cause an in-out unit to operate in an unpredictable fashion. Other unassigned addresses are reserved for possible use at a later date.

Stop instructions. The si operation is also used to provide a "stop," either to stop the computer or to stop any in-out unit which does not stop automatically (that is, a magnetic tape unit or the photoelectric reader). si 0 will stop the computer. si 1 will stop the computer only if the "Conditional Stop" switch is ON. si (any assigned address) will stop any in-out unit which may be running, without stopping the computer; however, if no in-out unit need be selected by this si stop instruction, the unique designation si 630 (octal) or si 408 (decimal) should be used, both for program clarity and for safety of operation.

Possible changes in rc and rd operations. It is possible that at some future date the address sections of the rc and rd instructions may specify the address to which control should be transferred if the in-out unit is not ready to carry out the required recording or reading process. It is therefore advisable to set the address section of each rc and rd instruction equal to the address of the register containing the instruction. For example, register x would contain rc x or rd x.

Punched paper tape. The conventional form of a 16-digit word on punched paper tape is known as "556" form (distinct from a previous form termed "5-5-6"). The binary digits (numbered 0 through 15) are physically distributed on the tape in the following manner:

hole no.	(1)	(2)	(3)		(4)	(5)	(6)	(7)
one word	0	1	2	•	3	4	X	
	5	6	7	•	8	9	Y	
	10	11	12	•	13	14	15	
				feed holes				

Positions X and Y are normally unpunched to aid in visually reading the tape. However, they may contain the same information as positions 5 and 10 respectively. The word-by-word reading modes of the in-out system are devised to correctly assemble into a 16-digit word a word punched in 556 form. Each line of tape which contains information must have the 7th hole position punched. If this were not done, the tape reader could not distinguish a line of significant zeros (which it must read) from a line of blank tape (which it must ignore). The omission of the 7th hole then allows the feature of punched visual identification numbers which will be ignored by the reader.

Delays and alarms associated with the read mode. While an in-out unit is operating in the read mode, if the computer attempts to perform any in-out instruction (normally an rd) before a word or character has arrived in IOR, it must wait until the word or character arrives. If the computer has not cleared IOR (by an rd instruction) before the succeeding word arrives in IOR from an in-out unit, an in-out alarm results. In connection with programming for in-out units, the only type of alarm which occurs is an in-out alarm. All cases in which in-out alarms can occur are specifically noted in the discussion of each unit.

AUXILIARY MAGNETIC DRUM (NOT YET AVAILABLE)

Function of the auxiliary drum. The auxiliary magnetic drum provides 24,576 registers of "intermediate speed storage," where each register can store a 16-digit binary word. The computer can transfer a word to or read a word from any drum register, in a manner similar to the way it does in electrostatic storage.

Register numbering. The registers on the drum bear consecutive addresses from 0 to 24,575. The registers are in 12 groups along the drum, each group consisting of 2048 registers. A drum address is specified by a 16-digit binary word, of which digit 0 is immaterial, digits 1-4 specify the group number, and digits 5-15 specify the storage address. Within any register group, the storage addresses are treated modulo 2048. For example, a block transfer starting at address 2047 will deal in turn with registers 2047, 0, 1, 2, etc., in the same register group.

Access time. To gain access to a specific register on the drum takes, on the average, 8 milliseconds, equal to the time for one-half revolution of the drum. An additional 12 milliseconds seconds delay occurs if the register lies in a group different than the group last used.

Register selection. The next drum address to be selected is determined by the si instruction and by any necessary portions of the contents of AC at the time the si is executed. The si instruction may call for a new group number or a new initial storage address, or neither, or both. When a new group number is needed, it is taken from digits 1-4 of AC. When a new initial storage address is needed, it is taken from digits 5-15 of AC. The group selected on the drum remains selected until an si instruction specifically calls for a change of group. The next storage address selected will be one greater than the storage address most recently referred to unless an si instruction specifically calls for a new initial storage address. To provide for all the cases above, there are four possible ways for an si instruction to specify a register:

1) Select no new group or initial address

2) Select new group only
3) Select new initial storage address only
4) Select both new group and new initial storage address

In addition, the si instruction selects reading or recording.

Recording single words on the auxiliary drum. Programming for recording on the auxiliary drum is as follows:

si a — Selects the auxiliary drum and the record mode. If the si instruction calls for a new group number, it is selected in accordance with the contents of digits 1-4 of AC. If the si calls for a new initial storage address, it is selected in accordance with the contents of digits 5-15 of AC. The computer cannot perform another in-out instruction until the group change, if any, requiring 12 milliseconds, is completed.

rc-- — Records the contents of AC at the next address called for by the si instruction, or at the next consecutive address following the last address at which a word was recorded. The computer cannot perform another in-out operation until the in-out equipment completes the recording process, which takes an average of 8 milliseconds and a maximum of 16 milliseconds. An rc instruction is required for each word to be recorded. As many rc instructions as necessary may be used before the next si instruction. Any number of instructions other than in-out instructions may precede each rc.

Recording by block-transfer instruction. A bo instruction may be substituted for a series of rc instructions. The address of the bo must be the initial address of the block to be taken from ES, and $\pm$ n, the number of words to be recorded, must be stored times 2^{-15} in AC. The block transfer will require an average of 8 milliseconds and a maximum of 16 milliseconds for the first word to be recorded, and 64 microseconds for each additional word. If the block transfer involves both registers 2047 and 0, in that sequence, an additional 16 milliseconds is required to complete the transfer. Any sequence of rc and bo instructions may follow a single si.

Reading from the auxiliary drum. Programming for reading from the auxiliary drum is as follows:

si a — Selects the auxiliary drum and the read mode. If the si instruction calls for a new group number, it is selected in accordance with the contents of digits 1-4 of AC. If the si calls for a new initial storage address, it is selected in accordance with digits 5-15 of AC. Reads into IOR the word from the chosen drum address. The time required to obtain the word is an average of 8 milliseconds and a maximum of 16 milliseconds, plus an additional 12 milliseconds if a group change is necessary. One, and only one, rd or bi instruction must intervene between this and the next si instruction.

rd-- — Transfers the word in IOR to AC, then clears IOR.

Reading by block-transfer instruction. A bi instruction may be substituted for an rd instruction. The address of the bi must be the initial address of the block of registers in ES to which the words will be transferred, and $\pm$ n, the number of words to be read, must be stored times 2^{-15} in AC. Each bi must be preceded by an si. The block transfer will require an average of 8 milliseconds and a maximum of 16 milliseconds for the first word to be read, and 64 microseconds for each additional word. If the block transfer involves both registers 2047 and 0, in that sequence, an additional 16 milliseconds is required to complete the transfer.

Zero-length block transfers on bi and bo. The use of a bi instruction calling for the transfer of a block zero words in length will result in one word being read but not transferred. The reading of the word actually is initiated by the preceding si instruction, hence one word is already read by the time the bi is ready to be performed. If the bi calls for the transfer of no words, the word already read is simply discarded. Zero-length block transfers on bo will always be performed correctly, i.e. no recording will take place.

si addresses for auxiliary drum. When the auxiliary drum becomes available, the si addresses will be as follows:

READ MODE:

1)	no address specified	si 700 octal	or	si 448 decimal
2)	select new group	si 701 octal	or	si 449 decimal
3)	select new initial address	si 702 octal	or	si 450 decimal
4)	select new group and address	si 703 octal	or	si 451 decimal

RECORD MODE:

1)	no address specified	si 704 octal	or	si 452 decimal
2)	select new group	si 705 octal	or	si 453 decimal
3)	select initial address	si 706 octal	or	si 454 decimal
4)	select new group and address	si 707 octal	or	si 455 decimal

CAMERA

Action of the camera. The selection of the camera by an si instruction results in the following cycle of operation, termed an "index cycle":

1) Close shutter
2) Advance film one frame
3) Open shutter

If shutter is initially closed, the first step is omitted.

Programming the index cycle. The index cycle is effected by the single instruction si 500 (octal) or si 320 (decimal). 220 milliseconds will elapse after the si instruction before the computer performs the next in-out instruction.

Manual controls. Three push-buttons, labeled CLOSE, OPEN, and INDEX, on the camera control panel provide for manual control of the camera. Depressing the INDEX button advances the film one frame; if the shutter is initially open, it is automatically closed during the period required to advance the film. A remote push-button unit is also available, which may be plugged into any one of several jacks. Depending upon whether the shutter is initially open or closed, one of the following sequences of operation will result.

> If the shutter is initially closed, depressing the button opens the shutter; releasing the button closes the shutter and advances the film.
>
> If the shutter is initially open, depressing the button has no effect; releasing the button closes the shutter, advances the film, and reopens the shutter.

CLOCK

Use of the "clock," or "time register." The "clock" is provided to indicate increments of time, counting in one-quarter second units from 0 to 32767 (136.53) minutes) and then starting over. It is a 16-digit flip-flop register of which the sign is always positive, and it is located (arbitrarily) at address 31 (decimal) in Test Storage. The contents of the clock may be read out by an instruction (ex, ca, cs, ad, su, cm, sa, ad, ao, mr, mh, dv, dm) in ES (but not in TS); information cannot be transferred to the clock by the computer. The clock is reset to zero by a pushbutton.

DELAYED OUTPUT VIA MAGNETIC TAPE

Delayed-output units. Where printed page or punched paper tape output is desired, computer time can be conserved by the use of the delayed-output unit. The 6-digit binary characters to control the printer or the punch are recorded on magnetic tape by the computer and the tape is later run through the delayed-output unit. The tape for delayed output can be recorded at the rate of 100 characters per second. An 800-foot reel of magnetic tape will store about 32,000 characters, which can be recorded in a minimum of 5.2 minutes and can be printed or punched out in about 75 minutes.

Tape unit connections. There are two magnetic tape units, designated 1A and 1B, associated with the delayed-output equipment. Either one of the units may be connected to the computer while the other is connected to the delayed-output equipment. The connections may be interchanged by a manual switch.

Programming for delayed output. In order to record on magnetic tape a series of characters for later automatic printing or punching, the following conventions must be observed:

1) A full 16-digit word must be recorded on magnetic tape to store each 6-digit binary character; each character occupies digit positions 2 through 7 of the recorded word, the contents of the other digit positions being immaterial. The necessary binary characters for the "delayed printer" are identical to those used for the printer (see the Flexowriter Code).

2) The temporal separation of the words on magnetic tape must be no less than 7 milliseconds. A 12-millisecond separation will automatically be achieved if the individual words are recorded as blocks each 1 word long. If recording time is to be a minimum, the program must "count" a 7-millisecond delay between individual recorded words.

3) It is advisable (but not necessary) to provide a Flexo-unit stop character as the last recorded character so that the automatic printing or punching equipment may operate unattended.

LIGHT GUNS

Action of a light gun. A light gun signal generated by the display of a point on a scope (see SCOPES) is transmitted immediately to IOR (which has been reset to zero by the display instruction). The signal causes ones to appear in two digit positions of IOR: namely, in the sign digit position, indicating that a signal has been received, and in the digit position to which the light gun is connected. Several light guns may send signals to IOR simultaneously.

Programming for light gun inputs. To determine if a light gun signal has occurred, it is necessary to program an rd after the point has been displayed and before another in-out instruction. The rd will bring the contents of IOR into AC, a cp instruction will examine the sign digit to see if any signal has been received, and successive sf or cl instructions will determine which light guns generated the signal.

Present light gun equipment. At present, the one available light gun is permanently connected to the sign digit of IOR, and can be connected by a four-position switch to any one of digit positions 1, 2, 3, and 4 of IOR. At some future date there will be more light guns, which may or may not be tied to specific digits of IOR.

MAGNETIC TAPE

Action of magnetic-tape units. The magnetic-tape units will record and read 16-digit binary words. An individual block is of arbitrary length, the start of the block being identified by a block marker automatically recorded. In addition to provisions for recording and reading, a mode termed "re-record" searches for a block marker and then switches to the record mode. Previously recorded information is automatically erased from a tape which is running in the record mode. A tape unit which is instructed to stop will continue to coast for approximately 6 milliseconds, but will not affect the recorded data which passes under the heads during this coasting period. A tape unit running at full speed and instructed to reverse direction will continue moving in the original direction for approximately 6 milliseconds, but will immediately being erasing and counting the delay for a block space if recording, or counting a delay and then searching for a block marker if reading or re-recording. The tape unit is a free-running unit: once started by an si instruction, it runs free until stopped by another si instruction.

Automatic assembly and disassembly of words. A 16-digit word is in actuality recorded as eight pairs of digits on magnetic tape, the word being automatically disassembled by digit pairs in IOR. On reading, the word is automatically assembled (by successive shifts left) by digit pairs in IOR. The word will be assembled properly only if the tape is running in the same direction as it was when recorded. If the tape is read in the direction opposite to that in which it was recorded, the resulting words must be unscrambled by a special subroutine.

Recording. Programming for recording a block of words is as follows:

si m — Selects the tape unit and starts the unit in forward or reverse, depending on the address m. An interblock space 12 milliseconds long is generated on the tape, then a block marker is automatically recorded. The computer cannot perform another in-out instruction until this 12-millisecond period has elapsed.

rc-- — Records on tape the 16-digit word contained in AC. 2.5 milliseconds must elapse before the computer can perform another in-out instruction. An rc is required for each word to be recorded. As many rc instructions as necessary may be used before the next si instruction. Any number of instructions other than in-out instructions may precede each rc.

si m — Identical to the si m above, for the purpose of erasing any previously recorded data at the end of the block. If the tape has been previously erased so that there is no possibility of old data occurring at the end of a newly-recorded block, this instruction and the succeeding si n may be omitted.

si n — Selects the same unit for recording, but in the opposite direction. The tape will coast in the original direction for approximately 6 milliseconds; it will then reverse, erase the block marker recorded by the preceding si m and move the newly-erased space under the heads. After 12 milliseconds a block marker will be recorded. This si instruction must follow within 16 milliseconds of the previous si.

si-- Stops the tape unit. Any si instruction which has been assigned a function will stop the tape unit, but if the program does not require a specific si instruction, use the unique designation si 630 (octal) or si 408 (decimal). The maximum safe interval between this si and the preceding si n, if any, is 12 milliseconds.

Recording by block-transfer instruction. The system is not designed to permit recording on magnetic tape by the block-transfer instruction.

Re-recording. Re-recording is similar to recording except that the unit runs for approximately 5 milliseconds and then begins the recording process after passing the next block marker instead of beginning to record immediately irrespective of the tape position. Only one block marker precedes the newly-recorded block.

Reading. Programming for reading of words is as follows:

si m Selects the tape unit and starts the unit in forward or reverse, depending upon the address of m. After waiting for 5 milliseconds, the computer reads into IOR the first word after the next block marker. The amount of time required for this process will depend on the distance of the next block marker from the reading heads. This si instruction must not be followed by another si without at least one intervening rd or bi instruction.

rd-- Transfers the contents of IOR to AC, then clears IOR in preparation for receiving the next word from tape. As many successive rd instructions will be needed as there are words to be read from tape. Assuming that the words were recorded at maximum density (one word every 2.5 milliseconds) a pair of digits will be read to IOR at intervals of approximately 300 microseconds. The computer must execute an rd instruction often enough to extract a word from IOR and clear IOR before the first pair of digits of the next recorded word arrives from the tape unit; otherwise an in-out alarm will result. To stop reading before the end of a recorded block has been reached, give an instruction to stop the tape unit within 2 milliseconds after the last desired word has been read; otherwise an in-out alarm may result. Any instructions other than in-out instructions may precede each rd.

si-- Stops the tape unit. Any si instruction which has been assigned a function will stop the tape unit, but if the program does not require a specific si instruction, use the unique designation si 630 (octal) or si 408 (decimal).

Reading by block-transfer instruction. A bi instruction may be substituted for a series of rd instructions. The address of the bi must be the initial address of the block of registers in ES to which the words will be transferred, and $\pm$ n, the number of words to be read, must be stored times 2^{-15} in AC at the time the computer executes the bi. Any sequence of rd and bi instructions may follow a single si.

Zero-length block transfer on bi. The use of a bi instruction calling for the transfer of a block zero words in length will result in one word being read but not transferred. The reading of the word actually in initiated by the preceding si instruction, hence one word is already read by the time the bi is ready to be performed. If the bi calls for the transfer of no words, the word already read is simply discarded.

Skipping blocks. The re-record instruction can perform an auxiliary function: that of making possible the skipping of any number of blocks, in either forward or reverse. Each si instruction to re-record causes the tape unit to search for the next block marker and to switch to the record mode as soon as the block marker is found. Since the record mode erases previously recorded data, however, another si instruction must switch the unit out of the record mode in time to avoid erroneously erasing. The maximum permissible interval between the si to re-record and the si to switch out of the record mode is dependent on the distribution of data on the tape, but in no case will it be less than 5 milliseconds.

si addresses for magnetic tape units. The si addresses for the magnetic tape units are as follows:

UNITS 2 AND 3 NOT YET AVAILABLE

Unit	0	1A, 1B*	2	3
Re-record, forward	100 octal 64 decimal	110 octal 72 decimal	120 octal 80 decimal	130 octal 88 decimal
Re-record, reverse	101 octal 65 decimal	111 octal 73 decimal	121 octal 81 decimal	131 octal 89 decimal
Read, forward	102 octal 66 decimal	112 octal 74 decimal	122 octal 82 decimal	132 octal 90 decimal
Read, reverse	103 octal 67 decimal	113 octal 75 decimal	123 octal 83 decimal	133 octal 91 decimal
Record, forward	106 octal 70 decimal	116 octal 78 decimal	126 octal 86 decimal	136 octal 94 decimal
Record, reverse	107 octal 71 decimal	117 octal 79 decimal	127 octal 87 decimal	137 octal 95 decimal

* associated with delayed-output equipment

MECHANICAL TAPE READER

Action of the mechanical tape reader. The mechanical tape reader "reads" the 6-digit binary combination punched in a line of paper tape and transmits it to the right-hand six digit places of IOR. In the line-by-line mode, each reading operation reads one line of tape and forms a word of which the left-hand ten digits are zero and the right-hand six digits correspond to the binary combination punched in the tape. In the word-by-word mode, each reading operation reads three lines of tape, and assembles (by successive shifts left in IOR) a 16-digit word from the digits punched in the tape in 556 form. The mechanical tape reader does not need to be stopped by an si instruction.

Programming for line-by-line mode. Programming for reading in the line-by-line mode is as follows:

si r Selects the mechanical reader.

rd-- Reads the next 6-digit character from paper tape into the right-hand six digit positions of AC via IOR, and clears IOR in preparation for receiving the next character. The contents of digits 0-9 of AC will be zero, and the contents of digits 10-15 of AC will correspond to the binary combination read from tape. In this mode with the mechanical reader, the computer requires 106 milliseconds to execute each rd instruction. As many successive rd instructions are necessary as there are lines of tape to be read. Any number of instructions other than in-out instructions may precede each rd.

Reading line-by-line by the block-transfer instruction. A bi instruction may take the place of a series of rd instructions. The address of the bi must be the initial address of the block of registers in ES to which the words will be transferred, and $\pm n$, the number of lines to be read, must be stored times 2^{-15} in AC. The time required to execute the block transfer is the same as the total time required to perform the rd instructions it replaces. Any sequence of rd and bi instructions may follow a single si.

Programming for word-by-word mode. Programming for reading in the word-by-word mode is as follows:

si r Selects the mechanical reader.

rd-- Reads the next three lines of tape (which must be punched in 556 form) and assembles them via IOR into a 16-digit word in AC, and clears IOR in preparation for receiving the next word. In this mode with the mechanical reader, the computer requires 318 milliseconds to execute each rd instruction. As many successive rd instructions are necessary as there are words to be read from tape. Any number of instructions other than in-out instructions may precede each rd.

Reading word-by-word by the block-transfer instruction. A bi instruction may take the place of a series of rd instructions. The address of the bi must be the initial address of the block of registers in ES to which the words will be transferred, and ± n, the number of words to be read, must be stored times 2^{-15} in AC. The time required to execute the block transfer is the same as the total time required to perform the rd instructions it replaces. Any sequence of rd and bi instructions may follow a single si.

Zero-length block transfer on bi. The use of a bi instruction calling for the transfer of a block zero words in length will result in one word being read but not transferred. The reading of the word actually is initiated by the preceding si instruction, hence one word is already read by the time the bi is ready to be performed. If the bi calls for the transfer of no words, the word already read is simply discarded.

si addresses for the mechanical reader. The si addresses for the mechanical tape reader (Flexowriter Input Unit #0) connected in "normal" fashion are as follows:

line-by-line:	si 200 (octal)	or	si 128 (decimal)
word-by-word:	si 202 (octal)	or	si 130 (decimal)

PHOTOELECTRIC TAPE READER

Action of the photoelectric reader. The photoelectric tape reader, abbreviated hereafter "PETR," "reads" the 6-digit binary combination punched in a line of paper tape and transmits it to the right-hand six digit places of IOR. In the line-by-line mode, each reading operation reads one line of tape and forms a word of which the left-hand ten digits are zero and the right-hand six digits correspond to the binary combination punched in the tape. In the word-by-word mode, each reading operation reads three lines of tape and assembles (by successive shifts left in IOR) a 16-digit word from the digits punched in the tape in 556 form. PETR is a free-running unit; that is, it runs free until stopped by an si instruction.

Substitution of mechanical reader for PETR. A switch will soon be available to connect the mechanical reader in place of the PETR whenever it is necessary to go instruction-by-instruction in a program which calls for the PETR. At other times the mechanical reader will be connected in its normal place, with its own selection addresses.

Programming for line-by-line mode. Programming for reading in the line-by-line mode is as follows:

si r — Starts PETR. Reads the first 6-digit character from tape into the right-hand six digits of IOR. This si instruction must not be followed by another si without at least one intervening rd or bi instruction.

rd-- Transfers contents of IOR to AC, then clears IOR in preparation for receiving the next character. The contents of digits 0-9 of AC will be zero, and the contents of digits 10-15 of AC will correspond to the binary combination read from tape. As many successive rd instructions are necessary as there are lines of tape to be read. If there are no intervening lines of blank tape, a 6-digit character will arrive at IOR every 7 milliseconds. The computer must execute an rd instruction often enough to extract a word from IOR and clear IOR before the next character arrives from the reader; otherwise an in-out alarm will result. Any instructions other than in-out instructions may precede each rd.

si-- Stops the reader. Since the reader passes about three-quarters of an inch of tape after it has been ordered to stop, it should not be stopped except where at least one and one-half inches of blank tape have been provided, otherwise information which coasts by the reading head will be lost. Any si instruction which has been assigned a function will stop the reader, but if the program does not require a specific si instruction, use the unique designation si 630 (octal) or si 408 (decimal).

Reading line-by-line by block-transfer instruction. A bi instruction may take the place of a series of rd instructions. The address of the bi must be the initial address of the block of registers in ES to which the words will be transferred, and $\pm$ n, the number of words to be read, must be stored times 2^{-15} in AC. The time required for the block transfer is the same as the total time required to perform the rd instructions it replaces. Any sequence of rd and bi instructions may follow a single si.

Programming for word-by-word mode. Programming for reading in the word-by-word mode is as follows:

si r Starts PETR. Reads the next three lines of tape (which must be punched in 556 form) and assembles them into a 16-digit word in IOR. This si instruction must not be followed by another si without at least one intervening rd or bi instruction.

rd-- Transfers contents of IOR to AC, then clears IOR in preparation for receiving the next character. The contents of AC will correspond to the 16-digit word originally punched on tape. As many successive rd instructions are necessary as there are words to be read from tape. If there are no intervening lines of blank tape, a 6-digit character will arrive in IOR every 7 milliseconds. The computer must execute an rd instruction often enough to extract a word from IOR and clear IOR before the next character arrives from the reader; otherwise an in-out alarm will result. Any instructions other than in-out instructions may precede each rd.

si-- Stops the reader. Since the reader passes about three-quarters of an inch of tape after it has been ordered to stop, it should not be stopped except where at least one and one-half inches of blank tape have been provided, otherwise information which coasts by the reading head will be lost. Any si instruction which has been assigned a function will stop the reader, but if the program does not require a specific si instruction, use the unique designation si 630 (octal) or si 408 (decimal).

Reading word-by-word by block-transfer instruction. A bi instruction may take the place of a series of rd instructions. The address of the bi must be the initial address of the block of registers in ES to which the words will be transferred, and $\pm$ n, the number of words to be read, must be stored times 2^{-15} in AC. The time required for the block transfer is the same as the total time required to perform the rd instructions it replaces. Any sequence of rd and bi instructions may follow a single si.

Zero-length block transfer on bi. The use of a bi instruction calling for the transfer of a block zero words in length will result in one word being read but not transferred. The reading of the word actually is initiated by the preceding si instruction, hence one word is already read by the time the bi is ready to be performed. If the bi calls for the transfer of no words, the word already read is simply discarded.

si addresses for the photoelectric reader. The si addresses for the photoelectric reader (Flexowriter Input Unit #1) are as follows:

line-by-line:	si 211 (octal)	or si 137 (decimal)
word-by-word:	si 213 (octal)	or si 139 (decimal)

PRINTERS

Action of the printer. Each character to be printed or machine function to be performed (for example, carriage return) requires that the computer send to the printer a 6-digit binary character from the left-hand six digit places of AC or of a storage register. Each key on the printer is actuated by a unique code character. The printer utilizes only 51 of the 64 possible code combinations, and it will ignore without consequence the remaining combinations. The computer-controlled printers will also ignore the "stop" code.

The Flexowriter Code. The 6-digit code, known as the "FL" Flexowriter Code, is assigned arbitrarily by the manufacturer. The code is given in the accompanying tables. Table 1 is in alphanumerical sequence and Table 2 is in numerical sequence of binary code characters.

Programming for printer operation. The printing of alphanumerical characters and the performance of machine functions is accomplished by the following sequence of instructions:

si t Selects the printer designated by the address t. The printer will remain selected until the next si instruction is executed.

rc— Actuates the printer key corresponding to the 6-digit code character contained in digits 0-5 of AC. A time (listed below) equal to that required for the printer to respond to the most recent character must elapse before the computer can perform the next in-out instruction. An rc instruction is required for each character to be printed or machine function to be performed. As many rc instructions as necessary may be used before the next si instruction. Any number of instructions other than in-out instructions may precede each rc.

Printing via the block-transfer instruction. If the Flexowriter codes for a group of characters to be printed are stored in sequence and in the left-hand 6-digit places in a block of consecutive registers, a bo instruction may be substituted for a series of rc instructions. The address of the bo must be the initial address of the block of registers, and $\pm$ n, the number of registers in the block, must be stored times 2^{-15} in AC at the time the bo instruction is executed. The time required for the block transfer to the printer will be the same as the total time required to execute the rc instructions it replaces. Any sequence of rc and bo instructions may follow a single si.

Printer response times. The approximate times required for the printer to carry out various processes are listed below:

print any alphanumerical character or symbol, space, color change, upper and lower case shifts	.135 milliseconds
back space.	.180 milliseconds
tabulation and carriage return. . .	.200 to 900 milliseconds

si addresses for printers. The si addresses for the printers are as follows: UNIT NO. 3 NOT YET AVAILABLE

Flexo Output Unit #1 (Room 222):	si 215 (octal) or si 141 (decimal)
Flexo Output Unit #2:	si 225 (octal) or si 149 (decimal)
Flexo Output Unit #3:	si 235 (octal) or si 157 (decimal)

PUNCH

Action of the punch. Each line of digits to be punched on tape is transmitted to the punch from the left-hand 6 digit-places of IOR. In the line-by-line mode, each recording operation punches one line of tape corresponding to the contents of digits 0-5 of IOR. In the word-by-word mode, each recording operation punches three lines of tape in 556 form (by successive shifts left in IOR) corresponding to the word in IOR.

Programming for line-by-line mode. Programming for punching in the line-by-line mode is as follows:

si p Selects the punch, and prepares to punch or suppress the 7th hole, according to the address p. The punch will remain selected until the next si instruction is executed.

rc— Punches in one line on paper tape the 6-digit binary combination corresponding to the contents of digits 0-5 of AC. The 7th hole position is automatically punched, or not, according to the mode determined by the most recent si instruction. 93 milliseconds must elapse before the computer can perform the next in-out instruction. An rc is required for each line of tape to be punched. As many rc instructions as necessary may be used before the next si instruction. Any number of instructions other than in-out instructions may precede each rc.

Punching line-by-line by block-transfer instruction. If the characters to be punched are stored in sequence and in the left-hand 6 digit-places in a block of consecutive storage registers, a bo instruction may be substituted for a series of rc instructions. The address of the bo must be the initial address of the block of registers, and $\pm$ n, the number of registers in the block, must be stored times 2^{-15} in AC at the time the bo instruction is executed. The time required for the block transfer to the punch will be the same as the total time required to execute the rc instructions it replaces. Any sequence of rc and bo instructions may follow a single si.

Programming for word-by-word mode. Programming for punching in the word-by-word mode is as follows:

si p — Selects the punch, and prepares to punch or suppress the 7th hole, according to the address p. The punch will remain selected until the next si instruction is executed.

rc-- — Punches in 556 form (in three lines) the 16-digit binary combination corresponding to the contents of AC. The 7th hole position is automatically punched, or not, according to the mode determined by the most recent si instruction. 280 milliseconds must elapse before the computer can perform the next in-out instruction. An rc is required for each word to be punched in three lines on tape. As many rc instructions as necessary may be used before the next si instruction Any number of instructions other than in-out instructions may precede each rc.

Punching word-by-word by block-transfer instruction. If the words to be punched are stored in sequence in a block of consecutive storage registers, a bo instruction may be substituted for a series of rc instructions. The address of the bo must be the initial address of the block of registers, and $\pm$ n, the number of registers in the block, must be stored times 2^{-15} in AC at the time the bo instruction is executed. The time required for the block transfer to the punch will be the same as the total time required to execute the rc instructions it replaces. Any sequence of rc and bo instructions may follow a single si.

si addresses for punch. The si addresses for the punch are as follows:

line-by-line normal:	si 205 (octal) or si 133 (decimal)
line-by-line, 7th hole suppressed:	si 204 (octal) or si 132 (decimal)
word-by-word, normal:	si 207 (octal) or si 135 (decimal)
word-by-word, 7th hole suppressed:	si 206 (octal) or si 134 (decimal)

SCOPES

Selection of scope displays. The computer program will specify by the address of the si instruction a particular "scope intensification line." Any scope connected to the selected line will display a point on each of the succeeding display instructions. A bank of toggle switches at each scope unit permits the connection of the scope to any one or more of the 16 intensification lines.

Scope deflection. The left-hand 11 digits of AC at the time a display instruction is given determine the direction and amount of deflection. The positive direction of horizontal deflection is to the right and positive vertical deflection is upward. The value $1 - 2^{-10}$ or its negative will produce the maximum deflection.

Display of a single point. The display of a single point is programmed by the following sequence of instructions:

si s — Selects the scope intensification line designated by the address s. Sets the horizontal deflection of all scopes to a value corresponding to the contents of digits 0-10 of AC.

rc-- — Sets the vertical deflection of all scopes to a value corresponding to the contents of digits 0-10 of AC. Intensifies a point on all scopes which are connected to the intensification line selected by the above si instruction. 100 microseconds will elapse before the computer performs the next in-out instruction. Any number of instructions other than in-out instructions may precede each rc. Each point to be displayed is programmed in a similar manner.

Display of vertical lines. The horizontal deflection is set up by any si instruction (including those which do not refer to scopes) and remains unchanged until a new si instruction is executed. Similarly, the vertical deflection is set up by any rc instruction (while a scope line is selected) and remains unchanged until a new rc instruction is executed. Hence a vertical line may be displayed simply by a single si to set the horizontal deflection followed by a succession of rc instructions to set up the vertical deflections and display the individual points on the vertical line. After each rc, 100 microseconds must elapse before the computer can perform the next in-out instruction.

si addresses for scope lines. The si addresses designating scope intensification lines are as follows:

scope line	octal address	decimal address	scope line	octal address	decimal address
0	si 600	si 384	8	si 610	si 392
1	si 601	si 385	9	si 611	si 393
2	si 602	si 386	10	si 612	si 394
3	si 603	si 387	11	si 613	si 395
4	si 604	si 388	12	si 614	si 396
5	si 605	si 389	13	si 615	si 397
6	si 606	si 390	14	si 616	si 398
7	si 607	si 391	15	si 617	si 399

TABLE 1. THE "FL" FLEXOWRITER CODE

Alphanumerical Sequence

Lower Case	Upper Case	Character 123456	Decimal Value	Octal Value	Lower Case	Upper Case	Character 123456	Decimal Value	Octal Value
a	A	000110	6	6	0	0	111110	62	76
b	B	110010	50	62	1	1	010101	21	25
c	C	011100	28	34	2	2	001111	15	17
d	D	010010	18	22	3	3	000111	7	7
e	E	000010	2	2	4	4	001011	11	13
f	F	011010	26	32	5	5	010011	19	23
g	G	110100	52	64	6	6	011011	27	33
h	H	101000	40	50	7	7	010111	23	27
i	I	001100	12	14	8	8	000011	3	3
j	J	010110	22	26	9	9	110110	54	66
k	K	011110	30	36	|	_	000101	5	5
l	L	100100	36	44	space bar		001000	8	10
m	M	111000	56	70	=	:	001001	9	11
n	N	011000	24	30	+	/	001101	13	15
o	O	110000	48	60	color change		010000	16	20
p	P	101100	44	54	.	)	010001	17	21
q	Q	101110	46	56	,	(	011001	25	31
r	R	010100	20	24	-	-	011101	29	35
s	S	001010	10	12	back space		100011	35	43
t	T	100000	32	40	tabulation		100101	37	45
u	U	001110	14	16	carr. return		101001	41	51
v	V	111100	60	74	stop		110001	49	61
w	W	100110	38	46	upper case		111001	57	71
x	X	111010	58	72	lower case		111101	61	75
y	Y	101010	42	52	nullify		111111	63	77
z	Z	100010	34	42					

TABLE 2. THE "FL" FLEXOWRITER CODE

Binary Numerical Sequence

Decimal Value	Octal Value	Character 123456	Lower Case	Upper Case	Decimal Value	Octal Value	Character 123456	Lower Case	Upper Case
0	0	000000	not used		32	40	100000	t	T
1	1	000001	not used		33	41	100001	not used	
2	2	000010	e	E	34	42	100010	z	Z
3	3	000011	8	8	35	43	100011	back space	
4	4	000100	not used		36	44	100100	l	L
5	5	000101	|	_	37	45	100101	tabulation	
6	6	000110	a	A	38	46	100110	w	W
7	7	000111	3	3	39	47	100111	not used	
8	10	001000	space bar		40	50	101000	h	H
9	11	001001	=	?	41	51	101001	carr. return	
10	12	001010	s	S	42	52	101010	y	Y
11	13	001011	4	4	43	53	101011	not used	
12	14	001100	i	I	44	54	101100	p	P
13	15	001101	+	/	45	55	101101	not used	
14	16	001110	u	U	46	56	101110	q	Q
15	17	001111	2	2	47	57	101111	not used	
16	20	010000	color change		48	60	110000	o	O
17	21	010001	.	)	49	61	110001	stop	
18	22	010010	d	D	50	62	110010	b	B
19	23	010011	5	5	51	63	110011	not used	
20	24	010100	r	R	52	64	110100	g	G
21	25	010101	1	1	53	65	110101	not used	
22	26	010110	j	J	54	66	110110	9	9
23	27	010111	7	7	55	67	110111	not used	
24	30	011000	n	N	56	70	111000	m	M
25	31	011001	,	(	57	71	111001	upper case	
26	32	011010	f	F	58	72	111010	x	X
27	33	011011	6	6	59	73	111011	not used	
28	34	011100	c	C	60	74	111100	v	V
29	35	011101	-	-	61	75	111101	lower case	
30	36	011110	k	K	62	76	111110	0	0
31	37	011111	not used		63	77	111111	nullify	

REFERENCE LIST OF si ADDRESSES

All of the presently assigned si addresses are listed below in numerical sequence, together with very brief notations of their functions.

AD = Auxiliary magnetic drum

BD = Buffer magnetic drum

CA = Camera

FI = Flexowriter input unit

FO = Flexowriter output unit

MT = Magnetic tape

SC = Scope intensification line

ST = Stop

Address Oct.	(Dec.)	Unit Class	Ser.	Mode	Address Oct.	(Dec.)	Unit Class	Ser.	Mode
0	(0)	ST	-	stop comp.	137	(95)	MT	3	rcd rev
1	(1)	ST	-	cond. stop	200	(128)	FI	0	l-by-l
100	(64)	MT	0	rer fwd	202	(130)	FI	0	w-by-w
101	(65)	MT	0	rer rev	204	(132)	FO	0	l-by-l;supp 7
102	(66)	MT	0	read fwd	205	(133)	FO	0	l-by-l;no sup
103	(67)	MT	0	read rev	206	(134)	FO	0	w-by-w;supp 7
106	(70)	MT	0	rcd fwd	207	(135)	FO	0	w-by-w;no sup
107	(71)	MT	0	rcd rev	211	(137)	FI	1	l-by-l
110	(72)	MT	1	rer fwd	213	(139)	FI	1	w-by-w
111	(73)	MT	1	rer rev	215	(141)	FO	1	-
112	(74)	MT	1	read fwd	220	(144)	FI	2	l-by-l
113	(75)	MT	1	read rev	221	(145)	FI	2	b-by-l
116	(78)	MT	1	rec fwd	222	(146)	FI	2	w-by-w
117	(79)	MT	1	rec rev	223	(147)	FI	2	b-by-w
120	(80)	MT	2	rer fwd	225	(149)	FO	2	-
121	(81)	MT	2	rer rev	230	(152)	FI	3	l-by-l
122	(82)	MT	2	read fwd	231	(153)	FI	3	b-by-l
123	(83)	MT	2	read rev	232	(154)	FI	3	w-by-w
126	(86)	MT	2	rcd fwd	233	(155)	FI	3	b-by-w
127	(87)	MT	2	rcd rev	235	(157)	FO	3	-
130	(88)	MT	3	rer fwd	500	(320)	CA	-	-
131	(89)	MT	3	rer rev	600	(384)	SC	0	-
132	(90)	MT	3	read fwd	601	(385)	SC	1	-
133	(91)	MT	3	read rev	602	(386)	SC	2	-
136	(94)	MT	3	rcd fwd	603	(387)	SC	3	-

Address Oct.(Dec.)	Unit Class	Ser.	Mode	Address Oct.(Dec.)	Unit Class	Ser.	Mode
604 (388)	SC	4	-	617 (399)	SC	15	-
605 (389)	SC	5	-	630 (408)	ST	-	unit stop
606 (390)	SC	6	-	700 (448)	AD	-	rd: -
607 (391)	SC	7	-	701 (449)	AD	-	rd: new gp
610 (392)	SC	8	-	702 (450)	AD	-	rd: new addr
611 (393)	SC	9	-	703 (451)	AD	-	rd: gp& addr
612 (394)	SC	10	-	704 (452)	AD	-	rc: -
613 (395)	SC	11	-	705 (453)	AD	-	rc: new gp
614 (396)	SC	12	-	706 (454)	AD	-	rc: new addr
615 (397)	SC	13	-	707 (455)	AD	-	rc: gp& addr
616 (398)	SC	14	-				

REFERENCES

Further details concerning the WWI system may be found in the following documents:

E-466 Operation of the In-Out Element
E-469 Display Facilities in the Final In-Out System
E-473 Input Program, September, 1952
E-479 Basic Conversion Program, September, 1952
E-482 Operation of Magnetic Tape Units
E-499 Operation of Block Transfer Orders

M-1516 Use of the Interim Magnetic Tape Print-Out Equipment
M-1591 Special Input Program, T-2000
M-1624-1 Short Guide to Coding and Whirlwind I Operation Code

R-127-1 Whirlwind I Block Diagrams
R-180 Functional Description of the Whirlwind I Computer

being written { Paper Tape Facilities
Auxiliary Magnetic Drum
Buffer Magnetic Drum

Signed: P R Bagley
P. R. Bagley

Approved: C. R. Wieser
C. R. Wieser

6345
Engineering Note E-516

Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: COMPREHENSIVE SYSTEM OF SERVICE ROUTINES

To: S. & E. C. Group and Group 61

From: H. Uchiyamada

Date: December 17, 1952

Abstract: The Comprehensive System of Service Routines provides for conversion by WWI to binary form from Flexowriter-coded perforated tapes prepared according to conventions which have been chosen to facilitate the task of coding programs for WWI. In addition to straightforward conversion of function letters and decimal addresses, the Comprehensive System (CS) provides for (1) use of floating addresses for which assignment of final storage locations is made by the computer (this has the important advantage of permitting insertions and deletions of instructions without extensive renumbering in the program) (2) automatic selection of Input/Output and Programmed Arithmetic (PA) interpretive subroutines which eliminates to a considerable degree the time wasted in handling tapes and the possible errors involved (3) automatic cycle control (patterned after the Manchester B-tube) available within the PA routines which will reduce the need for using uninterpreted WWI instructions within an interpreted program and which will generally facilitate programming (4) the handling of generalized decimal numbers (gdn) of the form $\pm z \times 2^{\gamma i} \times 10^{\delta i}$ which enables the programmer to express numerical data in whatever form is best suited to the particular calculation.

Comment: The author has acted, in the main, as editor of this E-note. Sections have been written by Eric Mutch, John Frankovich, Frank Helwig and Edwin Kopley. The CS as a whole represents the work of many people in the Scientific and Engineering Computation Group. This note is intended as a reference manual, not as an introductory presentation of programming techniques and conventions, which will be available later.

Index for Engineering Note E-516

Table of Contents

I. Introduction

A program is an ordered sequence of words, written with the intention of having it typed on paper tape in the (new) Flexo-code and inserted in WWI by the intermediary of the Comprehensive Conversion Program (CCP).

A word is a finite ordered sequence of syllables. Normally all the syllables forming a word must be separated by a plus or minus sign, but plus signs may be omitted wherever there is no danger of ambiguity. Details of this and other rules governing the assembly of syllables will be given in Section II. Any word made up of one or more syllables must be followed by a terminating character. There are four possible terminating characters giving four possible ways in which the conversion program will treat the word. These terminating characters and their functions will be described in Section II. A given word is meaningful, from the conversion program's viewpoint only if the words, or syllables, respectively, are chosen in a manner not contrary to any of the rules. Any combination of words or syllables not forbidden by the rules will be accepted by the conversion program. Special words will be described later in this section. A single length word is represented in WWI by a 16 binary digit array.

Syllables may be divided into two classes, namely, constant syllables and parametric syllables. The class of constant syllables includes operations, integers and octal numbers. The class of parametric syllables includes preset parameters, relative address, temporary storage, and floating address.

Constant Syllables

Operations are of two kinds, namely, WWI orders and interpreted orders. The WWI operations or orders (ca, cs,---slh, slr, srh, etc.) are described in detail in M-1624. The interpreted orders (ica, ics, etc.) will be found listed with their functions under Section III on Programmed Arithmetic (PA).

Integers may be positive or negative decimal integers or the literal integers, b or c. The decimal integers used are 0, 1, 2,---, 32767 with an implicit factor of 2^{-15} and no decimal point. The literal integers serve a specific purpose which will be described under Section III on PA.

Octal numbers are of the form $d_o.d_1d_2d_3d_4d_5$ where d_o, the sign digit, is either 0 or 1 and where $d_1...d_5$ are the octal digits having one of the values 0, 1, 2, 3, 4, 5, 6, 7. A 1. indicates the start of a negative octal number, the remaining digits being the sevens complement of the absolute magnitude of the number. If an octal number occurs as a syllable in a word, it must always be the first syllable, i.e. only one octal number syllable can occur in any word. An example of a positive octal number is 0.04573. In order to obtain the negative of this number one must change the 0. to 1. and also get the sevens complement of the five octal digits following the sign digit; thus the negative becomes 1.73204.

Parametric Syllables

Preset parameters are of the form $\alpha_1\alpha_2\#$, where α_1 is u, p or z depending on whether the parameter is of the type universal (assigned particular meaning and never used for anything else), personal (can be used by anyone to mean anything desired) or subroutine (for parameter in subroutines) respectively; α_2 is any letter of the alphabet except o and 1; and # is any decimal number of the form 1, 2, 3, ..., 255, with initial zero suppressed.

A relative address is one which is used for writing instructions within a subroutine or within any block of instructions with addresses relative to the start of the block (that is, as if the block started in register zero). Such relative addresses are obtained by including an "r" in the address of the instruction, e.g. ca 35r (which consists of the three syllables ca + 35 + r).

The single lower case letter "t" indicates the zero-th register of a block of temporary storage. Its value must be assigned in the same way as for a preset parameter. See Section II on preset parameters.

A floating address is one which enables a programmer to write his instructions so that they refer to the words of his program and not to the locations of those words in storage.

Special Words+

The following are different groups of special words:

Program title words: TAPE, MOD, PARAMETER, suffixed by additional information
Output words:* TOA, FOR etc. (See Section IV under Input/Output) perhaps suffixed by additional information and perhaps preceded by an i
Number system indicators: MULTIPLE, SINGLE, (m,n) where m and n are integral numbers. For details see Section III under P.A.
Entry to and exit from PA routines: IN,* OUT*
Word called a fence: |||...||| (i.e. 25 vertical bars)
Words: DITTO, START AT, i START AT, the last two of which are suffixed by the starting address
Denial of need for a PA interpretive routine: NOTPA
Special words which are ignored: LSR (library of subroutines), END OF SUBROUTINE

The number system (m,n) indicates a number, m-binary digits long with n the number of binary digits in the exponent of 2, and the number is of the form $z = x \cdot 2^y$ where x is a m binary digit number and y is a n binary digit number. A single length number with a fixed point would be a (15,0) number. An example of a single length floating point number would be (15,15). An example of a multiple-length number with a fixed point would be (30,0). An example of a multiple-length number with a floating point is (30-j,j) where $1 \leq j \leq 14$. For a detailed description of multiple length numbers see Section III on PA. Single length numbers with fixed point are adequately handled by the WWI operations. Multiple-length and single length floating point numbers are handled by the interpretive operations for which see Section III on PA.

+ All special Words must be terminated by a tab or carriage return.

* Only these Special Words occupy storage registers.

II. Programming

Terminating Characters and their Functions

Any word made up of one or more syllables must be followed by a terminating character. There are four possible terminating characters giving four possible ways in which the conversion program will treat the word.

Tab (--→|) or Carriage Return (↲)	=	"word to be stored" indicator. This causes the word to be stored in the register determined by the current address indicator, unless the word is <u>preceded</u> by an equals sign, for which see below.
Vertical bar (|)	=	address assignment indicator. This causes the current address indicator to be set to the value corresponding to the preceding word. Thus the following word to be stored will be placed in the register thus indicated regardless of consecutivity.
Comma (,)	=	floating and/or relative address assignment indicator (see paragraphs on relative and floating addresses below)
Equals sign (=)	=	parameter assignment indicator. This causes the parameter immediately preceding the equals sign to be set to the value following it (which will be terminated by <u>a</u> tab or carriage return). If no word follows the equals sign (i.e. if the next symbol is a tab or carriage return) the parameter will be assigned the value zero.

Absolute Addresses

At the start of a program and at any point thereafter a decimal integer followed by a vertical bar (e.g. 96|) indicates the location into which the next word is to be placed. In the absence of any further indication words will be stored consecutively; in the absence of an initial indication words will be stored consecutively starting in register 32 (decimal). Note that this conversion program does <u>not</u> permit the use of octal <u>addresses</u>.

Relative Addresses

Instructions within a subroutine or within any block of instructions may be written with addresses relative to the start of the block. Such relative addresses are obtained by including an "r" in the address. This causes the content of a special register known as the relative address indicator (r.a.i) to be added to the instruction during conversion. The r.a.i. may be set at the beginning of the block by the symbols 0r, which cause it to be set to the value of the current address--i.e. the address into which the next word will be put. If an integer n precedes the letter r instead of the zero the r.a.i. will be set to a value equal to the current address minus the integer n; e.g. if the current address is 90 the symbols 5r, will set the r.a.i. to 85. Note that a comma following a floating address will also set the r.a.i. (For details see the following paragraph on floating addresses.) The current address indicator may be set to a desired relative value at any point in a program by punching that value followed by the letter r and a vertical bar; e.g. 35r| will cause the next word to be stored in 35r regardless of consecutivity.

Floating Addresses

As already stated a floating address system is designed to enable a programmer to write his instructions so that they refer to the words of his program and not to the locations of those words in storage.

For example, consider the following set of instructions with fixed addresses:

```
32| ca 41
33| ad 100
34| ts 41
35| ca 42
36| ad 100
37| ts 42
38| ca 43
39| ad 100
40| ts 43
41| ca 101
42| mr 102
43| ts 103
44| cp 32
```

Seven of these instructions refer to the locations of other instructions within the group. If any instructions (or words) were to be added to or deleted from this set, a considerable amount of renumbering would be necessary, in general. A floating address system removes the need for this, by labelling each word to which reference is made by a floating address label. The floating address is of the form α#, where α is any lower-case letter of the alphabet except o and l, and where # is any integer of the form 1, 2, 3,..., 255 with initial zeros suppressed. This floating address, without the comma, is then used as the address section of any instruction which is to refer to the word so labelled. Thus the above program might become:

```
f3,  ca m9
     ad 100
     ts m9
     ca h5
     ad 100
     ts h5
     ca b2
     ad 100
     ts b2
m9,  ca 101
h5,  mr 102
b2,  ts 103
     cp f3
```

Note that floating addresses may be used in any order and that words referring to a floating address may occur either earlier or later than the word labelled by the corresponding floating address. Thus insertions into or deletions from such a program may be made without any renumbering or any alterations to the existing words.

The current address is the address of the register into which the next word will go. If the next word occupies several registers, then this is the address of the first register of the word. When a floating address is read, the conversion program records it, together with the current address, as an entry in a special table. The word following is then stored away in the normal manner--i.e. in the location specified by the current address. At a later stage in the conversion--when all the information to be converted has been read--all words referring to floating addresses have added to them the relevant entries from this table. The letter and number(s) forming a floating address may be chosen at will (within the limits already set on floating address labels) but care must be taken that the sum over all letters of the maximum numbers used for each letter does not exceed 255 (e.g. if only floating addresses a3, a17, d9, x31, x100 and z5 were used in a given program, the condition would be satisfied because 17 + 9 + 100 + 5 = 131 < 256). The comma following a floating address serves also as a reference for relative addresses which follow, by setting the relative address indicator to the value of the current address indicator (c.a.i.).

Examples

```
(Absolute address)--->  34|  ca g7 <--- (floating)
(Floating address)--->  g7,  sp 73 <--- (absolute)
                             ts 2r <--- (relative)
(Relative address)--->  4r|  si 7a2<--- (floating)
(Floating address)--->  a2,  + 3
                             -.0055
```

The words in this example would be converted to:

```
34| ca 35
35| sp 73
36| ts 37
37|  --       (contains +0)
38|  --       (contains +0)
39| si 47
40| +3
41| -.0055
```

A word not itself labelled by a floating address may be referred to in floating address fashion relative to a preceding or following word which has a floating address. Thus the word "si 7a2" in the above example refers to the seventh word after the word with the floating address "a2". The same word could be referred to by the floating address 12g7. It is of interest to note in this respect that a2 = 5g7 and g7 = -5a2 (+ is implicit between -5 and a2). Note that no additions or deletions may be made between a word referred to by such means and the word carrying the floating address without a certain amount of renumbering. Corrections may be made to words already labelled by floating addresses, and to the words following them, by preceding each corrected word by the relevant floating address terminated by a vertical bar instead of by a comma, e.g.

```
 g7| sp 72
1a2| -.0065
```

would amend the second and last orders of the above example.

Preset Parameters

The three classes of preset parameters, (universal, personal, and subroutine) have already been mentioned in the introduction. A preset parameter consists of two lower case letters followed by a decimal integer less than 256 but greater than zero. The second letter may be any letter other than o or 1. The first letter is used to distinguish the three classes of parameters; i.e. u for universal parameter, p for personal parameter and z for subroutine parameter. (Note that the letter s could not be used to indicate a subroutine parameter owing to the fact that the conversion program would confuse an sa parameter with an sa WWI operation, etc.) Care must be taken that the sum over all parameter letter pairs of the maximum numbers used for each letter pair must not exceed 40 (e.g. if only parameters pa2, za5, za7, pd7, zg4, ug6, ug8 and zz11 were used in a given program, the condition would be satisfied because 2+7+7+4+8+11 = 39<41. If the single lower case letter "t" were used anywhere in the program one more would have to be added to the sum which must not exceed 40. In the example given above if a "t" was used anywhere in the program the condition would still be satisfied since 39+1 = 40<41). A value may be assigned to a preset parameter by a word consisting of the parameter followed by an equals sign and the value to be assigned terminated by a tab or carriage return. After assignment any number of parameters may be added to or subtracted from any word. Preset parameters may be assigned values which depend on other preset parameters. They may also be assigned values which depend on floating addresses. Subroutine library tapes will begin with the symbols ↑LSR↓ followed by the catalog number and the title of the subroutine. After the title the various parameters needed by the subroutine will be listed, each followed by an equals sign, a stop character and a tab or carriage return. Thus, when copying a library tape onto a program tape, parameter values may be inserted by hand each time the Flexowriter stops. If the value of any parameter is zero, nothing need be inserted and it is only necessary to restart the Flexowriter.

Examples, illustrating point made above on preset parameters, follow:

um3=+3	universal parameter
ca71+um3	word becomes ca74
pm3=0.00020	personal parameter
sℓr+pm3	word becomes sℓr16
zm3=rs0	subroutine parameter
zs2=pm3+um3	subroutine parameter
cs7-zm3	word becomes ca7
sℓrzs2	word becomes sℓr19

The value of a temporary storage parameter is assigned in the same way as for a preset parameter; e.g. t=190 or t=pn3 or t=f3. To refer to a temporary storage register in an instruction, the fourth for example, the symbols 3t are used; e.g. ca3t.

Rules for forming words out of syllables

1) No other syllable may occur in a generalized decimal number but the generalized decimal number and the terminating character. A generalized decimal number is of the form $\pm d_1 d_2 {-}{-}{-} d_k . d_{k+1} {-}{-}{-} d_m x 2^{\gamma_i} x 10^{\delta_i}$ where $0 \leq k \leq m \leq 18$ and γ_i and δ_i are integers, signed if negative, otherwise not signed, and such that the final number is restricted by the number system indicated by the programmer.

2) Only one octal number syllable can occur in any word, i.e. the octal number syllable must always be the first syllable.

3) A word, address assignment, parameter assignment, or floating address assignment can be found by the sum formed by "special add" of successive syllables contained in them. (Note that the value thus obtained depends upon the sequence in which the syllables are written.)

4) A plus or minus sign preceding a syllable, indicates that the value of the syllable is to be multiplied by +1 or -1 respectively before being added into the word value.

5) A plus or minus sign should always be used when there is an ambiguity in the meaning of a syllable or pair of syllables. (For examples of ambiguities see list of ambiguities.)

6) Rules concerning the use of single letters:

   i) t is considered exactly as a preset parameter, and is usually used to indicate the zero-th register of a block of temporary storage registers.

   ii) b has the value of the address of the buffer storage register in PA routines.

   iii) c adds a value to the word sufficient to change an interpreted instruction into a cycle count interpreted instruction and should be used only with its, iex, ica, ics, iad, isa, imr, idv, isp.

   iv) r is the relative address and is given a value each time a comma occurs.

   r = current address - stem.

   A word containing the terminal character "," and at most one floating address syllable and one integer syllable, is called a <u>floating address assignment</u>, e.g. "7g9,". The stem of a floating address assignment or parameter assignment is the integer (if it exists) which precedes the lower case letter in the floating address assignment or parameter assignment. In the example above (7g9) 7 is the stem.

7) Whenever a "," occurs, the floating address in the floating address assignment is set equal to the current address less the stem.

8) Whenever an "=" occurs, the parameter in the parameter assignment is set equal to the following word less the stem.

9) A starting address word consists of a START AT or i START AT, suffixed by any word, i.e. the starting address including a tab or carriage return.

Rules for forming a program out of words

1) A fence (at least 25 vertical bars) must occur initially and terminally in a program.

2) The initial word of a program will go into the initial register of storage (i.e. register 32) and successive words will go into successive registers unless an address assignment is made. An address assignment consists of constant (except octal numbers) and/or parametric syllable(s) followed by a vertical bar. A definite address is one where the value is explicitly known. An indefinite address is one which depends upon floating addresses or parameters, i.e. only implicitly known. The current address is said to be indefinite, following an indefinite address assignment, and is said to be definite as soon as a definite address assignment is made; but is called indefinite again if another indefinite address assignment is made. If an address assignment (definite or indefinite) is made, the word following such an address will go into the register indicated by the address assignment. (Note that in the case of a definite address assignment the current address is given directly, whereas in the case of an indefinite address assignment the current address may be found indirectly). If no initial address assignment is made, the current address is considered to be definite.

3) No floating address assignments may be made while the current address is indefinite.

4) The special word "i START AT" must occur just before the last word.

5) Titular special words usually occur immediately after the initial fence, but may occur anywhere.

6) A fence must occur both before and after any output or titular special word.

7) A word containing the terminal character "=" and at most one parameter syllable and one integer syllable, is called a parameter assignment, e.g. "5pc10=". The word following a parameter assignment, less the stem of the parameter assignment, is the value given to the indicated parameter. For example, if the word following the above parameter assignment is 7 (i.e. 5pc10=7), then pc10=7-5, which says in effect that the parameter is assigned the value 2.

8) A generalized decimal number will be converted to the number system indicated by the last preceding number system indicator; i.e. SINGLE means that the number will be converted to the (15,0) system, and MULTIPLE to the system determined by the preceding (m,n), otherwise to (15,0).

9) Words occupy one register of storage, generalized decimal numbers (m+n)/15 registers, output special words and IN and OUT one register each. No other kinds of words occupy any register of storage.

10) The special word DITTO, followed by a tab (--->|) or carriage return (⤶) must be preceded by a word or generalized decimal number and followed by an address assignment. The word or generalized decimal number preceding the DITTO will be ditto'd up to but not including the address

indicated; e.g. 131| DITTO, would cause the word or generalized decimal number preceding DITTO, to be stored in the registers up to but <u>not</u> including 131.

11) The special words LSR--, END OF SUBROUTINE--, OCTAL--, and DECIMAL--, including all words that follow these special words up to tab or carriage return are ignored by the conversion program. (One result is that octal addresses are not permitted).

12) A parameter must be assigned before it is used.

<u>List of common ambiguities</u>

cℓc	vs	clc,	write cl+c	if the floating address cl is meant
sℓc	vs	slc,	write sl+c	" " " " sl " "
aol	vs	a01,	write al	i.e. initial zeros must be suppressed
bol	vs	b01,	write bl	
sol	vs	s01,	write sl	

Some ambiguities of the conversion program are not obvious to the programmer. In particular, single letters may not be written without preceding and following it by a plus or minus sign; e.g. not tca, but ±t±ca
not imrtc, but imr±t±c
To avoid difficulties always use a + between two syllables. The + may always be omitted between function letters and the next syllable.

III. Programmed Arithmetic

Number Systems and Preliminary Definitions

In the following discussion we shall frequently refer to (m,n) numbers where (m,n) = (30,0) or (15,15) or (30-j,j), j = 1, 2,..., 14. We now define these numbers.

(i) A (30,0) number is a 30 digit binary number with the binary point at the left-hand end of the number. Such numbers are stored in two consecutive registers, say q and q+1, with the most significant part of the number being contained in register q. We shall refer to this number as "the (30,0) number contained in "location" q."

(ii) A (15,15) number is a number which has been expressed in the form

$$z = x \cdot 2^y$$

where x is a 15 binary digit number such that $1/2 \leq x < 1$ or x=0 and y is a 15 binary digit integer. Such numbers are stored in two consecutive registers, say q and q+1. The number x is stored in register q and the number y is stored in register q+1. We shall refer to this number as the (15,15) number contained in "location" q.

(iii) A (30-j,j), j = 1, 2,..., 14, number is a number which has been expressed in the form

$$z = x \cdot 2^y$$

where x is a 30-j binary digit number such that $1/2 \leq x < 1$ or x=0 and y is a j-digit binary integer. Such numbers are stored in two consecutive registers, say q and q+1. The 15 most significant digits of x are stored in register q and the 15-j least significant digits of x are stored in the right-hand 15-j digits of register q+1. The integer y is stored in the left-hand j digits of register q+1. The sign digit of register q+1 refers to the sign of y. We shall refer to this number as the (30-j,j) number contained in "location" q.

On the basis of the above definitions it should be noted that ordinary calculations on WWI are in the (15,0) number system. (30,0) and (15,0) numbers shall be referred to as fixed (binary) point numbers. (15,15) and (30-j,j) numbers shall be referred to as floating (binary) point numbers.

Interpretive Subroutines

(m,n) interpretive subroutines shall mean a particular group of coded programs whose purpose is to facilitate computation using (m,n) numbers. These enable the programmer to write coded programs using (m,n) numbers which are in many ways analogous to ordinary WWI coded (15,0) programs. Such programs, when called into action, take "interpreted instructions" (more strictly, program parameters written as instructions) one at a time from consecuitve storage registers and perform the designated single address operations as defined by the interpreted instruction code. (For a complete list of interpretive operations and their functions see end of Section III.)

A <u>multiple register accumulator</u> (MRA) is used in place of the AC in many interpreted instructions. The MRA is not a special register as is the AC, but rather is a group of 3 ordinary storage registers contained within the interpretive subroutine.

Entry to and Exit from Interpretive Subroutines

Entry to the interpretive subroutine is accomplished by means of the (15,0) word IN. This word is changed into a (15,0) sp instruction by the CS which transfers control from the program to the proper register in the interpretive subroutine. The instructions following the word IN are then performed as interpreted instructions, e.g.

```
32| IN
33| ica50 }  This program forms the sum of the (m,n) numbers in
34| iad52 }  locations 50 and 52
..  ...
```

Exit from the interpretive subroutine is accomplished by means of an interpreted instruction sp. In this particular case the interpreted instruction, sp, and the (15,0) WWI instruction, sp, have the same binary value. As an example we have

```
32| IN
33| ica50
34| iad52
35| sp 60  }
..  ...    }  (15,0) WWI operation is resumed at register 60
60| ca 100 }
..  ...
```

Since it is frequently desired to resume (15,0) WWI operation at the register following the interpreted sp the special word OUT has been included in the conversion vocabulary. If p is the register containing the word OUT, then the special word is converted to an sp(p+1). The previous example can now be written as

```
32| IN
33| ica50
34| iad60
35| OUT
36| ca 100  (15,0) WWI operation is resumed at register 36.
```

Generalized Decimal Numbers

Several words are included in the CS to facilitate the insertion of (m,n) numbers into the computer.

The most general decimal number which can be converted and stored by the CS has the form

$$\pm\, d_1 d_2 \ldots d_k . d_{k+1} \ldots d_m \times 2^{\gamma} \times 10^{\delta}$$

Such numbers are first converted by the CCP into the integer

$$\pm\, d_1 d_2 \ldots d_k \ldots d_m .$$

The associated exponent of 2 is γ and the associated exponent of 10 is $\delta - m + k$. This result is then further processed in accordance with the last special word (m,n) which appears in the program. This special word causes the conversion program to convert all subsequent generalized decimal numbers into (m,n) numbers unless it is superseded by another special word (m_2,n_2). In the case of (30,0), (15,15) and (30-j,j) numbers the components of the number are stored in consecutive registers. The special word (15,0) gives us of course a single register number.

As an example, to store the (24,6) numbers 2 and 5 in consecutive locations write

```
      (24,6)
  32| +2.
  34| +5.     but 34| is not necessary.
```

It should be emphasized here that all generalized decimal numbers must contain at least a sign and a decimal point.

Two applications of the special word (m,n) are handled by the use of further special words.

The first of these is the special word SINGLE. All generalized decimal numbers, converted after this word appears in a program, are converted to (15,0) numbers.

The second of these is the special word MULTIPLE. All generalized decimal numbers, converted after this word appears in a program, are converted to (m_1,n_1) numbers, where (m_1,n_1) is the last special word (m,n) which appears on the tape. It should be noted that the word MULTIPLE in a tape will be redundant unless the special word SINGLE occurs between it and the last (m,n) on tape.

An example of the use of these words is

```
(24,6)
 32| +2.  }
 34| +5.  } Converted as (24,6) numbers
SINGLE
 36| +.2  }
 37| +.5  } Converted as (15,0) numbers
MULTIPLE
 38| +2.  }
 40| +5.  } Converted as (24,6) numbers
```

It is assumed for the most part that a generalized decimal number is of a magnitude commensurate with the number system into which it is being converted. If the number is not commensurate with the number system, an alarm may occur or an incorrect number occurs.

Cycle Control

The cycle control block of an interpretive subroutine is designed to facilitate the writing of cyclic programs and to permit a certain amount of "red tape" to be handled in the interpretive mode.

The heart of the cycle control block is the cycle control register pair. This is actually two storage registers located in the interpretive subroutine. These registers are called the index register, whose contents is "a," and the comparison register, whose contents is "b."

The following interpreted instructions are now defined:

Int. Inst.	Function
icr m (cycle reset)	Set the index register to +0 and the comparison register to +m.
ict y (cycle count)	Increase the index register by one and form the quantity $\big\|\,\|a+1\| - \|b\|\,\big\| - 0$. If the quantity is $>$ 0 interpret next the instruction in register y. If the quantity is = -0, ignore this instruction and reset the index register to +0.

If now one of the interpreted instructions ca, cs, ad, su, mr, dv, ts, ex, sp is written in the form

ixy 100 c or ixy 100 + c

the interpretive subroutine first forms the instruction

ixy (100 + 2a)

and then executes this instruction. The quantity 100 + 2a is formed instead of 100 + a since we deal mainly with arithmetic operations on 2 register numbers.

This procedure is best explained by a simple example. Suppose we wish to transfer the (24,6) numbers in 100, 102, 104, and 106 to registers 200, 202, 204, 206. We could then write

```
32| icr 4       Set up for four cycles
33| ica 100 c   Pick up C(100 + 2a)      a = 0, 1, 2, 3
34| its 200 c   Store in 200 + 2a        a = 0, 1, 2, 3
35| ict 33      Go thru 4 cycles.
```

Since it will not always be desired to operate on (m,n) numbers stored in consecutive locations we now define the following interpretive instructions

Int. Inst.	Function
ici m (cycle increase)	Increase the contents of the index register by +m
icd m (cycle decrease)	Decrease the contents of the index register by +m

As an example of the use of the ici, let us write a program which transfers the (24,6) numbers in registers 100, 104, 108 and 112 into registers 200, 204, 208, and 212. We have

```
32| icr 8       Set up for 4 cycles
33| ica 100 c   Pick up C(100 + 2a)    a = 0, 2, 4, 6
34| its 200 c   Store in 200 + 2a      a = 0, 2, 4, 6
35| ici 1       Increase contents of index register by 1
36| ict 33      Go thru 4 cycles
```

Since most programs usually contain cycles within cycles, the following interpretive instruction, which effectively permits one to use more than one cycle control register pair, is added to our code to enable these more complicated programs to be treated effectively.

Int. Inst.	Function
icx y (cycle exchange)	Exchange the contents of the index register with C(y) and exchange the contents of the comparison register with C(y+1)

To illustrate the use of this instruction, suppose that it is desired to form four scalar products. There are two sets, each with four four-dimensional vectors. The coefficients of each vector are (24,6) numbers. The coefficients of the first set of four vectors are stored in four blocks whose addresses start with 100, 108, 116 and 124. The coefficients of each vector are stored in one block. The coefficients of the second set of four vectors are stored in four blocks whose addresses start with 200, 208, 216 and 224. Scalar products will be formed with the first vector of the first set and the first vector of the second set; the second vector of the first set and the second vector of the second set; etc. It is desired to store the results of the four scalar products in a block starting with address 300. Register 400 is a register used to store the temporary sum. The instructions are as follows:

```
32| icr 16  }
33| icx 60  }    Set up for 16 cycles
34| icr 4   }
35| icx 70  }    Set up for 4 cycles
36| icr 4        Set up for 4 cycles
37| ica 51  }
38| its 400 }    Clears register 400
39| icx 60
40| ica 100 c }  a = 0, 1, 2, 3
41| imr 200 c }
42| iad 400   }  Forms scalar product
43| its 400   }
44| ici 1        Increase index register by 1
45| icx 60
46| ict 39       Go through 4 cycles
47| icx 70
48| ica 400   }
49| its 300 c }  Stores scalar product
50| ict 35       Go through 16 cycles
51| +.0
```

Finally, the following interpreted instructions are added to facilitate the handling of "red tape" while in the interpretive mode

Int. Inst.	Function
iat y (add and transfer)	Add the contents of the index register to the C(y) and store the result in the index register and register y.
iti y (transfer index digits)	Transfer the right 11 digits of the index register into the right 11 digits of register y.

These instructions primarily serve as a means of transferring the contents of the index register into a given storage register. Since the icr, ici and icd instructions enable one to set and change the contents of the index register, this register can be looked upon as an interpretive analogue of the single length, fixed point AC, with <u>iti</u> analogous to td, etc.

The Buffer Register

Although 2 register are used to store a (30-j,j) number, 3 registers are used for the MRA to avoid the time consuming operation of packing the last 15-j digits of the number and the j digits of the exponent into a single register after each interpreted instruction. A further advantage is gained in that any sequence of arithmetic operations is performed using 30 digits for the number and 15 digits for the exponent. This provides in effect a (30,15) system. The results of computation are combined into (30-j,j) number only when the C(MRA) is stored by the instructions *its* and *iex*.

The buffer register can be used in any of the instructions

icab, icsb, itsb, iexb, iadb, isub, imrb

In all of these cases "b" should be considered to represent a 3 register (30,15) location. Each of the instructions is then carried out as the corresponding instruction in a (30,15) interpretive subroutine would be carried out.

It should be emphasized that the above words represent the complete vocabulary available using the buffer symbol b.

The buffer can be used to store intermediate results in a cyclic program and thus rounding off can be avoided until after the final cycle.

Automatic Assembly of Interpretive Subroutines

Interpretive subroutines for computation in the (30,0), (15,15) and (30-j,j) number systems have been incorporated into the CCP in such a way that the type of subroutine and the features of this subroutine which are called for by the programmer in the process of writing his program are automatically punched out on 5-56 tape.

The kind of interpretive subroutine selected by the CCP will be determined by the value of the last (m,n) appearing on tape, e.g. if this is (30,0), the (30,0) interpretive subroutine will be selected. The corresponding (m,n) subroutine is then punched out onto paper tape if any interpretive instruction, imy, is used in the program. However, the special word NOTPA (which means NOT Programmed Arithmetic) appearing anywhere on the tape overrides the effect of writing the interpretive instructions and generalized numbers, and no PA subroutine is automatically selected. NOTPA is used if a programmer wishes to convert (m,n) numbers and use an interpretive subroutine which is not part of the CCP or not to use any interpretive subroutine.

Particular interpretive subroutines are further specialized in accordance with the words appearing in a program. If the single letters b or c are used in any of the instructions in the program, then the corresponding buffer and cycle control subblocks in the particular PA selected are punched out. If these letters are not used the corresponding subblocks are not punched out. Similarly, if an idv instruction is used in a program, the division subblock is punched out. These specializations are made so that parts of the subroutine which are not used will not be read into storage.

The interpretive subroutines are automatically stored by the CCP in a block of registers ending in register 1056. The initial address of the block is found by adding up the lengths of the several subblocks punched out and subtracting the result from +1057.

A table of subblocks and their lengths follows:

Subblock		Words necessary on tape for read in	Length
(30-j,j)			
PA	Buffer	final (30-j,j), b	39
	PA	final (30-j,j), ixy	199
	Cycle Count	final (30-j,j), c	57
	Divide	final (30-j,j), idv	26
(15,15)			
PA	PA	final (15,15), ixy	113
	Cycle Count	final (15,15), c	57
	Divide	final (15,15), idv	9
(30,0)			
PA	PA	final (30,0), ixy	135
	Cycle Count	final (30,0), c	57

Interpretive Operations and their functions

Interpreted Instruction	Function
ica*y (* refers to footnote and is not part of the instruction)	Clear the MRA and add into it the (m,n) number in location y.
ics*y	Clear the MRA and subtract from it the (m,n) number in location y.
iad*y	Add the (m,n) number in the MRA to the (m,n) number in location y and leave the sum in the MRA.
isu*y	Subtract from the (m,n) number in the MRA the (m,n) number in location y and leave the difference in the MRA.
imr*y	Multiply the (m,n) number in the MRA by the (m,n) number in location y and leave the product in the MRA.
idv*+y	Divide the (m,n) number in the MRA by the (m,n) number in location y and leave the quotient in the MRA.
its*y	Transfer the (m,n) number in the MRA to location y.
iex*y	Exchange the (m,n) number in the MRA with the (m,n) number in location y.
isp*y	Interpret next the instruction in register y.
sp y	Resume (15,0) WWI operation at register y.
icp*y	If the (m,n) number in the MRA is negative interpret next the instruction in register y; if positive, ignore this instruction.
ita*y	Transfer the address p+1 into the right 11 digits of register y, leaving the left 5 digits unchanged; p being the address of the isp or icp most recently interpreted.
icr m	Cycle Reset--set the index register to +0 and the comparison register to +m.
ict y	Cycle Count--increase the index register by one and form the quantity $\left\| \|a+1\| - \|b\| \right\| -0$. If this quantity is >0, interpret next the instruction in register y. If the quantity is =-0, ignore this instruction and reset the index register to +0.
ici m	Cycle Increase--increase the contents of the index register by +m.

* This interpretive operation is analogous to the (15,0) WWI operation obtained by dropping the initial i from the letter triple which designates it. The binary equivalent of the interpretive operation will <u>not</u> however be equal to the binary equivalent of the corresponding (15,0) WWI operation.

\+ Not available on (30,0).

icd m	Cycle Decrease--decrease the contents of the index register by +m.
icx y	Cycle Exchange--exchange the contents of the index register with the contents of register y and exchange the contents of the comparison register with the contents of register y+1.
iat y	Add and transfer--add the contents of the index register to the contents of register y and store the result in the index register and register y.
iti y	Transfer index digits--transfer the right 11 digits of the index register into the right 11 digits of register y.

IV. INPUT/OUTPUT

Introduction

The output media currently available for use with the In/Out routine consist of typewriters, punches, oscilloscopes and magnetic tape units. The latter may be used to record data for subsequent print out on a magnetic typewriter or as auxiliary storage devices. The oscilloscope may be used in any of three ways:

a) as a curve plotting instrument

b) to display information in binary form

c) as a numeroscope displaying alphabetical or digital characters (i.e. "alphanumeric" characters) in any desired layout.

Following are the relative speeds of the several media for recording alphanumeric characters and also their characters/line limits:

a) Typewriter — 8 characters/sec. — 160 characters/line max.

b) Scope — 150 characters/sec. — 64 characters/line max.

c) Magnetic Tape - to be used with Magnetic Typewriter — 250 characters/sec. — 90 characters/line max.

The In/Out routine is called into use by three upper case letters. The first specifies the equipment to be used, the second states whether information is to be fed into or out of the computer and the third specifies the type of information. The letters used are the initial letters of the following words:

Drum	In	Alphameric (alphanumeric)
Magnetic Tape	Out	Binary
Punch		Curve
Scope		
Typewriter		
Reader		

Examples of In/Out Instructions

TOA will print alphameric characters on the typewriter
SOC will display a curve on the scope
MIB will transfer binary information into the computer from magnetic tape.

A typical example of an output instruction is

$$\text{iTOA+p123.1234sx}2^{11}\text{x}10^{-2}$$

When the In/Out routine is called upon, it will handle the word currently in the AC or MRA. When a number expressed in any number system other than (15,0) is to be dealt with, the calling-in letters must be preceded by the lower case letter i so that the number will be interpreted. Thus iTOA will call in the output routine to print the contents of the MRA on the typewriter. At present, the following number systems are available; (30-n,n) with scale factor, (30-n,n) without scale factor, (15,0) with scale factor, (15,0) with binary point at extreme left, (15,0) with binary point at extreme right.

When the In/Out routine is required to print, display, or punch a number, the calling-in letters must be followed by a specimen number of the following general form where the numbers in parentheses refer to paragraphs below:

$$\underset{(1)}{\pm}\underset{(2)}{\alpha}\ \#\#\#\ldots\#\ \underset{(3)}{.}\ \#\#\ldots\#\ \underset{(4)}{\beta_i}\ \underset{(5)}{\times 2^{\gamma_i}}\ \underset{(5)}{\times 10^{\delta_i}}$$

The components of the number have the following meanings: (Note that in the following description the word "print" is used to mean print, punch or display, depending upon the medium previously selected).

(1) + = print the number preceded by its sign

- = print the number preceded by its sign if the number is negative, otherwise just print the number

sign omitted = print the number with no sign

<u>note</u>: By "omitted" we mean that nothing at all is written. We do <u>not</u> mean that the word "omitted" is written.

(2) (α is a lower case letter)
(By initial zeros we mean initial zeros at the left of the decimal point.)

If α is i — initial zeros are <u>i</u>gnored in printing and the first significant digit of the number is printed on the extreme left of the column.

If α is p — initial zeros are <u>p</u>rinted as spaces.

If α is omitted — initial zeros are printed.

If α is n — the number is <u>n</u>ormalized before printing, e.g., all numbers are multiplied by such a power of 10 that the first non-zero significant digit will always be in the same relative position with respect to the decimal point. <u>This cannot be used with (15,0) output</u>.

The actual digits of the numerical part of the specimen number are immaterial; they merely serve to indicate the number of digits which the programmer desires to have printed on each side of the decimal point. Thus iTOA + p347.6210s x 2^{-3} x 10^5 would indicate that the programmer wanted 3 digits to the left and 4 digits to the right of the decimal point and the numbers would be printed in the form ###.####. However, if α is n the number would be printed in the form ###.####x10^u which is the normalized case.

(3) If a decimal point is indicated, — it will be printed in the position indicated.

If a decimal point is omitted, — none will be printed. This is used in printing integers.

If a decimal point is replaced by r, — no decimal point will be printed but the r indicates where a decimal point would have been placed had there been one.

The latter facility would be of practical use in the case of decimal fractions in which it is desired to save printing time by omitting decimal points. <u>Unless one indicates a decimal point or replaces the decimal point by an r, the entire number will be treated as though it were an integer.</u>

(4) (**β** is a lower case letter)

The symbol(s) β_i specify the character(s) with which the printed number is to be terminated:

If β_i is s	we get one <u>s</u>pace
If β_i is ss	we get two <u>s</u>paces
If β_i is sss	we get three <u>s</u>paces
If β_i is ssss	we get four <u>s</u>paces
If β_i is c	we get a <u>c</u>arriage return
If β_i is t	we get a <u>t</u>ab
If β_i is omitted	we get no terminating character
If β_i is f	we get <u>f</u>ormat, i.e., the terminating character will be determined by the layout section of the In/Out routine which is in turn controlled by the Format Specification. (See paragraph on Format Specification)

(5) a) If the number is to be printed as a decimal fraction, then $\gamma = 0$, $\delta = 0$.

b) If the number is to be printed as a decimal integer, then $\gamma = 15$, $\delta = 0$.

c) Every factor <u>must</u> be preceded by a lower case x.

d) Any number of factors may be utilized, i.e., $2^{\gamma_1} \times 10^{\delta_1} \times 10^{\delta_2} \times 2^{\gamma_2}$ etc. with the following restriction: $\left|\sum \gamma_i\right|, \left|\sum \delta_i\right| \leq 127$

e) Whenever a factor such as 2^{γ} or 10^{δ} has a zero exponent, that factor may be omitted.

f) If any factor has an exponent of 1, the 1 may be omitted.

g) The exponents γ_i, δ_i are signed if negative, and not signed if positive.

Examples of the use of output instructions in the (30-n,n) system follow:

ex 1: Let the MRA contain the octal number 0.6277574516 with an exponent of 15 (octal).
Thus the number = $0.6277574516 \times 2^{15}$ (octal)
This is equivalent to $+.796812369 \times 2^{13}$ and to $+.652748693 \times 10^4$(decimal).
Let the output order be iTOA+nl.2345678c
Then the typewriter would print out +6.5274869/+03 where the number at the left of the slash is decimal and the number at the right is its exponent of 10. Thus the number is actually $+6.5274869 \times 10^3$.

ex 2: Let the MRA contain the octal number 1.1500203261 with an exponent of 15 (octal).
Thus the number = $1.1500203261 \times 2^{15}$ (octal).

This is equivalent to $-.796812369 \times 2^{13}$ and to $-.652748693 \times 10^{4}$ (decimal).
Let the output order be iSOA - $12.3456s \times 10^{-5}$
Then the 'scope would display -00.0652 sp. (see note below) where sp. means that a space is provided for on the 'scope.
note: At present no provision is made for rounding off to -00.0653.

In/Out Order Repeated

A specimen number need not be designated each time the In/Out routine is called in. If the calling-in letters are not followed by anything, then the In/Out routine will provide exactly the same set up as it furnished for the last In/Out specification. By exactly the same we mean that if one wrote iSOA following an iTOA + i12.345s $\times 2^{-4} \times 10^{6}$, he would automatically get iTOA + i12.345s $\times 2^{-4} \times 10^{6}$. If the programmer wants the same In/Out order as the last one except for the calling-in letters, he must write out the In/Out order in its entirety.

Format Specification

The In/Out routine contains a layout section which may be set by the special word:

FOR $\alpha \times \beta \times \gamma$

a) this word must precede any output order for which it is to be used

b) the entire word FORMAT may be written instead of FOR, if desired

c) α represents the number of words/line. (maximum is 15)

d) numbers α, β, γ should be separated by lower case x

e) β represents the number of spaces between words (maximum is 6)
If a tab is desired between words then set $\beta = 7$

f) γ represents the number of words per block (typewriter)
γ represents the number of words per frame ('scope)

The maximum γ is 511. However, if the programmer has more than 511 words to be printed, the block counter becomes automatically reset after each block is completed.

ex 1: Let us suppose that the programmer wishes to type 2500 words in blocks of 400. If he specifies that $\gamma = 400$, then he will automatically get 6 blocks of 400 words each and a seventh block of 100 words. The blocks will be separated by 2 carriage returns. In order to get the final 100 words as a separate block one must heed the following note.
Note: provision is made for one automatic carriage return at the beginning of the Format routine and two at the end of a block. However, the programmer should provide carriage returns at the end of his print-out if that doesn't coincide with the end of a block. This carriage return order is described in the Special Characters section.

ex 2: Let us consider ex. 1 if the scope were being used instead of the typewriter. The only difference is the restriction on the number of lines per frame which is 36. However, if the programmer requested 8 words/line, 400 lines/block, he would get 288(8x36) words on one frame and 112(400-288) words on the next frame since provision is made for automatic indexing at the end of 36 lines and at the end of a block. Thus the programmer would

get altogether 6 frames of 288 words each, 6 frames of 112 words each and one frame of 100 words. However, the last frame of 100 words will be obtained only if the following note is heeded.
Note: The programmer must provide the order FRAME in order to index the camera at the end of any particular display since it is unlikely that the end of a display will coincide with the filling up of a frame or the end of a block. An automatic index is provided at the beginning of the display routine.

Special Characters

a) One may obtain a -, +, ., s (space), t (tab), c (carriage return) at any time by merely using the call-in letters followed by any one of the above six.

exs: TOA + gives a + on the typewriter
SOA c gives a carriage return on the 'scope

b) The order COL continues the 'scope display in the next column, at the top of the frame.

The order FRA takes a picture, and sets the camera up for the next frame.

One may use the entire word COLUMN, FRAME instead of COL, FRA respectively but all letters must be upper case.

V. Conclusion

At present the CS is entirely on paper tape. Strides have been made in the direction of replacing some of the paper tapes with magnetic tapes. The latter transition will depend to a considerable extent upon the availability of magnetic tape units. At present only one magnetic tape unit is available whereas it is considered that three tape units is the optimum number for the efficient use of the CS. It is planned to store the CS permanently in the magnetic drum as soon as the drum is available. Post-mortems (PM) and Mistake Diagnosis (MD) routines will be incorporated into the CS in the near future. As soon as new In/Out routines are prepared, they will be incorporated into the CS.

Signed H. C. Uchiyamada
H. C. Uchiyamada

Signed E S Kopley
E. S. Kopley

Approved CWA.
C. W. Adams

Index for Engineering Note E-516

SUBJECT: POLICY ON OUTSIDE USERS OF WHIRLWIND I

To: Applications Group Staff

From: C. W. Adams

Date: September 29, 1952

Abstract: Computer time assigned to the Scientific and Engineering Applications Group for use in general purpose work will be strictly accounted for. Much of this time will be made available to outside users without charge if they will program their own problems and submit adequate written reports on their work. Assignment of time to individual users will be made by an Allocation Panel, composed of laboratory staff members, with appeal to the MIT Committee on Machine Methods of Computation when necessary. Assigned time will be guaranteed as the minimum amount of good time each user will receive within a bi-weekly period. Different categories of users will have quotas on the maximum amount of time to be assigned. Assignments will be made every two weeks for no more than ten weeks in advance. Extensive records and reports will be published on a bi-weekly basis.

Contents:

Availability

The Whirlwind I Computer is ordinarily operated 24 hours a day, 7 days a week. Installation of new equipment and maintenance occupies considerable machine time. About half the remainder, i.e., about 40 hours per week, is made available under ONR sponsorship for general purpose scientific and engineering computation (S&EC). This time occurs in periods ordinarily of one to four hours duration throughout the week according to a schedule which varies slightly from week to week. Most of the time for S&EC work occurs over weekends and during the early morning hours (0300 to 0700). Deducting time needed for S&EC staff work, demonstrations, emergency repairs, etc., some 20 hours per week remain which may be safely promised to outside users. The time is assigned in small amounts to various categories of users, without charge, under the conditions and according to the criteria described in this note.

Conditions

An applicant must usually

1. provide a complete written description of the proposed problem;

2. obtain the signature of a member of the MIT Faculty attesting the value of the solution and the validity of the proposed method of solution;

3. establish, with the aid of a member of the S&EC group staff if necessary, the feasibility of solving the problem on Whirlwind I within a reasonable amount of computer time and calendar time (less than 10 hours and less than 10 weeks are ordinarily reasonable);

4. be willing to prepare the computer program for the solution of the problem himself, or provide a member of his organization to do so, assuming reasonable aid in learning and in programming from the S&EC group staff;

5. be willing to prepare a brief progress report in writing every two weeks and a final report in writing at the completion of the problem;

6. release all information about the problem for publication, with due credit but without military or commercial secrecy restrictions.

Categories

The categories listed below have been established to aid in insuring that certain types of more numerous or more influential applicants do not entirely deprive other types of applicants, such as thesis students, from using the computer. Certain upper limits have been set on the amount of time available to each category and time assigned to each applicant is charged against his category. The requests of the various applicants are therefore judged in comparison to others in the same category only. In cases where an applicant can fall equally well into more than one category, the choice will be made to the advantage of the applicant. Categories are:

Academic courses, thesis and student research (S)
Machine Computation Committee Fellowships (F)
Academic and DIC Departmental research (M)
ONR projects (N)
Digital Computer Laboratory research (D)
Extended commitments (E)
Governmental and Industrial Laboratory research (G)

Criteria

Ordinarily applicants will be assigned either enough time to satisfactorily complete their projects or no time at all. Occasionally, a reduced amount of time will be offered when this seems justifiable and useful. Preference in the allocation of time will be given principally according to the following criteria:

1. utility of the general method of solution to science and engineering generally;

2. utility of the specific solution to the field in general;

3. utility of the specific solution to the applicant in particular;

4. magnitude of the problem relative to the importance of the solution;

5. efficiency and estimated cost (if charge were to be made) of using Whirlwind I relative to efficiency and cost of other possible means of solution;

6. availability to the applicant of other means for solution;

7. concreteness of the problem and proposed method of solution;

8. magnitude of problem by absolute standards;

9. efficiency of proposed use of computer by absolute standards;

10. reputation of the applicant generally and as established in any previous use of Whirlwind I.

Procedure in Applying for Use of WWI

The following procedure will ordinarily be followed in applying for the use of the Whirlwind computer:

1. The applicant will complete and submit to Professor Charles Adams form DL-518, Description of Problem Proposed for Solution on the MIT Whirlwind I Computer. The form when received will be duplicated by Ozalid process. Handwritten copy, if submitted, will be typed and a print returned to the applicant for verification. Detailed instructions for completing the form are provided. (see form and instructions attached)

2. A member of the Scientific and Engineering Applications Group will be selected by Adams to contact the applicant and arrange a meeting at which the problem can be discussed and the feasibility of the solution established. At this time also a reasonable estimate of the computer time involved will be made and added to the Description, along with any necessary amplification, corrections and staff-member comments.

3. The proposal, if feasible, will be submitted to the S&EC Time Allocation Panel at its next bi-weekly meeting for assignment of time. Ordinarily, the applicant need not be present at the panel meeting, but in some cases he will be asked to present and justify his problem and to answer questions orally.

4. When the applicant and the panel cannot reach an agreement, arrangements will be made to refer the question to the MIT Committee on Machine Methods of Computation (chairman: Professor P. M. Morse). The Committee recommendation will ordinarily be accepted as the final decision.

Mechanics of Allocation

The Allocation Panel will assign to each accepted applicant certain amounts of computer time to meet the estimated need in each of the next five bi-weekly periods. It is emphasized that an assignment represents total time per period, not a given hour on a given day. Priorities within a period and carry-overs to later periods are discussed below.

In assigning time, care will be taken not to commit all assignable time very far in advance. Ordinarily only about 20% of the assignable time for any given period will be assigned at any one bi-weekly meeting of the Allocation Panel. In no event will any assignment be made more than 10 weeks in advance except for a problem in the extended commitment category (to which are assigned a uniform number of hours per period for one year in advance only on recommendation of the Committee on Machine Methods of Computation). Thus time for a given problem will ordinarily be available as soon as it can be effectively used, and the problem should ordinarily be completed within 10 weeks of the assignment of time. If major changes of schedules or any increases in assigned time are necessary, a formal request must be made and the entire situation reconsidered by the Allocation Panel.

Allocations, once made by the Allocation Panel, will be adhered to as closely and as fairly as possible. Neither the Digital Computer Laboratory nor its staff members will be held legally or morally responsible when assigned time has to be postponed, reduced or withdrawn due to circumstances beyond the direct control of those involved. No promises of time hinted at, implied or specifically made by any person not representing the Allocation Panel or the Committee on Machine Methods may be taken as a definite commitment. In particular, no allocation of time more than 10 weeks in advance may be considered as a definite commitment except in the extended commitment category, in which time is assigned one year in advance.

Preparation of Programs

The applicant will be solely reponsible for preparing his own program to obtain an efficient solution of his problem on the machine. A staff member will be assigned only to advise him on specific points and give him whatever help he must have in preparing the program. He may call briefly on the advice or service of any other staff member in the S&EC group when necessary. He may not expect the laboratory staff to prepare detailed programs or do detailed checking for him.

Each staff member will schedule an individually-prescribed number of office hours and will ordinarily be available for consultation only during those hours. Appointments and schedules will be coordinated by the S&EC Group Staff Coordinator, Room 218, Barta Building. Records of the amount of staff time used in connection with each problem will be kept by the Staff Coordinator.

Training in Programming

In addition to the consulting facilities described above, the S&EC Group will provide (1) an up-to-date reference manual, (2) an elementary training program, and (3) an advanced seminar, all on programming for Whirlwind I.

The manual will describe the computer, techniques of programming, and the generally-adopted procedures for using the standard service routines for input, output, extra-precision and/or floating point operation, function evaluation and mistake location.

The elementary training program will consist of a series of six or seven lectures of two-hour duration presented over a period of two weeks by members of the staff. The program will be repeated every two weeks, excluding academic vacations, except whenever there is insufficient demand.

The seminar will meet once every two weeks to discuss new developments in Whirlwind I hardware or techniques, to hear new suggestions, to receive reports from a committee on new suggestions, and to discuss any questions or suggestions of general interest.

The manual and training program will be available to anyone listed as a programmer on an approved problem. Others may be included by special request to Professor Adams or to the Committee on Machine Methods of Computation. The seminar will be open to everyone. The overall program will be under the direction of the Training Supervisor, aided by the Training Coordinator, Room 218, Barta Building.

Preparation of Tapes

Preparation of punched paper tapes to introduce programs and data into the computer is a clerical procedure handled by the Tape Preparation Room, supervised by the S&EC Group Tape Preparation Coordinator, under the direction of the Operations Supervisor, Room 218, Barta Building. A reasonable amount of tape preparation is implicit with an allocation of computer time. In some cases special arrangements can be made by which an applicant may make tapes of his own, but this is not usual procedure. Records of the amount of tape preparation used in connection with each problem will be kept by the Tape Preparation Coordinator. Requisition forms must be used whenever a new tape is to [illegible] be prepared or an old one modified. Ordinarily, all requests are filled within one day. Rush jobs, so marked, are handled in two hours or less. Five minutes is deducted from the computer time assignment for each rush tape job.

Operation of the Computer

To increase efficiency, all computer operation on S&EC work is performed by one of a group of young operators who specialize in this job. Programmers are not permitted to operate the computer for themselves. They may specially instruct the operators in advance or on the spot if desired, subject to all usual time limitations given below. Computer operations are scheduled and supervised by the S&EC Group Computer Coordinator, under the direction of the Operations Supervisor.

Precedence in Program Performance

When a program has been submitted to the Tape Room, the programmer may immediately request the performance of that program. He must submit a request form in which he designates, among other things, the length of time which his program is expected to take. If this is less than five minutes, he may mark his program "short run." All requests are submitted to the Computer Coordinator in Room 218, Barta Building, where they are numbered, marked with the date and hour, and placed in sequence as received. During each available computer hour, the programs are performed in the order requested, except that

1. in any hour, requests marked "rush" are dealt with first, in the order received;

2. in any daytime hour, all short runs are performed before any long runs (however, a short run is taken from the machine as soon as it runs over five minutes, the five minutes used being charged against the assigned time);

3. any request may be delayed (but in no case advanced), if desired, until a particular time or until the programmer arrives to watch it run;

4. requests for which no assigned time is available are kept in a separate sequence and performed in that sequence only after all assigned-time requests are completed. Rush and short runs are given no preferential treatment. No unassigned-time request may exceed one hour of computer time. If more than one request for unassigned time is received from a single programmer, the others are kept in special sequence and inserted into the main sequence only after the first request has been filled.

Assignment

The applicant's "account" of assigned time in a given period will be charged for every second his problem is on the computer. In case of computer malfunctions, the operator will start the problem over again with no charge for the false start unless the operating time before the failure exceeded five minutes, in which case the excess over five minutes will be charged. When all assigned time for a particular applicant during a given biweekly period has been used up, his requests for further operations will be held until unassigned time becomes available as described above or until the start of the next period in which he has time assigned.

Reassigned Time

Assigned time unused at the end of a period will automatically be multiplied by a reassignment factor of .5 and added to the assignment for the following week. If the time was unused because the computer was unavailable due to malfunctioning the reassignment factor will 1.0.

Whenever an applicant realizes that he cannot effectively use his time during a given period, he may relinquish any amount of it in writing immediately and apply to the Allocation Panel for reassignment. Time relinquished far enough in advance to make it available for other assignment will be reassigned to later periods as desired (where possible) with a reassignment factor of .75 or greater: time relinquished during a period will be reassigned to later periods as desired (where possible) with an reassignment factor of .5 or greater. (Form DL-527, attached)

Direct exchanges of assigned time between applicants are permissible. These should be reported immediately to the Computer Coordinator, Room 218, to permit suitable changes of record. The ratio of exchange need not be one for one. In some cases the Allocation Panel or the Operations Supervisor may offer relinquished time to other applicants in exchange for relinquishing the same amount (or less if circumstances dictate) of later assigned time.

Rush Jobs

A request for computer operation marked "rush" will be performed before all routine requests. A rush job is charged against assigned time at a rate of double time plus five minutes per job.

Forfeiture

Whenever no unfilled requests for assigned time are waiting, computer time is given without assignment to any requests for which assigned time was not available. When any appreciable amount of such time is given without assignment, all assigned time outstanding for that period may be proportionately reduced to compensate (down to a minimum of 50%).

Records

Careful records are kept by the Computer Coordinator, both cumulatively and biweekly, of time requested, of time assigned, of time used, and of assigned time lost due to unused time, relinquished time, rush jobs and forfeiture. These records supplement the records of staff time and tape preparation time.

Bi-Weekly Reports

A biweekly report must be submitted on form DL-525 on or before noon on alternate Thursdays. Assigned time for the following weeks will be deducted at a rate of 10 minutes time for each working hour or fraction of tardiness. These reports should be brief but should be comprehensible without specialized vocabulary (especially of code symbols) or extensive knowledge of previous status. Above all, the report must explain the disposition of the computer time, staff time, and tape preparation time which have been charged against the problem. The Panel or the Committee may at any time suspend any problem for which justification seems inadequate until the applicant has reconvinced the Panel or Committee of the validity of his work.

Final Reports

At the completion or abandonment of a problem, the applicant must prepare a final report describing completely the problem, the methods of solution tried, the difficulties encountered, the final program used, and the results obtained. In the case of problems running for more than three months, a quarterly progress report must also be prepared in which the previous 6 or 7 bi-weekly reports, along with the description of the problem, must be condensed into a single, integrated report for the quarter. Failure to prepare a final report within four weeks of the completion or abandonment of a problem may prejudice consideration of future requests for computer time.

Signed CWAdams
C. W. Adams

Attached:
DL-518
DL-526
DL-527
DL-525

INSTRUCTIONS FOR THE PREPARATION OF FORM DL-518

a. Fill out answers to all questions except those which, after serious deliberation, you feel unqualified to answer before discussing the matter with a member of the Digital Computer Laboratory staff.

b. Use a typewriter for all words, black ink for equations and symbols if possible. In any event, do not use blue pencil or light blue ink as this does not reproduce on Ozalid.

c. Add to the form a supplementary statement, typed on unwatermarked semi-translucent white paper, describing in some detail the problem, its origin, its history to date, its importance, and giving names of researchers interested in it and published references if any.

d. The meaning of the individual questions is given below.

1. Give full name with title (e.g., Mr., Miss, Mrs., Dr., Prof., Col., etc.) of person primarily responsible for wanting the problem done. Then CROSS OUT either "associated with" or "representing" to indicate the type of affiliation and give name of organization and department, with further detail if necessary. Do not use initials unless they can be readily interpreted. Do not use SELF. A student in math would cross out "representing" and write "MIT Math Dept." A student in math assigned to the work as part of a Machine Methods Committee Fellowship would cross out "associated with" and write "MIT Math Dept., ONR Fellow."

2. Give a title which describes the physical problem as briefly as possible, followed by and not depending on a brief mention of the mathematical problem involved. For example, RLC Circuit Transient Response; second order linear differential equation.

3. Summarize the physical and mathematical problem in words and/or equations as concisely as possible.

4. List the symbols which represent the independent variables (type I), the parameters or numbers which remain constant during one solution but are to be varied from one solution to the next (type P), the dependent variables (type D), and the constants (type C). Give their meanings if obvious. Arrange them in order of type I, P, D or C and label each. Indicate the range, i.e., the quotient of maximum divided by minimum possible values, of each. Indicate the number of different discrete values each quantity is expected to assume during the calculation.

5. Name and/or describe the numerical procedures to be used.

6. Indicate the number of significant digits which are necessary (needed) and those which are sufficient (desired) to insure useful results. Cross out two of the three choices (known, estimated, guessed).

7. Indicate how many complete or partial numerical solutions have already been attained by hand or using other computing aids. If none, write NO. If any, indicate about how many man (or girl) minutes were required on the average per complete solution. Describe the equipment used (e.g., "2 girls

DESCRIPTION OF PROBLEM PROPOSED FOR SOLUTION ON THE MIT WHIRLWIND I COMPUTER
(see instruction on separate sheet)

1. Submitted by: associated with / representing:

2. Names:

3. Brief description of complete problem with indication of the part to be solved by WWI:

4. Symbol, Meaning, Type, Range, No.

5. Basic numerical procedures to be used, with approximate number of repetitions of each:

6. Number of significant decimal digits: ___ desired ___ needed (known, estimated, guessed)

7. Numerical solutions have been obtained for ___ cases, each requiring about ___ minutes, by means of:

8. An analytic solution to the problem cannot be obtained because:

9. Programmer(s): Name, Position, Business Address, Phone, Field of Interest, Degree, Date, Experience, Time Available

10. Reference(s): Name, Position, Business Address, Phone, Field of Interest

11. SCHEDULE

Period						Total
Programming hours						
Computer MD hours						
Performance hours						

DL-518

--->send to ROOM 218, BARTA BUILDING

SUBJECT: REQUEST FOR WWI TIME FOR SCIENTIFIC AND ENGINEERING COMPUTATION

To: Charles W. Adams, chairman, S&EC Group Time Allocation Panel
MIT Digital Computer Laboratory, Cambridge 39, Massachusetts

From:

Date:

Title or Number of Problem:

applicant check/and fill in blanks: a or b	Action of Panel (to be filled in at Panel Meeting)
a)____I submit herewith form DL-518 describing a problem which I have discussed in detail with__________ __________of the S&EC Group Staff b)____I hereby relinquish_____of the hours which were assigned to me for the period ending___________,195__	Date_______________, 195__ Members present a | b | c | no opinion
I request computer time (re)assigned as follows: for biweekly period ending __________________, 195__, ____hours __________________, 195__, ____hours __________________, 195__, ____hours __________________, 195__, ____hours __________________, 195__, ____hours TOTAL ____hours Comments:	________ (a) request disapproved ________ (b) referred to Committee ________ (c) assigned as follows former assignment | NEW ASSIGNMENT Comments:

DL-527

BIWEEKLY REPORT ---→send to ROOM 218, BARTA BUILDING

SUBJECT: PROGRESS ON SCIENTIFIC AND ENGINEERING COMPUTATION PROBLEM # ____________

To: J. W. Forrester, Director, MIT Digital Computer Lab., Cambridge 39, Mass.

From:

Covering the two-week period up to 1200, Thursday, ________________, 195__

Names of Programmers	Time Spent	Whirlwind I Time (to be filled in by Computer Coordinator)			
		used	____	used to date	____
		rush jobs	____	rush to date	____
		assigned	____	assigned to date	____
		unused	____	for next period	____
Associated DCL Staff Member		carried		assigned ahead	
				Numbers	Total
		requests processed		____	____
____________________		tapes prepared		____	____
		number of words		____	____

Date initated __________________, 195__; Date to be terminated ______________, 195_

Problem: ☐ Described in Biweekly Report ______________, 195__
☐ See attached sheet (Description should be as brief and non-technical as possible)

(Prepare A,B,C coherently enough to be directly incorporated into the DCL Biweekly.)

A. Results: (describe, indicate significance & what fraction of job is represented)

(a) Productive computer time { computation ____ hr ____ min; output ____ hr ____ min }

B. Difficulties (be very explicit-name exact trouble and cause);

(b) Time lost to program difficulties ____ hr ____ min

(c) Time lost to clerical difficulties ____ hr ____ min

(Sum of (a), (b), (c) should equal net time used)

C. Plans for future:

Expect to complete problem ___ weeks ahead of / behind schedule.

DL-525

WWI PERFORMANCE REQUEST Being Modified

Prob. No. ____ Date ______________ Submitted by ______________

☐ Rush MIT Phone ______ Home ______

Input

☐ 5-56 tape
☐ basic direct flexo
☐ CS direct flexo
☐ 5-56 and CS direct flexo

Programmer

☐ need not } be present
☐ must
approved by ____ time ______

Output

operates within ___ sec.
runs ___ min.

☐ printer, no. lines ____
☐ punch, no. inches ____
☐ scope & camera, no. frames ____
☐ magnetic tape, read ____
record ____
unit no. ____

Performance

☐ short run (5 min. or less)
☐ long run ____ min.

Program Should

☐ stop automatically on ______
☐ be stopped manually after ____ min.

INSTRUCTIONS: (list the sequence in which they are to be operated)

Performed by ______________

Date ______________

DL-324-4

Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: TAPE PREPARATION REQUISITIONS

To: S. & E. C. Group and Group 61

From: M. C. Mackey

Date: December 22, 1952

Abstract: A new tape preparation form is now in use. This memorandum should answer any questions on how this form should be filled out and submitted.

Since the form has been generalized in order to satisfy both the Scientific and Engineering Computation and the Air Defense groups, it seems advisable to issue instructions on how Tape Requisitions are to be submitted.

Tape Numbers

The tape number must be written in the proper place on the requisition.

See the sections dealing with Mods and parameters for information on numbering them. A requisition covers either a mod or a parameter. One requisition may not be used for both.

Problem Numbers

Space for Problem number is allowed for the S. & E. C. group. This must be filled in by members of this group or any outsiders working with this group.

Modifications

Mod numbers start with zero and are increased by 1 for each additional mod. If the registers and their contents which are to be typed constitute a complete program, please indicate this by checking the block provided.

If these registers are changes in a previous program they are attached to this tape. The programmers must indicate the tape and mod to which they wish this information attached. The same is true of additions to a tape.

Parameters

Parameter tapes are always kept separate from the main program. They should be numbered starting with zero and increased by 1 for each new parameter. If the programmer wishes to change certain registers in a parameter, he must use the next highest parameter number and indicate the parameter tape to which he wishes these changes attached. As far as our records are concerned, there is no other provision made for modifications on parameters.

Addresses

One of the blocks dealing with the addresses should be checked depending on whether they are octal or decimal.

Form

The form in which the information is to be typed should be indicated. Since the Direct Read-in Program (for tapes left in Standard form) occupies registers 1250 thru 2037 (octal), a program must be converted to 5-56 form if it occupies any of these registers.

The Comprehensive System of Service Routines (CS) will process tapes using floating addresses, generalized decimal numbers and interpretive routines.

The number of registers to be punched manually in 5-56 form may not exceed 4.

Other Information

The author's name, room and telephone number should be given.

The register to which control is transferred after read-in should be placed after the words START AT on the requisition. If your program starts at an interpreted order be sure to put an "i" before the words START AT.

Because of the lack of space for filing tapes and manuscripts, our motto has become "When in doubt, throw it out!" All previous mods or parameters (depending on which the request is for) will be discarded unless they are circled in the space provided on the tape preparation requistion.

Programmers may request that the tape be completed by a certain time by filling in the blank labeled "Needed by" in the upper left-hand corner of the requisition. However, this does not mean that the tape will be ready at this time. All requests are filled in order of receipt and the time needed to prepare a tape will vary according to the work load at that time.

Clear, legible manuscripts and a requisition with the necessary information carefully indicated will help us to operate quickly and efficiently.

Signed M. C. Mackey
M. C. Mackey

Approved C. W. Adams
C. W. Adams djcl

TAPE PREPARATION REQUISITION

Rec'd at ______________

Needed by ______________

Prep. Time ______________

Tape # ________ Mod # / Param # ________

Author ________ Rm ______ Ext ______

Problem # ______ Approved by ______

☐ This is a complete tape

☐ Attach to ________

☐ Octal addresses are used

☐ Decimal addresses are used

☐ Type and leave in Standard Form

☐ Type and convert to 5-56 Form

☐ Type for CS Conversion

☐ Type in Subroutine Form { ☐ Basic / ☐ CS }

☐ Punch manually in 5-56 Form

☐ Type exactly as indicated

t= ____________ (address of zero temp. reg.

START AT ______________

Special Instructions ______________

Discard all previous Mods / Param. except those circled 0 1 2 3 4 5 6 7 8 9

Typed by ______________

Printed by ______________

Duplicated by ______________

Proofread by ______________

Corrected by ______________

Filed by ______________

DL-590-2

Tape Preparation Complaint

Being Modified

Tape # ________

Programmer ________________ Typist ________________ Date ________

Errors:

☐ Error in typewritten copy

at reg. ________, read ________ instead of ________.

at reg. ________, read ________ instead of ________.

☐ additional errors marked on attached program print.

☐ Error in 5-5-6 tape as marked on the tape

☐ incorrect character

☐ omitted character

☐ improper structure

Cause: ________________ Investigated by ________________ Date ________

The tape ☐ was / ☐ was not checked.

☐ manual error, operator ________________

☐ machine error, punch ________________ typewriter ________________

☐ manual duplication error, operator ________________

☐ machine duplication error, punch ________________ reader ________________

☐ no punched tape error

Summary: (Error occurred on (date) ________ at (time) ________)

☐ transient, reader ________

☐ transient, punch ________

☐ transient, typewriter ________

☐ unknown cause (theory: ________________________________

____

________________________________).

☐ equipment, breakdown ________

☐ transient, computer ________

☐ manual, operator ________

Additional Comments:

FORM DL-251

POWERS OF TWO

2^x	x	2^{-x}
2	1	.5
4	2	.25
8	3	.125
16	4	.0625
32	5	.03125
64	6	.01562 5
128	7	.00781 25
256	8	.00390 625
512	9	.00195 3125
1024	10	.00097 65625
2048	11	.00048 82812 5
4096	12	.00024 41406 25
8192	13	.00012 20703 125
16384	14	.00006 10351 5625
32768	15	.00003 05175 78125

2^x	x	2^{-x}
65536	16	. 00001 52587 89062 5
1 31072	17	.* 76293 94531 25
2 62144	18	.* 38146 97265 625
5 24288	19	.* 19073 48632 8125
10 48576	20	.* 09536 74316 40625
20 97152	21	.* 04768 37158 20312 5
41 94304	22	.* 02384 18579 10156 25
83 88608	23	.* 01192 09289 55078 125
167 77216	24	.* 00596 04644 77539 0625
335 54432	25	.* 00298 02322 38769 53125
671 08864	26	.* 00149 01161 19384 76562 5
1342 17728	27	.* 00074 50580 59692 38281 25
2684 35456	28	.* 00037 25290 29846 19140 625
5368 70912	29	.* 00018 62645 14923 09570 3125
10737 41824	30	.* 00009 31322 57461 54785 15625

2^x	x	2^{-x}
21474 83648	31	.* 00004 65661 28730 77392 57812 5
42949 67296	32	.* 00002 32830 64365 38696 28906 25
85899 34592	33	.* 00001 16415 32182 69348 14453 125
1 71798 69184	34	.** 58207 66091 34674 07226 5625
3 43597 38368	35	.** 29103 83045 67337 03613 28125
6 87194 76736	36	.** 14551 91522 83668 51806 64062 5
13 74389 53472	37	.** 07275 95761 41834 25903 32031 25
27 48779 06944	38	.** 03637 97880 70917 12951 66015 625
54 97558 13888	39	.** 01818 98940 35458 56475 83007 8125
109 95116 27776	40	.** 00909 49470 17729 28237 91503 90625
219 90232 55552	41	.** 00454 74735 08864 64118 95751 95312 5
439 80465 11104	42	.** 00227 37367 54432 32059 47875 97656 25
879 60930 22208	43	.** 00113 68683 77216 16029 73937 98828 125
1759 21860 44416	44	.** 00056 84341 88608 08014 86968 99414 0625
3518 43720 88832	45	.** 00028 42170 94304 04007 43484 49707 03125
7036 87441 77664	46	.** 00014 21085 47152 02003 71742 24853 51562 5
14073 74883 55328	47	.** 00007 10542 73576 01001 85871 12426 75781 25
28147 49767 10656	48	.** 00003 55271 36788 00500 92935 56213 37890 625
56294 99534 21312	49	.** 00001 77635 68394 00250 46467 78106 68945 3125
1 12589 99068 42624	50	.*** 88817 84197 00125 23233 89053 34472 65625
2 25179 98136 85248	51	.*** 44408 92098 50062 61616 94526 67236 32812 5
4 50359 96273 70496	52	.*** 22204 46049 25031 30808 47263 33618 16406 25
9 00719 92547 40992	53	.*** 11102 23024 62515 65404 23631 66809 08203 125
18 01439 85094 81984	54	.*** 05551 11512 31257 82702 11815 83404 54101 5625
36 02879 70189 63968	55	.*** 02775 55756 15628 91351 05907 91702 27050 78125
72 05759 40379 27936	56	.*** 01387 77878 07814 45675 52953 95851 13525 39062 5
144 11518 80758 55872	57	.*** 00693 88939 03907 22837 76476 97925 56762 69531 25
288 23037 61517 11744	58	.*** 00346 94469 51953 61418 88238 48962 78381 34765 625
576 46075 23034 23488	59	.*** 00173 47234 75976 80709 44119 24481 39190 67382 8125
1152 92150 46068 46976	60	.*** 00086 73617 37988 40354 72059 62240 69595 33691 40625
2305 84300 92136 93952	61	.*** 00043 36808 68994 20177 36029 81120 34797 66845 70312 5
4611 68601 84273 87904	62	.*** 00021 68404 34497 10088 68014 90560 17398 83422 85156 25
9223 37203 68547 75808	63	.*** 00010 84202 17248 55044 34007 45280 08699 41711 42578 125
18446 74407 37095 51616	64	.*** 00005 42101 08624 27522 17003 72640 04349 70855 71289 0625
36893 48814 74191 03232	65	.*** 00002 71050 54312 13761 08501 86320 02174 85427 85644 53125

*= 00000