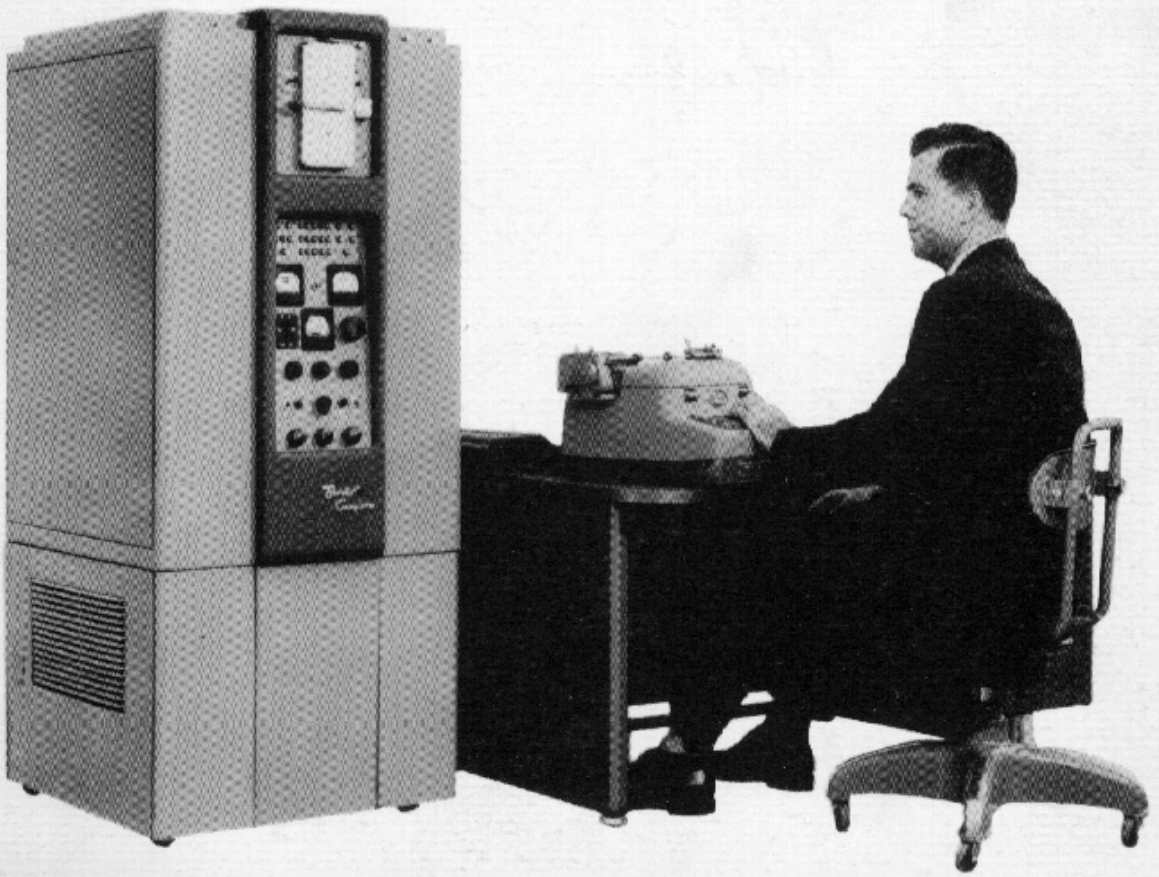
CODING MANUAL FOR THE

Bendix G-15

GENERAL PURPOSE DIGITAL COMPUTER



SUMMARY OF CONTENTS

THE G-15 COMPUTER

The Bendix G-15 is a powerful, compact, internally programmed, digital computer. The basic unit provides a complete, general purpose, computing system in a single cabinet. A photo-electric tape reader and a tape punch are built into the machine. A special typewriter is provided for control and can be used for input and output.

The system is expandable by means of numerous accessories.

Information can be expressed in decimal notation during input and output. Internally during computation, information exists in serial, binary form.

The design of the G-15 permits efficient use of a variety of programming systems. In some of these systems, such as "Intercom 1000" and "Pogo," much programming detail is done by the computer.

Intercom 1000 is an interpretive system. A program is written in simple form and stored in the memory of the computer in that form. During computation the computer examines the simple commands one at a time, and then executes a series of basic machine commands for each stored, simplified command.

Pogo is a compiling system. A program written in simple form is changed by the computer into a new program made up of basic machine commands which the computer records on punched tape or magnetic tape. The program on tape may then be re-entered into the computer for execution whenever desired.

Intercom 1000 and Pogo are described in other manuals. This manual describes basic machine programming in which a command is written for each operation performed by the computer. In basic machine coding, programmers can utilize the full power of the computer.

STORAGE OF INFORMATION IN THE G-15

The commands to be obeyed by the computer and the numerical information on which the commands operate are held in the computer's memory.

BITS AND WORDS

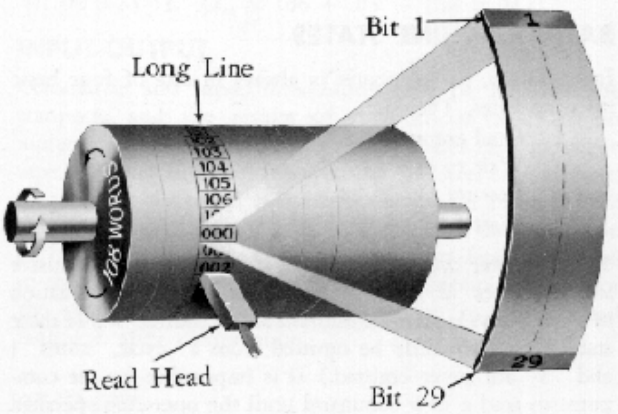
The internal memory of the G-15 is a rotating magnetic drum on which information is stored serially in binary form. The smallest unit of information, either a "one" or a "zero," is called a "bit" (an abbreviation of binary digit). A "word" of information contains 29 bits and may represent either a command or a number. If the 29 bits represent a number, one bit, in the least significant portion of the word, represents a positive or negative sign. The other 28 bits represent the number's absolute value in binary form. All numbers are assumed to be greater than -1 and less than $+1$ in internal computation; that is, the binary point, which corresponds to the decimal point for decimal numbers, is assumed to be at the beginning of the word.

LONG LINES

Most of the information stored on the drum is in 20 channels, or "long lines," numbered from 0 through 19. Each long line is able to hold 108 words numbered from 0 through 107. There are, therefore, 29 times 108 or 3132 bits of information in a long line, and there are 2160 words in the 20 long lines.

DRUM ADDRESSES

To locate a word of information stored on the drum it is only necessary to specify the number of the channel the word is in, and its position in the channel. The number 07:38, for example, would designate word 38 in line 7. Such a location number is called an "address." An address in which the word position is an odd number is called an "odd" address; an address in which the word position is an even number is called an "even" address.



PORTION OF MAGNETIC MEMORY DRUM

Figure 1

DRUM ACCESS

The information in a line is sensed or "read" by a motionless "head" as shown in Figure 1. A specific word is read, and hence available for use, once per drum revolution, or once every 108 "word times." A word time, the time required

for 29 bits of information, or one word, to pass under a reading head, is equal to .27 milliseconds. Word availability is cyclic: word 00 is available for use immediately after word 107. The drum cycle thus requires 108 times .27, or 29 milliseconds. The average access time for a long line is one half of this figure or 14.5 milliseconds.

SHORT LINES

In order to provide information storage, the contents of which will be made available for processing more often than once per drum revolution, eight "rapid-access" lines of storage are provided on the drum. Four of these lines, identified as "short lines" and numbered 20 through 23 inclusive, are able to hold four words of information each. The words are numbered 0, 1, 2 and 3 in each line. Each word in a short line is available for processing once every four word times of drum revolution. Therefore,there are 27 (108 divided by 4) opportunities to read a word stored in a short line during a single drum cycle. In a short line also, word availability is cyclic: word 0 is available immediately after word 3. Each of the 16 words in the four short lines has a maximum access time of 4 times .27, or 1.08, milliseconds. Thus, average access time is .54 milliseconds.

REGISTERS

The four other rapid-access lines are called "registers" and are named ID, PN, MQ and AR. The ID, PN and MQ registers can each hold two words of information; each word of their contents is available every other word time. The AR register holds one word of information which is accessible during any word time.

BASIC MACHINE STATES

In operation, the computer is always in one of four basic machine states. They are:

1. Read command
2. Wait to execute
3. Execute
4. Wait next command

The computer cycles through these four mutually exclusive states. (States 2 and 4 represent delays. Minimization of these delays is called minimum access coding. While these states may frequently be omitted from a cycle, states 1 and 3 are never omitted.) It is impossible for the computer to read a new command until the operation specified by the previous command has been completed. (Input-output operations are exceptions to this rule, as will be explained in a later section.) Similarly, the operation specified by a command cannot be initiated until the conclusion of the word time during which it is read.

FUNCTIONS OF ARITHMETIC REGISTERS AND COMMAND LINES

ARITHMETIC REGISTERS

The AR and PN registers are "accumulators"; that is, each has associated electronic circuitry that enables it to perform addition and subtraction. The MQ, ID, and PN registers are particularly useful in multiplication and division. The MQ register holds the *M*ultiplier in the former operation and the *Q*uotient in the latter; the ID register holds the multipl*I*cand in multiplication or the *D*enominator in division; the PN register holds the *P*roduct or the *N*umerator.

COMMAND LINES

Commands must be executed from one of eight lines, called command lines, or from register AR; the lines are long lines 0, 1, 2, 3, 4, 5, 19 and short line 23. A command or number may be stored in any memory location, but a command must be transferred to one of the above listed lines or register AR before it can be obeyed by the computer.

Any one of the eight command lines may be selected by the program itself or by the operator from the keyboard of the typewriter. Once a command line has been selected, all subsequent commands must be taken from this line until (a) the operator selects a new command line, or (b) a special command taken from the command line causes a new line to be selected. Another special command may be used to cause the word in the AR register to be obeyed as a command, but command control normally reverts in this case to the line from which this special command has been taken.

PROGRAM PREPARATION ROUTINE

The set of commands and numbers that constitute a program is entered into the memory in decimal form. A previously inserted "service" program then automatically converts the set of commands and numbers to binary form before use. The standard service program is called the Program Preparation Routine. It is supplied in a punched tape magazine along with the computer. The insertion of this routine into the computer is almost entirely automatic.

The Program Preparation Routine is described in detail in the G-15 Operating Manual.

COMMAND STRUCTURE OF THE G-15

FORM OF G-15 COMMAND

T or L_k	N	C	S	D

A command in the G-15 specifies the nature of the operation to be performed, the addresses of words necessary to perform the operation, and the address of the next command. As shown in the Standard Command List, combinations of C, S and D specify the nature of the operation. T and S or T and D specify the address of the word operated on (address of the operand). N specifies the word position in the command line of the next command. Commands not requiring any operand address, such as input-output commands, are coded with the symbol L_k. This unique command

structure combines the advantages of both a short command word and easily programmed minimum access coding (since the address of the next command is always specified).

Modifications are permissible in the standard command codes in order to make programming more compact and more adaptable to certain problems. The allowable changes are described in the sections following the description of the Standard Command Codes.

TERMS USED IN STANDARD COMMAND LIST

T and S or T and D represent the address of the operand. S or D is the channel or line in which the operand is located, and T is the position in the line. The designation S is used when the address is one which is being used as a *source* of information for the operation; the designation D is used when the operation specifies that the address is to be a *destination* for information. S or D is expressed as a two-digit number, 00 to 31. N and T assume decimal values between 0 and 107, but are represented in command codes by two digit numbers as shown in Table I.

N specifies the word location in the command line of the next command to be followed. N should not be equal to 107 since a command should not be obeyed from word 107 of the command line. The command line from which commands are being obeyed remains the same until an order to select a new command line is executed. After the normal turn-on procedure for the computer has been followed, the first command will be obeyed from word 00 of line 23.

Commands in which it is not necessary to specify any operand address are coded with the symbol "L_k" where k is a numerical subscript. L_k signifies a number equal to the sum of the word position of the command (L) and the subscript k. L_0 then signifies a number equal to word position of the command itself, and L_2 a number two larger than the word position of the command. If the resulting number is greater than 107, a value of 108 must be subtracted from the number.

The one, two and three address forms of the G-15 command are shown below.

ADDRESSES IN G-15 COMMAND

In some commands one address is specified. The address is the location of the next command.

L_k	N			

L_k is a number equal to the word position of the command + k. N is word position of next command.

In some commands two addresses are specified. One is the address of the operand; the other is the location of the next command.

T	N			D

or

T	N		S	

D and T or S and T is operand address. N is word position of next command.

In the remaining commands three addresses are specified. Two are the addresses of the operand; the third is the location of the next command.

T	N		S	D

D and T is the operand destination address. S and T is the operand source address. N is word position of next command.

The blank spaces shown in the four command forms contain code numbers specifying the nature of the command.

The following examples demonstrate the coding of commands using the Standard Command List.

Example a: It is desired to write a command to add the contents of line 14, word 53 to Register AR. The next succeeding command is to be taken from word 55 of the command line. Since the code for "Add to AR" is T N 1 S 29 in the Standard Command List, the command is written as "53 55 1 14 29."

Example b: It is desired to write the command "Test for Overflow" in word location 19 of line 00. The next succeeding command is to be obeyed from word location 25 of the command line. Since the code for "Test for Overflow" is "L_2 N 0 29 31," the command is written as "21 25 0 29 31." ($L_2 = 19 + 2 = 21$)

Example c: It is desired to write the command "Clear Multiplication and Division Registers" in word 106 of line 17. The next succeeding command is to be obeyed from word 09 of the command line. Since the code in the Standard Command List is "L_3 N 0 23 31," the command is written as "01 09 0 23 31." ($L_3 = 106 + 03 - 108 = 01$)

INPUT-OUTPUT

Commands and input information may be fed into the computer, and the results of computation read out in numerous forms. Typewriter, punched paper tape, magnetic tape, and punched card input-output facilities are available. In addition, provision is made for a Special Input-Output Register: The register would be designed for the conversion of input information, in a different form from that used in the G-15, to proper form before computation, and for the reconversion of output information to the external language after computation.

READ-OUT

Either an input or an output operation may proceed while the computer is carrying out computation. This is accomplished by operating the input-output section independently of the rest of the computer. When a command is given to initiate an input or an output operation, the computer stays in the "execute" state long enough to transfer a code number to the input-output unit, then continues its basic cycle. The input-output unit interprets the code number as an instruction, and after obeying the instruction, clears the number from a static register.

The read-out of information is from either line 19 or the AR register. The order of read-out is from highest num-

bered to lowest-numbered word and from most significant to least significant digit. (Output proceeds from word location 107 in line 19: The contents of word 107 are read out, and all other words in the line are automatically shifted to the next higher-numbered word locations. A repetition of the process then causes the read-out of the next word in the line.)

OUTPUT FORMAT

The format of output from the G-15 is controlled by the programmer, who may use two modes of operation, alphanumeric or numeric.

In the alphanumeric mode, the programmer types in each character, symbol and control function desired for the output format. For example, if a tab is required following certain information on the output copy, then during input the tab key is pressed after this information is typed in.

In the numeric mode, information is punched or typed out under control of a block of four words. (See "Output Format Control," page 22.) These format control words contain codes, inserted by the programmer, that determine the manner in which output information is to be interpreted as well as the location of periods, tabs and carriage returns. When the standard Program Preparation Routine is in service, an automatic means for compiling format control words is provided.

READ-IN

Information read into the computer is read into line 19. (Input proceeds from the lower-numbered word positions of the line: Four words of information are read into locations 00 through 03; all words in line 19 are then shifted to the next higher-numbered group of four-word locations and the process repeated. The shifting process is automatic and need not be programmed.)

READY

Before any input-output operations may be initiated, the computer must be in the "ready" state. The "ready" state is defined as the state the computer is in after it has fully completed its last input-output operation, and therefore is ready for the next one. If two or more input-output operations are to be carried out in one program, the command, "Test for 'Ready' of Input-Output," should precede all but the first operation.

Line 23 is used by the computer during input or during magnetic tape output; therefore, line 23 should not be programmed for storage during times which may coincide with the execution of those operations.

BREAK-POINT OPERATION

The computer may be programmed to come to a halt after performing a specific command at the option of the programmer. To cause a halt, a minus sign (−) is added as a suffix to the command and the "Compute" switch is set to the "BP" position.

The "Compute" switch has three positions: "BP," or break-point; the middle position, which is off; and "GO," which is the normal position in which computation occurs. If the switch is set to "GO," computation will proceed and any coded break-points in commands (minus signs) will be disregarded. If the switch is set to "BP," computation will proceed but will cease after a command coded for break-point is obeyed.

To re-continue computation after a break-point has been reached, the switch is set to off and then again to "BP" or "GO." The computer will then obey the command in the "N" address of the break-point command.

TABLE I

COMMAND CODE NUMBERS

NUMBERS DESIRED IN N, T OR L POSITION	NUMBER WRITTEN IN COMMAND CODE
0 ↓	00 ↓
99	99
100	u0
101	u1
102	u2
103	u3
104	u4
105	u5
106	u6
107	u7
108	u8
109	u9
110	v0
111	v1
112	v2
113	v3
114	v4
115	v5
116	v6

STANDARD COMMANDS

		T or L_k	N	C	S	D	
Information Transfer							
Transfer word between addresses (neither address a two-word register)		T	N	0	S	D	
Transfer to or from two-word register used as temporary storage		T'	N	1	S	D	
Arithmetic Operations							
Clear and add to AR		T	N	1	S	28	
Clear and add absolute value to AR		T	N	2	S	28	
Clear and subtract from AR		T	N	3	S	28	
Add to AR		T	N	1	S	29	
Add absolute value to AR		T	N	2	S	29	
Subtract from AR		T	N	3	S	29	
Store sum or difference from AR		T	N	1	28	D	
Clear multiplication and division registers		L_3	N	0	23	31	
Load multiplicand, denominator or number to be shifted right		T	N	0	S	25	(T odd)
		T	N	6	S	25	(T even)
Load multiplier or number to be shifted left or normalized		T	N	0	S	24	(T odd)
		T	N	6	S	24	(T even)
Load numerator		T	N	0	S	26	(T odd)
		T	N	6	S	26	(T even)
Multiply		56	N	0	24	31	
Divide		57	N	1	25	31	
Store product		T	N	0	26	D	(T odd)
Store quotient		T	N	0	24	D	(T even)
Normalize MQ		v2	N	0	27	31	
Shift MQ left and ID right under control of command		T_p	N	1	26	31	
Shift MQ left and ID right under control of AR		v2	N	0	26	31	
Conditional Transfer of Control							
Test for non-zero		T	N	0	S	27	
Test if sign of AR negative		L_2	N	0	22	31	
Test for overflow		L_2	N	0	29	31	
Test for "Ready" state of Input-Output		L_0	L_0	0	28	31	
Command Channel Selection							
Select command line and mark	w	T	N	C	21	31	
Select command line and return		L_2	L_1	C	20	31	
Take next command from AR		L_2	N	0	31	31	
Extract Operations							
Extract and copy into ID the bits from PN that correspond to "one" bits of word T in Line 02	w	T	N	3	23	31	
Extract "one" bits of word in Line 21 that correspond to "one" bits of same-numbered word in Line 20		T	N	0	31	D	
Extract "one" bits of word in Line 21 that correspond to "zero" bits of same-numbered word in Line 20		T	N	0	30	D	
Special Commands							
Halt		L_2	N	0	16	31	
Ring bell		L_1	N	0	17	31	
Input-Output							
Permit alphanumeric type-in		L_5	N	4	12	31	
Permit numeric type-in		L_2	N	0	12	31	
Read punched tape		L_2	N	0	15	31	
Read magnetic tape		L_2	N	C	13	31	
Print AR in alphanumeric mode		L_5	N	4	08	31	
Print AR in numeric mode		L_2	N	0	08	31	
Print Line 19 in alphanumeric mode		L_5	N	4	09	31	
Print Line 19 in numeric mode		L_2	N	0	09	31	
Punch Line 19 on tape		L_2	N	0	10	31	
Write on magnetic tape	w	00	N	C	01	31	
Input-Output Control							
Reverse punched tape		L_2	N	0	06	31	
Search magnetic tape, forward		L_{16}	N	C	05	31	
Search magnetic tape, reverse		L_{16}	N	C	04	31	
Write file code on magnetic tape		L_5	N	C	30	31	

Commands for use of Differential Analyzer DA-1, Punched Card Couplers CA-1 and CA-2, Universal Code Accessory AN-1, and the special Input-Output Registers are described in Technical Bulletins.

DESCRIPTION OF STANDARD COMMANDS

Information Transfer

Transfer Word Between Addresses (neither address a two-word register) T N 0 S D

The information in word T of line S will be transferred to word T of line D, replacing the original contents of address D•T. Word S•T is not destroyed by the transfer.

Lines S and D may be long lines, short lines, or register AR, identified by their code numbers.

If a transfer is between a short line and a long line, the word-number of the short line that is involved in the transfer may be determined by dividing the word number in the long line by 4; the *remainder* is the word-number in the short line. Note that the contents of any address may be transferred to any other address with a different word time with two commands: transfer from the source address to AR (code T N 0 S 28), and transfer from AR to the destination address (code T N 0 28 D).

Transfer To or From Two-Word Register Used as Temporary Storage T N 1 S D

The information in address S•T is transferred to address D•T replacing the contents of address D•T. The word in S•T is not destroyed by the transfer. If T is an even number, the transfer will be to or from the even-numbered word in the two-word register. If T is an odd number, the transfer will be to or from the odd word in the two-word register. Because the number in the two-word register is not in the form of absolute value and sign this command is designed to be used twice: first as a transfer from a memory location to a two-word register and then as a transfer from the two-word register to another memory location (not another two-word register). The second transfer will automatically restore the number to the form of absolute value and sign.

The code numbers used to identify the two-word registers are:

MQ 24
ID 25
PN 26

Arithmetic Operations

Clear and Add to AR T N 1 S 28

The AR register will be cleared to zero. The number in address S•T will be transferred to the AR register in preparation for an addition or a subtraction. This command is to be used *only* in preparation for an addition or subtraction.

Clear and Add Absolute Value to AR T N 2 S 28

The AR register will be cleared to zero. The absolute value of the number in address S•T will be transferred to Register AR.

Clear and Subtract from AR T N 3 S 28

The number in Address S•T will be subtracted from zero and transferred to Register AR, replacing the original contents of Register AR. This command should be followed by either another Arithmetic Operation involving Register AR, or by the command "Shift MQ left and ID right under control of AR."

Add to AR T N 1 S 29

The number in address S•T will be added to the number in Register AR. Since the sum in Register AR may not be in the form of absolute value and sign, this command should be followed by a "Store Sum or Difference from AR" command, which will automatically put the stored sum in the form of absolute value and sign.

Add Absolute Value to AR T N 2 S 29

The absolute value of the number in Address S•T will be added to the number in Register AR.

Subtract from AR T N 3 S 29

The number in address S•T will be subtracted from the number in Register AR. Since the difference in Register AR may not be in the form of absolute value and sign, the command must be followed by a "Store Sum or Difference from AR" command which will automatically put the difference in the form of absolute value and sign.

Store Sum or Difference from AR — T N 1 28 D

The result of an addition or subtraction will be transferred from Register AR to address D•T. This order should be used only to store the result of an addition or subtraction.

Clear Multiplication and Division Registers — L_3 N 0 23 31

The contents of the MQ, ID and PN registers will be cleared to zero.

Load Multiplicand or Denominator

T N 0 S 25 (T odd)
T N 6 S 25 (T even)

The number in address S•T will be transferred to the ID register where it will serve as either the multiplicand in multiplication or as the denominator in division. If T is an odd number, use T N 0 S 25; if T is even, use T N 6 S 25. This command must be given before the "Load Multiplier" command for multiplication or before the "Load Numerator" command for division. (ID should not be loaded from the PN or MQ registers.) If T is even, S must be less than 28; and the information stored in Register AR before the command was obeyed will be lost.

Load Multiplier

T N 0 S 24 (T odd)
T N 6 S 24 (T even)

The number in address S•T will be transferred to the MQ register where it will serve as the multiplier for multiplication. If T is an odd number, use T N 0 S 24; if T is even, use T N 6 S 24. (MQ should not be loaded from the ID or PN registers.) If T is even, S must be less than 28; and the information stored in Register AR before the command was obeyed will be lost.

Load Numerator

T N 0 S 26 (T odd)
T N 6 S 26 (T even)

The number in address S•T will be transferred to Register PN where it will serve as the numerator for division. If T is an odd number, use T N 0 S 26; if T is even, use T N 6 S 26. (PN should not be loaded from the MQ or ID registers.) If T is even, S must be less than 28; and the information stored in Register AR before the command was obeyed will be lost.

Multiply — 56 N 0 24 31

The number in Register ID is multiplied by the number in Register MQ and the product is stored in Register PN. The word location of the "multiply" command should be odd; therefore, the number in the N position of the immediately preceding command should be odd. The contents of the ID and MQ registers are altered by the "multiply" command.

Divide — 57 N 1 25 31

The number in Register PN is divided by the number in Register ID and the quotient is stored in Register MQ. The word location of the "divide" command must be odd. The numerator in Register PN must be smaller in magnitude than the denominator in Register ID. The contents of Register PN are altered by the "divide" command.

Store Product — T N 0 26 D (T odd)

The product in Register PN is transferred to address D•T. D should not be a two-word register; the address T must be an odd number. (The most significant 28 bits of the product are in the "odd" word of Register PN.)

Store Quotient — T N 0 24 D (T even)

The quotient in Register MQ is transferred to address D•T. D should not be a two-word register; the address T must be an even number. (The quotient is the "even" word in Register MQ.)

Normalize MQ — v2 N 0 27 31

The contents of Register MQ will be shifted left until there are no zeros preceding the most significant "one" bit in the register. A count of the number of zeros lost will be added to the contents of Register AR. The word location of this command (that is, the N of the preceding command) should be odd. Since there may be two words in MQ, both words will be shifted until there is a "one" digit in the most significant digit position of the odd word location. (If the odd word contains all zeros, the shifting will be terminated by a "one" digit that was originally in the even

word location. In the latter case, the AR register will have been incremented 29 more times than the number of zeros preceding the first "one" in the even word of Register MQ.)

Shift MQ Left and ID Right
Under Control of Command T_p N 1 26 31

The contents of the MQ register will be shifted left and the contents of the ID register will be simultaneously shifted right under control of the integer T_p. T_p is made equal to twice the number of bit positions that the contents of MQ and ID are shifted. The word location of this command should be odd. (Bits shifted out of the odd word of MQ or the even word of ID are lost; zeros are introduced into the least significant bit of MQ and the most significant bit of ID during each shift.)

Shift MQ Left and ID Right
Under Control of AR v2 N 0 26 31

The contents of Register MQ will be shifted left and the contents of Register ID will be shifted right for a number of bit positions determined by the contents of AR. The number of desired shifts must be first transferred from a memory location to Register AR by use of the order "Clear and Subtract from AR." The word location of the shift command should be odd.

Conditional Transfer of Control

Test for Non-Zero T N 0 S 27

If the contents of address S•T are zero, the next command will be taken from word N of the command line; if the contents of address S•T are not equal to zero, the next command will be taken from word N + 1 of the command line.

Test if Sign of AR Negative L_2 N 0 22 31

If the contents of the AR register are positive or equal to zero, the next command will be taken from N. If the contents of the AR register are negative, the next command will be taken from N + 1.

Test for Overflow L_2 N 0 29 31

If, as a result of addition, subtraction or division, the contents of the AR, PN or MQ registers have, since this command was previously given, tried to equal or exceed one in magnitude, the next command will be taken from N+1. If the contents of the registers have remained between +1 and −1, the next command will be taken from N.

Test for "Ready"
state of Input-Output L_0 L_0 0 28 31

If the input-output circuitry is ready to handle new information, the next command will be taken from word L_1 of the command line. If the input-output circuitry is still processing earlier information, this test will be repeated as many times as necessary until the information has been completely processed.

Command Channel Selection

Select Command Line and
Mark ("Mark") w T N C 21 31

The next command will be taken from word N of line C (C may be 0, 1, 2, 3, 4, 5, 6 or 7. C values 0 through 5 represent lines 0 through 5. A "C" of 6 represents line 19. A "C" of 7 represents line 23). All subsequent commands will be taken from line C until another command channel is specified. Address T is remembered by the computer to be used if the "Return" command below is given.

Select Command Line and
Return ("Return") L_2 L_1 C 20 31

The next command will be taken from word T of line C, where T is the address specified in the "Mark" command, and line C is specified by the "Return" command. (This command will be used as an exit command from a sub-routine.) All subsequent commands will be taken from line C until another command channel is specified. If a break-point is placed in this command and BP operation is used, or if one-step operation (single-cycling) is used, the next command will be taken from command line C, word T; the halt will occur after execution of the command which follows the return command.

Take Next Command
from AR L_2 N 0 31 31

The next command will be taken from register AR. After the command in AR has been obeyed, the computer will return to the command line containing the "Take Next Command from AR" command.

Extract Operations

Extract and Copy into ID the bits from PN that correspond to "one" bits of word T in Line 02 — w T N 3 23 31

The bits from PN that correspond in position to "one" bits of word T in line 02 are extracted and copied into ID. These bit positions in PN are cleared. In positions corresponding to "zero" bits in line 02, ID will be cleared and the contents of PN will be unchanged. If T is even, the even words of PN and ID are affected; if T is odd, the odd words of PN and ID are affected.

Extract "One" Bits of Word in Line 21 that Correspond to "One" Bits of Same-Numbered Word in Line 20 — T N 0 31 D

The bits of a specified word in short line 21 are compared to the bits of the same numbered word in short line 20. Wherever there are "one" bits in corresponding bit positions in both words, a "one" in that bit position is transferred to address D•T; in other bit positions zeros are transferred. (The specified word in the short lines is determined by dividing T by four: the remainder (0, 1, 2, or 3) is the specified word position.)

Extract "One" Bits of Word in Line 21 that Correspond to "Zero" Bits of Same-Numbered Word in Line 20 — T N 0 30 D

The bits of a specified word in short line 21 are compared to the bits of the same numbered word of short line 20. In each bit position where a "one" in line 21 corresponds to a "zero" in line 20, a "one" bit is transferred to address D•T; in other bit positions, zeros are transferred. (The specified word in the short lines is determined by dividing T by four: the remainder (0, 1, 2, or 3) is the specified word position.)

Special Commands

Halt — L_2 N 0 16 31

Computation in the computer will come to a halt. To resume operation, the "compute" switch must be switched to the middle position and back to the "GO" or "BP" position. Computation will resume with the command in word position N.

Ring Bell — L_1 N 0 17 31

A single note chime is sounded, signalling that the point of computation at which the command is located has been reached. The command must not be executed more often than once every three drum cycles.

Input-Output

Permit Alphanumeric Type-In — L_5 N 4 12 31

After this command is executed, alphanumeric information may be typed into the computer. Each typewriter key pressed will enter an 8-bit code into Line 23.

Permit Numeric Type-In — L_2 N 0 12 31

After this command is given, numeric information may be typed into the computer. Each numeric key pressed will enter a 4-bit code into Line 23.

Read Punched Tape — L_2 N 0 15 31

The paper tape will be moved in the forward direction and read into Line 19 until a stop code appears. Tape is read at approximately 250 characters per second.

Read Magnetic Tape — L_2 N C 13 31

The magnetic tape is run in the forward direction and is read into Line 19 until a STOP code appears on the tape. File codes on the tape have no effect. If this command:

(1) Follows a "Search Magnetic Tape" command for any tape unit, at least 16 drum cycles must elapse after the computer has reached the READY state before the "Read" command is given.
(2) Follows a "Write on Magnetic Tape" command, at least 4 drum cycles must elapse after the READY state is reached before the "Read" command is given.
(3) Follows a "Write File Code" command, at least 4 drum cycles must elapse after the execution of the file code command before the "Read" command is given.
(4) Follows a "Read Magnetic Tape" command, at least 15 word times must elapse after the computer has reached the READY state before the second "Read" command is given.

The number of the tape unit, 0, 1, 2, or 3, is put in the "C" position of the command.

Print AR in
Alphanumeric Mode L_5 N 4 08 31

The contents of Register AR will be typed out. The order of read-out is from most significant bit to least significant bit. Read-out proceeds until AR is cleared or until an 8-bit code group is encountered in which the first 4 bits are zeros.

Print AR in Numeric Mode* L_2 N 0 08 31

The contents of Register AR will be typed out under control of format words held in Line 03 (see Appendix). The order of read-out is from most significant bit to least significant bit. If standard format is used, AR will be cleared to zero.

Print Line 19 in
Alphanumeric Mode L_5 N 4 09 31

The contents of Line 19 will be typed out, eight bits for each character. The order of read-out is from highest-numbered (word 107) to lowest-numbered word and from most significant (bit 29) to least significant bit. Read-out continues until Line 19 contains all zeros or until an 8-bit code group is encountered in which the first 4 bits are zeros.

Print Line 19 in
Numeric Mode* L_2 N 0 09 31

The contents of Line 19 will be typed out under control of format words held in Line 02 (see Appendix). The order of read-out is from highest-numbered to lowest-numbered word and from most significant to least significant digit. Line 19 is cleared to zero.

Punch Line 19 on Tape L_2 N 0 10 31

The contents of Line 19, starting with word 107, will be punched on tape under control of format words held in Line 02 (see Appendix). Line 19 is cleared to zero.

Write on
Magnetic Tape w 00 N C 01 31

The contents of Line 19 will be written on magnetic tape at the rate of 430 sexadecimal characters per second. When Line 19 is empty a stop code will be written on the tape and the computer will assume the "READY" state. If this command follows a "Search Magnetic Tape" command for any tape unit, at least 16 drum cycles must elapse after the READY state is reached before the "Write" command is given.

If this command follows an earlier "Write on Magnetic Tape" or "Read Magnetic Tape" command, at least 15 word times must elapse after the READY state is reached before the "Write" command is given.

*In the numeric mode, with the Punch switch ON, simultaneous typeouts and punchouts will occur.

Input-Output Control

Reverse Punched Tape L_2 N 0 06 31

The paper tape is run in the reverse direction for one block length. A block length is the space on the tape between two stop codes.

Search Magnetic
Tape Forward L_{16} N C 05 31

The magnetic tape is run in the forward direction until a file code appears. The tape stops in front of the block following the file code. The information is *not* read into the memory. If this command follows a "Write on Magnetic Tape" command, at least 4 drum cycles must elapse after the READY state is reached before the "Search" command is given.

If this command follows a "Write File Code" command, at least 4 drum cycles must elapse after execution of the file code command before the "Search" command is given.

If this command follows a "Search Magnetic Tape" or "Read Magnetic Tape" command, at least 15 word times must elapse after the computer has reached the READY state before the "Search" command is given.

Search Magnetic
Tape Reverse L_{16} N C 04 31

The magnetic tape is run in the reverse direction until a file code appears on the tape. The tape stops in the blank area before the file code. The timing instructions listed under "Search Magnetic Tape Forward" apply here also.

Write File Code on
Magnetic Tape L_5 N C 30 31

A file code is written on the tape in Tape Unit "C." The tape is not moved.

This command should be preceded by a series of commands which move the tape slightly in order to provide a small blank area before the file code is recorded. Such a series is:

w	00	01	C	01	31
	03	03	0	00	31
	02	02	0	00	00

The command "Write File Code" would then be put in word position 02.

EXAMPLE 1
ADDITION AND SUBTRACTION

Problem: Write a program to find and store the sum of a + b − c.

Storage Locations: a is in line 05, word 6.
b is in line 22, word 2.
c is in line 16, word 32.
sum is to be put in line 09, word 104.

Program:

Word Position of Command	Command Code					Explanation
	T	N	1	S	28	*Clear and add*
00	06	07	1	05	28	AR is cleared and a (S = 05, T = 06) is added to AR.
	T	N	1	S	29	*Add*
07	10	18	1	22	29	b (S = 22, T = 10) is added to AR. The quantity a + b is now in AR. (10 ÷ 4 gives a remainder of 2, the word number of b in short line 22.)
	T	N	3	S	29	*Subtract*
18	32	61	3	16	29	c (S = 16, T = 32) is subtracted from AR. The quantity a + b − c is now in AR.
	T	N	1	28	D	*Store*
61	u4	u5	1	28	09	The sum in AR is stored in line 09, word 104 (D = 09, T = 104 = u4).

In this and other examples, the box contains the standard command code. Below the box is shown the command in the form to be written on the coding sheet for entry in the computer.

A typical coding sheet is shown accompanying Example 5. Symbol "P" on the coding sheet specifies a prefix which is sometimes used. The locations, 0 through u6, on the left side of the coding sheet indicate the storage locations that have been used.

EXAMPLE 2
MULTIPLICATION

Problem: Write a program to find and store the product of a times b.

Storage Locations: a is in line 04, word 61.
b is in line 14, word 76.
product is to be put in line 12, word 35.

Program:

Word Position of Command	Command Code					Explanation
	L_3	N	0	23	31	*Clear*
52	55	58	0	23	31	The multiplication and division registers are cleared. (L_3 = word position of command plus subscript = 52 + 03 = 55)
	T	N	0	S	25	*Load ID*
58	61	65	0	04	25	a (In address S = 04, T = 61) is loaded as the multiplicand. Since T is odd, the form of this command for "T odd" is used.
	T	N	6	S	24	*Load MQ*
65	76	79	6	14	24	b (In address S = 14, T = 76) is loaded as the multiplier. Since T is even, the form of this command for "T even" is used.
	56	N	0	24	31	*Multiply*
79	56	32	0	24	31	a, the multiplicand, is multiplied by b, the multiplier.
	T	N	0	26	D	*Store*
32	35	36	0	26	12	The product is stored in line 12, word 35 (D = 12, T = 35).

EXAMPLE 3
DIVISION

Problem: Write a program to find and store the quotient of a divided by b.

Storage Locations: a is in line 09, word 103.
b is in line 10, word 83.
quotient is to be put in line 15, word 62.

Program:

Word Position of Command	Command Code					Explanation
	L_3	N	0	23	31	*Clear*
71	74	74	0	23	31	The multiplication and division registers are cleared (L_3 = word-position of command + subscript = 71 + 03 = 74).
	T	N	0	S	25	*Load ID*
74	83	88	0	10	25	b (in address S = 10, T = 83) is loaded as the denominator. Since T is odd, the form of this command for "T odd" is used.
	T	N	0	S	26	*Load PN*
88	u3	u5	0	09	26	a (in address S = 09, T = 103 = u3) is loaded as the numerator. Since T is odd, the form of this command for "T odd" is used.
	57	N	1	25	31	*Divide*
u5	57	60	1	25	31	a, the numerator, is divided by b, the denominator.
	T	N	0	24	D	*Store*
60	62	63	0	24	15	The quotient is stored in line 15, word 62 (D = 15, T = 62).

MODIFICATION OF COMMAND CODES

In the interests of simple, concise programming the G-15 command structure has been made flexible rather than rigid: The codes in the Standard Command List may be modified to permit computation on double-length numbers; multi-word transfers and multi-word operations on information may be ordered by a single command; true minimum access coding may be made almost automatic; many operational short-cuts may be programmed. The added diversification is obtained by altering the command codes in the manners shown below.

Double-Length Numbers

A double-length number is one, with too many digits for a single address, which is held in two consecutive word positions. A double-length number consists of 57 bits and sign rather than 28 bits and sign. The more significant 29 bits of the number are stored at an odd-numbered address; the less significant 28 bits and sign are stored at the even address numbered one less than that holding the more significant bits. To specify that a command should apply to two consecutive addresses, the contents of which are to be considered a double-length number, a value of 4 is *added* to the digit of the command code that follows N. Such a command is called a double-precision command. The addresses affected by the order would then be the address in the T position of the command and the next consecutive address, namely addresses T and T + 1. (T must be even).

EXAMPLE: The command 36 N 4 14 07 would cause the contents of words 36 and 37 in line 14 to be transferred to word positions 36 and 37 of line 07.

All commands in the groups "Information Transfer" and "Extract Operations" may be treated as double-precision commands. All commands in the group "Arithmetic Operations," except those involving Register AR, may be made to be double-precision commands. Register AR cannot be used for double-length addition or subtraction since the register can hold but one word. Register PN is used for double-precision addition or subtraction in almost the same manner as AR is used for the single-precision operation. The necessary new commands to accomplish this are:

Clear and add to PN (double-precision)	T	N	5	S	26
Add to PN (double-precision)	T	N	5	S	30
Add absolute value to PN (double-precision)	T	N	6	S	30
Subtract from PN (double-precision)	T	N	7	S	30
Store Sum or Difference from PN (double-precision)	T	N	5	26	D

In using these commands T must be an even number.

To enter double-precision numerators, denominators, multipliers and multiplicands, use the corresponding single precision commands except for C which should be 4.

The commands "multiply" and "divide" assume a slightly different form if double-precision:

Multiply (double-precision)	v4	N	0	24	31
Divide (double-precision)	v6	N	1	25	31

Multiplication

In multiplication one bit of the multiplier is processed every two word-times beginning with the most significant bit. Therefore, if less than a full precision product (28 or 57 bits for the single and double precision cases, respectively) is required, the multiplication may be terminated early. To do so, and make a consequent saving in computation time, a number should be placed in the "T" position of the command that is equal to twice the number of bits precision desired in the product.

Every two word-times that the "multiply" command is executed, the contents of Register MQ are shifted left one bit and the contents of Register ID are shifted right one bit.

Division

In division one bit of the quotient is developed every two word-times the command is permitted to operate by the number in its "T" position. The most significant bit of the quotient is developed first and is stored in the less significant end of the even word in Register MQ. Every following pair of word-times that the "division" command is executed, the contents of Register MQ are shifted left one bit position, and an additional quotient bit of lesser significance is developed in the register. Consequently, if the division is terminated early (i.e., before 28 or 57 bits on the single or double precision cases respectively have been developed) the quotient appears scaled down by the factor 2^{-n} where n is the number of bits not developed to full significance. This often means that no time is saved in not fully developing a quotient even when not all of it is needed. An automatic round-off is included in the division process within the computer: the least significant numerical bit in the quotient is always made to be a one.

Division Overflow

When the division command is executed for an odd number of word-times, an overflow occurs if a "one" bit is in the most significant bit position of the even word in MQ just prior to the last word-time the command is operative. When the division command is executed for an even number of word-times, an overflow occurs if a "one" bit is in the most significant bit position of the odd word in MQ just prior to the last pair of word-times the command is operative. Note that as a consequence an overflow will occur when the quotient is one or greater for either a single-precision command operating for 57 word-times or a double-precision command operating for 116 word-times.

Normalize

The numbers in the registers specified by the "Normalize" command in the Standard Command List may be shifted one digit position each two word times. Since the number in the "T" position of the command indicates the number of word-times the command is permitted to be operative, a smaller number may be written in the "T" position if the programmer knows that the maximum number of permissible shift times will not be necessary. There will be a consequent saving in computation time.

Effect of Changing the "C" Value in Shift and Normalize Commands

A different value of C in the command "Normalize MQ" or "Shift MQ Left and ID Right Under Control of Command" changes the effect of the command on the contents of Register AR.

NORMALIZE MQ

C value of:

0 . . . . . See page 7.

1 . . . . . Operation is the same as with a "C" of 0 except that the contents of Register AR are unchanged. A count of the number of zeros lost is not added to the contents of Register AR.

2 . . . . . Operation is the same as with a "C" of 1.

3 . . . . . Operation is the same as with a "C" of 1.

SHIFT MQ LEFT AND ID RIGHT
UNDER CONTROL OF COMMAND

C value of:

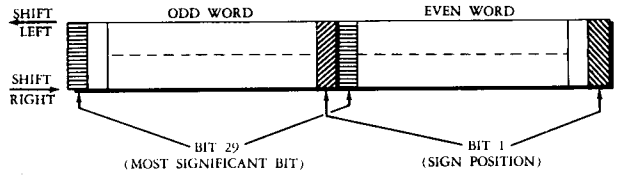
1 . . . . . See page 8.

2 . . . . . Operation is the same as with a "C" of 1.

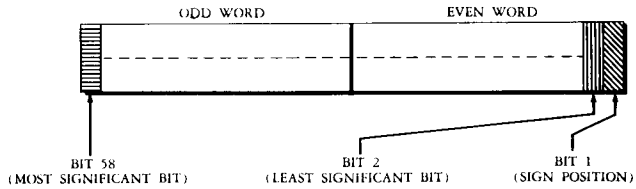
3 . . . . . Operation is the same as with a "C" of 1.

0 . . . . . Operation is the same as with a "C" of 1 except that the contents of Register AR are changed. A number equal to the number of shifts $\left(\frac{T_D}{2} \cdot 2^{-28}\right)$ is added to the contents of Register AR. Register AR must not be forced to exceed its capacity $(1\text{-}2^{-28})$ by the incrementing of its contents, and should not initially contain a negative number.

2 SINGLE PRECISION NUMBERS
IN 2 WORD REGISTER



1 DOUBLE PRECISION NUMBER
IN 2 WORD REGISTER



CONTENTS OF 2 WORD REGISTER

Figure 2

Multi-Word Operations

A multi-word operation is one which may be specified by a single command but which applies to a number of consecutive words in a drum channel rather than a single word. All commands in the groups, "Information Transfer" and "Extract Operations" may be treated as multi-word operations. "Add to AR" and "Subtract from AR" in the group "Arithmetic Operations" may be treated as multi-word operations.

Block Commands

The command form which specifies a multi-word operation is called a "block" command. A block command becomes effective for the word in S or D numbered one greater than the word position of the block command itself, and remains effective for every word up to, but not including, the word number in the T position of the command.

(Symbolically, this may be expressed as word positions L_1 to T-1 inclusive.) To specify a command to be a block command a "u" is added as a prefix to the command word.

EXAMPLE: The command u 47 N 1 S 29, if located in position 41 of the command line, would cause the contents of consecutive addresses S•42 to S•46 inclusive to be added to the contents of the AR register.

Other Transfer Commands

Two additional orders are useful in permitting certain transfers to be done by a single command which would otherwise

require two commands:

Interchange Storage with AR T N 2 S D

The contents of address S•T will be transferred to Register AR, and the original contents of Register AR will be transferred to address D•T. Addresses S•T and D•T may be long lines or short lines; neither may be a register.

Transfer Word Between Addresses With One Word Delay u L_3 N 2 S D

The word in address $S \cdot L_1$ is transferred to address $D \cdot L_2$. Address $S \cdot L_2$ is transferred to Register AR. The original contents of Register AR are transferred to address $D \cdot L_1$. Addresses in the command may be long lines or short lines; they may not be registers.

Programmed Alteration of Commands

A command may be altered during the course of computation by the program itself. An arithmetic value can be added to, or subtracted from, a portion of a command so that the nature of the command is changed. The arithmetic value which performs this function is called an increment.

The increment, if previously known by the programmer, can be stored in the memory of the computer. Or the increment can be determined by internal computation. In either case, its value should neither be negative nor cause the portion of the command it modifies to exceed its limits. For example, if the T portion of a command is affected, T should not be forced below 0 or above 107.

If the value of the increment is known, it can be entered into the computer as a command word by use of the procedure specified for entering a command. The modifying increment is placed in those digit positions of the word that correspond to the portion of the command to be altered. For example, the increment 00 02 0 00 00 would modify the N portion of a command.

The value of an increment may not be initially known. It may be necessary to alter a command in a manner which is a function of some intermediate result obtained during computation. In this case, after the increment has been internally computed, it is shifted left into those digit positions which will modify the proper portion of the command. The correct number of left shifts for modifying the various portions of a command are given in the table below. The table assumes that the number to be shifted appeared originally as a positive integer stored in the least significant digits of the word.

Portion of Command to be Incremented	Required Number of Left Shifts
D	0
S	5
N	12
L or T	20

The commands used for altering a command are:

Enter Command into AR for Alteration T N 0 S 28

S•T is the address of the command being altered.

Either:

Alter Command by Adding Increment T N 0 S 29

Or:

Alter Command by Subtracting Increment T N 3 S 29

S•T is the address of the altering increment.

The modified command can be executed immediately from Register AR or can be stored in the memory. The command for the latter operation is:

Store Altered Command From AR T N 0 28 D

D•T is the address in which the command is stored.

Minimum Access Coding

The term "minimum access coding" refers to the addressing of a set of commands in a manner such that, when the computer is instructed to search for a specific address, the waiting time for that address to appear under the drum "read" heads is reduced to a minimum. Ideally, the drum address at which one operation ends should immediately precede the address of the next command. This ideal situation may be programmed on the G-15. To do so, it is necessary to keep track of word positions and the number of words required to perform the operation indicated by the command.

The number of word times required for the execution of the commands in the Standard Command List are tabulated below.

Type of Command	*Word Times for Execution*
Commands Coded with "T" in Standard List.	
Single-precision, not block command, not command to load multiplier, multiplicand, denominator or numerator.	1
Double-precision, not block command, not command to load multiplier, multiplicand, denominator or numerator.	2
Load multiplier, multiplicand,denominator or numerator and T odd, single-precision.	1
Load multiplier, multiplicand, denominator or numerator and either T even, or T odd, double-precision.	2

Type of Command	*Word Times for Execution*
Block commands.	Number in T position $-L_1$
Commands coded with L and numerical subscript K in Standard List.	K-1
Command coded with neither T nor L and a numerical subscript (specifically, "multiply," "divide," "normalize," and the shift commands).	First two digits of command code.

The word position in which a command is read from the drum must precede the word positions during which it is executed as the two operations cannot be simultaneous. Therefore, if a command is a normal command, which includes in its code a specific address S•T or D•T, the number in the T position of the command should be one greater than the number of the command itself (Symbolically: $T = L_1$) for minimum access coding. The number put in the N position of the command (which is the number of the next command) should be the *next consecutive* address after the addresses specified by the command. (Symbolically, if $T = L_1$: $N = L_2$ for single-precision commands; $N = L_3$ for double-precision commands.)

For commands made to be block operations, or those not normally coded with the symbol "T," the number in the N position of the command should be the *next consecutive* address after the number of word-times required to perform the command are added to the number of the command itself. (Symbolically: L_0 + WORD-TIMES REQUIRED TO CARRY OUT COMMAND + 1 = N. It must also be remembered that word 00 follows word 107.)

Building Additional Commands

By modifying their codes, a great many additional commands may be written for the G-15. In this manner, a command can be "custom-made" to fit a specific purpose. The programmer is not limited, as in other computers, to a fixed command list. Building a command to suit a particular purpose from G-15 command components is described in other Bendix Computer publications.

EXAMPLE 4

COMPLETE MINIMUM ACCESS CODING

Problem: Re-write the program of Example 2 so that operand addresses as well as command locations obey the considerations for minimum access coding.

Program:

Word Position of Command	Command Code					Explanation
	L_3	N	0	23	31	*Clear*
52	55	55	0	23	31	
	T	N	6	S	25	*Load ID*
55	56	58	6	04	25	The command is read during word-position 55. Therefore, drum access time can be reduced to zero if the operand, a, were stored in word-position 56. Two word times are required for the execution of this command since T is even. The command will be executed during word-positions 56 and 57. N is then made equal to 58.
	T	N	0	S	24	*Load MQ*
58	59	61	0	14	24	The command is read during word-position 58. Therefore, drum access time can be reduced to zero if the operand, b, were stored in word-position 59. The command will be executed during word position 59. N would be made equal to 60 except the next command, multiply, must be in an odd location. Therefore, N is made equal to 61.
	56	N	0	24	31	*Multiply*
61	56	10	0	24	31	The command is read during word-position 61, and is executed for the next 56 word times. N is therefore made to be the word position immediately following the end of the execution of the command, or 61 + 56 + 1 = 118. Since this figure is greater than 107, a value of 108 must be subtracted. N = 118 − 108 = 10.
	T	N	0	26	D	*Store*
10	11	12	0	26	12	The command is read during word-position 10. Since T must be odd in this command, the command is executed during the next available word-position, 11, and the product is stored in word-position 11 of line 12.

EXAMPLE 5

SUM OF THE SQUARES OF 50 NUMBERS

Problem: Write a program to sum up the squares of fifty 3-digit numbers. The numbers may range in value: $0 \leq N \leq 999$.

The fraction point for these numbers is assumed to be 14 places (bit positions) to the right. In the sum of the squares, the fraction point is assumed to be 28 places to the right, at the extreme end of the word.

Storage Locations:

n_1	is in	line 10, word 01
n_2	is in	line 10, word 02
.	.	.
.	.	.
.	.	.
.	.	.
n_{50}	is in	line 10, word 50

A sexadecimal code, w000000, is stored in line 00, words 52, 57 and 99. This code functions as a "flag"; when the test for non-zero indicates a zero condition, then all the numbers will have been processed.

Program:

Word Position of Command	Command Code					Explanation
	T	N	0	S	D	*Transfer word between addresses*
98	99	00	0	00	28	The code is copied from 00:99 into AR.
	u L_3	N	2	S	D	*Transfer word between addresses with one-word delay*
00	u 51	51	2	10	10	Each time this command is executed precession occurs, that is, the contents of 10:01 are copied into 10:02, the contents of 10:02 into 10:03 . . . the contents of 10:49 into 10:50. The contents of AR are copied into 10:01 and the contents of 10:50 are copied into AR.
	T	N	3	S	29	*Subtract from AR*
51	52	53	3	00	29	The code stored in 00:52 is subtracted from the number in AR.
	T	N	0	S	27	*Test for non-zero*
53	54	55	0	28	27	AR is tested for non-zero. If the contents of AR are non-zero, the next command is taken from word 56 (N+1). If the contents of AR are zero, the next command is taken from word 55 (N).
	T	N	1	S	29	*Add to AR*
56	57	58	1	00	29	The code in 00:57 is added to AR to restore its contents to the value held there before the subtraction.
	L_3	N	0	23	31	*Clear multiplication and division registers*
58	61	61	0	23	31	The contents of the MQ, ID and PN registers are cleared to zero.
	T_{odd}	N	0	S	25	*Load multiplicand*
61	63	64	0	28	25	The multiplicand in AR is loaded into the ID register.
	T_{odd}	N	0	S	24	*Load multiplier*
64	65	67	0	28	24	The multiplier in AR is loaded into the MQ register.

Program:

Word Position of Command	Command Code					Explanation
	56	N	0	24	31	*Multiply*
67	28	96	0	24	31	The contents of ID are multiplied by the contents of MQ, that is, the number is squared. There are 14 bits of significance in the multiplier. Two word-times are required for each bit to form the full 28-bit precision product.
	T	N	0	S	D	*Transfer word between addresses*
96	97	00	0	26	28	The product is copied from the PN register to AR. The next command is taken from word 00, the beginning of the loop. Precession occurs again, the contents of AR are copied into 10:01, and the contents of 10:50 are copied into AR. After the last iteration, the square of each number will be stored in 10:01 through 10:50.
	T	N	1	S	28	*Clear and add to AR*
55	01	01	1	10	28	When the code is subtracted from itself (AR = O), all the numbers have been processed. AR is cleared and the contents of 10:01 are added to AR.
	u T-1	N	1	S	29	*Block add to AR*
01	u 51	54	1	10	29	This command is a block command effective for the word numbered one greater than the location of the command, word 02, and operative up to T-1, or word 50. The sum of the squares are now in AR.

Bendix Computer — Los Angeles 45, California

Page ___ of ___

Date: _____

G-15

PROGRAM PROBLEM: Example 5: Sum of the Squares of 50 Numbers

Prepared by _____

Line 0

~~0~~	~~1~~	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	~~51~~
~~52~~	~~53~~	54	~~55~~
~~56~~	~~57~~	~~58~~	59
60	~~61~~	62	63
~~64~~	65	66	~~67~~
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
~~96~~	97	~~98~~	~~99~~
u0	u1	u2	u3
u4	u5	u6	

L	P	T or Lk	N	C	S	D	BP	NOTES
98		99	00	0	00	28		Transfer "code" from 00:99 to AR
99								Code, w000000, stored in 00:99
00	u	51	51	2	10	10		Transfer word between addresses with 1-word delay
51		52	53	3	00	29		Subtract code from AR
52								Code, w000000, stored in 00: 52
53		54	55	0	28	27		AR = 0 → Test AR for zero
56		57	58	1	00	29		AR ≠ 0 → Add code in 00: 57 to AR
57								Code, w000000, stored in 00:57
58		61	61	0	23	31		Clear multiplication and division registers
61		63	64	0	28	25		Load ID from AR
64		65	67	0	28	24		Load MQ from AR
67		28	96	0	24	31		Multiply (14 bits precision = 28 word times)
96		97	00	0	26	28		Place product in AR
55		01	01	1	10	28		Clear and add 10:01 ←
01	u	51	54	1	10	29		Add 10:02 thru 10:50 to AR

APPENDIX

WRITING COMMANDS IN DIRECT MACHINE LANGUAGE

All information, stored in and processed by the computer, is in binary form. Commands may be written, and information entered and removed, in the decimal form described in this manual by use of the standard Program Preparation Routine; commands and numbers are automatically converted to binary form. The Program Preparation Routine is described in the "Operating Manual for the G-15."

If desired, commands may be entered into the computer in binary form, in direct machine language, by not using the Program Preparation Routine. The 29 bits of the command would then be written in the manner below. The bits are numbered from 1 to 29, from right to left. Internally, all commands have this form in the computer.

BINARY POSITION

(from left to right)

29 ..... 1 normally; 0 if command is block command.

28-22 ..... T or L as a 7-bit binary number

21 ..... 0 normally; 1 if command is coded for break-point operation.

20-14 ..... N as a 7-bit binary number.

13-12 ..... C as a 2-bit binary number (if command is double-precision, subtract 4 from C before insertion in command)

11-7 ..... S as a 5-bit binary number

6-2 ..... D as a 5-bit binary number

1 ..... 0 for single precision; 1 for double precision.

For the manner in which binary information may be entered directly into the computer, see "To Control Internal Operation from Typewriter" in the Bendix G-15 Operating Manual.

A NOTE CONCERNING THE PROGRAM PREPARATION ROUTINE

The standard Program Preparation Routine which makes the decimal-to-binary conversion has also been designed to interpret certain commands, which have been written without the prefix "u," as block commands:

Standard commands which end with the code digits "31" are transformed to block commands by the Program Preparation Routine.

Normal commands in a location such that T is equal to L_1 will be transformed by the Program Preparation Routine into the form of a block command operative on the same addresses. The transformation reduces computation time. Because of internal design there is a delay of one word time after L_0 before a normal command can be operative. This inherent delay does not exist for a block command. Therefore, if T is equal to L_1 and the command were not transformed, there would be a delay of one drum revolution before the command would be executed.

The automatic transformation to block command form can be prevented, if desired, by adding the prefix "w" to the command.

OUTPUT FORMAT CONTROL

Format control for alphanumeric information typed out of line 19 or AR is obtained by typing in each character and control function desired for the output copy. For example, if a company name followed by a carriage return is needed in the output format, then during input the name of the company and the carriage return are typed in where required. When alphanumeric information is punched on paper tape or written on magnetic tape, the numeric mode of operation is used.

Information read out of the computer in numeric mode is under the control of four words held in either line 02 or 03. Words 00 through 03 of line 02 are used to hold data which determine the form of information typed out in numeric mode or punched out from line 19; similarly, words 00 through 03 of line 03 hold data controlling the form of numeric information typed out of AR.

The format characters used in the numeric mode, and their functions, are described below. Each format character is represented by a three-bit code shown in Table II. These characters are used serially. Word 107 (u7) of line 19 is the first to be processed; correspondingly, the first format character is taken from the three most significant bits of word 03. Succeeding format characters are taken from successive three-bit groups.

Sign: This character causes bit one of the most significant word that has not yet been read out (the bit which holds sign information) to be typed out as a space if "zero" or as a "—" sign if "one." In each word the sign type-out should precede the digit type-out. (Internally, there is no shift of information in the output line as a result of sign read out.)

Digit: This character causes the information in the four most significant binary places of the highest numbered word in line 19, or in Register AR, that has not yet been read, to be read out as a single decimal or sexadecimal digit. The digit that is read is lost. (Internally, information in the line is shifted in the more significant direction by four bits.)

Period: A period character causes a period (decimal point) to be typed without affecting the information held in the output line.

Tab: A tab character causes the tab key to be pressed; the tab must come at the end of each complete number. (Internally, the output line is shifted one bit in the more significant direction. Compensation is thus made for the absence of a shift during sign read-out.)

Carriage Return: This character effects the same changes in the information line as the Tab while causing the carriage return key to be pressed.

Wait: A Wait character in the format will inhibit the typeout of the corresponding four-bit digit. It is useful in preventing the printing of digits which lack significance.

End: The End character resets the input-output control signals to the ready state. If line 19 is being typed, the End character will automatically be changed to a Reload character if any non-zero information remains in line 19.

Reload: In the event that more than 38 format characters are needed, the format may be reloaded by the Reload character. This allows for the typing of up to a full line of 108 words with a single command and without stopping computation. Note that the Reload character never appears in lines 02 and 03, but results only from a changed End character.

Two standard formats are often used: the 28 bits representing the magnitude of a binary number are grouped into seven four-bit digits, preceded in output by the sign. The standard format for numeric output from line 19 provides for the typeout of four words, separated by tabs and followed by a carriage return. Corresponding codes are punched on tape. The standard format for numeric typeout from AR causes the sign of AR to be typed followed by seven digits and a carriage return. The proper control words in lines 02 and 03 for standard format may be automatically entered in the manner described in the G-15 Operating Manual.

TABLE II

FORMAT CHARACTERS FOR NUMERIC MODE

Character	Binary Code
Digit	000
End	001
Carriage Return	010
Period	011
Sign	100
Reload	101
Tab	110
Wait	111

INDEX

Offices:

BOSTON 16

607 Boylston Street
COngress 2-9110

CHICAGO 11

919 N. Michigan Avenue
MIchigan 2-6692

CLEVELAND 13

55 Public Square
CHerry 1-7789

DALLAS 1

1511 Bryan Street
RIverside 7-8805

DENVER 3

655 Broadway
Suite 910
ALpine 5-1403

DETROIT 37

12950 West Eight Mile Road
JOrdan 6-8789

HUNTSVILLE, ALA.

Holiday Office Center
Memorial Parkway, South
539-8471

KANSAS CITY 11, MO.

3430 Broadway
VAlentine 1-8681

LOS ANGELES

291 S. La Cienega Blvd.
Beverly Hills, California
OLeander 5-9610

NEW YORK 17

205 East 42nd Street
Room 1205
ORegon 9-6990

SAN FRANCISCO

1330 Broadway
Suite 1121
Oakland 12, California
GLencourt 2-3664

TULSA 14

1754 Utica Square
RIverside 3-6485

WASHINGTON 6, D. C.

1000 Connecticut Avenue, N.W.
STerling 3-0311

CANADA

Computing Devices of Canada
P. O. Box 508
Ottawa 4, Ontario, Canada
TAlbot 8-2711

OTHER COUNTRIES

Bendix International Division
205 E. 42nd Street
New York 17, New York
MUrray Hill 3-1100

Bendix Computer Division
LOS ANGELES 45, CALIFORNIA

THE Bendix CORPORATION

April 1961

Publication APG-01571-3
Formerly T 3-3