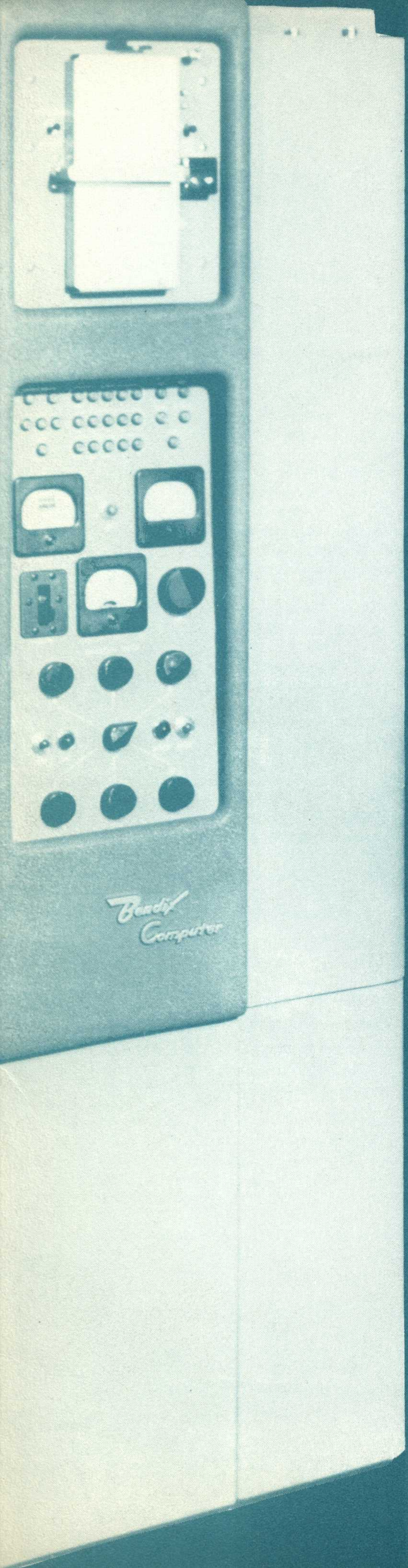


POGO COMPILER FOR THE BENDIX G-15 GENERAL PURPOSE DIGITAL COMPUTER

POGO, Program Optimizer for G-15 Operation, prepares an optimum machine program from simple commands.

Preliminary Manual · January 1959

THE CONTENTS IN BRIEF

PROGRAMMING WITH POGO

Introduction

The Pogo Compiler permits a programmer, who is not familiar with Bendix G-15 machine language programming, to prepare a machine language program for the computer. Therefore, the advantages of power and speed inherent in machine language programming may be obtained with a minimum of training.

A program is prepared from the simplified commands listed on pages 3 through 5. They are similar in form to commands for the Intercom 1000 programming system.

The compiler transforms each simplified command into a series of machine language commands and selects memory locations for the machine language commands which minimize computation time. The compiler then punches the machine language program on paper tape. The program may be executed at any time from the paper tape.

Information is handled in the form of decimal fractions; each fraction consists of seven digits and sign.

Each simplified command specifies a single address. An arithmetic command operates on the contents of the specified address and on the contents of an accumulator.

Memory

The memory of the G-15 Computer consists of twenty long channels numbered from 00 through 19 and four short rapid-access channels numbered from 20 to 23.

Each long channel has a capacity of 108 words. The word positions in each channel are numbered from 00 to 99 and u0 to u7. Each short channel has a capacity of four words; the word positions are numbered from 00 to 03.

The combination of a channel number and a word position number is called an address. It is usually abbreviated ADDR.

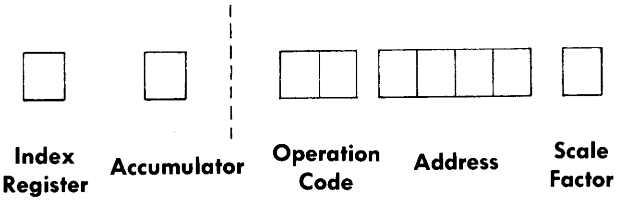
Word position u7 should not be used in an address.

The Pogo Compiler can place programs in long channels 06 through 18. The channels 00 through 05 contain facilities for control and check-out of the machine language program. Channel 19 may be used to store data generated by the program.

Command Structure

A simplified command consists of seven, eight or nine digits.

FORM OF A COMMAND



Every command contains the seven digit positions shown to the right of the dotted line. The two digit positions to the left of the dotted line are optional.

An arithmetic operation - addition, subtraction, multiplication or division - involves two numbers. One of the numbers is held in the location specified by the address portion of the command; the other number is held in one of a group of special memory locations called "accumulators".

Accumulators

The G-15 programmer using the Pogo Compiler has seventeen different accumulators available for his use. The accumulator is specified in the position of the command code indicated above under "FORM OF A COMMAND" by a digit from 0 through 9, u, v, w, x, y or z. If no accumulator number is explicitly specified in that position, the command operates on Accumulator AR.

All accumulators are addressable locations that can be put in the ADDR portion of a command. Accumulator numbers and their locations are:

A	ADDRESS
0	2000
1	2001
2	2002
3	2003
4	2100
5	2101
6	2102
7	2103
8	2200
9	2201
u	2202
v	2203
- - - - - - -	- - - - - - -
w	2300
x	2301
y	2302
z	2303
(AR)	2800

The accumulators below the dotted line, w, x, y, z and AR, do not retain their contents during an input operation.

The contents of Accumulator AR are destroyed when any other accumulator is used. The result of an "add" or "subtract" operation on any accumulator appears in Accumulator AR as well as the accumulator specified in the command.

Command Sequence

Commands are obeyed in the order in which they are listed on the coding sheet; the programmer numbers the commands sequentially, beginning with command number 001. In computer terminology, after obeying a command the computer automatically "transfers control" to the command at the next consecutive command number.

Control may be transferred to a command other than the next one in sequence by writing a transfer of control command. A Transfer of Control command does not specify an address in the memory of the computer, but specifies the number of the next command to be obeyed. Control may be transferred conditionally or unconditionally; if conditional, the transfer may be based on the nature of the contents of any accumulator, or may be based on the existence of an accumulator overflow.

If control is transferred unconditionally, the location of the transfer of control command can be automatically remembered by the computer. Other commands are provided to return to the point in the program from which control was transferred. Eight levels of such marked transfers of control and return are possible. Such commands are particularly useful for the incorporation in a program of subroutines written from simplified commands.

Another command permits subroutines written in machine language form to be incorporated into a program. At the conclusion of the execution of the machine language subroutine, control is automatically transferred back to the command immediately after the Perform Subroutine command.

Index Registers

Each command used with an index register operates on the contents of an address determined by adding a selected value to the address specified in the command. The command itself is left unchanged in the memory of the computer.

The same index register may be used with any number of commands. The compiler provides eleven index registers in order that separate groups of commands in a program may have their effective addresses directly modified in different ways. The number designating the index register is put in the K position of each command in which the address is to be modified.

An index register may be numbered from 1 to 9, u or v.

Any location in the memory, except for locations in channels 00 through 05, may be specified to be an index register and may be assigned an index register number. The instruction for doing so is listed on page 6.

If a long channel is used to hold an index register, the index register should be placed in word position 96, 97, 98, 99, u0, u1, u2 or u3 in the channel in order to minimize computation time.

Note that any of the accumulators except AR may be assigned to be index register locations. Accumulators are particularly useful as index registers since they may be incremented by a single "add" command or decremented by a single "subtract" command, and since the result of an "add" or "subtract" operation is also held in Accumulator AR.

The following three commands, for example, can be used to permit execution of a series of commands a fixed number of times, with a different set of addresses each time.

Add increment to Index Register K

Subtract limit from duplicate incremented value in AR

If AR is negative, transfer control to first command modified by Index Register K

Notice that three constants may be needed for each index register. They are the initial value of the index register, the incrementing value, and the limiting value.

Constants

Constants and other numerical values may be placed in memory locations specified by the programmer. The numerical values are put in channels not used to hold compiled machine language commands.

In writing a program, an arbitrary constant may be inserted at any point and specified to be the operand of an arithmetic command.

Scale Factors

Numerical values during computation are fractional; they range between -1 and +1. The computer indicates an "overflow" if either of these limits is reached or exceeded.

The programmer can specify that the numerical result of a command be multiplied by, or be divided by, a power of ten. The result of the command can be multiplied by 10^0 to 10^7, or be divided by 10^1 to 10^7. The exponent of 10 is called the scale factor in the command. The scale factor is preceded by the symbol "/" if it indicates division. In order to avoid overflow the numerical result of a command must be between -1 and +1 before being modified by the scale factor.

For example: If the numerical result, before scaling, of an arithmetic command is .0023456, and the command has a scale factor of 0, the final result would also be .0023456.

If the scale factor were 2, the final result would be .2345600.

If the scale factor were /2, the final result would be .0000234.

If the scale factor were 3, the final result would be .3456000; and the computer would indicate an overflow since the true value after multiplication by 10^3 is 2.3456000.

COMMAND LIST

In the commands on these pages, K is the index register number. K may be a digit from 1 to 9, u or v. The K position in a command is left blank if no index register is used.

"A" is the accumulator number. A may be a digit from 0 through 9, u, v, w, x, y or z. If Accumulator AR is to be used by the command, and if no index register is used, the A position of the command is left blank. However, if both an index register and Accumulator AR are used in the same command, "w 8" must be written in the A position of the command.

"S" is the power of 10 by which the numerical result of the command in which it appears is multiplied or divided. S may range from 0 through 7, indicating multiplication by 10^0 to 10^7, or S may range from /1 to /7 indicating division by 10^1 to 10^7.

ADDR is a location in the memory of the computer.

CMN is the numerical position of a simplified command on the programming sheet.

Arithmetic Commands

Clear and Subtract — K A 40 ADDR S

The contents of Accumulator A are replaced by the contents of address ADDR with the sign reversed.

Subtract — K A 41 ADDR S

The contents of address ADDR are subtracted from the contents of Accumulator A. The difference replaces the previous contents of Accumulator A.

If the magnitude of the result is equal to or greater than one, the overflow indicator will be set.

If an index register is used, the accumulator can not be AR.

Clear and Add — K A 42 ADDR S

The contents of Accumulator A are replaced by the contents of address ADDR.

Add — K A 43 ADDR S

The contents of address ADDR are added to the contents of Accumulator A. The sum replaces the previous contents of Accumulator A.

If the magnitude of the result is greater than or equal to 1, the overflow indicator will be set.

If an index register is used, the accumulator can not be AR.

Multiply K A 44 ADDR S

The contents of Accumulator A are multiplied by the contents of address ADDR. The product replaces the previous contents of Accumulator A.

Divide K A 48 ADDR S

The contents of Accumulator A are divided by the contents of address ADDR. The quotient replaces the previous contents of Accumulator A.

If the magnitude of the result is greater than or equal to one, the overflow indicator will be set.

Clear and Add Absolute Value K A 45 ADDR S

The contents of Accumulator A are replaced by the absolute value of the contents of address ADDR.

Store K A 49 ADDR S

The contents of Accumulator A are stored at location ADDR, replacing the previous contents of ADDR.

Copy Two Words K A 62 ADDR 0

The contents of Accumulators A and A + 1 are replaced by the contents of addresses ADDR and ADDR + 1. Accumulator A can not be AR. Both A and ADDR must be even numbers.

Copy Four Words K A 14 ADDR 0

The contents of Accumulators A, A - 1, A - 2 and A - 3 are replaced by the contents of addresses ADDR, ADDR - 1, ADDR - 2 and ADDR - 3.

The number ADDR + 1 must be divisible by four without a remainder. A must be Accumulator 3, 7, v or z.

Examples

The command to add the contents of address 1569 to Accumulator AR would be "43 1569 0".

The command to add the contents of address 1569 to Accumulator AR and to multiply the result by 10^7 would be "43 1569 7".

The command to add the contents of address 1569 to Accumulator AR and to divide the result by 10^2 would be "43 1569/2".

The command to add the contents of address 1569 to Accumulator 8 and to divide the result by 10^2 would be "8 43 1569/2".

The command to add to Accumulator 8 the contents of a memory location determined by adding the contents of Index Register 7 to 1569, and to divide the result by 10^2, would be "7 8 43 1569/2".

Transfers of Control

Transfer Control Unconditionally 29 0CMN 0

Control is transferred to command number CMN.

Transfer Control if Accumulator Positive or Zero A 20 0CMN 0

Control is transferred to command number CMN only if the contents of Accumulator A are not negative. If the contents of Accumulator A are negative the next command obeyed will be the one in normal sequence.

Transfer Control if Accumulator Negative A 22 0CMN 0

Control is transferred to command number CMN only if the contents of Accumulator A are negative. If the contents of Accumulator A are zero or positive the next command obeyed will be the one in normal sequence.

Transfer Control if Accumulator Zero A 23 0CMN 0

Control is transferred to command number CMN only if the contents of Accumulator A are zero. If Accumulator A contains a non-zero quantity, the next command obeyed will be the one in normal sequence.

Transfer Control on Overflow 21 0CMN 0

If the overflow indicator has been previously set by an arithmetic operation, control is transferred to command number CMN. Otherwise, the next command in the sequence will be obeyed. The overflow indicator is reset.

Mark Place and Transfer Control 28 0CMN M

Control is transferred to command number CMN. The command number of the "Mark Place and Transfer Control" command is stored in special register M.

M may be any digit from 0 to 7.

Return to Marked Place 18 0000 M

Control is transferred to the command immediately following the last "Mark Place and Transfer Control" command which has the same value of M as this command.

Input Commands

Read Paper Tape 55 ADDR 0

A block of information, of 108 words or less, is photo-electrically read from punched tape and entered into the channel in the memory specified by the first two digits of ADDR. The last two digits of ADDR must be 00.

Information in channel 19 is destroyed by this command.

Permit Decimal Type-In K w8 51 ADDR S

Computation will halt. If a seven digit decimal fraction, followed by "tab s", is typed by the operator, the number will be stored in location ADDR and computation will be continued.

Output Commands

Type Decimal Number and Tab K 0 33 ADDR 0

The contents of address ADDR are typed out as a decimal fraction. The typewriter carriage moves to the next tab stop.

ADDR can not be 2800.

Type Decimal Number and Return Carriage K 0 38 ADDR 0

The contents of address ADDR are typed out as a decimal fraction. The typewriter carriage moves to the beginning of the next line.

ADDR can not be 2800.

Position Typewriter Paper K 0 30 TB CR 0

Paper in the typewriter carriage is automatically positioned by the execution of CR carriage returns followed by TB tabs.

CR is a two-digit number ranging from 00 to 99. TB is a two-digit number ranging from 00 to 99.

Punch Paper Tape 39 ADDR 0

The contents of the channel determined by the first two digits of ADDR are punched on tape. The last two digits of ADDR must be 00.

The contents of channel 19 are destroyed by this command.

Special Commands

Ring Bell 32 0000 0

The bell chimes once.

Enough time must physically elapse between the execution of successive Ring Bell commands for the gong to return to its rest position.

Halt 67 0000 0

Computation is halted. Computation may be continued with the next command in sequence by moving the Compute switch from GO to the center position and back to GO.

Permit Manual Operation 25 0000 0

Computation is halted. The computer is put in position to accept the instructions listed in Table II from the typewriter keyboard.

Perform Subroutine 27 ADDR 0

The machine language subroutine, which begins at location ADDR, is executed. The operand must be in Accumulator w. After execution of the subroutine, control is transferred to the command immediately following the "Perform Subroutine" command.

Information in channel 19 is destroyed by this command.

Convert Decimal Fraction 25 0254 0

A decimal fraction is converted to the form necessary for internal computation. The decimal fraction to be converted must be in Accumulator w. The result appears in Accumulator AR.

This command is used only for decimal fractions that have been read from punched paper tape not prepared by the computer. At all other times the conversion of decimal input information to the necessary internal form is automatic.

PREPARATION OF THE MACHINE LANGUAGE PROGRAM

To Prepare a Coding Sheet

1. For each index register to be established, write K 0 07 ADDR 0. K is the index register number from 1 to 9, u or v. ADDR is the memory location to be used for that index register.

   As the remaining lines on the coding sheet are written, number them consecutively beginning with command number 1. Do not number lines which contain a numerical value to be entered into the computer.

   For each constant to be included for use by an index register write an applicable set of codes from Table I. (The constants associated with an index register represent its initial value, its incrementing value, and its limiting value. See example on page 8.)

2. Write the commands of the program, one command to a line, using the codes listed in the Command List. Continue to number the commands consecutively.

   The numerical value zero may be entered into an accumulator at any point in the program by writing the command A 42 2900 0. Memory location 2900 always contains zero.

   A numerical value may be stored in an optimum memory location for use by an arithmetic command in the program. To do so, write the arithmetic command with 05u8 in the ADDR position. On the next line write the numerical value in the form of a seven digit decimal fraction with a minus sign if negative and preceded by the letter v.

Example: To have the decimal number -.314159 available for use in Accumulator 3 at a specific point in the program, write the command 3 42 05u8 0 at that point. On the next line write v-3141590. Remember that this last line on the coding sheet does not have a command number associated with it.

3. At the end of the program write one of the commands that halt computation.

4. To enter data into a specific set of consecutive memory locations in a channel not used to hold machine language commands write:

```
09 0000 0
06 0000 0
0y 0000 0
11 CH00 0
0x 0000 0
10 00WD 0
```

   CH is the channel in which the constants are to be placed. WD is the word position in the channel of the first constant.

   Write the constants to be stored in consecutive word positions - one on each line. Each constant should be in the form of a seven digit decimal fraction preceded by the prefix "v". The constant should include a minus sign if negative. Do not write more than 40 constants.

   Write on the next line 3 0u 0000 0.

   To enter data into a different set of consecutive memory locations in the channel, repeat the directions in step 4 beginning with the fifth code line: 0x 0000 0. Repeat as often as necessary for additional groups of data.

   Write on the next line 09 0000 0.

   All unused locations in each channel in which data has been entered will contain zero except for locations 00, u4 and u7. Location u7 should not be used for data storage.

5. To enter data in additional channels, repeat the directions in step 4 beginning with the second code line: 06 0000 0.

To Accelerate a Program

When programmed commands are read into the computer, each is transformed by the compiler into a series of machine language commands with assigned memory locations. The compiler begins by putting the machine language equivalent of command number 1 in locations which start at channel 06 word position 00.

During compiling, the empty word positions in the channel are examined, and the machine language commands are put into available word positions which minimize computation time. After a channel is full, compiling automatically continues in the next consecutive channel. Usually about 35 programmed commands correspond to one full channel of machine language commands.

As a channel fills with machine language commands, word position allocation by the compiler becomes less effective in reducing computation time.

The computation time of a program which does not require the full memory capacity of the computer may be reduced by writing the code 05 0000 0 at selected intervals on the coding sheet. This code causes the computer to start compiling in the next channel immediately instead of filling the current channel. The code may be interjected after the coding sheet is prepared, since it uses no command number.

TABLE I

CODES FOR ENTERING INDEX REGISTER CONSTANTS

A constant which represents the initial value, the incrementing value, or the limiting value of an index register can be placed in any memory location. To do so, copy one of the forms below on the coding sheet. ADDR is the location in which the constant is to be stored.

If the constant is zero:

w 42 2900 0
w 49 ADDR 0

The previous contents of Accumulator w are destroyed by this operation.

If the constant is 9 or less, but greater than zero; and is for indexing the word-position portion of ADDR:

w 42 05u8 0	
w 0C00000	C is the one-digit constant
w 49 ADDR 0	

The previous contents of Accumulator w are destroyed by this operation.

If the constant is any value, and is for indexing the word-position portion of ADDR:

w 62 2900 0	
w 42 05u8 0	
u 00000CT	CT is the two-digit constant
w 6v 0020 0	
w 49 ADDR 0	

The previous contents of Accumulators w and x are destroyed by this operation.

If the constant is any value and is for indexing the channel portion of ADDR:

w 62 2900 0	
w 42 05u8 0	
u 00000CT	CT is the two-digit constant
w 6v 0005 0	Omit this code if indexing "Store" or input command
w 49 ADDR 0	

The previous contents of Accumulators w and x are destroyed by this operation.

If the constant is of any value, and is for indexing both word-position and channel portions of ADDR:

w 62 2900 0	
w 42 05u8 0	
u 00000CT	The two-digit constant CT is for word-position indexing
w 6v 0020 0	
y 62 2900 0	
y 42 05u8 0	
u 00000CT	The two-digit constant CT is for channel indexing
y 6v 0005 0	Omit this code if indexing "Store" or input command
y 43 2300 0	
y 49 ADDR 0	

The previous contents of Accumulators w, x, y and z are destroyed by this operation.

An Illustrative Example Which Uses an Index Register

A coding sheet is prepared below for a program which squares ninety-one numbers, finds their sum, and stores the result.

The ninety-one numbers are held in memory locations 1600 to 1690. The sum is stored in location 1500.

CMN	COMMAND	*Notes*
	4 0 07 2100 0	Accumulator 4, which is at 2100, is established to be Index Register 4.
1	w 42 2900 0	The initial value of the index register is set equal to zero.
2	w 49 2100 0	
3	w 42 05u8 0	The constant, 1, which will increment the index register, is placed in location 1796.
	w 0100000	
4	w 49 1796 0	
5	w 62 2900 0	The constant, 91, which will limit incrementation of the index register, is placed in location 1797.
6	w 42 05u8 0	
	u 0000091	
7	w 6v 0020 0	
8	w 49 1797 0	
9	2 42 2900 0	Accumulator 2 is cleared to zero. It will be used to hold the sum of the squares.
10	4 3 42 1600 0	The contents of the memory location determined by adding the contents of Index Register 4 to 1600 are copied into Accumulator 3.
11	3 44 2003/2	The number in Accumulator 3 is multiplied by itself (Accumulator 3 is located at address 2003). The result is divided by 100 in order to prevent the possibility of an arithmetic overflow in the next step.
12	2 43 2003 0	The contents of Accumulator 3 are added to the contents of Accumulator 2. This is the partial sum of the squares.
13	4 43 1796 0	The increment 1 is added to the contents of Accumulator 4; in this problem Accumulator 4 is also Index Register 4.
14	41 1797 0	The limiting value, 91, is subtracted from the duplicate incremented value in Accumulator AR.
15	22 0010 0	If AR is negative indicating the limit has not been reached, control is transferred to command number 10.
16	2 49 1500 0	The contents of Accumulator 2 (which is now the sum of the squares of the numbers in locations 1600 to 1690) are stored in location 1500.
17	67 0000 0	Halt.

In this example the index register number coincided with the accumulator number. It is convenient, but not necessary, to specify an index register location to be that accumulator which has the same number.

To Compile a Machine Language Program from the Coding Sheet

1. Put compiler magazine #1 on the photo-reader.

2. Put the Compute switch to the center position, Enable switch ON, type "p", put Enable switch off.

   Wait for the photo-reader light to remain off.

3. Put Compute switch to GO.

   Wait for the photo-reader light to remain off.

   The computer will return the typewriter carriage and type out 1.

4. Type the contents of the coding sheet. Type "tab s" after each line.

   After each tab s is typed the computer will return the typewriter carriage and type out the applicable command number.

   The computer will punch tape at intervals. Mark the locations on the coding sheet at which punching occurs for later reference. The tape being punched is a preliminary program tape.

   If a typing error is made, correct the error immediately in the manner described on page 10.

5. After all lines on the coding sheet have been typed, type "0w 0000 0 tab s".

   The computer will finish punching the preliminary program tape.

6. Rewind compiler magazine #1 and remove from photo-reader.

7. Put compiler magazine #2 on photo-reader.

8. Put Compute switch to center, Enable switch ON, type "p", put Enable switch off.

   Wait for the photo-reader light to remain off.

9. Put Compute switch to GO.

   Wait for the photo-reader light to remain off and for the tape in the magazine to automatically rewind.

10. Remove compiler magazine #2 from the photo-reader. Place the preliminary program tape which the computer has punched on the photo-reader.

11. Type "tab s".

    The computer will punch a length of tape.

12. Type "06 0000 0 tab s".

13. Type "0x 0000 0 tab s".

    The computer will read tape in the photo-reader at intervals and will punch tape at intervals. The tape being punched is the machine language program tape.

    The computer will type out the number of each programmed command followed by the memory locations of the first and last machine language commands corresponding to it.

    If the command number, typed out on the left, of a transfer of control command is greater by one hundred or more than the value of CMN to which it transfers control, the computer will halt. The computer will type out the value of CMN to which control is to be transferred. The operator looks at the previous compiler type-out and notes the first memory location listed opposite the value of CMN. He types that memory location, followed by "tab s", and compiling automatically continues.

    The computer will also halt if the operator violates either of two restrictions. It will halt if more than 999 programmed commands are typed; or, if the number of transfers of control to command numbers not yet typed exceeds twenty.

14. When the computer halts after it types out a value of CMN one greater than the last command number in the program, type "13 0000 0 tab s".

    The computer will finish recording the machine language program tape. Run out tape trailer and tear off the completed tape.

    The program has been compiled to begin in channel 06 word 00. The compiler service magazine must be used in order to execute the program.

To Correct Errors Made During Compiling

When compiler magazine #1 is being used to prepare a preliminary tape, errors can be corrected from the typewriter.

To correct an error made while typing a line from the coding sheet before the s key has been pressed at the end of that line:

Type "/ / tab s". Then type the correct code line.

If the s key has already been pressed:

Type "WD tab s" where WD is the estimated word position of the simplified command. The computer will type out the command.

WD is always an even number and can be estimated by doubling the number of simplified commands that have been typed since the computer last punched tape.

If the command typed out is the one in error, type its correct version followed by "tab s". If it is not the command in error, type "WD tab s" again in order to inspect some other word position. Enter the correction after the proper word position has been located.

To correct a numerical value which has been put into an accumulator by use of an arithmetic command with 05u8 in its ADDR position, locate the command by use of "WD tab s". Retype the command referring to 05u8 followed by the correct numerical value.

After corrections have been made, the typing of simplified commands from the coding sheets into the computer can be continued. To do so, locate, by typing "WD tab s", the last simplified command entered into the computer before corrections were made. Retype this command (in its new form if it is one which was corrected) and continue entering the remainder of the program.

Disregard command numbers typed out by the computer after corrections have been made as that type-out no longer indicates the stored value of CMN for each command.

To Load and Execute the Machine Language Program Prepared by the Compiler

1. Put the compiler service magazine on the photo-reader.
2. Put the Compute switch off, Enable switch ON, type "p". Put Enable switch off.
3. Put the Compute switch to GO.

   Wait for the photo-reader light to remain off.
4. Replace the compiler service magazine on the photo-reader with the machine language program tape.
5. Repeat steps 2 and 3.
6. Type "s c f".

   The computer is now in the manual operating mode. Any manual instruction listed in Table II on page 12 may now be typed.
7. To execute the program beginning at location CMN, determine the corresponding memory location ADDR from the first number typed out opposite CMN by the compiler. Type "y ADDR tab s".

Operating Notes

Information read from a compiled machine language tape into the computer is automatically checked for accuracy. The bell rings and the computer halts if a reading error occurs. If this happens, the tape should be rewound and then re-read into the computer.

Though numbers entered into and typed out of the computer consist of seven decimal digits, internal computation is carried out to slightly greater precision in order to minimize round-off error. When zero is entered into the computer in the form "v0000000" a very small numerical value may be entered into the round-off position of the number. This value, though effectively zero during computation, will not be considered equal to zero by the command "Transfer Control if Accumulator Zero". True zero can always be found at Address 2900. The command "3 42 2900 0", for example, will put true zero in Accumulator 3.

CHECK-OUT OF THE MACHINE LANGUAGE PROGRAM

When the machine language program is first executed, computation may be incorrect because of errors in its composition. The instructions listed in Table II permit execution of the program to be controlled and interrogated from the typewriter keyboard in order to locate the errors.

Computation can be halted at any time by putting the Compute switch to the center position.

One of the instructions in Table II inserts a breakpoint in a specified memory location. During computation, if the Compute switch is at BP instead of GO, the computer will halt after obeying a command in a location in which a breakpoint has been inserted.

After computation has halted, put the Enable switch ON, type "m s c f", put the Enable switch off and the Compute switch to GO or BP. Any instructions listed in Table II may now be typed and executed. To continue computation from the point at which it was interrupted type "z tab s".

Program Tracing

One of the keyboard instructions in Table II directs the computer to document the step by step execution of a program beginning at any specified memory location. The procedure is called "Tracing".

During tracing, after each set of machine language commands that correspond to a simplified arithmetic command is executed, the computer types out the ending memory location of the simplified command, the accumulator number, and the accumulator contents.

Commands which use Accumulator AR do not cause type-out. The command "Store" causes type-out only if ADDR in the command is the address of an accumulator other than AR.

The type-out after execution of each arithmetic command ceases when a breakpoint is reached and resumes when the next breakpoint is reached. Successive breakpoints alternately halt and resume type-out. This alternation permits type-out to be inhibited during execution of a machine language subroutine or other portion of the program for which no typed record is wanted.

If an overflow occurs when a program is being traced, the bell in the computer will ring until the overflow indicator is reset by the command "Transfer Control on Overflow".

The Check-Out Procedure

A general procedure for isolation of programming errors is given below. Use the keyboard instructions in Table II to perform the following steps.

1. Prepare a test solution for the coded problem.

2. Insert breakpoints at several critical points in the program. Execute each portion of the program, examining the numerical result at each breakpoint location. When the type-out is incorrect proceed to step 3 or step 4.

3. Insert additional breakpoints in the portion of the program in error. Again examine the numerical result at each breakpoint location.

4. Trace the section of the program which has an incorrect result and note the memory location of the first erroneous command. Identify the command by referring to the type-out of command memory locations obtained during compiling (the type-out obtained in step 13 on page 9).

5. Insert the proper intermediate numerical result at the point at which computation became incorrect. Continue computation from that point and repeat steps 2 through 4 in order to locate the next error.

   Repeat this step until all errors have been located and identified.

After the errors have been isolated, correct the coding sheet of simplified commands in the following manner:

For each erroneous command that can be replaced by a single simplified command, cross out the incorrect version and write the correct one in its place.

For each command that must be replaced by more than one simplified command, cross out the command and write a transfer of control to the end of the program. (Remember that command numbers at the end of the program must continue to be consecutive.) Write the correct set of commands at the end of the program. End the set of commands with a transfer of control back to the command number one greater than the command being replaced.

The procedure to obtain a correct machine language tape is:

1. Perform the first three steps on page 9.
2. Replace the magazine in the photo-reader with the preliminary program tape.
3. Type "2z 0000 0 tab s". The computer will read a group of simplified commands into the memory for correction.
4. Type "WD tab s" where WD is an even number corresponding to the estimated word position of the command to be replaced in the group read from tape. (See page 10 for use of this keyboard instruction.)
5. When the command to be changed has been located by typing "WD tab s"; type the replacing command.
6. Repeat steps 4 and 5 for each change in that group of commands that has been read into the memory.
7. Type "1w 0000 0 tab s". The computer will punch out the corrected group of simplified commands.
8. Repeat steps 3 through 7 until the entire preliminary tape in the photo-reader has been processed and all corrections and additions have been made. The additional commands at the end of the program are added by using "WD tab s" to locate the last simplified command in the uncorrected program. This last command can then be retyped followed by the new commands.
9. Type "0w 0000 0 tab s". The computer will finish punching a corrected preliminary program tape.
10. Follow the directions on page 9, beginning with step 6, to obtain a new machine language tape from the corrected preliminary program tape.

TABLE II

KEYBOARD INSTRUCTIONS FOR MACHINE LANGUAGE PROGRAM

Type Out Decimal Fraction — v ADDR tab s

The contents of location ADDR will be typed out as a seven digit decimal fraction.

Permit Decimal Type-In — -v ADDR tab s

If a number consisting of seven decimal digits and sign, followed by tab s, is typed by the operator, it will be inserted in location ADDR.

Type Out Hexadecimal Number — w ADDR tab s

The contents of location ADDR will be typed out as a seven digit hexadecimal number.

Permit Hexadecimal Type-In — -w ADDR tab s

If a number consisting of seven hexadecimal digits and sign, followed by tab s, is typed by the operator, it will be inserted in location ADDR.

Insert Breakpoint — -x ADDR tab s

A breakpoint will be inserted in location ADDR. ADDR must be the last location of a group of machine language commands which correspond to a simplified command.

Remove Breakpoint — x ADDR tab s

If a breakpoint has been inserted in location ADDR, it will be removed.

Execute Program — y ADDR tab s

Computation will proceed automatically beginning with the command in location ADDR. ADDR must be the first location of a group of machine language commands which correspond to a simplified command.

Trace Program — -y ADDR tab s

Computation will proceed automatically beginning with the next command in location ADDR. The program will be traced in the manner described on page 11 during execution. ADDR must be the first location of a group of machine language commands that correspond to a simplified command.

ADDR may be any address in the memory. For these instructions only, the address of AR is "0 1 u7".

SUMMARY OF COMMANDS

Arithmetic Commands

Clear and Subtract	K	A	40	ADDR	S
Subtract	K	A	41	ADDR	S
Clear and Add	K	A	42	ADDR	S
Add	K	A	43	ADDR	S
Multiply	K	A	44	ADDR	S
Divide	K	A	48	ADDR	S
Clear and Add Absolute Value	K	A	45	ADDR	S
Store	K	A	49	ADDR	S
Copy Two Words	K	A	62	ADDR	0
Copy Four Words	K	A	14	ADDR	0

Transfers of Control

Transfer Control Unconditionally			29	0CMN	0
Transfer Control if Accumulator Positive or Zero		A	20	0CMN	0
Transfer Control if Accumulator Negative		A	22	0CMN	0
Transfer Control if Accumulator Zero		A	23	0CMN	0
Transfer Control on Overflow			21	0CMN	0
Mark Place and Transfer Control			28	0CMN	M
Return to Marked Place			18	0000	M

Input Commands

Read Paper Tape			55	ADDR	0
Permit Decimal Type-In	K	w8	51	ADDR	S

Output Commands

Type Decimal Number and Tab	K	0	33	ADDR	0
Type Decimal Number and Return Carriage	K	0	38	ADDR	0
Position Typewriter Paper	K	0	30	TB CR	0
Punch Paper Tape			39	ADDR	0

Special Commands

Ring Bell			32	0000	0
Halt			67	0000	0
Permit Manual Operation			25	0000	0
Perform Subroutine			27	ADDR	0
Convert Decimal Fraction			25	0254	0

LOS ANGELES *(Home Office)*
5630 Arbor Vitae Street
Los Angeles 45, California
Telephone SPring 6-2220

CHICAGO
919 N. Michigan Avenue
Chicago 11, Illinois
Telephone MIchigan 2-6692

DALLAS
1511 Bryan Street
Dallas 1, Texas
Telephone RIverside 7-8805

WASHINGTON
1000 Connecticut Avenue, N.W.
Washington 6, D.C.
Telephone STerling 3-1508

NEW YORK
205 East 42nd Street
New York 17, New York
Telephone ORegon 9-6990

SAN FRANCISCO
2337 Shattuck Avenue
Berkeley 4, California
Telephone THornwall 3-5706

Bendix Computer DIVISION OF BENDIX AVIATION CORPORATION

CANADA
Computing Devices of Canada
P.O. Box 508
Ottawa 4, Ontario
Telephone PArkway 8-1761

ALL OTHER COUNTRIES
Bendix International
205 East 42nd Street
New York 17, New York
Telephone MUrrayhill 3-1100