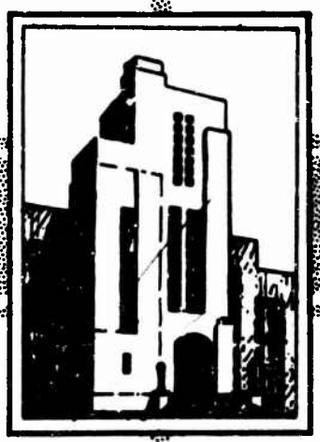


DEPARTMENT OF THE NAVY
DAVID TAYLOR MODEL BASIN

HYDROMECHANICS

o

AERODYNAMICS

o

STRUCTURAL MECHANICS

o

APPLIED MATHEMATICS

THE CXPQ COMPUTER
A PRELIMINARY DESCRIPTION FOR PROGRAMMING PURPOSES

By

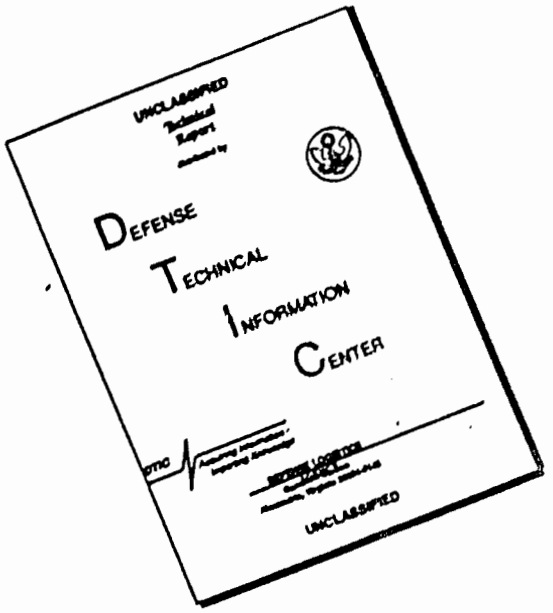
Barbara R. Sherard

Operations Research Division
Applied Mathematics Laboratory

July 1959

Report 1356

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

THE CXPQ COMPUTER
A PRELIMINARY DESCRIPTION FOR PROGRAMMING PURPOSES

by

Barbara R. Sherard

July 1959

Report 1356

CONTENTS

		Page
	INTRODUCTION	1
I.	COMPUTER CONFIGURATION	2
	1. Magnetic Core Storage	2
	2. Magnetic Tape Units	2
	3. Paper Tape	2
	4. Card Reader	3
	5. Magnetic Drum	3
	6. Machine Word Size	3
II.	CONTROL SECTION	4
	1. Instruction Format	4
	2. Index Registers	4
	3. Control Sequence	5
III.	ARITHMETIC SECTION	6
IV.	INSTRUCTION CODE	7
	1. Arithmetic Instructions	8
	2. Data Transfers	12
	3. Shifts	13
	4. Jumps	15
	5. Miscellaneous Instructions	17
	6. Logical	18
	7. Input-Output	18
V.	COMPUTER CONSOLE AND OPERATIONS	23
	Appendix	28
	Table of Instruction Codes	28
	Machine Codes	32
	Programming example	34

INTRODUCTION

This report is intended to be a preliminary programming manual for the Philco CXPQ Computer. This computer was developed under BUSHIPS Contract NObsr-72609, during the period 1955 to 1958. Members of the Operations Research Division, Code 830, David Taylor Model Basin, co-operated with BUSHIPS' Engineers in the logical and programming design of the computer. The design of the CXPQ was oriented for use in real time military control systems. It is inherently suitable for solving comp utational problems and for feasibility research programming in areas not requiring excessive quantities of input-output equipment.

The CXPQ computer will be installed in the latter half of 1959 at the David Taylor Model Basin. It will be assigned to the Operations Research Division of the Applied Mathematics Laboratory for use in research programming for the Operations Control Center Project. It will be maintained by the personnel of the Engineering and Development Division. The Operations Research Division intends to make the computer available to all groups involved in the Operations Control Center Project, to the other divisions of the Applied Mathematics Laboratory and to other laboratories of the David Taylor Model Basin, whenever possible.

The report contains a general description of the computer and sufficient information on the Instruction Code to permit programming to be performed. The information contained herein has been made available by the Philco Corporation in the form of Engineering Progress Reports and in the course of oral discussions. The author is grateful for the cooperation of the personnel in the Philco Corporation. The author holds herself responsible for any misinterpretation of the information thus obtained.

I. COMPUTER CONFIGURATION

The CXPQ is a large scale, fully transistorized, binary computer. It is a single address machine and operates in a parallel, asychronous mode.

1. Magnetic Core Storage

Access time.

Access to words in any unit of the core memory takes place in parallel in 12 microseconds. Words are read from, or stored in the memory in two cycles - a 5 microsecond read cycle, and a 7 microsecond write cycle. When a word is read from memory the read cycle reads and clears the memory location, and the write cycle restores the contents of the memory location. When a word is stored in memory, the read cycle clears the memory location and the write cycle stores the word in the accessed memory location.

Storage.

The present magnetic core contains 4096 words with tape, drum, and paper tape as auxiliary storage.

2. Magnetic Tape Units (Potter - Model 905)

The tape is Mylar, 1 inch wide, 1 mil thick and approximately 3600 feet long. The speed of the tape is 75 inches per second for read and write operations; for rewind operations the speed is 150 inches per second. The density of information on the tape is 200 bits per inch per channel. The start and stop times have not been accurately measured.

Each tape will consist of blocks of 128 words of 48 bits each. The blocks will be self-addressed and are not program addressable. The present system contains 3 magnetic tape units.

3. Paper Tape

The paper unit comprises:

(a) A Ferranti Reader which is capable of reading paper tape at a speed of 200 characters per second (transfer time of 40 milliseconds per word).

(b) A teletype punch which is capable of punching paper tape at a speed of 60 characters per second (access time of 133 milliseconds per word).

(c) A Flexowriter with a capability of punching and/or printing at a speed of 10 characters per second (transfer time of 800 milliseconds per word). It is capable of reading at the same rate. A provision is also made for direct input from the keyboard.

4. Card Reader and Punch (IBM 528)

The Model 528 will be used only in its card read and punch operations; all data processing will be carried on within the CXPQ system proper. The IBM 528 operates at a reading rate of 200 cards per minute (300 milliseconds per card) and a writing rate (punch) of 100 cards per minute (600 milliseconds per card).

5. Magnetic Drum

The magnetic drum is operated in the parallel mode with a storage capacity of 16, 384 words on four bands of 4096 words each, and with a total of 192 information tracks. The maximum access time is 34 milliseconds with a transfer time of 16 microseconds and a speed of 1740 revolutions per minute.

The four bands on the drum are continuously addressed such that:

4096 of band 2 follows 4095 of band 1
8192 of band 3 follows 8191 of band 2, etc.

The addressing is cyclic such that 0000 of band 1 follows 16, 384 of band 4.

6. Machine Word Size

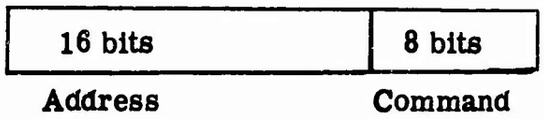
A word is composed of 48 binary digits, numbered from left to right. The word may be 8 binary coded characters, a 47-bit number with a sign bit, or an instruction word containing two instructions.

II. CONTROL SECTION

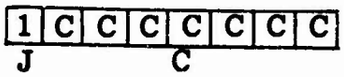
The control section selects and executes instructions in an ordered sequence; i.e., the left half followed by the right half of an instruction word until a jump instruction is executed.

1. Instruction Format

Each instruction contains 24 bits divided into a 16-bit address part and an 8-bit command part.

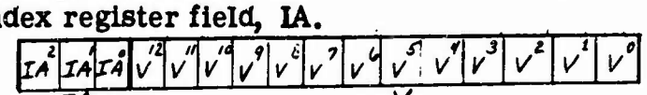


The command part is further sub-divided into a 7-bit command, C, and a function bit, J.



Some instructions require all 8 bits to define an instruction, and others are defined by the 7 bits modified by the function bit. (The function bit specifies which instruction is to be performed first in a jump instruction. If J is 0, the instruction in the left half of a word is executed first. If J is 1, the right instruction is executed first).

The address part is also subdivided into a 13-bit variable field, V, and a 3-bit index register field, IA.



V specifies the memory address or in the shift instruction V^1 through V^5 specifies the number of places to be shifted and IA specifies the index register involved. (If an index register is involved V is modified by IA).

2. Index Register

In the present system seven Index Registers are available. Each index register is a 12-bit register. Index Registers are specified in the following manner:

Index Register 1, $IA^0 = 1$
Index Register 2, $IA^1 = 1$
Index Register 3, $IA^0 = 1$ and $IA^1 = 1$

Index Register 4, IA^2 = 1, etc.

All instructions which involve an index register use the J bit of an instruction. (J = 0 refers to the left-hand instruction, J = 1 refers to the right hand instruction).

3. Control Sequence.

Instructions are sequenced through various control registers. The Program Register which is a 48-bit register, stores the selected pair of instructions to be executed. The Program Address Register - 13 bits - contains the address of the next instruction word. The Memory Address Register - 13 bits - contains the address of the memory location to be accessed.

Procedure:

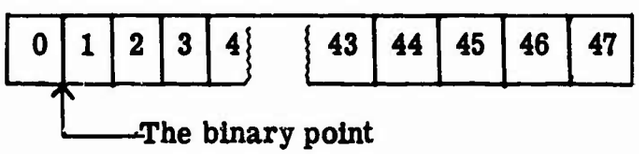
1. Contents of Program Address Register goes to Memory Address Register.
2. Contents of Memory Address Register goes to Program Register.
3. Contents of Program Address Register plus one goes to Program Address Register.
4. Left half of Program Register is executed.
5. Right half of Program Register is executed.
6. Steps 1 - 5 are repeated until a half instruction is executed, an error is detected, or a jump is effected.

III. ARITHMETIC SECTION

The arithmetic section comprises three 48-bit registers and are:

1. The Accumulator or A-register
2. The Multiplier - Quotient or Q-register
3. The Data or D-register (Distributor)
    a. Receives all data transferred between the memory and the arithmetic unit.
    b. Receives all data transferred between Arithmetic Registers.

Since the CXPQ is a fixed point machine, the binary point of a data word lies immediately to the right of the sign or zero position.



The maximum positive number is less than one - as far as the computer is concerned - and the smallest computer negative number is minus one. Any arithmetic result which would be outside the above limits produces overflow, which sets the overflow indicator. (The overflow indicator or flip flop is cleared at the beginning of the execution of all arithmetic operations with the exception of the overflow jump, which clears the overflow flip flop after the jump is performed). The computer ignores the occurrence of overflow unless one instructs the computer to make one cognizant of its occurrence.

IV. INSTRUCTION CODE

Nomenclature

A - A register - 48 bits

D - D register - 48 bits

Q - Q register - 48 bits

V - Memory location V - 48 bits - 12 bit address

IA - Index Register (subtractive process)

c( ) - Contents of a register or memory location

OF - The overflow flip flop - considered a one bit register

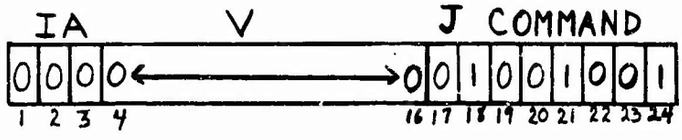
OVR - Considered a one bit register which remembers to inhibit clearing of overflow.

*PA - The place which remembers the address following the last jump - 12 bit address plus one bit for half words.

** - Indexable

Toggle Switch - External switch which is manually set and is on corresponding to a down position and off corresponding to an up position

Instruction codes are represented in quaternary code, (bits 17 - 24), e.g., the instruction add Q (1021) has the following format:



1. Arithmetic Instructions

** 1000 - Add V
Add the C(V) to C(A).

** 1001 - Add V and store
Add the C(V) to C(A) and store in V.

** 1002 - Clear and add V
Clear A to zero and add C(V).

** 1003 - Clear and add V and store
Clear A to zero and add C(V) and store in V.

** 1010 - Add magnitude of V
Add the magnitude of C(V) to the C(A).

** 1011 - Add magnitude of V and store
Add the magnitude of the C(V) to the C(A) and store in V.

** 1012 - Clear and add magnitude of V
Clear A to zero and add the magnitude of the C(V).

** 1013 - Clear and add magnitude of V and store
Clear A to zero and add the magnitude of the C(V) and store magnitude in V.

** 1020 - Add Q
Add the C(Q) to the C(A).

** 1021 - Add Q and store in V
Add the C(Q) to the C(A) and store in V.

** 1022 - Clear and add Q
Clear A to zero and add the C(Q).

** 1023 - Clear and add Q and store
Clear A to zero and add C(Q) and store in V.

** 1030 - Add magnitude of Q
Add the magnitude of the C(Q) to the C(A) and replace A with this sum.

** 1031 - Add magnitude of Q and store
Add the magnitude of the C(Q) to the C(A) and store in V.

** 1032 - Clear and add magnitude of Q
Clear A to zero and add the magnitude of the C(Q).

** 1033 - Clear and add magnitude of Q and store in V
Clear A to zero and add the magnitude of the C(Q) and store in V.

** 1100 - Subtract V
Subtract the C(V) from the C(A) and replace A with the difference.

** 1101 - Subtract V and store in V
Subtract the C(V) from the C(A), replace A with the difference and store in V.

** 1102 - Clear and subtract V
Clear A to zero and subtract the C(V) from the C(A).

** 1103 - Clear and subtract V and store in V
Clear A to zero and subtract the C(V) from the C(A) and store in V.

** 1110 - Subtract magnitude of V
Subtract the magnitude of C(V) from the C(A) and replace A with the difference.

** 1111 - Subtract magnitude of V and store in V
Subtract the magnitude of C(V) from the C(A), replace A with the difference and store in V.

** 1112 - Clear and subtract magnitude of V
Clear A to zero and subtract the magnitude of C(V) from the C(A).

** 1113 - Clear and subtract magnitude of V and store in V
Clear A to zero and subtract the magnitude of the C(V) from C(A) and store in V.

** 1120 - Subtract Q
Subtract the C(Q) from the C(A) and replace A with the difference.

** 1121 - Subtract Q and store in V
Subtract the C(Q) from the C(A), replace A with the difference and store in V.

** 1122 - Clear and subtract Q
Clear A to zero and subtract the C(Q) from the C(A).

** 1123 - Clear and subtract Q and store in V
Clear A to zero and subtract the C(Q) from the C(A) and store in V.

** 1130 - Subtract magnitude of Q
Subtract the magnitude of C(Q) from the C(A) and replace A with the difference.

** 1131 - Subtract magnitude of Q and store in V
Subtract the magnitude of C(Q) from the C(A), replace A with the difference and store in V.

** 1132 - Clear and subtract magnitude of Q
Clear A to zero and subtract the magnitude of the C(Q) from the C(A).

** 1133 - Clear and subtract magnitude of Q and store in V
Clear A to zero and subtract the magnitude of the C(Q) from the C(A) and store in V.

** 1200 - Multiply by V
The contents of Q are multiplied by the C(V) to yield a double length product.
The major product is left in the A-register and the minor product is left in the Q-register.
The sign of the A-register is repeated in the Q-register.

** 1201 - Multiply by V and store in V
Same as 1200 with the exception the major product is stored in V.

** 1202 - Multiply by V and round
The contents of Q are multiplied by the C(V) to yield a single rounded product. The major product (rounded) is in the A-register and the minor product plus 2^{-1} is in the Q-register.

Method: A one is always added to Q resulting in a carry over to position A if Q contains a one or increasing Q by one if position Q contains a zero.

** 1203 - Multiply by V, round and store
Same as 1202 with the exception the major product is stored in V.

** 1210 - Multiply by magnitude of V
The magnitude of the contents of V are multiplied by the C(Q) to yield a double length product.
The major product is left in the A-register and the minor product is left in the Q-register.

** 1211 - Multiply by magnitude of V and store in V
Same as 1210 with the exception the major product is stored in V.

** 1212 - Multiply by magnitude of V and round
The magnitude of the contents of V is multiplied by the C(Q) to yield a single rounded product.

Method: Same as 1202.

** 1213 - Multiply by magnitude of V, round and store in V.
Same as 1212 with the exception the single product (contents of A) is stored in V.

** 1220 - Multiply by A
Same as 1200, except A-register involved rather than V.

** 1221 - Multiply by A and store in V
Same as 1201, except A-register involved rather than V.

** 1222 - Multiply by A and round
Same as 1202, except A-register involved rather than V.

** 1223 - Multiply by A, round and store in V
Same as 1203, except A-register involved rather than V.

** 1230 - Multiply by magnitude of A
Same as 1210 except A-register involved rather than V.

** 1231 - Multiply by magnitude of A and store in V
Same as 1211 except A-register involved rather than V.

** 1232 - Multiply by magnitude of A and round
Same as 1212 except A-register involved rather than V.

** 1233 - Multiply by magnitude of A, round and store in V
Same as 1213 except A-register involved rather than V.

** 1300 - Divide by V
This instruction divides the contents of A by the contents of V and leaves the quotient in the Q-register and remainder in the A-register. The sign of the remainder is the same as the sign of the dividend.

** 1301 - Divide by V and store in V
Same as 1300 except the quotient is stored in V.

** 1302 - Divide by V and round
The contents of A are divided by the contents of V leaving the remainder in the A-register and a rounded quotient. In rounding Q is set equal to one.

**1303 - Divide by V, round and store in V.

** '310 - Divide A by the magnitude of V.

** 1311 - Divide A by the magnitude of V and store in V.

** 1312 - Divide A by the magnitude of V and round.

** 1313 - Divide A by the magnitude of V, round and store.

** 1320 - Divide A by Q
The contents of A are divided by the contents of Q leaving the remainder in A and the quotient in Q.

** 1321 - Divide A by Q and store in V.

** 1322 - Divide A by Q and round.

** 1323 - Divide A by Q, round and store in V.

** 1330 - Divide A by the magnitude of Q.

** 1331 - Divide A by the magnitude of Q and store in V.

** 1332 - Divide A by the magnitude of Q and round.

** 1333 - Divide A by the magnitude of Q, round and store in V.

2. Data Transfers

** 0102 - Transfer V to Q
Transfer the C(V) to Q-register.

** 0103 - Transfer V to D
Transfer the C(V) to the D-register.

** 0110 - Transfer A to V
Transfer the C(A) to V.

** 1210 - Multiply by magnitude of V
The magnitude of the contents of V are multiplied by the C(Q) to yield a double length product.
The major product is left in the A-register and the minor product is left in the Q-register.

** 1211 - Multiply by magnitude of V and store in V
Same as 1210 with the exception the major product is stored in V.

** 1212 - Multiply by magnitude of V and round
The magnitude of the contents of V is multiplied by the C(Q) to yield a single rounded product.

Method: Same as 1202.

** 1213 - Multiply by magnitude of V, round and store in V.
Same as 1212 with the exception the single product (contents of A) is stored in V.

** 1220 - Multiply by A
Same as 1200, except A-register involved rather than V.

** 1221 - Multiply by A and store in V
Same as 1201, except A-register involved rather than V.

** 1222 - Multiply by A and round
Same as 1202, except A-register involved rather than V.

** 1223 - Multiply by A, round and store in V
Same as 1203, except A-register involved rather than V.

** 1230 - Multiply by magnitude of A
Same as 1210 except A-register involved rather than V.

** 1231 - Multiply by magnitude of A and store in V
Same as 1211 except A-register involved rather than V.

** 1232 - Multiply by magnitude of A and round
Same as 1212 except A-register involved rather than V.

** 1233 - Multiply by magnitude of A, round and store in V
Same as 1213 except A-register involved rather than V.

** 1300 - Divide by V
This instruction divides the contents of A by the contents of V and leaves the quotient in the Q-register and remainder in the A-register. The sign of the remainder is the same as the sign of the dividend.

** 1301 - Divide by V and store in V
Same as 1300 except the quotient is stored in V.

** 1302 - Divide by V and round
The contents of A are divided by the contents of V leaving the remainder in the A-register and a rounded quotient. In rounding Q is set equal to one.

**1303 - Divide by V, round and store in V.

** 1310 - Divide A by the magnitude of V.

** 1311 - Divide A by the magnitude of V and store in V.

** 1312 - Divide A by the magnitude of V and round.

** 1313 - Divide A by the magnitude of V, round and store.

** 1320 - Divide A by Q
The contents of A are divided by the contents of Q leaving the remainder in A and the quotient in Q.

** 1321 - Divide A by Q and store in V.

** 1322 - Divide A by Q and round.

** 1323 - Divide A by Q, round and store in V.

** 1330 - Divide A by the magnitude of Q.

** 1331 - Divide A by the magnitude of Q and store in V.

** 1332 - Divide A by the magnitude of Q and round.

** 1333 - Divide A by the magnitude of Q, round and store in V.

2. Data Transfers

** 0102 - Transfer V to Q
Transfer the C(V) to Q-register.

** 0103 - Transfer V to D
Transfer the C(V) to the D-register.

** 0110 - Transfer A to V
Transfer the C(A) to V.

0112 - Transfer A to Q
Transfer the C(A) to the Q-register

0120 - Transfer Q to V
Transfer the C(Q) to V

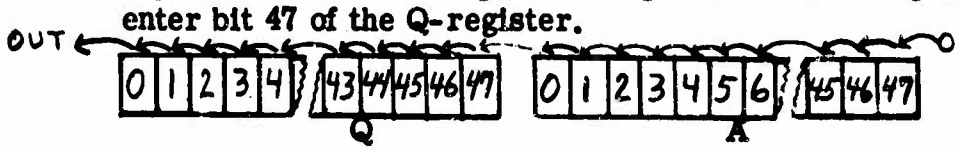
0130 - Transfer D to V
Transfer the C(D) to V

3. Shifts

It is possible to use two types of shifts, namely:

a. Ordinary - treats every bit in the effected register or registers
b. Numerical (sign shift) - treats words in such a way as to preserve the sign.

0201 - Ordinary shift left Q and A
The contents of the Q and A registers are shifted left V places. Bits shifted past the sign bit of the A-register enter bit 47 of the Q-register.



0203 - Sign shift left Q and A
The contents of Q_{1-47} and A_{1-47} are shifted left V places. Bits shifted past bit 1 of the A-register enter bit 47 of the Q-register.

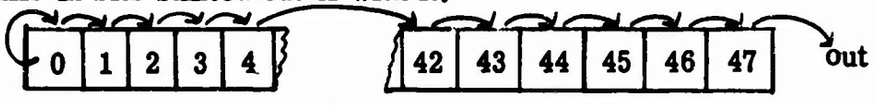
0210 - Ordinary shift right A
The contents of $A_{0, 1-47}$ are shifted right V places. Bits shifted past bit 47 of the A-register are lost.

0220 - Ordinary shift right Q
The contents of $Q_{0, 1-47}$ are shifted right V places. Bits shifted past bit 47 of the Q-register are lost.

0211 - Ordinary shift left A
The contents of the $A_{0, 1-47}$ are shifted left V places. Bits shifted past the sign bit of the A-register are lost.

0221 - Ordinary shift left Q
The contents of Q_0, 1-47 are shifted left V places.
Bits shifted past the sign bit of the Q-register are lost.

0212 - Sign shift right A
The contents of A_0, 1-47 are shifted right V places.
Unlike an ordinary right shift which fills in bits shifted out of with zeros, the sign right shift takes the sign bit and fills in bits shifted out of with it.

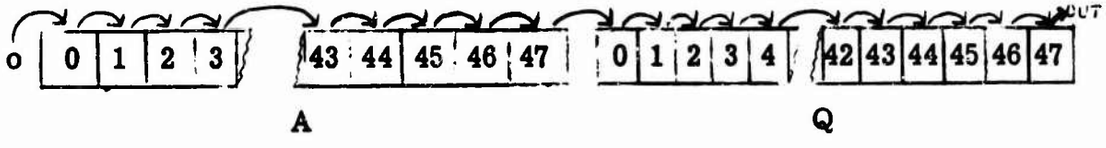


0222 - Sign shift right Q
The contents of Q_0, 1-47 are shifted right V places.
(Method same as 0212)

0213 - Sign shift left A
The contents of A_{1-47} are shifted left V places.
Zero replaces bit 47 of the A-register when a bit is shifted out of A_{47}. Bits shifted out of bit 1 are lost.

0223 - Sign shift left Q
The contents of Q_{1-47} are shifted left V places.
(Method same as 0213)

0230 - Ordinary shift right A and Q
The contents of A_0, 1-47, and Q_0, 1-47, are shifted right V places.
Bits shifted out of A_{47} enter bit 0 of the Q-register.
Bits shifted out of Q_{47} are lost.



0231 - Ordinary shift left A and Q
The contents of A_0, 1-47 and Q_0, 1-47 are shifted left V places.
(Method same as 0230)

0232 - Sign shift right A and Q
The contents of A_0, 1-47 and Q_{1-47} are shifted right V places.
The A-register is treated in the same manner as the instruction sign shift right A-register with the exception bits shifted out of bit A_{47} enter bit 1 of the Q-register. Bits shifted out of Q_{47} are lost.

0233 - Sign shift left A and Q
The contents of A_1-47 and Q_{1-47} are shifted left V places. Bits shifted out of Q_1 enter A_{47} and bits shifted out of A_1 are lost.

4. Jumps

** 0300 - Jump (unconditional)
Jump to either the right or left-hand instruction and store the address of the next instruction following the jump instruction in the *PA.

** 0301 - Jump if Overflow flip flop equals 0.
If the overflow flip-flop equals 0, jump to the specified memory location. If jump occurs store address in *PA and set the 1 bit register OVR equal to 0. This in effect ends the inhibiting of clearing the overflow flip flop on arithmetic instructions.

** 0303 - Jump if overflow flip flop equals 1.
If overflow flip equals 1, jump to the specified memory location. If jump occurs store address in *PA and set the OVR equal to zero.

** 0302 - Breakpoint jump
Jump to the specified memory locations if the toggle switch is off. If toggle switch is on, halt and jump to the location specified by the instruction when the advance key is depressed.

** 0310 - Jump if A equals 0
Jump to the specified memory location if the contents of $A_{0,1-47}$ equals zero. If not proceed to the next instruction in sequence.

** 0311 - Jump if A is positive
Jump to the specified memory location if the contents of A are positive. If not proceed to the next instruction in sequence.

** 0312 - Jump if A equals D
If the contents of the A-register equals the contents of the D-register, jump to the specified location and store address following jump in the *PA. If test is not satisfied proceed to the next instruction in sequence.

** 0313 - Jump if A is minus.
If A_0 is equal to one, jump to the specified memory location and store address following jump in the *PA. If A_0 is unequal to one, proceed to the next instruction in sequence.

** 0320 - Jump if low order position of Q equals 0
If Q_{47} is equal to zero, jump to the specified address and store address following the jump in the *PA. (The Q-register is circular shifted right 1 bit following the jump). If Q_{47} is unequal to zero proceed to the next instruction in sequence.

** 0321 - Jump if Q is positive
If Q_0 is equal to zero, jump to the specified address and store address following the jump in *PA. (The Q-register is circular shifted left 1 bit). If Q_0 (sign bit) is unequal to zero, proceed to the next instruction in sequence.

** 0322 - Jump if low order position of Q equals 1
If Q_{47} is equal to one, jump to the specified address and store address following jump in the *PA. (The Q-register is circular shifted right 1 bit following the jump).

** 0323 - Jump if Q is minus
If Q_0 is equal to one, jump to the specified address and store the address following the jump in the *PA. (The Q-register is circular shifted left 1 bit). If Q_0 (sign bit) is unequal to one, proceed to the next instruction in sequence.

** 0331 - Jump if D is positive
If D_0 is equal to zero, jump to the specified address and store the address of the next instruction following the jump in the *PA. If D_0 is unequal to zero, proceed to the next instruction in sequence.

** 0333 - Jump if D is negative
If D_0 is equal to one, jump to the specified address and store the address of the next instruction following the jump in the *PA. If D_0 is unequal to one, proceed to the next instruction in sequence.

NOTE:

1. A jump is made to the left or right-hand instruction according to the function bit. For example, if one wishes to jump to a right-hand instruction the instruction 0300 becomes 2300.

2. Whenever the Q-register is involved in a jump, the contents of the Q-register are shifted circularly after the jump sensing is complete.

5. Miscellaneous Instructions

0000 - Halt
If the function bit, J, is equal to 0, halt.
If J = 1 and the toggle switch is on, the instruction becomes a breakpoint halt.

0001 - Transfer control to Input-Output Controls.

0002 - Inhibit clearing overflow flip flop.
Sets the 1 bit register (OVR) which remembers to inhibit clearing of the overflow flip flop equal to one and the overflow flip flop equal to zero.

0003 - Set overflow flip flop equal to one if the contents of the specified index register are equal to the address portion of memory location V; i.e., the address portion of the memory location specified by V and the function bit, J.

0010 - Add the contents of the address portion of the specified memory location to the contents of the specified index register and replace the index register with this sum.

0011 - Subtract the contents of the address portion of the specified memory location to the contents of the specified index register and replace the index register with this difference.

0012 - Transfer the contents of the specified index register to the address portion of the specified memory location. For example: if it is desired to transfer the contents of the specified index register to the address of the right-hand instruction of memory location V, 0012 becomes 2012.

0013 - Transfer the contents of the address portion of memory location V (left or right half depending on J) to the specified index register.

** 0020 - Address substitution
Bring the address (from *PA) following the last jump to specified memory location (left or right half depending on J).

0021 - Add A to D
Add the contents of D to the (A) and replace A with the sum.

** 0022 - Increment V by 1
The address portion of the specified memory location (left or right half depending on J) is increased by one.

0023 - Subtract D from A
The contents of D are subtracted from the C(A) and A is replaced by the difference.

6. Logical Instructions

** 0030 - And Q to V and store in D
Each bit of the Q-register is matched with the corresponding bit of V. When the corresponding bit of both Q and V is a one, a one replaces the contents of that position in the D-register. When the corresponding bit of Q or V is a zero, a zero replaces that position in the D-register. The contents of Q and V are unchanged.

** 0031 - And Q and V and store in A.
Same as 0030 with the exception the A and D-1 registers contain the results.

** 0032 - Or D to V
Each bit of the D-register is matched with the corresponding bit of V. When the corresponding bit of either or both is a one, a one replaces the contents of that position in V. When the corresponding bit of D and V is a zero, a zero replaces that position in V. The contents of D are unchanged.

** 0033 - Clear V
Zeros replace the contents of the specified memory location.

7. Input-Output

When the input-output instruction transfer to input-output controls (0001) occurs in the program register, it is assumed that the D-register contains a 48-bit input-output instruction to be executed. The input-output word is divided into four fundamental parts, as follows:

$D\ (2^{\circ}$ through $2^{-15}) = D_s$ (16 bits)
$D\ (2^{-16}$ through $2^{-23}) = D_u$ (8 bits)
$D\ (2^{-24}$ through $2^{-39}) = D_n$ (16 bits)
$D\ (2^{-40}$ through $2^{-47}) = D_f$ (8 bits)

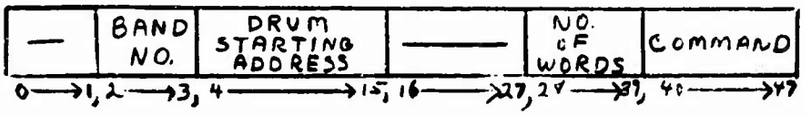
D_s stores the starting address of the magnetic drum or magnetic tape, if either is involved.

D_u designates the tape or drum unit involved.

D_n determines the quantity of information involved such as - number of words, number of magnetic tape blocks, or number of punched cards.

D_f resembles the command portion of an instruction, in that it designates the input-output function that is to be performed.

A. Type A



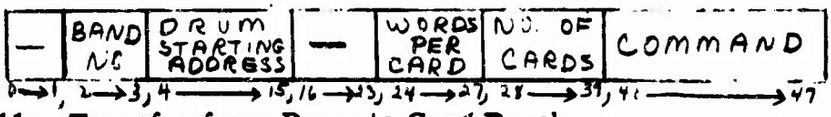
0102 - Transfer from Core to Drum
This instruction transfers the specified number of words from core to the specified drum address and band number.

0201 - Transfer from drum to core
This instruction transfers the specified number of words starting at the designated drum address and band number to core.

0210 - Transfer from Drum to Paper Tape
This instruction transfers the specified number of words starting at the designated drum address and band number to the paper tape reader.

1002 - Transfer from Paper Tape Reader to Drum

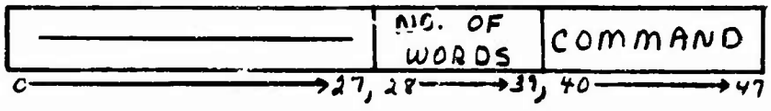
B. Type B



0211 - Transfer from Drum to Card Punch
This instruction punches on IBM cards the specified number of words starting at the designated drum address and band number.
(From one to two words may be punched on one card).

1102 - Transfer from Card Reader to Drum

C. Type C



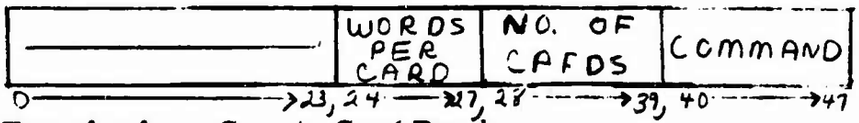
0110 - Transfer from Core to Paper Tape Punch.
This instruction punches onto paper tape a specified number of words starting at the designated memory address.

1001 - Transfer from Paper Tape Reader to Core.

0113 - Transfer from Core to Flexowriter.

1301 - Transfer from Flexowriter to Core.

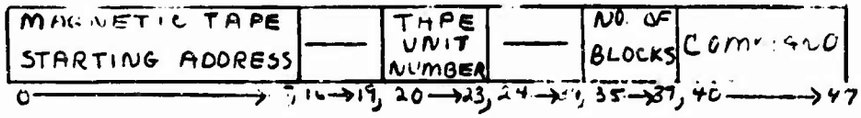
D. Type D



0111 - Transfer from Core to Card Punch.
This instruction punches a specified number of words on cards from the designated memory locations or locations.

1101 - Transfer from Card Reader to Core.

E. Type E



2001 - Transfer from Magnetic Tape to Core, Forward, Mode 1.
This instruction reads the specified number of blocks into the computer and then returns control to the computer. If the first block address is not the specified address, a search is automatically made for the specified address and when found continues to read forward.

3001 - Transfer from Magnetic Tape to Core, Forward, Mode 2
(Opposite of 2001. See Note 3 for explanation of Mode 2).

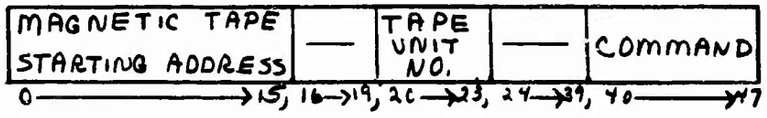
2201 - Transfer from Magnetic Tape to Core, Reverse, Mode 1.
This instruction starts the tape in reverse, verifies that the first block address is the specified address, and read the specified number of blocks into the computer. If the first block is not the specified address, a search is automatically made for the specified block address.

3201 - Transfer from Core to Magnetic Tape, Reverse, Mode 2.

0120 - Transfer from Core to Magnetic Tape, Forward, Mode 1.
This instruction verifies that the first block address is the specified address and writes the specified number of blocks on the designated tape unit number. If the first block address is not the specified address, the unit will automatically search for the specified address.

0130 - Transfer from Core to Magnetic Tape, Forward, Mode 2.

F. Type F



2020 - Search forward on Magnetic Tape, Mode 1.
The tape is brought to a position such that when started in a forward direction the first block read is the specified block. This is preparatory to reading forward.

3020 - Search forward on Magnetic Tape, Mode 2.

2220 - Search reverse on Magnetic Tape, Mode 1.
The tape is brought to a position such that when started in a reverse direction the first block is the specified block. This is preparatory to reading reverse.

3220 - Search reverse on Magnetic Tape, Mode 2.

2022 - Rewind Magnetic Tape.
Runs the tape in reverse until the end of tape is reached at which time the tape stops.
Any number of tapes may be rewound at the same time.
A single tape unit which is not being rewound is free to operate with the computer while the other tape is rewinding.

0301 - Transfer from Toggle Register to Core.
This instruction transfers the contents of the toggle register to the specified memory address.
(Toggle Register) goes to D-register and core.

NOTE:

1. Writing on magnetic tape can occur in a forward direction only.

2. The memory address for any input-output instruction is specified in the address portion of the input-output instruction (0001).

3. The two modes on magnetic tape have the following meaning:

    a. Mode 2 indicates if a parity check error occurs the computer halts

    b. Mode 1 indicates if a parity check error occurs the error is ignored. On writing in preparation of block addresses, the block that is written is read back for parity check - if an error exists, the area that is physically bad is not assigned a block address and the next block is assigned the specified address. On reading the block is ignored.

4. Whenever an input-output instruction refers to core memory the A-Register is automatically cleared to zero after control has been transferred to the I/0 control; - the reason being that a check sum is automatically computed in the A-Register and is available to the programmer when the I/0 controls have been released.

V. COMPUTER CONSOLE AND OPERATION

A. Registers and Displays

Program Register

An instruction word can be manually entered directly into the Program Register by means of the key switches, numbered corresponding to bit positions.

PR is cleared to zero by the CLEAR switches alongside the display neons. Four switches enable changing information in only part of the register; i.e., the address or instruction parts of the program register.

D Register

Manual entry directly into the D Register is provided by CLEAR and key switches similar to those for PR.

A and Q Registers

The A and Q registers are manually accessible by transfers from D. Manual transfer switches (to the left of the display neons for the A and Q registers) control the transfer from D.

Manual transfers from A or Q to D are controlled by the switches to the right of the register display neons.

Toggle Register

This register, whose contents are manually selected, can be transferred to the D register and core under control of the program; namely, the transfer from Toggle Register to Core (0301). The TR consists of 48 two-position toggle switches having a binary value of "0" in the down position.

M Register

The M Register is used solely for displaying the contents of any single memory location. The location is manually selected by the Memory Preset toggle switches on the console.

PA, MA, Index Register Displays

These console neons are connected to the registers indicated and display their contents.

MP Switches

These select a memory location for display in the M register. The down position of these switches is the binary "0" position. When the address selected by the MP switches coincides with the address used by the program, or program control, an option of stopping the computer is available.

MA=MP

The switch immediately to the right of the Memory Preset toggle switches and labelled ON, OFF, will cause the computer to halt if in the ON position and MA=MP. The computer will halt at the end of the memory cycle.

Overflow

The neon is lit when the overflow flip-flop is in the one state, indicating overflow. If the switch is on, the computer will stop at the end of the instruction during which overflow occurred, unless the following instruction is an overflow jump instruction.

Faults

If the Command fault light is on, it is the result of an illegitimate Command code in the instruction.

Memory fault is the result of incorrect temperature range within the core stack.

Input-output fault is a type due to a control failure in any of the I-0 equipment that prevents it from completing its operation. Generally, computer operations are not stopped. Thus, the faulty sector cannot do succeeding operations having failed to complete one. However, other types of I-0 equipment can be operated.

Breakpoint Switch

When the breakpoint switch is in the "on" position, the computer will stop before performing the breakpoint instruction.

Stop Key

When stop is depressed the computer will stop before performing the next instruction.

Advance key

The Advance key sets a flip-flop when depressed providing a steady advance signal; i.e., sequence each instruction in a logical order.

B. Operational Modes

The mode selection switches, on the lower section of the console, determine the operation rate of the basic machine cycle. In the RUN mode the computer operates continuously, from one program cycle to the next, in the absence of fault. In the step mode the computer stops each cycle. The stop key is used to manually intervene in the operation. The computer is stopped at the completion of the current program cycle when the stop key is depressed.

The Advance key, when depressed, will initiate the RUN mode of operation, if the RUN key was previously depressed. If the Step key is depressed, the Advance key is used to start the computer off on each cycle. The Advance key is ineffective when Stop is depressed. In the Step Mode the Step Oscillator can be substituted for the Advance key to make the program cycle repetition rate that of an oscillator whose frequency can be varied between one and ten pulses per second.

C. Operating examples

1. To initiate a program:

   a. Depress the stop key
   b. Manually enter into the D-Register the input/output instruction, - cards to core, magnetic tape to core, etc.
   c. Manually enter into the PR the instruction transfer control to input/output with the proper address.
   d. Depress the Run key.
   e. Depress the Advance key.

2. To change information in the A-Register when the computer has halted as a result of an error:

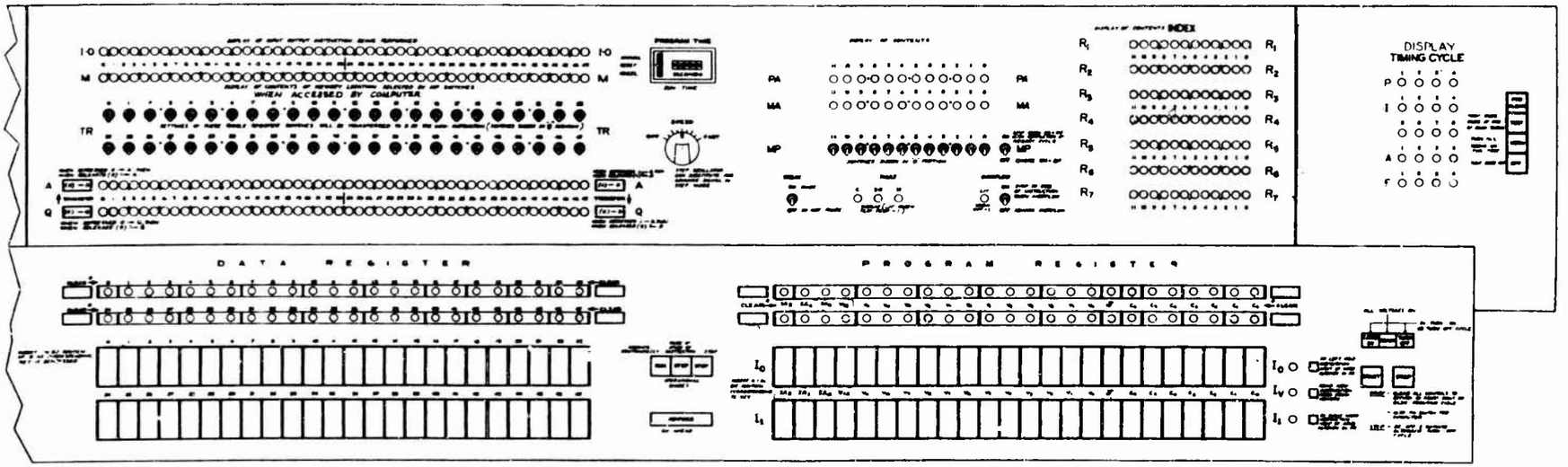
   a. Clear D-Register
   b. Manually enter into the D-Register the desired information
   c. Depress the transfer switch (D to A)
   d. Depress the Advance key

3. To perform new instructions before proceeding to the next instruction in sequence (the computer has halted or the Stop key has been depressed):

   a. Put computer in the step mode
   b. Clear PR
   c. Manually enter into PR the desired instruction(s)

d. Depress I_0 - left half (if desired)
e. Depress I_1 - right half (if desired)
f. Depress I_V - which will display in PR the next pair of instructions to be performed

(The sequence of 4-6 may be rearranged to suit specific needs).



CXPR Console

27

0000 Halt if J=0 BPT Halt if J=1			0300 Jmp	1000 $(A)+(V)\longrightarrow A$	1100 $(A)-(V)\longrightarrow A$	1200 $(Q)\times(V)\longrightarrow A, Q$	1300 $(A)/(V)\longrightarrow Q$
0001 I/O CONTROL TRANSFER		0201 Ordinary shift left Q and A	0301 Jmp if OF=0	1001 $(A)+(V)\longrightarrow V$	1101 $(A)-(V)\longrightarrow V$	1201 $(Q)\times(V)\longrightarrow V$	1301 $(A)/(V)\longrightarrow V$
0002 Inhibit clearing OVR	0102 $(V)\longrightarrow Q$		0302 Halt and Jmp if BPT on: Jmp if BPT off	1002 $(V)\longrightarrow A$	1102 $-(V)\longrightarrow A$	1202 $(Q)\times(V)_R\longrightarrow A$	1302 $(A)/(V)_R\longrightarrow Q$
0003 if $(IA)=(V_A)$, $1\longrightarrow OF$	0103 $(V)\longrightarrow D$	0203 Sign shift left Q and A	0303 Jmp if OF=1	1003 $(V)\longrightarrow A$ and V	1103 $-(V)\longrightarrow V$	1203 $(Q)\times(V)_R\longrightarrow V$	1303 $(A)/(V)_R\longrightarrow V$
0010 $IA+V_A\longrightarrow IA$	0110 $(A)\longrightarrow V$	0210 Ordinary shift right A	0310 Jmp if (A) =0	1010 $(A)+\lvert(V)\rvert\longrightarrow A$	1100 $(A)-\lvert(V)\rvert\longrightarrow A$	1210 $(Q)\times\lvert(V)\rvert\longrightarrow A, Q$	1310 $(A)/\lvert(V)\rvert\longrightarrow Q$
0011 $IA-V_A\longrightarrow 1_A$		0211 Ordinary shift left A	0311 Jmp if $A\neq 0$	1011 $(A)+\lvert(V)\rvert\longrightarrow V$	1111 $(A)-\lvert(V)\rvert\longrightarrow V$	1211 $(Q)\times\lvert(V)\rvert\longrightarrow V$	1311 $(A)/\lvert(V)\rvert\longrightarrow V$
0012 $(IA)\longrightarrow V_A$	0112 $(A)\longrightarrow Q$	0212 Sign shift right A	0312 Jmp if (A) =(D)	1012 $\lvert(V)\rvert\longrightarrow A$	1112 $-\lvert(V)\rvert\longrightarrow A$	1212 $(Q)\times\lvert(V)\rvert_R\longrightarrow A$	1312 $(A)/\lvert(V)\rvert_R\longrightarrow Q$
0013 $(V_A)\longrightarrow IA$		0213 Sign shift left A	0313 Jmp if A< 0	1013 $\lvert(V)\rvert\longrightarrow V$	1113 $-\lvert(V)\rvert\longrightarrow V$	1213 $(Q)\times\lvert(V)\rvert_R\longrightarrow V$	1313 $(A)/\lvert(V)\rvert_R\longrightarrow V$
0020 $(*PA)\longrightarrow V_A$	0120 $(Q)\longrightarrow V$	0220 Ordinary shift right Q	0320 Jmp if $Q_{47}=0$	1020 $(A)+(Q)\longrightarrow A$	1120 $(A)-(Q)\longrightarrow A$	1220 $(Q)\times(A)\longrightarrow A, Q$	1320 $(A)/(Q)\longrightarrow Q$
0021 $(A)+(D)\longrightarrow A$		0221 Ordinary shift left Q	0321 Jmp if $Q\neq 0$	1021 $(A)+(Q)\longrightarrow V$	1121 $(A)-(Q)\longrightarrow V$	1221 $(Q)\times(A)\longrightarrow V$	1321 $(A)/(Q)\longrightarrow V$
0022 $(V_A)+1\longrightarrow V_A$		0222 Sign shift right Q	0322 Jmp if $Q_{47}=1$	1022 $(Q)\longrightarrow A$	1122 $-(Q)\longrightarrow A$	1222 $(Q)\times(A)_R\longrightarrow A, Q$	1322 $(A)/(Q)_R\longrightarrow Q$
0023 $(A)-(D)\longrightarrow A$		0223 Sign shift left Q	0323 Jmp if $Q< 0$	1023 $(Q)\longrightarrow V$ and A	1123 $-(Q)\longrightarrow V$	1223 $(Q)\times(A)_R\longrightarrow V$	1323 $(A)/(Q)_R\longrightarrow V$
0030 $(Q)\cap(V)\longrightarrow D$	0130 $(D)\longrightarrow V$	0230 Ordinary shift right A and Q		1030 $(A)+\lvert(Q)\rvert\longrightarrow A$	1130 $(A)-\lvert(Q)\rvert\longrightarrow A$	1230 $(Q)\times\lvert(A)\rvert\longrightarrow A, Q$	1330 $(A)/\lvert(Q)\rvert\longrightarrow Q$
0031 $(Q)\cap(V)\longrightarrow A$		0231 Ordinary shift left A and Q	0331 Jmp if $D\neq 0$	1031 $(A)+\lvert(Q)\rvert\longrightarrow V$	1131 $(A)-\lvert(Q)\rvert\longrightarrow V$	1231 $(Q)\times\lvert(A)\rvert\longrightarrow V$	1331 $(A)/\lvert(Q)\rvert\longrightarrow V$
0032 $(D)\cup(V)\longrightarrow V$		0232 Sign shift right A and Q		1032 $\lvert(Q)\rvert\longrightarrow A$	1132 $-\lvert(Q)\rvert\longrightarrow A$	1232 $(Q)\times\lvert(A)\rvert_R\longrightarrow A$	1332 $(A)/\lvert(Q)\rvert_R\longrightarrow Q$
0033 Clear V		0233 Sign shift left A and Q	0333 Jmp if $D< 0$	1033 $\lvert(Q)\rvert\longrightarrow V$	1133 $-\lvert(Q)\rvert\longrightarrow V$	1233 $(Q)\times\lvert(A)\rvert_R\longrightarrow V$	1333 $(A)/\lvert(Q)\rvert_R\longrightarrow V$

ARITHMETIC ORDERS

ORDER	EXPLANATION	V	D	A	Q	ORDER	EXPLANATION	V	D	A	Q
1000	(A) +(V)⟶A	(V)	(V)	(A) + (V)	(Q)	1100	(A) -(V)⟶A	(V)	(V)	(A) -(V)	(Q)
1001	(A) +(V)⟶V	(A) + (V)	(A) + (V)	(A) + (V)	(Q)	1101	(A) -(V)⟶V	(A) -(V)	(A) -(V)	(A) -(V)	(Q)
1002	(V)⟶A	(V)	(V)	(V)	(Q)	1102	-(V)⟶A	(V)	(V)	-(V)	(Q)
1003	(V)⟶A and V	(V)	(V)	(V)	(Q)	1103	-(V)⟶V	-(V)	-(V)	-(V)	(Q)
1010	(A) + |(V)|⟶A	(V)	(V)	(A) + |(V)|	(Q)	1110	(A) - |(V)|⟶A	(V)	(V)	(A) - |(V)|	(Q)
1011	(A) + |(V)|⟶V	(A) + |(V)|	(A) + |(V)|	(A+ |(V)|	(Q)	1111	(A) - |(V)|⟶V	(A)- |(V)|	(A)- |(V)|	(A)- |(V)|	(Q)
1012	|(V)|⟶A	(V)	(V)	|(V)|	(Q)	1112	- |(V)|⟶A	(V)	(V)	- |(V)|	(Q)
1013	|(V)|⟶V	|(V)|	|(V)|	|(V)|	(Q)	1113	- |(V)|⟶V	- |(V)|	- |(V)|	- |(V)|	(Q)
1020	(A) +(Q)⟶A	(V)	(Q)	(A) +(Q)	(Q)	1120	(A) -(Q)⟶A	(V)	(Q)	(A) -(Q)	(Q)
1021	(A) +(Q)⟶V	(A) +(Q)	(A)+(Q)	(A) +(Q)	(Q)	1121	(A) -(Q)⟶V	(A) -(Q)	(A) -(Q)	(A) -(Q)	(Q)
1022	(Q)⟶A	(V)	(Q)	(Q)	(Q)	1122	-(Q)⟶A	(V)	(Q)	-(Q)	(Q)
1023	(Q)⟶V and A	(Q)	(Q)	(Q)	(Q)	1123	-(Q)⟶V	-(Q)	-(Q)	-(Q)	(Q)
1030	(A) + |(Q)|⟶A	(V)	(Q)	(A) + |(Q)|	(Q)	1130	(A) - |(Q)|⟶A	(V)	(Q)	(A) - |(Q)|	(Q)
1031	(A)+ |(Q)|⟶V	(A)+ |(Q)|	(A)+ |(Q)|	(A) + |(Q)|	(Q)	1131	(A) - |(Q)|⟶V	(A)- |(Q)|	(A) - |(Q)|	(A) - |(Q)|	(Q)
1032	|(Q)|⟶A	(V)	(Q)	|(Q)|	(Q)	1132	- |(Q)|⟶A	(V)	(Q)	- |(Q)|	(Q)
1033	|(Q)|⟶V	|(Q)|	|(Q)|	|(Q)|	(Q)	1133	- |(Q)|⟶V	- |(Q)|	- |(Q)|	- |(Q)|	(Q)

ARITHMETIC ORDERS

ORDER	EXPLANATION	V	D		Q	ORDER	EXPLANATION	V	D	A	Q
1200	$(Q)\times(V)\longrightarrow A, Q$	(V)	(V)	(Q) x (V) Major Product Not rounded	(Q) x (V) Minor Product	1300	$(A)/(V)\longrightarrow Q$	(V)	(V)	Remainder	Quotient
1201	$(Q)\times(V)\longrightarrow V$	(Q) x (V)	(Q) x (V)			1301	$(A)/(V)\longrightarrow V$	(A)/(V)	(A)/(V)	Remainder	Quotient
1202	$(Q)\times(V)_R\longrightarrow A$	(V)	(V)	(Q) x (V) Major Product Rounded	(Q) x (V) Minor Product $+2^{-1}$	1302	$(A)/(V)_R\longrightarrow Q$	(V)	(V)	Remainder	Quotient Rounded
1203	$(Q)\times(V)_R\longrightarrow V$	$(Q)\times(V)_R$	$(Q)\times(V)_R$			1303	$(A)/(V)_R\longrightarrow V$	$(A)/(V)_R$	$(A)/(V)_R$	Remainder	
1210	$(Q)\times\|V\|\longrightarrow A, Q$	(V)	(V)	(Q) x |(V)| Major Product Not Rounded	Minor Product	1310	$(A)/\|V\|\longrightarrow Q$	(V)	(V)	Remainder	(A)/ |(V)| Not Rounded
1211	$(Q)\times\|V\|\longrightarrow V$	(Q) x |(V)|	(Q) x |(V)|		Minor Product	1311	$(A)/\|V\|\longrightarrow V$	(A)/ |(V)|	(A)/ |(V)|	Remainder	
1212	$(Q)\times\|V\|_R\longrightarrow A, Q$	(V)	(V)	(Q) x |(V)| Rounded	Minor Product $+2^{-1}$	1312	$(A)/\|V\|_R\longrightarrow Q$	(V)	(V)	Remainder	Quotient Rounded
1213	$(Q)\times\|V\|_R\longrightarrow V$	(Q) x |(V)|	(Q) x |(V)|		"	1313	$(A)/\|V\|_R\longrightarrow V$	$(A)/\|V\|_R$	$(A)/\|V\|_R$	Remainder	
1220	$(Q)\times(A)\longrightarrow A, Q$	(V)	(A)	(Q) x (A) Major Product	Minor Product	1320	$(A)/(Q)\longrightarrow Q$	(V)	(Q)	Remainder	(A)/(Q) Not Rounded
1221	$(Q)\times(A)\longrightarrow V$	(Q) x (A)	(Q) x (A)			1321	$(A)/(Q)\longrightarrow V$	(A)/ (Q)	(A)/(Q)	Remainder	
1222	$(Q)\times(A)_R\longrightarrow A, Q$	(V)	(A)	(Q) x (A) Rounded	Minor Product $+2^{-1}$	1322	$(A)/(Q)_R\longrightarrow Q$	(V)	(Q)	Remainder	Quotient Rounded
1223	$(Q)\times(A)_R\longrightarrow V$	$(Q)\times(A)_R$	$(Q)\times(A)_R$			1323	$(A)/(Q)_R\longrightarrow V$	$(A)/(Q)_R$	$(Q)/(Q)_R$	Remainder	
1230	$(Q)\times\|A\|\longrightarrow A, Q$	(V)	(A)	(Q) x |(A)| Major Product	Minor Product	1330	$(A)/\|Q\|\longrightarrow Q$	(V)	(Q)	Remainder	(A)/ |(Q)| Not Rounded
1231	$(Q)\times\|A\|\longrightarrow V$	(Q) x |(A)|	(Q) x |(A)|			1331	$(A)/\|Q\|\longrightarrow V$	(A)/ |(Q)|	(A)/ |(Q)|	Remainder	
1232	$(Q)\times\|A\|_R\longrightarrow A$	(V)	(A)	(Q) x |(A)| Rounded	Minor Product	1332	$(A)/\|Q\|_R\longrightarrow Q$	(V)	(Q)	Remainder	Quotient Rounded
1233	$(Q)\times\|A\|_R\longrightarrow V$	$(Q)\times\|(A)\|_R$	$(Q)\times\|(A)\|_R$			1333	$(A)\ \|Q\|_R\longrightarrow V$	$(A)/\|(Q)\|_R$	$(A)/\|(Q)\|_R$	Remainder	

ARITHMETIC ORDERS

ORDER	EXPLANATION	V	D	A	Q	ORDER	EXPLANATION	V	D	A	Q
1000	(A) +(V)⟶A	(V)	(V)	(A) + (V)	(Q)	1100	(A) -(V)⟶A	(V)	(V)	(A) -(V)	(Q)
1001	(A) +(V)⟶V	(A) + (V)	(A) + (V)	(A) + (V)	(Q)	1101	(A) -(V)⟶V	(A) -(V)	(A) -(V)	(A) -(V)	(Q)
1002	(V)⟶A	(V)	(V)	(V)	(Q)	1102	-(V)⟶A	(V)	(V)	-(V)	(Q)
1003	(V)⟶A and V	(V)	(V)	(V)	(Q)	1103	-(V)⟶V	-(V)	-(V)	-(V)	(Q)
1010	(A) + |(V)|⟶A	(V)	(V)	(A) + |(V)|	(Q)	1110	(A) - |(V)|⟶A	(V)	(V)	(A) - |(V)|	(Q)
1011	(A) + |(V)|⟶V	(A) + |(V)|	(A) + |(V)|	(A+ |(V)|	(Q)	1111	(A) - |(V)|⟶V	(A)- |(V)|	(A)- |(V)|	(A) - |(V)|	(Q)
1012	|(V)|⟶A	(V)	(V)	|(V)|	(Q)	1112	- |(V)|⟶A	(V)	(V)	- |(V)|	(Q)
1013	|(V)|⟶V	|(V)|	|(V)|	|(V)|	(Q)	1113	- |(V)|⟶V	- |(V)|	- |(V)|	- |(V)|	(Q)
1020	(A) +(Q)⟶A	(V)	(Q)	(A) +(Q)	(Q)	1120	(A) -(Q)⟶A	(V)	(Q)	(A) -(Q)	(Q)
1021	(A) +(Q)⟶V	(A) +(Q)	(A)+(Q)	(A) +(Q)	(Q)	1121	(A) -(Q)⟶V	(A) -(Q)	(A) -(Q)	(A) -(Q)	(Q)
1022	(Q)⟶A	(V)	(Q)	(Q)	(Q)	1122	-(Q)⟶A	(V)	(Q)	-(Q)	(Q)
1023	(Q)⟶V and A	(Q)	(Q)	(Q)	(Q)	1123	-(Q)⟶V	-(Q)	-(Q)	-(Q)	(Q)
1030	(A) + |(Q)|⟶A	(V)	(Q)	(A) + |(Q)|	(Q)	1130	(A) - |(Q)|⟶A	(V)	(Q)	(A) - |(Q)|	(Q)
1031	(A) + |(Q)|⟶V	(A) + |(Q)|	(A) + |(Q)|	(A) + |(Q)|	(Q)	1131	(A) - |(Q)|⟶V	(A)- |(Q)|	(A) - |(Q)|	(A) - |(Q)|	(Q)
1032	|(Q)|⟶A	(V)	(Q)	|(Q)|	(Q)	1132	- |(Q)|⟶A	(V)	(Q)	- |(Q)|	(Q)
1033	|(Q)|⟶V	|(Q)|	|(Q)|	|(Q)|	(Q)	1133	- |(Q)|⟶V	- |(Q)|	- |(Q)|	- |(Q)|	(Q)

ARITHMETIC ORDERS

ORDER	EXPLANATION	V	D	A	Q
1200	(Q) x (V) ⟶ A, Q	(V)	(V)	(Q) x (V) Major Product Not rounded	(Q) x (V) Minor Product
1201	(Q) x (V) ⟶ V	(Q) x (V)	(Q) x (V)		
1202	(Q) x (V) $_{R}$⟶ A	(V)	(V)	(Q) x (V) Major Product Rounded	(Q) x (V) Minor Product $+2^{-1}$
1203	(Q) x (V) $_{R}$⟶ V	(Q) x (V) $_{R}$	(Q) x (V) $_{R}$		
1210	(Q) x |(V)| ⟶ A, Q	(V)	(V)	(Q) x |(V)| Major Product Not Rounded	Minor Product
1211	(Q) x |(V)| ⟶ V	(Q) x |(V)|	(Q) x |(V)|		Minor Product
1212	(Q) x |(V)| $_{R}$⟶ A	(V)	(V)	(Q) x |(V)| Rounded	Minor Product $+2^{-1}$
1213	(Q) x |(V)| $_{R}$⟶ V	(Q) x |(V)|	(Q) x |(V)|		"
1220	(Q) x (A) ⟶ A, Q	(V)	(A)	(Q) x (A) Major Product	Minor Product
1221	(Q) x (A) ⟶ V	(Q) x (A)	(Q) x (A)		
1222	(Q) x (A) $_{R}$⟶ A, Q	(V)	(A)	(Q) x (A) Rounded	Minor Product $+2^{-1}$
1223	(Q) x (A) $_{R}$⟶ V	(Q) x (A) $_{R}$	(Q) x (A) $_{R}$		
1230	(Q) x |(A)| ⟶ A, Q	(V)	(A)	(Q) x |(A)| Major Product	Minor Product
1231	(Q) x |(A)| ⟶ V	(Q) x |(A)|	(Q) x |(A)|		
1232	(Q) x |(A)| $_{R}$⟶ A	(V)	(A)	(Q) x |(A)| Rounded	Minor Product
1233	(Q) x |(A)| $_{R}$⟶ V	(Q) x |(A)| $_{R}$	(Q) x |(A)| $_{R}$		

ORDER	EXPLANATION	V	D	A	Q
1300	(A)/(V) ⟶ Q	(V)	(V)	Remainder	Quotient
1301	(A)/(V) ⟶ V	(A)/(V)	(A)/(V)	Remainder	Quotient
1302	(A)/(V) $_{R}$⟶ Q	(V)	(V)	Remainder	Quotient Rounded
1303	(A)/(V) $_{R}$⟶ V	(A)/(V) $_{R}$	(A)/(V) $_{R}$	Remainder	
1310	(A)/ |(V)| ⟶ Q	(V)	(V)	Remainder	(A)/ |(V)| Not Rounded
1311	(A)/ |(V)| ⟶ V	(A)/ |(V)|	(A)/ |(V)|	Remainder	
1312	(A)/ |(V)| $_{R}$⟶ Q	(V)	(V)	Remainder	Quotient Rounded
1313	(A)/ |(V)| $_{R}$⟶ V	(A)/ |(V)| $_{R}$	(A)/ |(V)| $_{R}$	Remainder	
1320	(A)/(Q) ⟶ Q	(V)	(Q)	Remainder	(A)/(Q) Not Rounded
1321	(A)/(Q) ⟶ V	(A)/ (Q)	(A)/(Q)	Remainder	
1322	(A)/(Q) $_{R}$⟶ Q	(V)	(Q)	Remainder	Quotient Rounded
1323	(A)/(Q) $_{R}$⟶ V	(A)/ (Q) $_{R}$	(Q)/(Q) $_{R}$	Remainder	
1330	(A)/ |(Q)| ⟶ Q	(V)	(Q)	Remainder	(A)/ |(Q)| Not Rounded
1331	(A)/ |(Q)| ⟶ V	(A)/ |(Q)|	(A)/ |(Q)|	Remainder	
1332	(A)/ |(Q)| $_{R}$⟶ Q	(V)	(Q)	Remainder	Quotient Rounded
1333	(A) |(Q)| $_{R}$⟶ V	(A)/ |(Q)| $_{R}$	(A)/ |(Q)| $_{R}$	Remainder	

TRANSFER ORDERS

ORDER	EXPLANATION	V	D	A	Q
0102	(V)⟶Q	(V)	(V)	(A)	(V)
0103	(V)⟶D	(V)	(V)	(A)	(Q)
0110	(A)⟶V	(A)	(A)	(A)	(Q)
0112	(A)⟶Q	(V)	(A)	(A)	(A)
0120	(Q)⟶V	(Q)	(Q)	(A)	(Q)
0130	(D)⟶V	(D)	(D)	(A)	(Q)

SHIFT ORDERS

0201	Ordinary shift left Q and A	
0203	Sign shift left Q and A	
0210	Ordinary shift right A	
0211	Ordinary shift left A	
0212	Sign shift right A	
0213	Sign shift left A	
0220	Ordinary shift right Q	
0221	Ordinary shift left Q	
0222	Sign shift right Q	
0223	Sign shift left Q	
0230	Ordinary shift right A and Q	
0231	Ordinary shift left A and Q	
0232	Sign shift right A and Q	
0233	Sign shift left A and Q	

JUMP ORDERS

0300	Jmp	
0301	Jmp if OF=0	
0302	Halt and Jmp if BPT on, Jmp if BPT off.	
0303	Jmp if OF=1	
0310	Jmp if (A)=0	
0311	Jmp if A>0	
0312	Jmp if (A)=(D)	
0313	Jmp if A<0	
0320	Jmp if Q_{47}=0	
0321	Jmp if Q>0	
0322	Jmp if Q_{47}=1	
0323	Jmp if Q<0	
0331	Jmp if D>0	
0333	Jmp if D<0	

MISCELLANEOUS ORDERS

ORDER	EXPLANATION	V	D	A	Q
0000	Halt if J=0 BPT Halt if J=1				
0001	I/O Control Transfer				
0002	Inhibit Clearing OVR				
0003	If (IA)=(V_A), 1⟶OF				
0010	IA + V_A⟶IA				
0011	IA-V_A⟶I_A				
0012	(IA)⟶V_A				
0013	(V_A)⟶IA				
0020	(*PA)⟶V_A				
0021	(A)+(D)⟶A	(V)	(D)	(A)+(D)	(Q)
0022	(V_A)+1⟶V_A				
0023	(A)-(D)⟶A	(V)	(D)	(A)-(D)	(Q)
0030	(Q)∩(V)⟶D	(V)	(Q).(V)	(A)	(Q)
0031	(Q)∩(V)⟶A	(V)	(Q).(V)	(Q).(V)	(Q)
0032	(D)∪(V)⟶V	(D)U(V)	(D)	(A)	(Q)
0033	Clear V	0	(D)	(A)	(Q)

BPT ~ Breakpoint toggle switch

V_A ~ Address portion of V

MACHINE CODES

	Machine Code	Octal	Paper Tape Code
A	011000	30	110000
B	010011	23	100110
C	001110	16	011100
D	010010	22	100100
E	010000	20	100000
F	010110	26	101100
G	001011	13	010110
H	000101	05	001010
I	001100	14	011000
J	011010	32	110100
K	011110	36	111100
L	001001	11	010010
M	000111	07	001110
N	000110	06	001100
0	000011	03	000110
P	001101	15	011010
Q	011101	35	111010
R	001010	12	010100
S	010100	24	101000
T	000001	01	000010
U	011100	34	111000
V	001111	17	011110
W	011001	31	110010
X	010111	27	101110
Y	010101	25	101010
Z	010001	21	100010
0	011111	37	111110
1	101010	52	010101
2	111100	74	111001
3	111000	70	110001
4	110100	64	101001
5	110010	62	100101
6	110110	66	101101
7	111010	72	110101
8	110000	60	100001
9	011011	33	110110

MACHINE CODES (Cont'd)

Typewriter Operations

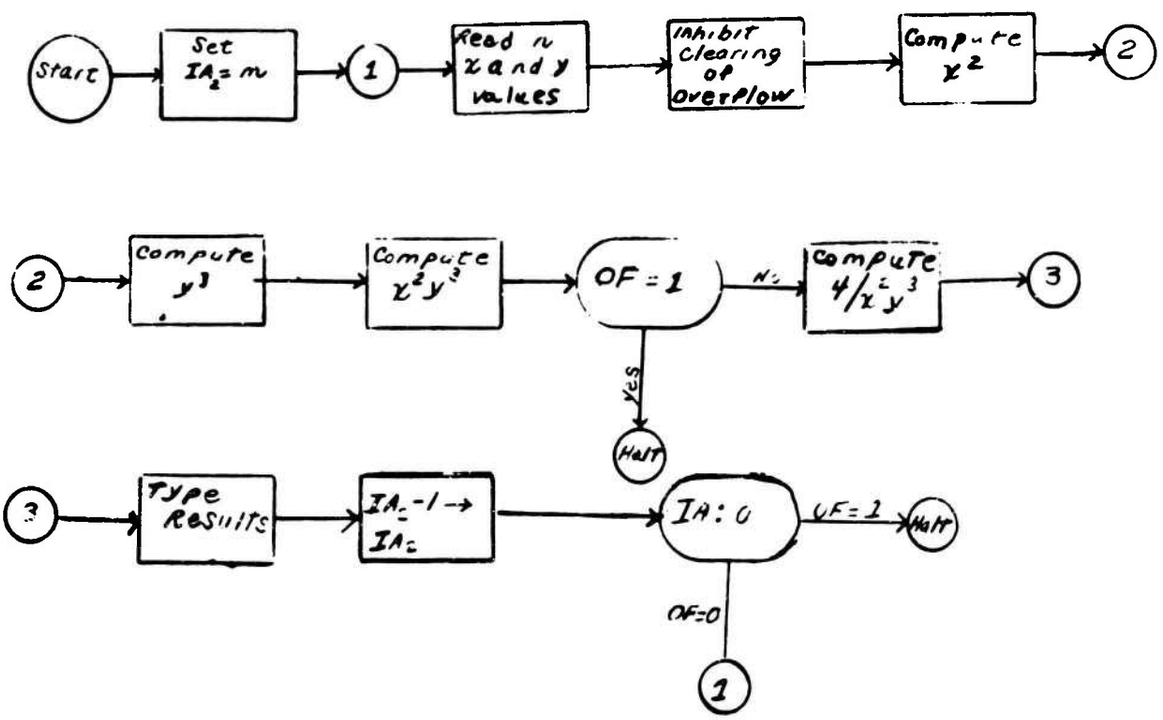
Space	000100	03	001000
Upper Case	100111	47	001111
Lower Case	101111	57	011111
Tab.	101001	51	010011
Carriage Return	100101	45	001011
Auto. Carriage Return	001000	10	010000

Symbols

UC	LC	Machine Code	Octal	Tape Code
(Open parenthesis	, Comma	100110	46	001101
) Closed parenthesis	. Period	100010	42	000101
· Multiply	= Equal	100100	44	001001
- Minus	- Hyphen or minus	101110	56	011101
/ Virgule	+ Plus	101100	54	011001
_ Underline	' Apostrophe	101000	50	010001
7 Level (Control)				0000001

SAMPLE PROGRAM

PROBLEM: Compute $4x^{-2}y^{-3}$ for n integral values of x and y



PHILCO CORPORATION, Gov't. & Industrial Division, Phila., 44, Pa. Computer Laboratory

Program Sheet 1 — Subroutine No. 1

Computer CXPQ — Program No. 1

Programmer BS — Date 7-24-59 — Page No. 95

Flow Box	Previous M. L.	M. L.$_{10}$	Ia	V_{10}	J	Command	Explanation
		0 I_0	2	0012	1	0013	Set $IA_2 = n$
		I_1	0	0013	0	0103	Transfer I/O instruction to D
		1 I_0	0	0014	0	0001	Transfer Control to I/O Controls
		I_1	0	0000	0	0002	Inhibit Clearing OF
		2 I_0	0	0014	0	0102	$x \rightarrow Q$
		I_1	0	0014	0	1200	$x \cdot x \rightarrow A$
		3 I_0	0	0017	0	0110	$(A) \rightarrow V$
		I_1	0	0015	0	0102	$y \rightarrow Q$
		4 I_0	0	0015	0	1200	$y \cdot y \rightarrow A$
		I_1	0	0015	0	0102	$y \rightarrow Q$
		5 I_0	0	0000	0	1220	$(A) \cdot (Q) \rightarrow A$
		I_1	0	0017	0	0102	$(V) \rightarrow Q$
		6 I_0	0	0000	0	1220	$(A) \cdot (Q) \rightarrow A$
		I_1	0	0012	0	0309	OF = 1, Yes
		7 I_0	0	0000	0	0112	$(A) \rightarrow Q$
		I_1	0	0016	0	1002	$4 \rightarrow A$
		8 I_0	0	0000	0	1322	$(A)/(Q)$
		I_1	0	0017	0	0120	$(Q) \rightarrow V$
		9 I_0	0	0018	0	0103	Transfer I/O instruction to D
		I_1	0	0017	0	0001	Transfer Control to I/O Controls
		10 I_0	2	0019	1	0011	$IA_2 - V_A \rightarrow IA_2$
		I_1	2	0019	0	0003	$IA_2 : V_A$
		11 I_0	0	0012	0	0303	OF = 1, Yes
		I_1	0	0000	1	0300	No
		12 I_0	0	0000	0	0000	Halt
		I_1	0	ne	0	0000	
		13 I_0					
		I_1	2	0001	0	1101	Card to Core

G & I Form No. 203

PHILCO CORPORATION, Gov't. & Industrial Division, Phila., 44, Pe. Computer Laboratory

Program Sheet 2 | Subroutine No.

Computer | Program No.

Programmer | Date | Page No. 36

Flow Box	Previous M. L.	M. L.	Ia	V	J	Command	Explanation
		14 I_0					X
		I_1					
		15 I_0					Y
		I_1					
		16 I_0					4
		I_1					
		17 I_0					V
		I_1					
		18 I_0	0	0000	0	0000	
		I_1	0	0001	0	0113	Core to flexowriter
		19 I_0	0	0000	0	0000	
		I_1	0	0001	0	0000	
		I_0					
		I_1					
		I_0					
		I_1					
		I_0					
		I_1					
		I_0					
		I_1					
		I_0					
		I_1					
		I_0					
		I_1					
		I_0					
		I_1					
		I_0					
		I_1					
		I_0					
		I_1					

G & I Form No. 203