APOLLO GUIDANCE COMPUTER

Information Series

ISSUE 2

MACHINE INSTRUCTIONS

FR-2-102A

10 March 1963

Changed 2 July 1964

CONTENTS

CONTENTS (cont)

ILLUSTRATIONS

ILLUSTRATIONS (cont)

TABLES

2-1. INTRODUCTION

2-2. This is the second issue of the AGCIS, which is published to inform the technical staff of MIT/IL and Raytheon about the Apollo guidance computer (AGC) subsystem. The various types of instructions provided for the AGC are defined in paragraphs 15-80 through 15-97 of Issue 15. The Machine Instructions are listed in table 15-12. This issue describes the execution of each Machine Instruction. Information pertaining to the Machine Instructions was originally taken from an advanced copy of MIT/IL Report R-393 and later updated.

2-3. A Machine Instruction is defined as a sequence of Actions (paragraph 15-14); each Action as a set of control pulses generated by the Sequence Generator (SG, Issue 5). Table 2-1 lists and defines those control pulses which are directly involved in the execution of a Machine Instruction. Actions are generated at a rate of 1.024 mc, or every 0.977 μsec. Twelve Actions make a subinstruction and take one MCT (memory cycle time). One or more subinstructions make one Machine Instruction. Table 2-2 lists the control pulses as generated at the various Actions of all subinstructions. Control pulse WP generated at Action 1 is needed for noise suppression in F memory (paragraph 15-36) and has little to do with the control of information flow.

2-4. EXECUTION OF MACHINE INSTRUCTIONS

2-5. The execution of Machine Instructions is described with the help of instruction flow charts. The format of such a chart, for instance figure 2-1, is similar to the machine organization diagram (figure 15-7). The F memory and the E memory are shown combined. The control registers are shown individually. Buffer-register B is shown with its direct (B) and its complement (C) side. The row marked "+1" symbolizes that circuitry of the Adder which, on command, adds the quantity plus-one to an operand entered into input register X or Y. The basic principle of the Adder operation has been described in paragraphs 15-27 through

TABLE 2-1

CONTROL PULSES

Pulse	Purpose
CI	Force carry into bit position 1 of Adder
CLG*	Clear (reset) bit position 15 through 0 of G △1
CTR	Decrement Loop Counter and set STAGE 2 at Action 12 if content of Loop Counter goes to zero △2
GP	Reset bit position 0 of G and enter into it the new parity bit generated by the parity pyramid. If c(S) = 0014, reset bit position 0 of OUT4 and enter generated parity bit there
KRPT	Clear Request Flip-Flop (in Program Interrupt Priority Control) that initiated program interrupt.
NISQ	Transfer contents of B_{16-13} to SQ at Action 12
RA	Read content of A into WA's
RB	Read content of B (direct side of B) into WA's
RC	Read content of C (complement side of B) into WA's
RB14	Read 020000 into WA's (a ONE into bit position 14)
RG, RG*	Read content of G into WA's △1 △3
RLP	Read content of LP into WA's △3
RP2	Reset bit position 0 of G and enter c(P2) into it
RQ	Read content of Q into WA's
RRPA	Read address provided by Program Interrupt Priority Control into WA's
RSA	Read specified address into register S
RSB	Read 100000 into WA's (a ONE into WA16 only)
RSC	Read content of addressed flip-flop register into WA's
RSCT	Read address provided by Counter Priority Control into WA's
RSTRT	Read STRT address into WA's
RU, RU*	Read content of U into WA's △1
RZ	Read content of Z into WA's
R1	Read 000001 into WA's
R1C	Read 177776 into WA's

TABLE 2-1

CONTROL PULSES (continued)

Pulse	Purpose
R2	Read 000002 into WAs
R22	Read 000022 into WAs
R24	Read 000024 into WAs
ST1	Set STAGE 1 at Action 12 △2
ST2	Set STAGE 2 at Action 12 △2
TMZ	Test for minus zero. Transfer contents of WAs to SQG and set BR2 if all bits are ONEs. Reset BR2 if all bits are ZEROs △2
TOV	Test for overflow or underflow. Transfer contents of WAs 16 and 15 to SQG and set BR2 in case of overflow, or set BR1 in case of underflow. Reset BR1 and BR2 for other conditions △2
TP	Test Parity. Allow incorrect parity to cause alarm if $c(S) \geq 0030$
TRSM	Test for RSM. Transfer c(S) to SQG and set STAGE 2 at Action 12 if c(S) = 0025 △2
TSGN	Test sign. Transfer content of WA16 to SQG and set BR1 if bit 16 is a ONE. Reset BR1 if bit 16 is a ZERO △2
TSGN2	Test sign. Transfer content of WA16 to SQG and set BR2 if bit 16 is a ONE. Reset BR2 if bit 16 is a ZERO △2
TSGN3	Test sign. Transfer content of WA16 to SQG and send signal to Program Priority Circuit if bit 16 is a ONE
WA	Reset A and write contents of WAs into A
WALP	Reset A and bit position 14 of LP and write contents of WAs into them △3
WB	Reset B and write contents of WAs into B
WG	Reset bit positions 15 through 1 of G and write contents of WAs into G △3
WG*	Write contents of WAs directly (without editing) into bit positions 15 through 1 of register G and the generated parity bit into bit position 0 △1 △3

TABLE 2-1

CONTROL PULSES (continued)

Pulse	Purpose
WLP	Reset LP and write contents of WAs into LP [3]
WP	Reset P and write contents of WAs into P
WP*	Write contents of WAs into P during last 3/4 μsec
WP2	Reset P2 and enter into it the new parity bit generated by parity pyramid. If c(S) = 0014, reset position 0 of OUT4 and enter c(P2) there
WQ	Reset Q and write contents of WAs into Q
WS	Reset S and write contents of WAs 12 through 1 into S
WSC	Reset addressed flip-flop register and write contents of WAs into it
WX, WX*	Write contents of WAs into X [1]
WY	Reset Y, X and CI flip-flop and write contents of WAs into Y [1]
WY*	Write contents of WAs into Y [1]
WZ	Reset Z and write contents of WAs into Z
WOVI	Inhibit program interruption at end of current instruction in case of overflow or underflow
WOVC	Increment or decrement OVCTR by executing-next PINC or MINC
WOVR	Deliver counter overflow or underflow to appropriate priority input as selected by the content of register S

NOTES:

[1] * indicates that read or write signal generated by that control pulse is 1 μsec long rather than 0.75 μsec

[2] BR1, BR2, STAGE 1, STAGE 2 and LOOP Counter will be discussed in a later issue when describing the SQG

[3] For editing operation see table 1-5

TABLE 2-5

Control Pulses for all Subinstructions

Instruction	Action 1 △3	Action 2	Action 3	Action 4	Action 5	Action 6	Action 7	Action 8	Action 9	Action 10	Action 11	Action 12	Figure Number (Page Number)
STD2	RZ WS WY CI WP	NO ACTION	CLG*	RU WZ	NO ACTION	NO ACTION	RSC RG WB WP	GP TP	RB WSC WG	NO ACTION	(NISQ) →	→	2-1 2-2 (2-7) (2-11)
TCO CODE 0 △1	RB WS WY CI WP	NO ACTION	CLG*	RA WOVI	NO ACTION	NO ACTION	RG RSC WB WP	GP TP RZ WQ	RB WSC WG	RU WZ	(NISQ) →	→	2-3 (2-14)
XCHO CODE 3 △1	RB WS WP	RA WP △4	CLG*	WP2	NO ACTION	NO ACTION	RSC RG WB WP	GP TP	RA WSC WG RP2	RB WA WOVI	(ST2)		2-4 (2-17)
CSO CODE 4 △1	RB WS WP	NO ACTION	CLG*	NO ACTION	NO ACTION	NO ACTION	RSC RG WB WP	GP TP	RB WSC WG	RC WA WOVI	(ST2)		2-5 (2-19)
TSO CODE 5 △1	RB WS WP	RA WB WP TOV	CLG*	NO ACTION - - - RZ WY CI	NO ACTION - - - RI RIC or WA WA	NO ACTION	NO ACTION - - - RU WZ	GP	RB WSC WG	RA WOVI	(ST2)		2-6 (2-21) - - - 2-7 (2-22)
MSKO CODE 7 △1	RB WS WP	RA WB	CLG*	RC WY	NO ACTION	NO ACTION	RSC RG WB WP	RU RC WA GP TP	NO ACTION	RA WB	RC WA WOVI (ST2)		2-8 (2-25)
ADO CODE 6 △1	RB WS WP	RA WY	CLG*	NO ACTION	NO ACTION	NO ACTION	RSC RC WB WP	CP TP RB WX	RB WSC WG	NO ACTION	RU WA WOVI WOVC (ST2)		2-9 (2-27)
NDX0 CODE 2 △1	RB WS WP	NO ACTION	CLG*	RA WOVI	NO ACTION	NO ACTION	RSC RG WB WP	GP TP	RB WSC WG	TRSM	(ST1)		2-10 (2-29)
NDX1 CODE 2 △1	RZ WS WY CI WP	NO ACTION	CLG*	RU WZ	NO ACTION	RB WY	RSC RC WB WP	GP TP RB WX	RB WSC WG	NO ACTION	RU WB WOVI (NISQ) →	→	2-11 (2-3)
CCS0 CODE 1 △1	RB WS WP	RZ WY	CLG*	NO ACTION	NO ACTION	RSC RG WB WP TSGN	RC TMZ - - - RB TMZ	GP TP ¦ RI WX GP TP - - - R2 WX GP TP ¦ RI R2 WX GP TP	RB WSC WG	RC WA ¦ RIC WA - - - RB WA ¦ RIC WA	RU (ST1) WZ		2-12 (2-33) ¦ 2-13 (2-34) - - - 2-14 (2-35) ¦ 2-15 (2-36)
CCS1 CODE 1 △1	RZ WS WY CI WP	NO ACTION	CLG*	RU WZ	RA WY CI	NO ACTION	RSC RG WB WP	RU WB GP TP	NO ACTION	RC WA WOVI	RG RSC WB (NISQ) →	→	2-16 (2-38)
SUO CODE 6 △1 △2	RB WS WP	RA WY	CLG*	NO ACTION	NO ACTION	NO ACTION	RSC RG WB WP	GP TP RC WX	RB WSC WG	NO ACTION	RU WA WOVI WOVC (ST2)		2-17 (2-43)
MP0 CODE 4 △1 △2	RB WS WP	RA WB CLG* TSGN	RSC WG	RB WLP - - - RC WLP	RLP WA	NO ACTION	RG WY WP - - - RG WB WP	GP TP - - - RC WY GP TP	RU WB TSGN2	RA WLP TSGN ¦ RA RB 14 WLP TSGN	WALP (ST1) or RU WALP (ST1) ¦ RI RIC WALP (ST1) or RU WALP (ST1)		2-20 (2-48) ¦ 2-21 (2-49) - - - 2-22 (2-50) ¦ 2-23 (2-51)

NOTES: △1 Indicates Instruction Code - not Subinstruction Code.

△2 With EXTEND preceding.

Control pulse symbols followed by * indicate that the pulse takes 1 μsec rather than 0.75 μsec (also indicated with S).

△3 WP at Action 1 for noise suppression during F memory readout

△4 RA and WP suppressed if F memory is addressed

TABLE 2-5 (cont)

Control Pulses for all Subinstructions

Instruction	Action 1 △3	Action 2	Action 3	Action 4	Action 5	Action 6	Action 7	Action 8	Action 9	Action 10	Action 11	Action 12	Figure Number (Page Number)
MP1 CODE 4 △1 △2	RA WY No WP	RLP WA TSGN	NO ACTION or RB WX	RA WLP	RLP (TSGN)	RU WALP	RA WY	NO ACTION or RB WX	RLP WA	RA WLP (CTR)	RU WALP (ST1)		2-24 (2-52)
MP3 CODE 4 △1 △2	RZ WS WY CI WP	RLP TSGN	CLG*	RU WZ	RA WY	NO ACTION or RB WX	RG RSC WB WP	RLP WA GP TP	RB WSC WG	RA WLP	RU WALP (NISQ)	→	2-25 (2-53)
DV0 CODE 5 △1 △2	RB WS WP	RA WB CLG* TSGN	RSC WG	RC WA - - - NO ACTION	R1 WLP - - - R2 WLP	RA WQ	RG WG WP (TSGN)	RB WA GP TP	RLP R2 WB ¦ NO ACTION	RB WLP ¦ RC WA	R1 WB (ST1)		2-28 (2-65) ¦ 2-29 (2-66) - - - 2-30 (2-67) ¦ 2-31 (2-68)
DV1 CODE 5 △1 △2	R22 WS WP	RQ WG	RG RSB WQ WY	RA WX	RLP (TSGN2)	NO ACTION	RU (TSGN)	NO ACTION - - - RU WQ	RB RSB WG - - - RB WG	RG WB (TSGN)	(ST1) or RC WA (ST2) or RB WA (ST2)		2-32 (2-69) - - - 2-33 (2-70)
RPT1	R24 WY WS CI WP	NO ACTION	CLG*	NO ACTION	NO ACTION	NO ACTION	NO ACTION	NO ACTION	RZ WG	RU WZ	(ST1) (ST2)		2-34 (2-82)
RPT3	RZ WS WP	RRPA WZ	RZ CLG* KRPT	NO ACTION	NO ACTION	NO ACTION	NO ACTION	NO ACTION	RB WSC WG	NO ACTION	(ST2)		2-35 (2-83)
RSM (NDX1 of NDX 0025)	R24 WS WP	NO ACTION	CLG*	NO ACTION	NO ACTION	NO ACTION	RG WZ	NO ACTION	NO ACTION	NO ACTION	(NISQ)	→	2-36 (2-85)
PINC	RSCT WS WP	NO ACTION	CLG*	R1 △5 WY	NO ACTION	RG* WX* WP CI △5	TP	WP	RU* CLG* WP*	RU* WG* WOVR	NO ACTION		2-37 (2-87)
MINC	RSCT WS WP	NO ACTION	CLG*	R1C WY	NO ACTION	RG* WX* WP CI △5	TP	WP	RU* CLG* WP*	RU* WG* WOVR	NO ACTION		2-38 (2-89)
SHINC	RSCT WS WP	NO ACTION	CLG*	WY	NO ACTION	RG* WY* WX* WP TSGN3	TP	WP	RU* CLG* WP*	RU* WG* WOVR	NO ACTION		2-39 (2-91)
SHANC	RSCT WS WP	NO ACTION	CLG*	WY	NO ACTION	RG* WY* WX* WP TSGN3	TP CI	WP	RU* CLG* WP*	RU* WG* WOVR	NO ACTION		2-41 (2-94)
OINC	WS WP	NO ACTION	CLG*	NO ACTION	NO ACTION	NO ACTION	RSC RG	NO ACTION	NO ACTION	NO ACTION	NO ACTION		2-42 (2-96)
LINC	WS WP	NO ACTION	CLG*	NO ACTION	NO ACTION	NO ACTION	WSC WG WP	GP	NO ACTION	NO ACTION	NO ACTION		2-43 (2-97)

NOTES: △1 Indicates Instruction Code - not Subinstruction Code.

△2 With EXTEND preceding.

Control pulse symbols followed by * indicate that the pulse takes 1 μsec rather than 0.75 μsec (also indicated with S).

△5 R1 suppressed at Action 4 and CI generated at Action 6 if register 0047, 50, 51, 52 or 53 is addressed.

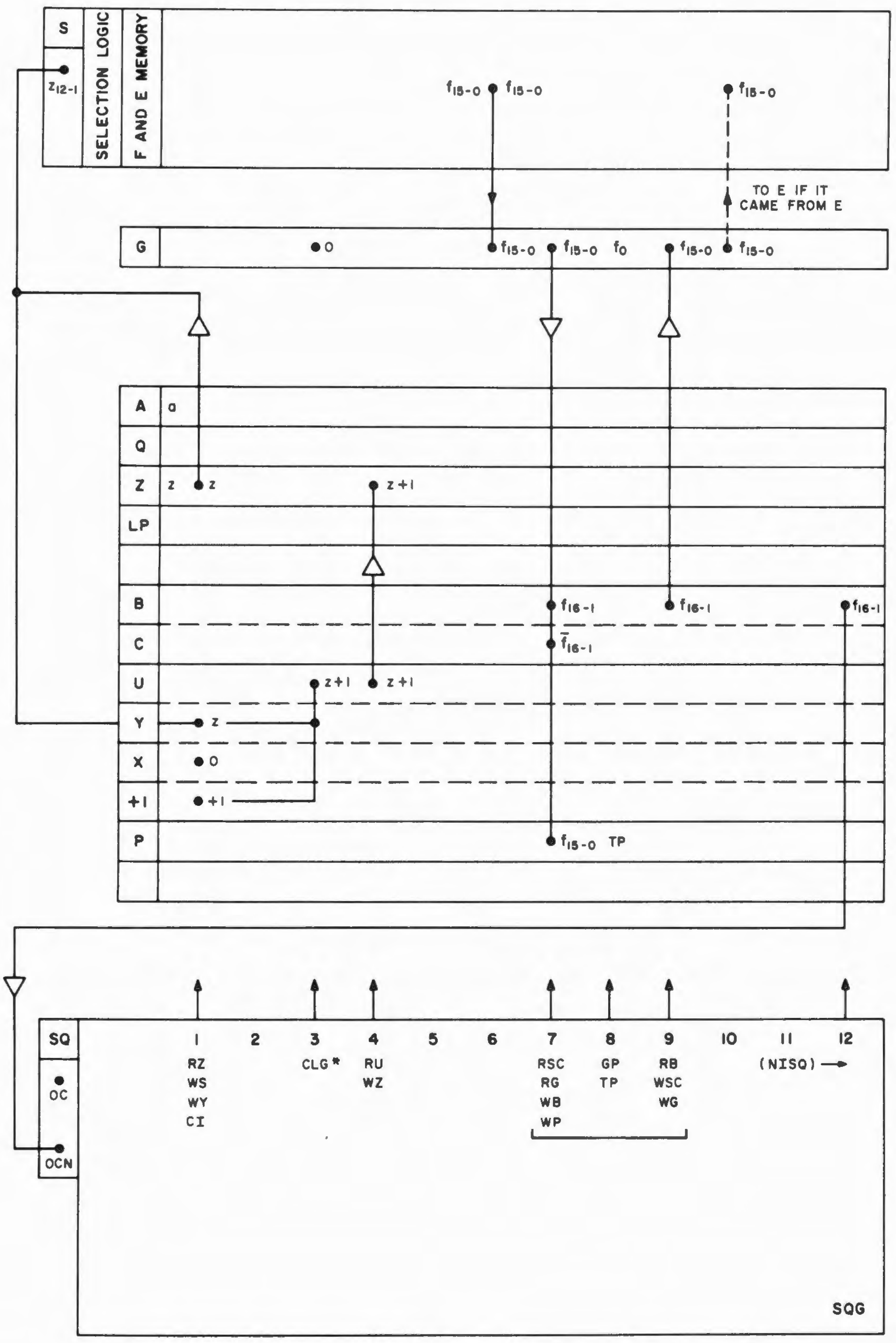

Figure 2-1. Subinstruction STD2 (Example for z $\geq$ 0020)

15-33. The write amplifiers (WA's) of figure 15-7 are symbolized in figure 2-1 by triangles placed into the flow lines. Registers are always signified by capital letters, their contents by small letters. Furthermore, $\bar{a}$ means complement of a. Control pulse symbols in parentheses indicate signals internal to the SQG. In the text, c(A) means content of A, b(A) means previous (before) content of (A), $\bar{c}$(A) indicates the complemented c(A), and c^e(A) indicates the edited c(A). Basic instructions and Extra Code Instructions, which belong in the category of Regular Instructions, are described first. Paragraphs 2-6 through 2-77 deal with normal cases, i. e. instructions of which the relevant address K refers to a location in the F or E memory. (Special attention is required for any address K which refers to an editing location.) Instructions with address K referring to registers A, Q, Z, and LP (special cases) are discussed in paragraphs 2-78 through 2-93 together with some other special cases of NDX K instructions.

2-6. BASIC INSTRUCTIONS

2-7. COMMON CHARACTERISTICS

2-8. A program portion written in basic machine language consists of a series of Basic Instructions and it is the responsibility of each instruction to make sure that its execution will be followed by the execution of another (the proper) instruction. The relevant address (figure 15-12) of each Basic Instruction (table 15-12) is normally used to specify the location of data to be worked with. In order to specify the sequence in which Basic Instructions are to be executed, the instructions are simply stored at locations in numerical order. By incrementing (by one) the address of an instruction presently executed, the location of the instruction to be executed next is defined. The address of the instruction to be executed next (the "next" address) is stored in register Z, the program counter. During the execution of an instruction, this address code in register Z is incremented (by one) in order to represent the proper address of the "next" instruction at the start of the subsequent instruction. In case the normal sequence of instruction is interrupted, the "next" address is stored in register Q for later use. Another duty of a Basic Instruction is to enter the entire code of the subsequent instruction (the instruction to be executed next) into register B so that it will be available during its execution. Finally, each Basic Instruction (in fact, all Regular Instructions) must enter the order code of the subsequent instruction into the SQ register in order to initiate its execution. (Involuntary Instructions cannot be written into a program and are free of the duties mentioned above.)

2-9. Most Basic Instructions consist of two subinstructions. Normally the first subinstruction carries out the specific task of the instruction, such as, adding data, transferring information, etc. The second subinstruction performs the common duties required of the instruction. Usually the second subinstruction is STD2 (Standard Subinstruction Two). Instruction flow chart figure 2-1 illustrates the execution of STD2 for $z \geq 0020$. In the box representing the sequence generator (SQG), four control pulses (RZ, WS, WY, and CI) are listed in the first column, indicating that these four pulses are generated at Action 1. The control pulses generated at each Action are listed in table 2-2. Pulse RZ reads the contents of register Z, i.e. the "next" address (z), into the write amplifiers. Note that if L is the location of the instruction being executed, then $z = c(Z) = (L + 1)$. Pulse WS resets (clears) the S register then writes the "next" address into it. Pulse WY clears input registers X and Y of the Adder, and writes c(Z) into Y. Control pulse CI forces a carry into the Adder, thereby causing a one to be added to z. The sum $(z + 1)$, i.e., the address "after the next" or the "second next" address, appears in output gates U with 3 μsec. Now, $(z + 1) = (L + 2)$. Control pulse CLG*, at Action 3, resets register G as symbolized by the 0. At Action 4, pulse RU reads quantity $(z + 1)$ to the write amplifiers, and pulse WZ clears Z and writes $(z + 1)$ into it.

2-10. The content of register S (z, entered at Action 1) causes the selection logic to gate the proper location for readout and write-in. If the address stored in S is equal to or larger than 0020 (all data are normally given in octal numbers), the corresponding location in the F or E memory will deposit its content (f or e) into register G during Action 6. If the address refers to a location in the E memory ($0020 \leq z \leq 1777$), the z drivers of the E memory are energized during Actions 10 through 12 and any content of G is written into the addressed location in the E memory. If the address stored in S is equal to or smaller than 0017, a flip-flop register is addressed for readout or write-in, as shown in figure 2-2 for the case of $z = 0001$.

2-11. Actions 7, 8, and 9, as shown in figures 2-1 and 2-2, are common to most subinstructions. This group of control pulses is referred to as the standard memory inquiry cycle (STMIC) and is symbolized by the bracket underneath. Control pulses RG, WB, and WP transfer data to be worked with or the subsequent instruction f from G to B and P. The complement of f appears at the C side of buffer-register B. Control pulse RSC has no effect unless a flip-flop register is addressed. Pulse GP gates the new parity bit (generated by the parity pyramid) for the word stored at register P into bit position 0 of G, as symbolized by f_o in G. Pulse TP (test parity) causes an alarm in case the parity of the word in register P is incorrect. The parity test is symbolized by TP in register P. (The operation of the Parity Block is described in paragraphs 15-34 through 15-36. In Issue 2 it is always assumed that the parity is correct and no consequences on incorrect parity are discussed. Control pulses RB and WG transfer the content of B into bit positions 1 through 15 of G. Pulse WSC has no effect since a flip-flop register has not been addressed.

2-12. It is worthwhile to remember that whenever a word is transferred from register G to any other flip-flop register except P, bit 15 of G is entered into bit positions 16 and 15 of the other register, as shown in table 1-6. Whenever a word is transferred from a flip-flop register to register G, bit 16 from the flip-flop register moves into bit position 15 of G and bit 15 of the flip-flop register gets lost. The parity bit location in position 0 of register G can only be transferred into position 0 of register P. Any new parity bit is fed directly into bit position 0

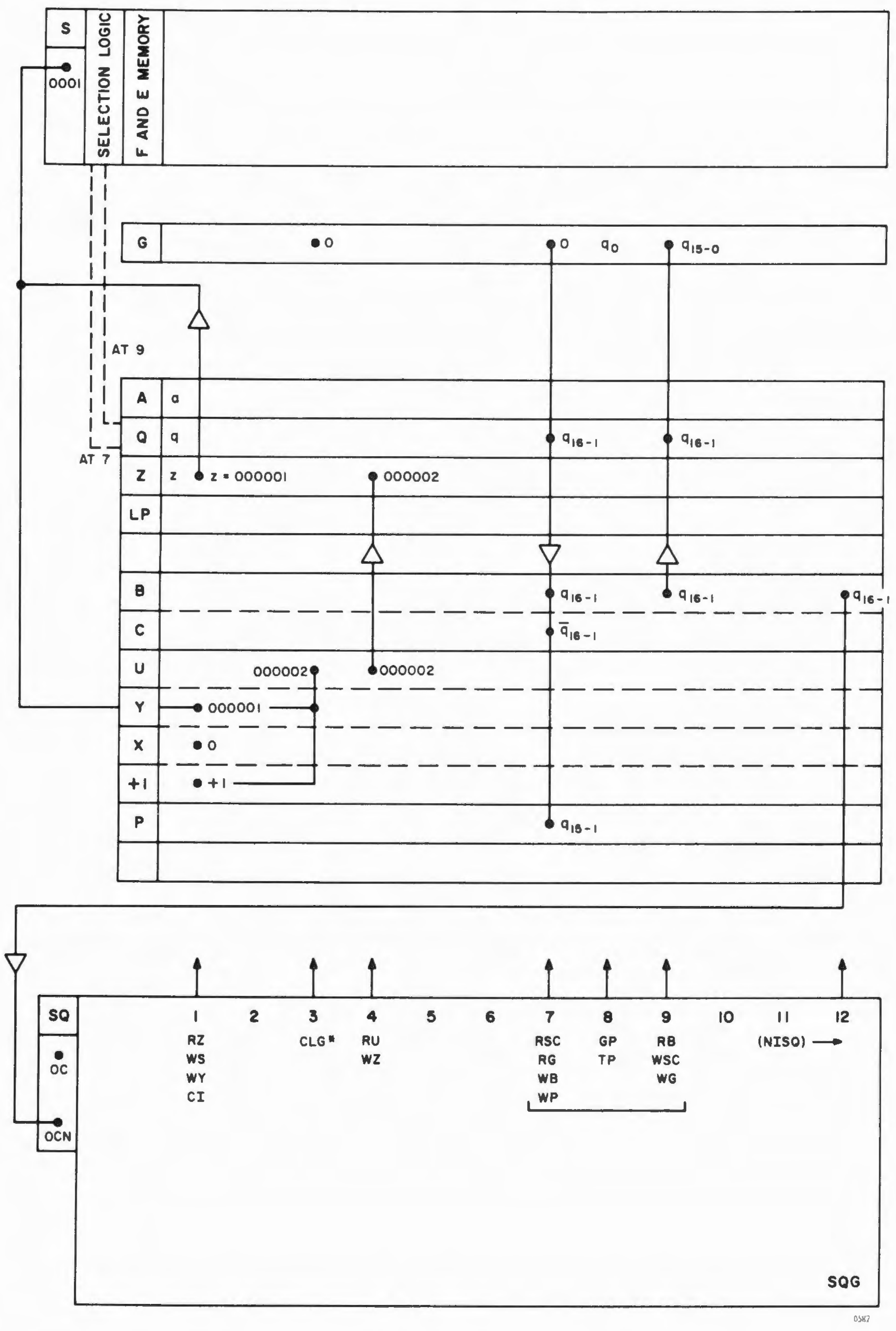

Figure 2-2. Subinstruction STD2 (Example for z = 0001)

of G. For example, the word 32614 stored in the F or E memory becomes 032614 when transferred into the accumulator (A), and word 62614 becomes 162614.

2-13. Control pulse NISQ generated at Action 11 causes the SQG to transfer (at Action 12, as indicated by an arrow) the order code OCN of the instruction to be executed next from bit positions 16 through 13 of B to SQ. If the transfer of data or of the subsequent instruction (paragraph 2-11) is not established by means of STD2 (i. e., if STD2 is not the second subinstruction), control pulse NISQ must be contained in another concluding subinstruction.

2-14. Figure 2-2 is very similar to figure 2-1 except that the addressed register is Q, a flip-flop register, instead of a location in the F or E memory. Control pulse RG at Action 7 has no real effect since register G is cleared at Action 3 and not loaded from the F or E memory thereafter. Since the address stored in S (0001) is lower than 0020, the selection logic signals the SQG to gate the proper flip-flop register for readout and write-in during Actions 7 and 9. This is indicated by dotted lines leading from the selection logic to register Q. Pulse RSC goes to all addressable flip-flop registers but is only able to read out the one which is addressed simultaneously. Control pulse WSC also goes to all addressable flip-flop registers, but only that register which is addressed at the same time is enabled to accept data. Since $c(S) \leq 0027$, control pulse TP is prevented from causing an alarm as stated in paragraph 15-35.

2-15. INSTRUCTION TC K (Transfer Control to K, Order Code 0)

2-16. TC K means: transfer program control to the instruction stored at location K. The entire operation TC K (for $0020 \leq K$) can be formulated as follows:

(1) Execute next the instruction located at K instead of at $z = (L + 1)$.

(2) Set $c(Z) = (TC\ K) + 1 = b(B) + 1$.

(3) Set $c(Q) = z = b(Z)$, $z = L + 1$.

(4) Restore $c(K) = b(K)$, if $0024 \leq K \leq 1777$.
If $0020 \leq K \leq 0023$, then $c(K) = b^e(K)$; b^e means before content edited, i.e. cycled or shifted.

The TC K instruction consists of only one subinstruction, TC0.

2-17. Figure 2-3 illustrates the execution of TC 6145 as an example. Assuming that the present instruction is located at address 2670 of the F memory, the command TC 6145 means that the instruction f located at 6145 shall be executed next instead of the instruction located at z = 2671, the "next" instruction of the present sequence of instructions. Instruction f might be the first instruction of another program, of a subroutine, etc. The address (z) of the "next" instruction is transferred from register Z to Q in order to be available in case a return to the original sequence of instructions is requested. The instruction TC Q transfers control to Q. After the execution of TC 6145, register Q contains 002671, which in fact is the instruction TC z, or TC 2671 in the case of the example, since the order code stored in bit positions 16 through 13 consists of four ZERO's.

2-18. The entire code of instruction TC 6145 (06145) was entered into register B during the execution of the previous instruction,and B now contains b = 006145. The order code 00 was entered into register SQ at Action 12 of the previous instruction. In so doing, the SQG was set to execute a TC instruction. At Action 1 the relevant address 6145 contained in bit positions 12 through 1 of B is transferred to S and the selection logic gates a location with

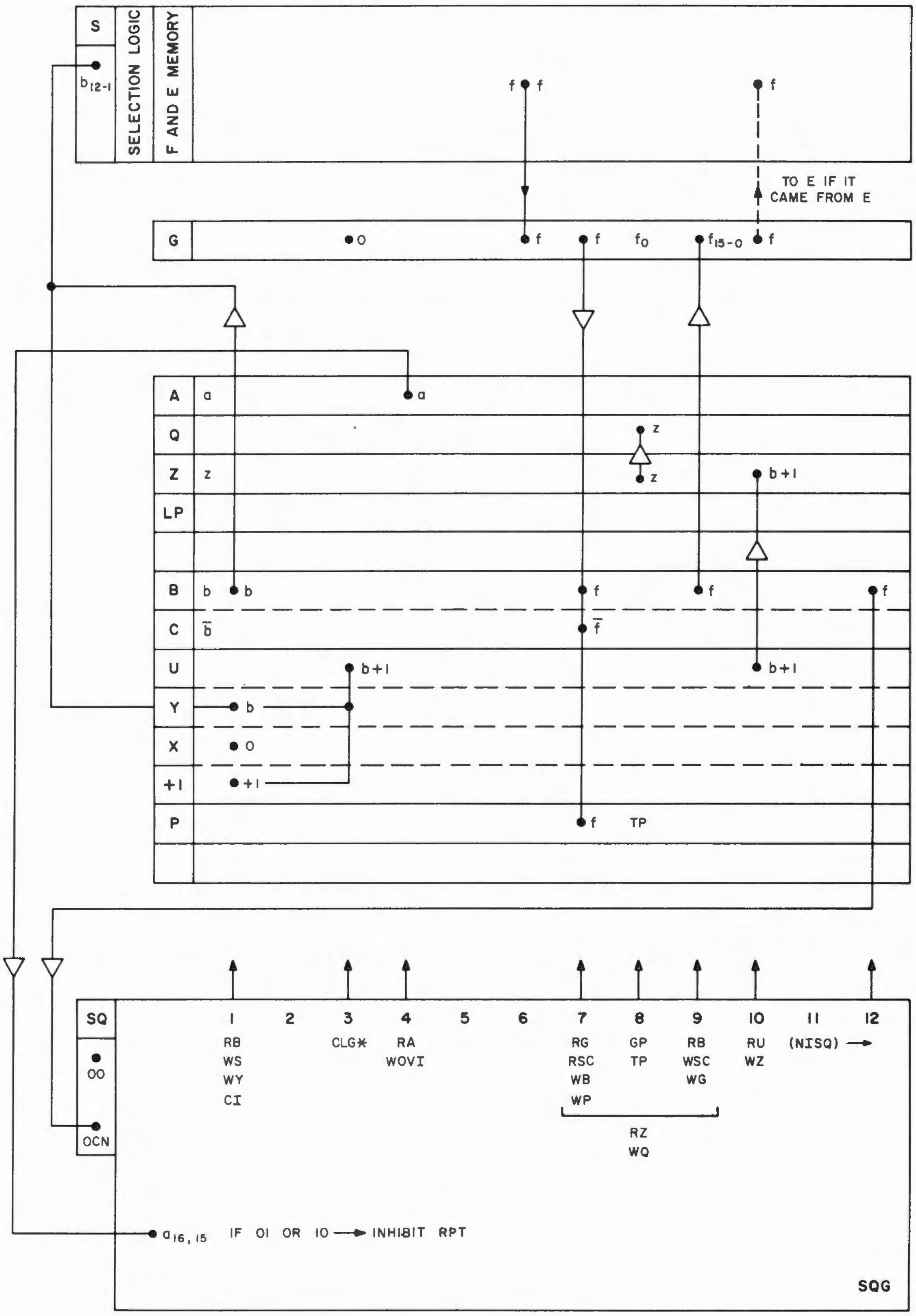

Figure 2-3. Subinstruction TC0

address 6145 for readout and write-in. The content of register BNK determines which location with address 6145 is selected. At the same time, the entire content of B is fed into Y, and X is cleared. Control pulse CI forces a one into the Adder, causing a one to be added to b. The sum (b + 1) = 006146 is available at U within 3 μsec. At Action 3, register G is cleared. At Action 4, bits 16 and 15 contained in A are transferred to the SQG for test of overflow or underflow. Assuming that both bits are ZEROs, or that both bits are ONEs, control pulse WOVI has no consequence. In case the two bits are not equal, interrupt at the end of the current instruction is inhibited in order to save the overflow bit. The overflow bit would be lost in the process of transferring the content of A to location 0024 as requested by a program interruption. Actions 7, 8, and 9 of instruction TC K are very similar to the STMIC with the exception that control pulses RZ and WQ are added. The purpose of RZ and WQ is to transfer "next" address z = 2671 from Z to Q. Instruction f located at address 6145 is transferred from the F memory into register G at Action 6, and from G into B and P at Action 7. At Action 8, the parity of word f is tested, an alarm caused in case of incorrect parity, and a new parity bit (f_o) for quantity f is generated and entered into G. At Action 9, instruction f is returned to G. At Action 10, (b + 1) = 006146 is entered into Z and becomes the "next" address 6146. At Action 12, order code OC of instruction f is entered into register SQ to initiate the execution of instruction f. (If K refers to a flip-flop register, the STMIC has an effect similar to that shown in figure 2-2.)

2-19. INSTRUCTION XCH K (Exchange Data with Location K, Order Code 3)

2-20. XCH K means: exchange data contained in the accumulator (A) with data stored at location K. If K represents a location in F memory, then XCH transfers data from K to A but data in K remains undisturbed. Therefore, optional code CAF (Clear and Add F) may be used. The entire operation XCH K (for $0020 \geq K$) can be formulated as follows:

(1) Set c(A) = b(K) and c(K) = b(A) if $0024 \leq K \leq 1777$.
If $2000 \leq K$, set c(A) = b(K) only.
If $0020 \leq K \leq 0023$, then c(K) = b^e(A).
Remember that bit 15 (overflow bit) of A gets lost and that bit 16 (sign bit) of A moves into bit position 15 of K. Bit 16 of K moves into positions 16 and 15 of A.

(2) Execute-next the instruction located at z = (L + 1).

The XCH K instruction consists of two subinstructions; XCH0 and STD2.

2-21. Figure 2-4 illustrates the execution of XCH 1063. The quantity (a) stored in A is to be exchanged with the quantity (e) stored at location 1063. During the execution of the previous instruction, the entire code of instruction XCH 1063 (31063) was entered into B (B now contains b = 031063), and order code 03 was entered into register SQ. At Action 1, the address contained in B is transferred to S and the selection logic gates location 1063 for readout and write-in. At Action 2, quantity a is read into P, and at Action 4, parity bit a_o, generated for quantity a, is entered into P2 for temporary storage. At Action 3, register G is cleared. Actions 7, 8, and 9 of subinstruction XCH0 are very similar to the STMIC with the exception that control pulse RP2 is added. Word e is transferred from location 1063 to G at Action 6, and from G into B and P at Action 7. At Action 8, the parity of word e is tested, an alarm caused in case of incorrect parity, and a new parity bit (e_o) for quantity e is generated and entered into G. At Action 9, quantity a and parity bit a_o are entered into G and written into location 1063 at Action 10. Also at Action 10, the quantity e is transferred from B to A, and bits 16 and 15 into the SQG for test of overflow or underflow. Since quantity e was taken from the E memory, normally no overflow or underflow in A is possible at that time. Control pulse ST2 causes the SQG to execute-next subinstruction STD2 in order to increment the program counter (Z) and to initiate the execution of the next instruction. (If K refers to a flip-flop register, Action 7 has an effect similar to that of Action 7 of figure 2-2. At Action 9, quantity a is written into the addressed flip-flop register, but a_o is written into G.)

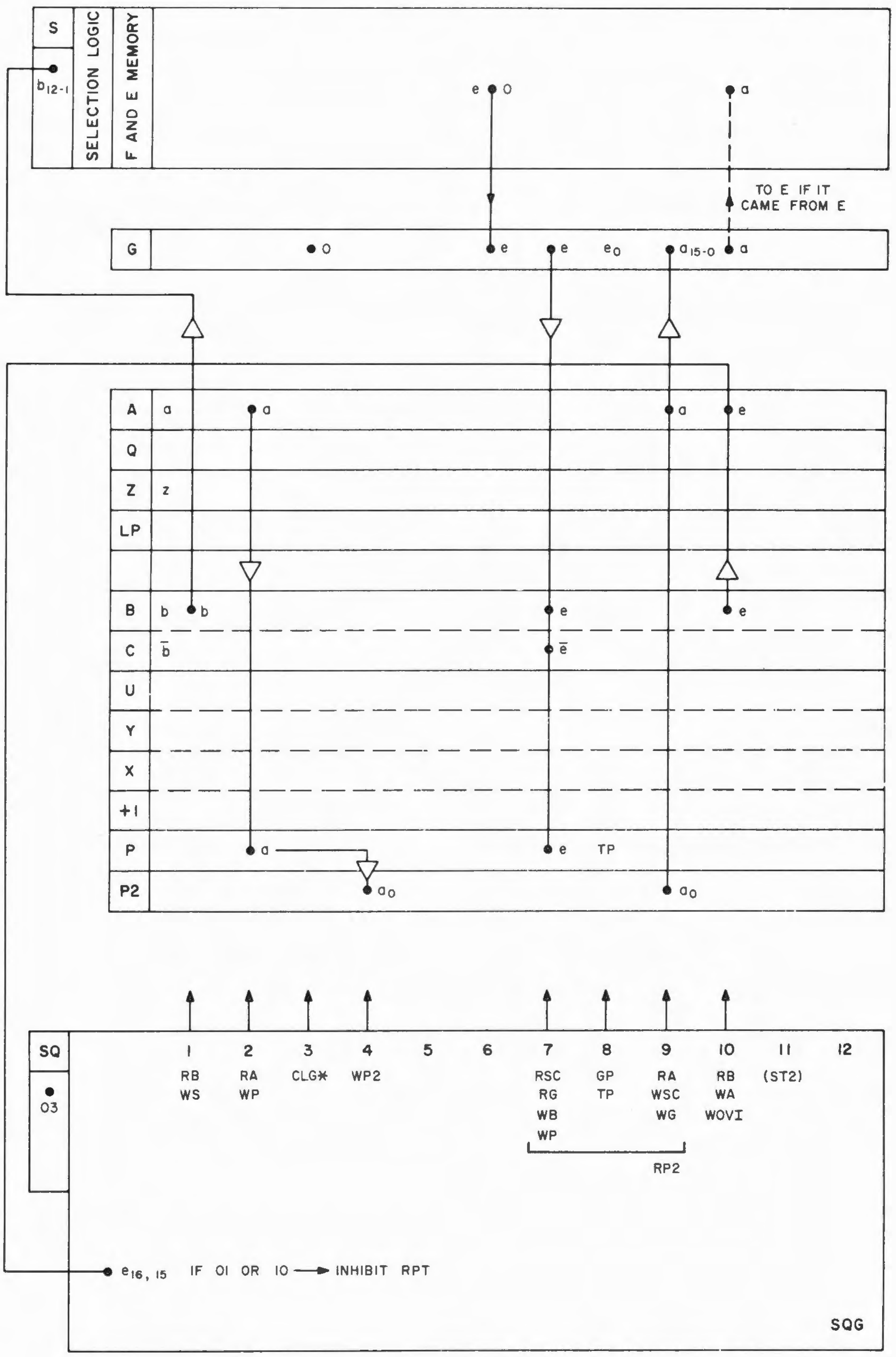

Figure 2-4. Subinstruction XCH0

2-22. INSTRUCTION CS K (Clear A and Subtract Data in K, Order Code 4)

2-23. CS K means: clear the accumulator (A) and enter into it the complemented value of the quantity stored at location K. The entire operation CS K (for $0020 \leq K$) can be formulated as follows:

(1) Set $c(A) = \overline{b}(K)$.

(2) Restore $c(K) = b(K)$ if $0024 \leq K \leq 1777$.
If $0020 \leq K \leq 0023$, then $c(K) = b^e(K)$.

(3) Execute-next the instruction located at $z = (L + 1)$.

The CS K instruction consists of two subinstructions; CS0 and STD2.

2-24. Figure 2-5 illustrates the execution of CS 1021. Quantity e located at 1021 is to be complemented and entered into A. During the execution of the previous instruction, the entire code of instruction CS 1021 (41021) was entered into B (B now contains b = 141021), and order code 14 was entered into register SQ. At Action 1, the address contained in B is transferred to S and the selection logic gates location 1021 for readout and write-in. At Action 3, register G is cleared, as usual. During the STMIC (Actions 7 to 9), data e is entered into B and P, and restored in G as usual. At Action 10, quantity $\overline{e}$ is entered into A by means of control pulses RC and WA. Control pulse WOVI transfers bits 16 and 15 of $\overline{e}$ into the SQG for test of overflow or underflow. Since quantity e was taken from the F or E memory, normally no overflow or underflow in A is possible at that time, therefore, WOVI is of no further consequence. Control pulse ST2 causes the SQG to execute-next subinstruction STD2, in order to increment the program counter (Z) and to initiate the execution of the next instruction. (If K refers to a flip-flop register, the STMIC has an effect similar to that shown in figure 2-2.)

2-25. INSTRUCTION TS K (Transfer Data to K, Order Code 5)

2-26. TS K means: transfer the content of the accumulator (A) to location

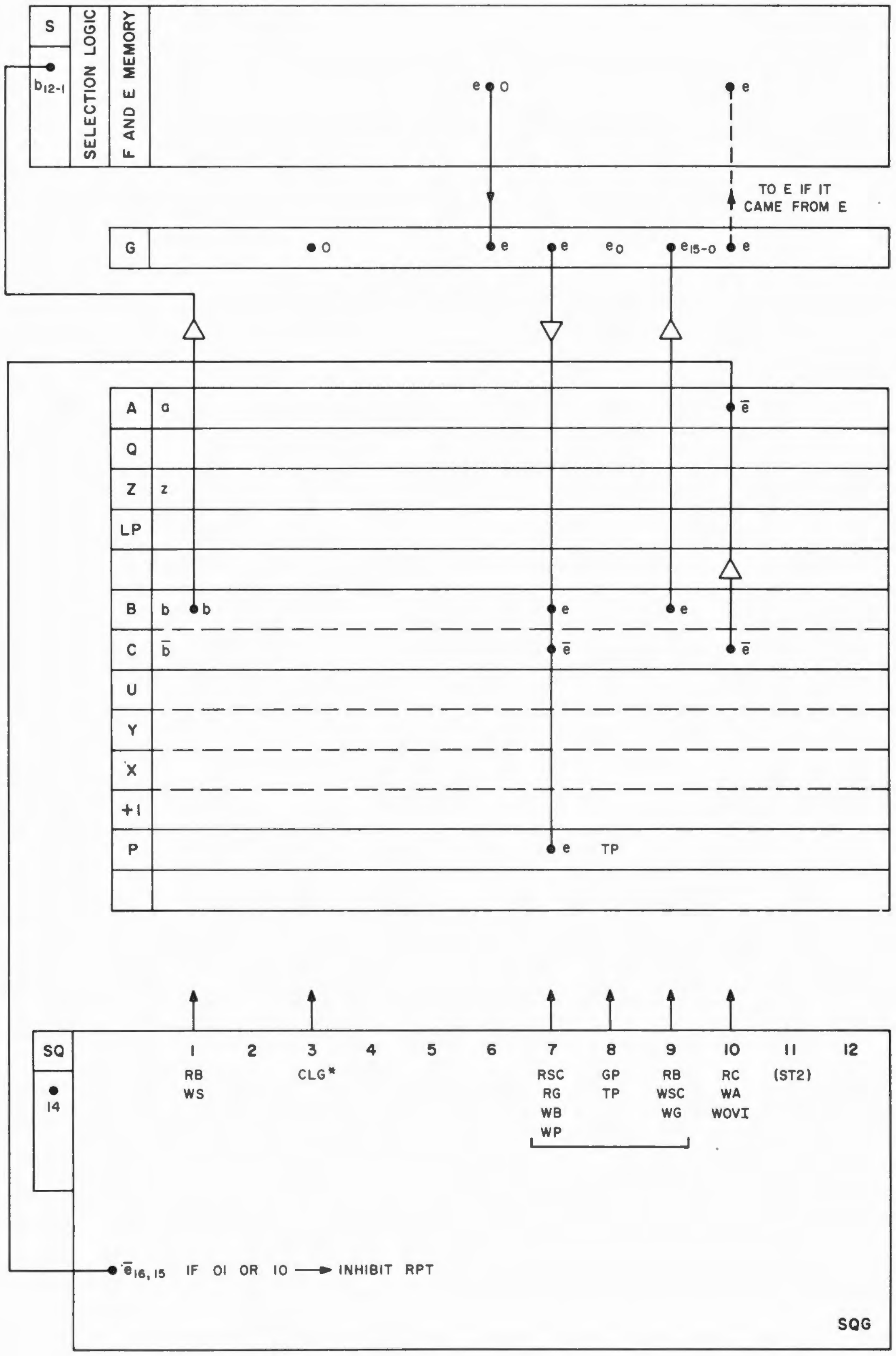

Figure 2-5. Subinstruction CS0

K. If K represents a location in the F memory, then TS K is equivalent to NOOP (No Operation). If A contains no overflow or underflow, execute-next the instruction at z = (L + 1). If A contains overflow or underflow, execute-next the instruction at (L + 2), and enter the value plus-one or minus-one respectively into A. The "skip on overflow" feature is used chiefly in multi-precision operations. The entire operation TS K (for $0020 \leq K$) can be formulated as follows:

(1) Set c(K) = b(A) for $0024 \leq K \leq 1777$.
If $0020 \leq K \leq 0023$, then $c(K) = b^e(A)$.
Remember that bit 15 of b(A) gets lost, and that bit 16 of b(A) moves into position 15 of c(K).

(2a) If $000000 \leq b(A) \leq 037777$, or
if $177777 \geq b(A) \geq 140000$, then
keep c(A) = b(A) and execute-next the instruction located at
z = (L + 1).

(2b) If $040000 \leq b(A) \leq 077777$, then
set c(A) = 000001 and execute-next the instruction located at
(L + 2).
Set c(Z) = b(Z) + 1.

(2c) If $137777 \geq b(A) \geq 100000$, then
set c(A) = 177776 and execute-next the instruction located at
(L + 2).
Set c(Z) = b(Z) + 1.

The TS K instruction consists of two subinstructions; TS0 and STD2.

2-27. Figures 2-6 and 2-7 illustrate the execution of TS 0611; figure 2-6 shows the execution without overflow, and figure 2-7 shows the execution with overflow. The quantity (a) stored in A is to be transferred to location 0611 in the E memory where the quantity e is presently located. During the execution of the previous instruction, the entire code of instruction TS 0611 (50611) was entered into register B (B now contains b = 150611), and order code 15 was

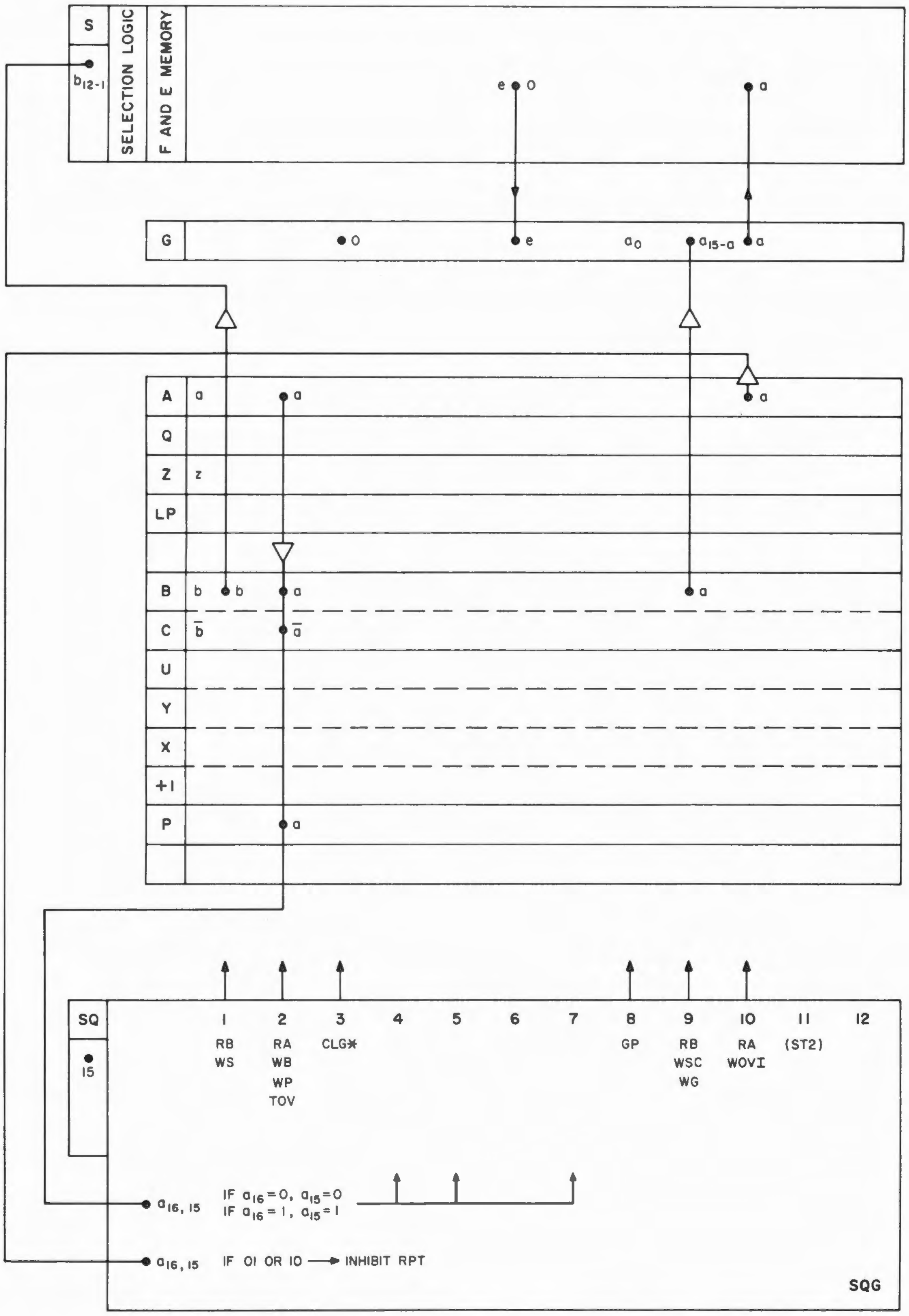

Figure 2-6. Subinstruction TS0 (without Overflow or Underflow in A)

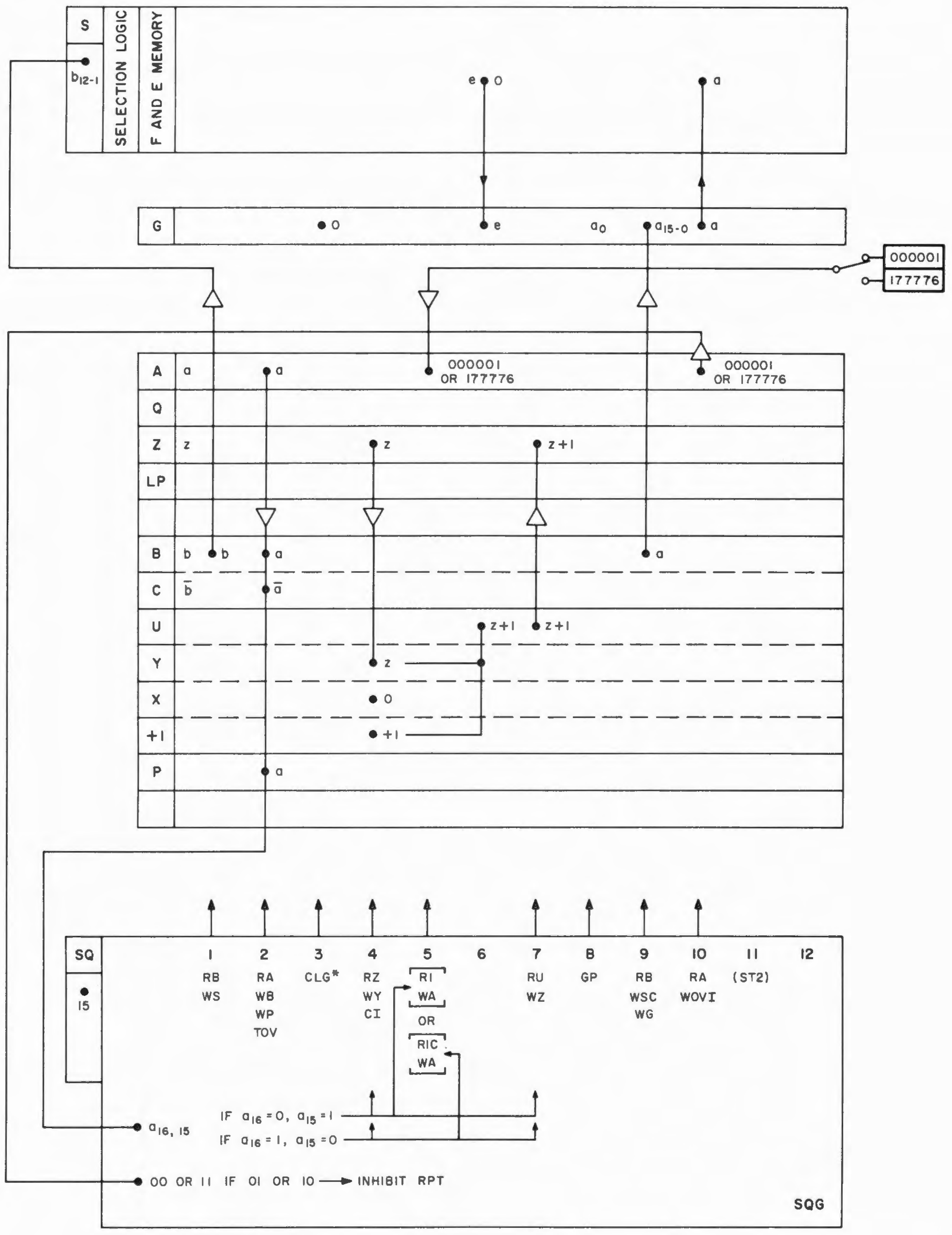

Figure 2-7. Subinstruction TS0 (with Overflow or Underflow in A)

entered into register SQ. At Action 1, the address contained in B is transferred to S and the selection logic gates location 0611 for readout and write-in. At Action 2, the entire content of A is entered into B and P. Bits 16 and 15 are brought into the SQG for test of overflow or underflow by means of control pulse TOV. At Action 3, register G is cleared. If there was not any overflow or underflow detected at Action 2, no control pulses are generated during Actions 4 through 7, as shown in figure 2-6. The parity bit, generated by the parity pyramid for data a, is entered into G at Action 8, and quantity a is entered into G at Action 9. At Action 10, bits 16 and 15 of quantity a are transferred into the SQG for test of overflow or underflow, as usual. Since no overflow or underflow is present in A of figure 2-6, Action 10 is of no further consequence. Control pulse ST2, at Action 11, causes the SQG to execute-next the subinstruction STD2 in order to increment the program counter (Z) and to initiate the execution of the next instruction. (If K refers to a flip-flop register, Action 9 has an effect similar to that of Action 9 of figure 2-2.)

2-28. If overflow or underflow was detected at Action 2, several control pulses are generated at Actions 4 through 7, as shown in figure 2-7. At Action 4, address z is entered into the Adder, incremented by one, and returned to Z at Action 7. In case of overflow in A, the word 000001 is written into A; in case of underflow, the word 177776. At Action 10, bits 16 and 15 of quantity 000001 or 177776 are entered into the SQG for the usual test of overflow or underflow, but Action 10 is again of no further consequence.

2-29. INSTRUCTION MSK K (Mask with Data from K, Order Code 7)

2-30. MSK K means: mask the content of location K with the content of the accumulator (A), i.e. perform an operation which has the same effect as applying the Boolean AND operation for each bit position of locations A and K. This is accomplished by applying $\overline{\overline{c}(A) \text{ OR } \overline{c}(K)}$ instead of c(A) AND c(K) for each bit position. The entire operation MSK K (for $0020 \leq K$) can be formulated as follows:

(1) Set c(A) = b(A) AND c(K).

(2) Restore c(K) = b(K) if $0020 \leq K \leq 1777$.

(3) Execute-next the instruction located at z = (L + 1).

The MSK K instruction consists of two subinstructions; MSK0 and STD2.

2-31. Figure 2-8 illustrates the execution of MSK 0700. The quantity e = 36000, located at K, is to be masked with the quantity a = 025252, contained in A. During the execution of the previous instruction, the entire code of instruction MSK 0700 (70700) was entered into B (B now contains b = 170700), and order code 17 was entered into register SQ. At Action 1, the address contained in B is transferred to S and the selection logic gates location 0700 for readout and write-in. At Action 2, quantity a is transferred to B and $\bar{a}$ is available at C. Register G is cleared at Action 3. At Action 4, the complemented quantity $\bar{a}$ is entered into Y for temporary storage in U. At Actions 7 and 8, quantity e is transferred to B and P, a new parity bit for quantity e is entered into G, and an alarm caused in case of incorrect parity. Furthermore, quantities $\bar{a}$ and $\bar{e}$ are read simultaneously and written into A, resulting in ($\bar{a}$ OR $\bar{e}$). At Action 10, the quantity ($\bar{a}$ OR $\bar{e}$) is complemented once more and becomes (a AND e) which is returned to A at Action 11. Control pulse WOVI inhibits program interruption at the end of the current instruction in case of overflow or underflow in A, as usual. Control pulse ST2 causes the SQG to execute-next subinstruction STD2 in order to increment the program counter (Z) and to initiate the execution of the next instruction. (If K refers to a flip-flop register, Action 7 of MSK0 has an effect similar to that of Action 7 of figure 2-2.)

2-32. By omitting the control pulses usually generated at Action 9, Instruction MSK does not change the content of register G after Action 7. Most other instructions change the content of G, if an editing operation (shift or cycle) is performed.

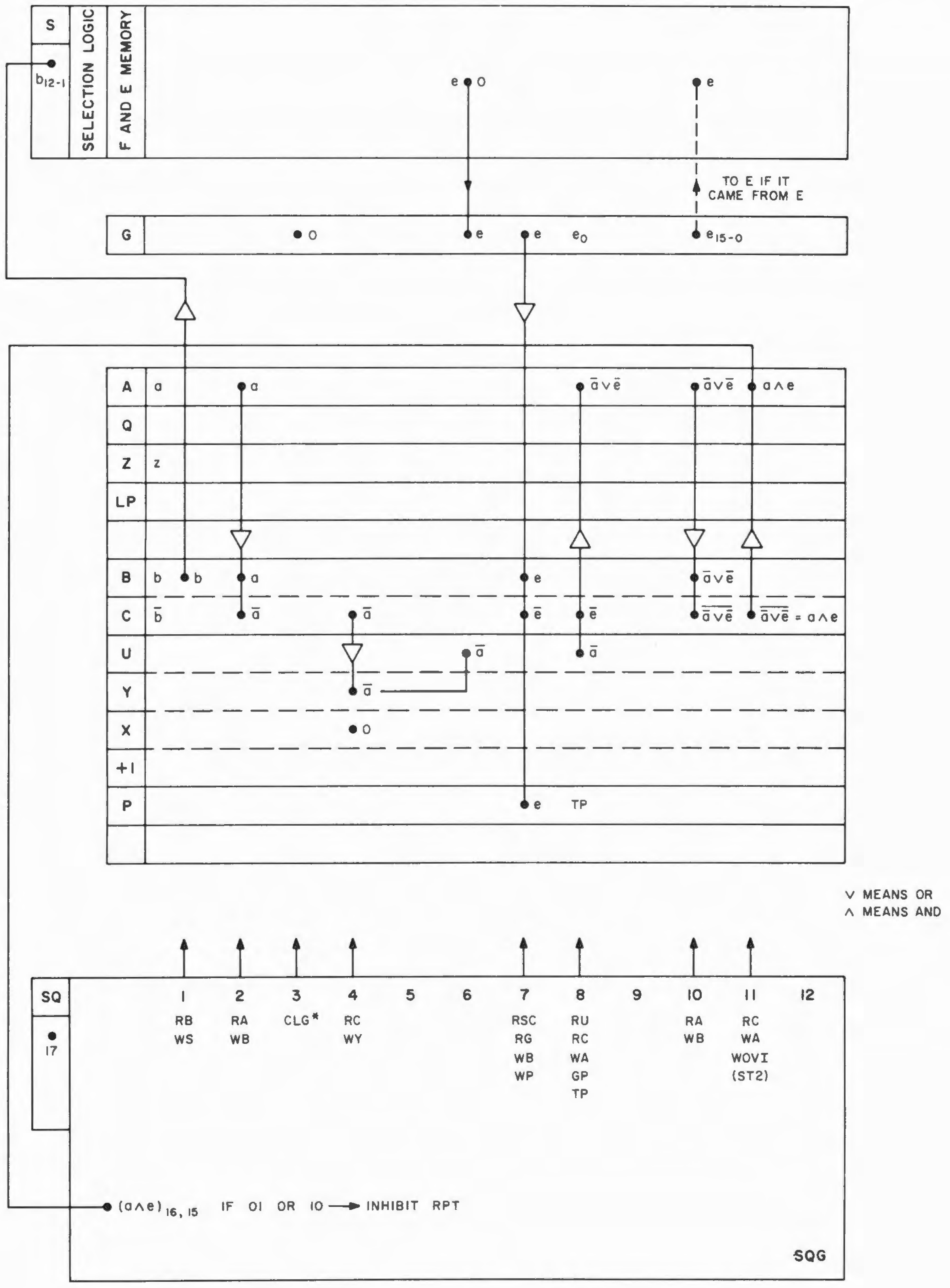

Figure 2-8. Subinstruction MSK0

2-33. INSTRUCTION AD K (Add Data from K and Count on Overflow or Underflow, Order Code 6)

2-34. AD K means: add the quantity located at K to the quantity contained in the accumulator (A). In case of overflow or underflow, increment or decrement the overflow counter (OVCTR). The method of addition applied is discussed in paragraph 15-30 through 15-33. The entire operation AD K (for $0020 \leq K$) can be formulated as follows:

(1) Set c(A) = b(A) + c(K).

(2a) In case of overflow, set c(OVCTR) = b(OVCTR) + 1 by executing PINC.

(2b) In case of underflow, set c(OVCTR) = b(OVCTR) - 1 by executing MINC.

(3) Restore c(K) = b(K) if $0024 \leq K \leq 1777$.
If $0020 \leq K \leq 0023$, then $c(K) = b^e(K)$.

(4) Execute-next the instruction located at z = (L + 1).

The AD K instruction consists of two subinstructions; AD0 and STD2. In case of overflow or underflow, it takes 3 MCTs to perform an addition.

2-35. Figure 2-9 illustrates the execution of AD 1043. Quantity e located at 1043 is to be added to the quantity (a) in A. During the execution of the previous instruction, the entire code of instruction AD 1043 (61043) was entered into B (B now contains b = 161043), and order code 16 was entered into register SQ. At Action 1, the address contained in B is transferred to S and the selection logic gates location 1043 for readout and write-in. At Action 2, quantity a is entered into Y. At Action 3, register G is cleared, as usual. The STMIC (included in Actions 7 to 9) deals with data a. The additional pulses, RB and WX, at Actions 7 to 9 transfer quantity e into X, and the arithmetic sum (a + e) is available at U within 3 μsec. At Action 11, control pulses RU and WA transfer the sum to A. Pulse WOVI enters bits 16 and 15 of (a + e) into the SQG for test of overflow or underflow. In case of overflow or underflow, a program interruption at the end of the current instruction is inhibited.

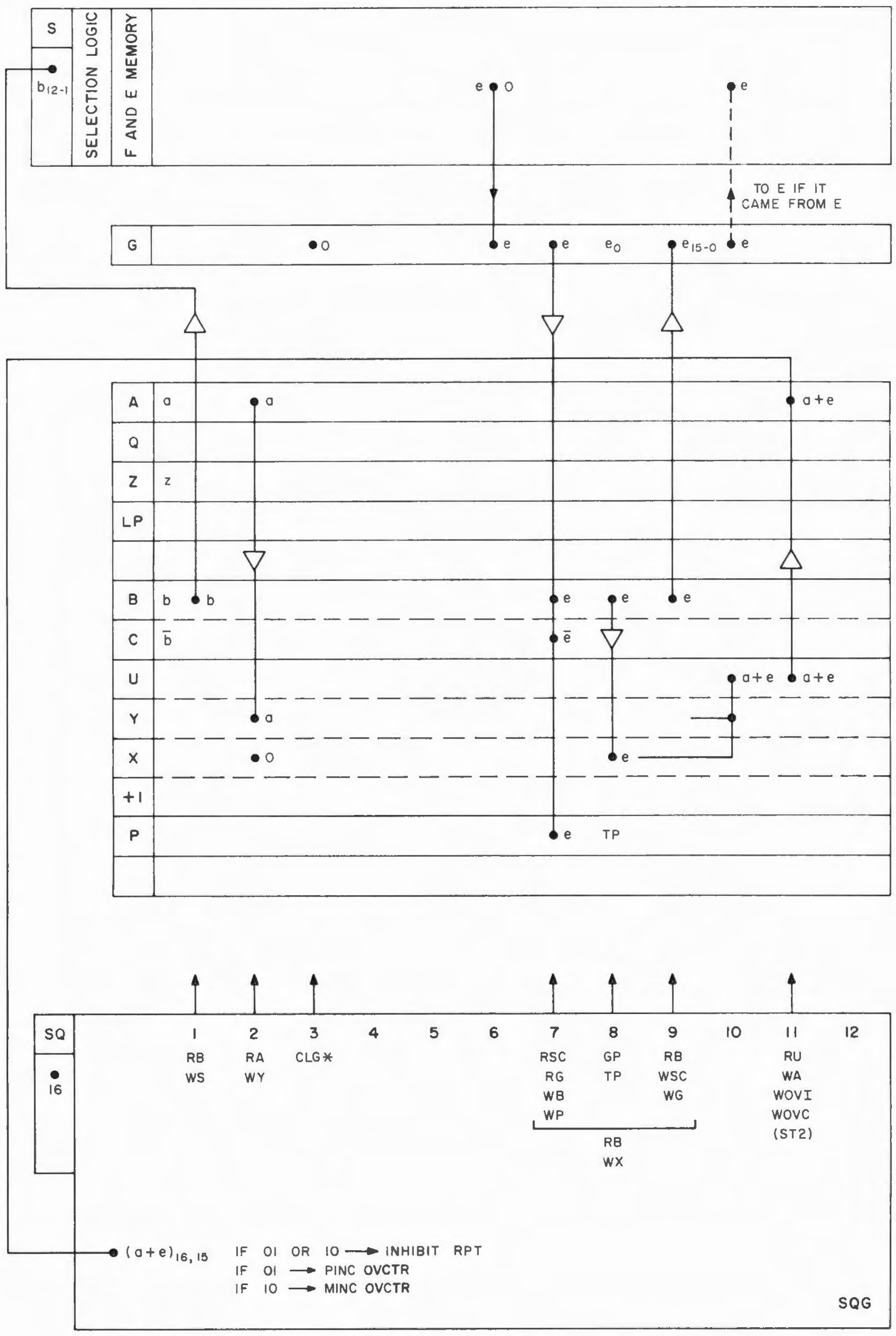

Figure 2-9. Subinstruction AD0

If overflow occurs, control pulse WOVC is sent to the Counter Priority Control. About 10 μ sec later the Counter Priority Control causes the SQG to initiate instruction PINC, in order to increment the OVCTR by one. Incrementing the OVCTR is executed at the end of subinstruction STD2. If underflow occurs, WOVC causes decrementing (MINC) the OVCTR by one. Control pulse ST2 causes the SQG to execute next subinstruction STD2 in order to increment the program counter (Z) and to initiate the execution of the next instruction. (If K refers to a flip-flop register, the STMIC has an effect similar to that shown in figure 2-2.)

2-36. INSTRUCTION NDX K (Index Next Instruction, Order Code 2)

2-37. NDX K means: use as the next instruction the arithmetic sum of the instruction located at the next address z = (L + 1) plus the quantity located at K. K may be any address except 0025. In case K = 0025, this is instruction RESUME. When no overflow or underflow of the four-bit order code occurs during addition, the new instruction remains a Basic Instruction. When underflow occurs, the new instruction is an Extra Code Instruction. The entire operation NDX K (for $0020 \leq K$, except K = 0025) can be formulated as follows:

(1) Set c(B) = c(z) + c(K), z = (L + 1).

(2) Restore c(K) = b(K), if $0024 \leq K \leq 1777$.
If $0020 \leq K \leq 0023$, then $c(K) = b^{e}(K)$.

(3) After the execution of the instruction put into B, execute next the instruction located at (L + 2), L being the location of the NDX instruction. If the instructions at (L + 1) and K are TC instructions (and if no overflow or underflow occurs during addition), the instruction to be executed next is not located at (L + 2) but at address c(L + 1) + c(K).

The NDX K instruction consists of two subinstructions: NDX0 and NDX1.

2-38. Figures 2-10 and 2-11 illustrate the execution of NDX 1174, this instruction being located at address 6534. Instruction AD 2103 is located at address z = 6535. Location K = 1174 contains the quantity e = 00011. The

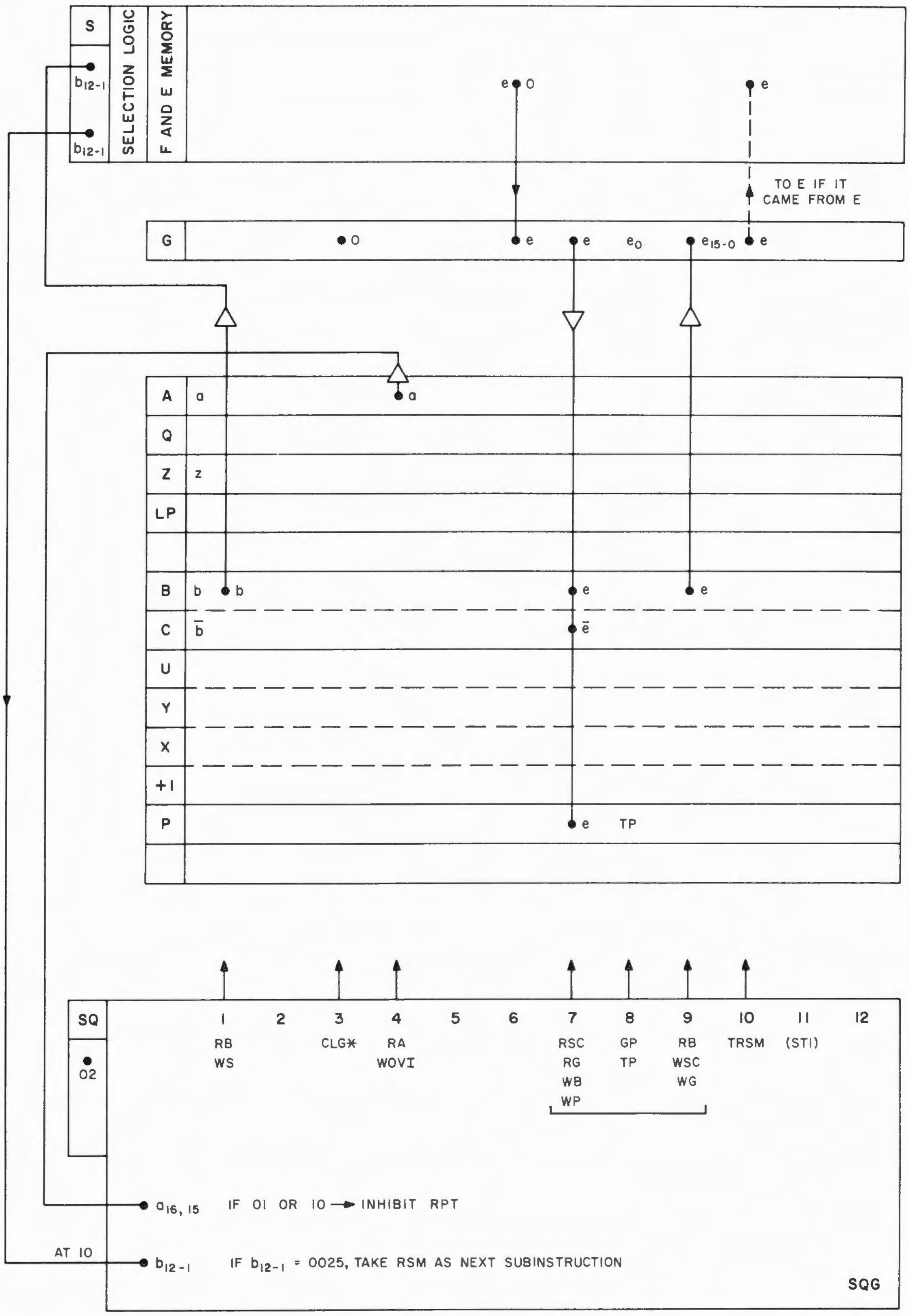

Figure 2-10. Subinstruction NDX0

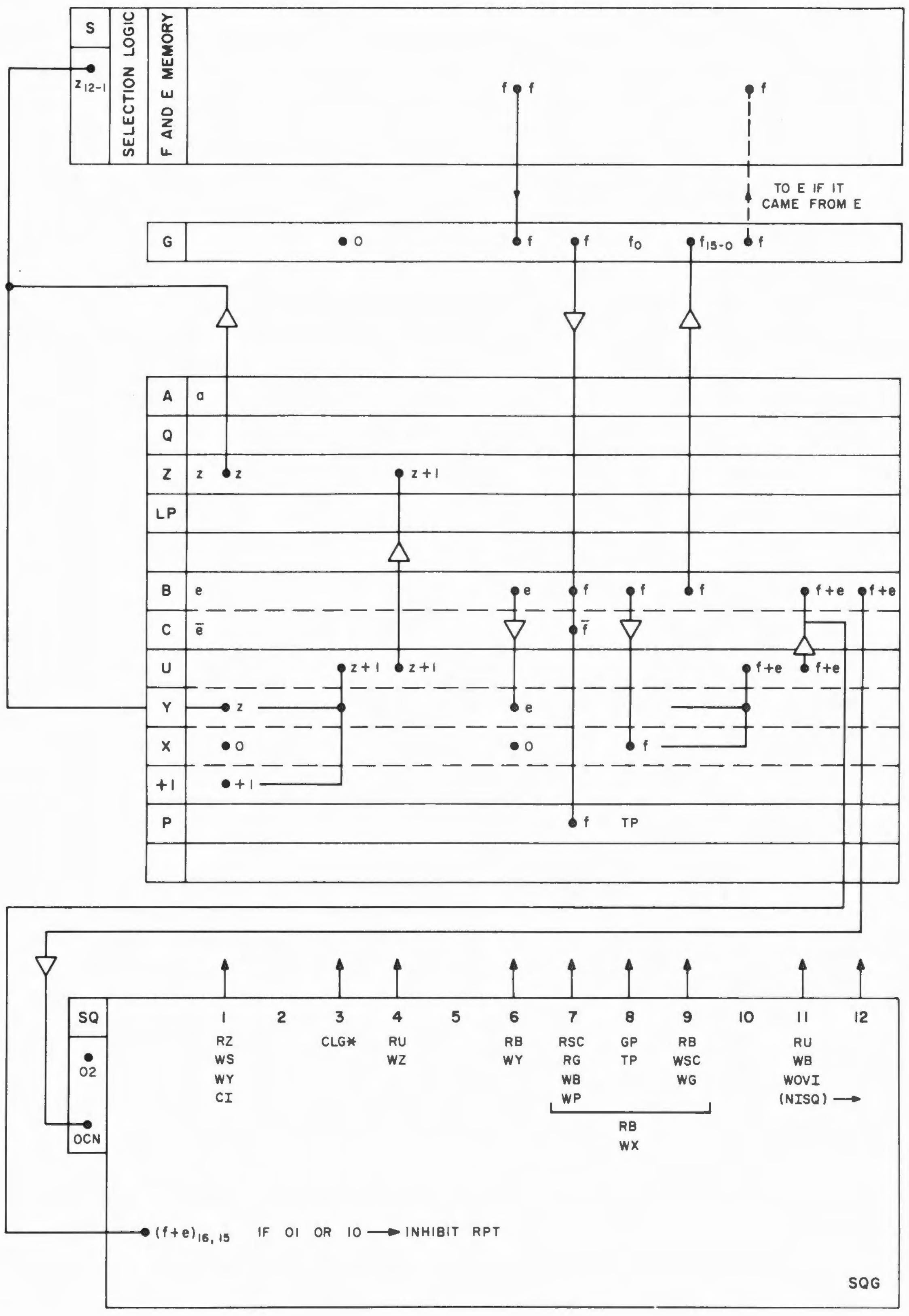

Figure 2-11. Subinstruction NDXI

instruction AD 2103 is to be indexed to AD 2114. During the execution of the previous instruction, the entire code of instruction NDX 1174 (21174) was entered into B (B now contains b = 021174), and order code 02 was entered into register SQ. At Action 1 of NDX0, the address contained in B is transferred to S and the selection logic gates location 1174 for readout and write-in. Register G is cleared at Action 3. At Action 4, a program interruption at the end of the current instruction is inhibited if there is any overflow or underflow in the accumulator. The STMIC (Actions 7 to 9) brings data e into B and P, generates a new parity bit (e_o), causes an alarm in case of incorrect parity, and restores word e in G. At Action 10, pulse TRSM tests whether the address contained in S is or is not 0025. Control pulse ST1, generated at Action 11, causes the SQG to execute-next subinstruction NDX1, if the address is not 0025, or subinstruction RSM, if the address is 0025. Subinstruction RSM will be discussed with the Involuntary Instructions. (If K refers to a flip-flop register, the STMIC has an effect similar to that shown in figure 2-2.)

2-39. At the start of NDX1, register Z still contains z but register B contains data e. The order code 02 is still stored in the SQ register. At Action 1, the address contained in Z is transferred to S and Y. The selection logic gates a location with address 6535 for readout, and the Adder increments z by one. The content of register BNK determines which location with address 6535 is selected. At Action 3, register G is cleared, and (z + 1) is transferred to Z at Action 4. At Action 6, quantity e is entered into Y from B. Actions 7, 8, and 9 of NDX1 are very similar to the STMIC with the exception that control pulses RB and WX are added. The instruction f = AD 2103 is entered into B (b becomes 162103) and P, an alarm caused in case of incorrect parity, a new parity bit for word f is generated, and word f is returned to G. At Action 8, instruction f is entered into X, and the new instruction (f + e) = 162114 = AD 2114 is available at U within 3 μsec. At Action 11, the new instruction is entered into B, and its order code OCN is brought into SQ at Action 12 to initiate the execution of the new instruction. Control pulse WOVI inhibits program interruption if there is any overflow or underflow in the sum (f + e). For Extra Code Instructions, (f + e) is the quantity with underflow.

2-40. INSTRUCTION CCS K (Count, Compare, and Skip with Data at K, Order Code 1)

2-41. CCS K means: examine the data located at K. If $c(K) > +0$, take as the subsequent instruction the one located at $z = (L + 1)$. If $c(K) = +0$, take the subsequent instruction from $(L + 2)$. If $c(K) < -0$, take from $(L + 3)$. If $c(K) = -0$, take from $(L + 4)$. K may be any address, except an address referring to F Memory. The entire operation CCS K (for $0020 \leq K < 2000$) can be formulated as follows:

(1a) If $00000 < c(K) < 37777$, set $c(A) = c(K) - 00001$ and execute-next the instruction located at $z = (L + 1)$.

(1b) If $c(K) = 00000$, set $c(A) = 000000$ and execute-next the instruction located at $(L + 2)$.

(1c) If $40000 < c(K) < 77777$, set $c(A) = \bar{c}(K) - 00001 = |c(K)| - 00001$ and execute-next the instruction located at $(L + 3)$.

(1d) If $c(K) = 77777$, set $c(A) = 000000$ and execute-next the instruction located at $(L + 4)$.

(2) Restore $c(K) = b(K)$ if $0024 \leq K \leq 1777$.
If $0020 \leq K \leq 0023$, then $c(K) = b^e(K)$.

The CCS K instruction consists of two subinstructions; CCS0 and CCS1.

2-42. Figures 2-12 through 2-16 and table 2-3 illustrate the execution of CCS 1051 being located at 6040. Content e of location 1051 is to be examined, and dependent on the result, the instruction to be executed next is to be taken from location 6041, 6042, 6043, or 6044. During the execution of the previous instruction, the entire code of instruction CCS 1051 (11051) was entered into B (B now contains b = 011051), and order code 01 was entered into register SQ. At Action 1 of CCS0, the address contained in B is transferred to S and

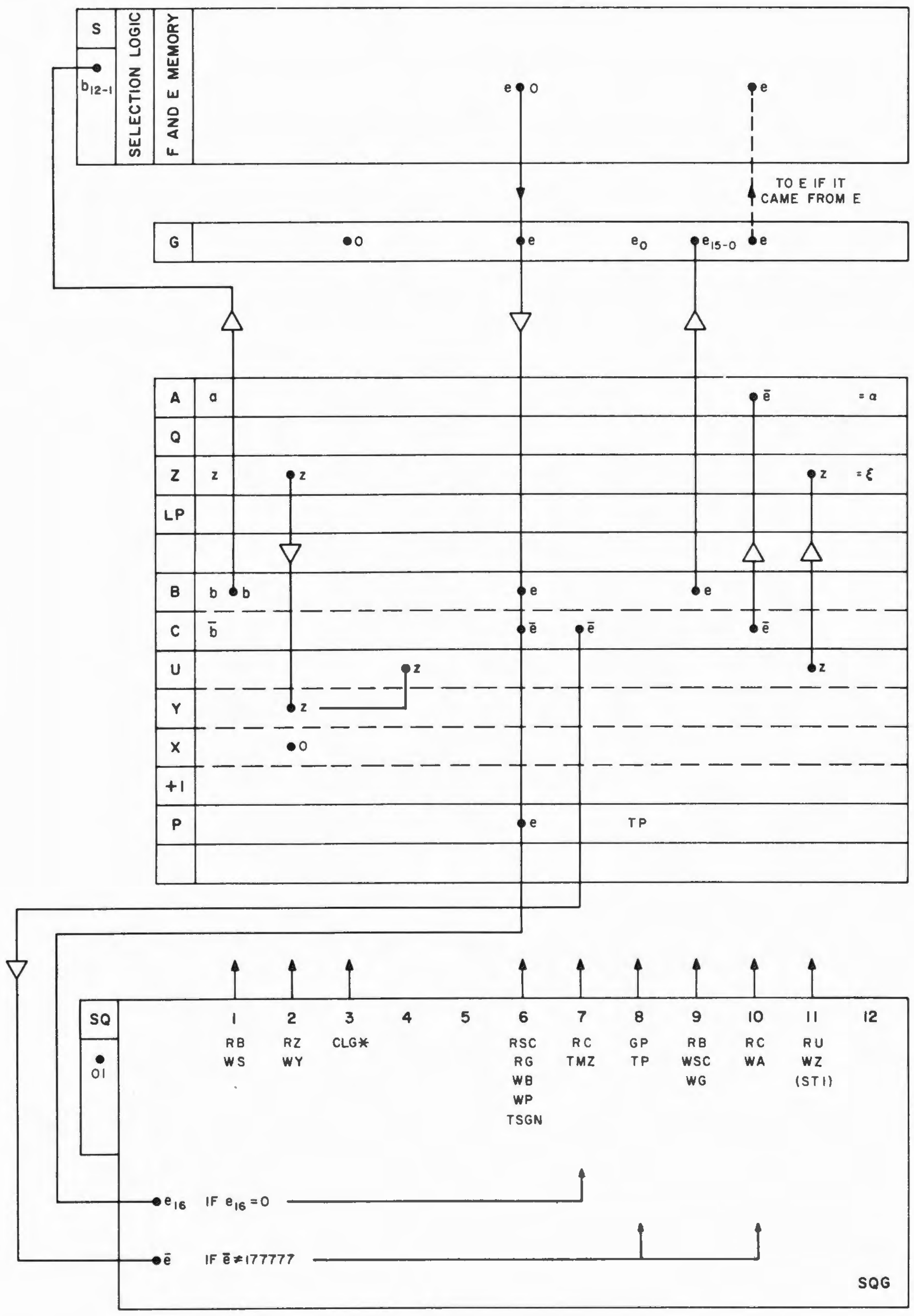

Figure 2-12. Subinstruction CCS0 (Example e > +0)

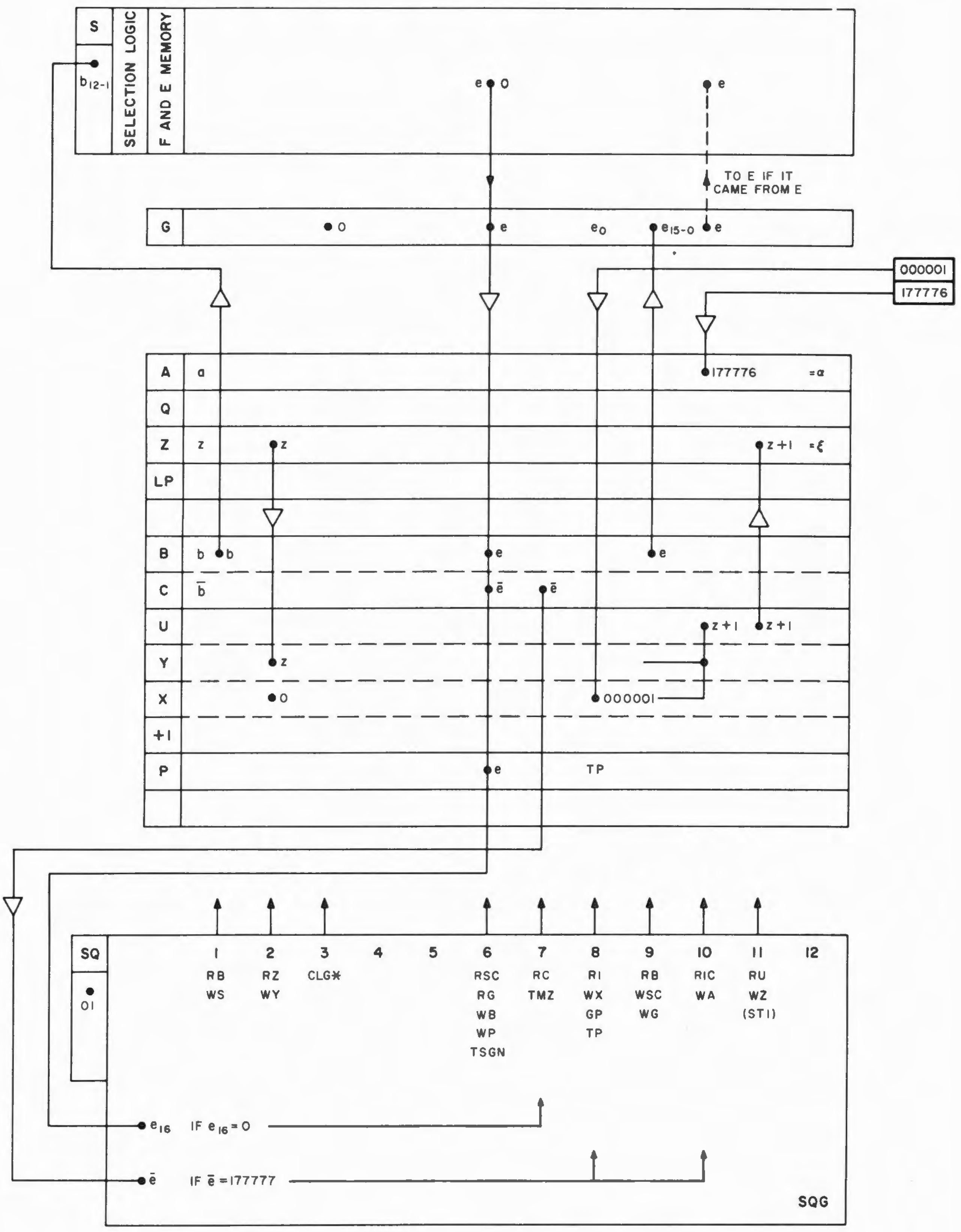

Figure 2-13. Subinstruction CCS0 (Example e = +0)

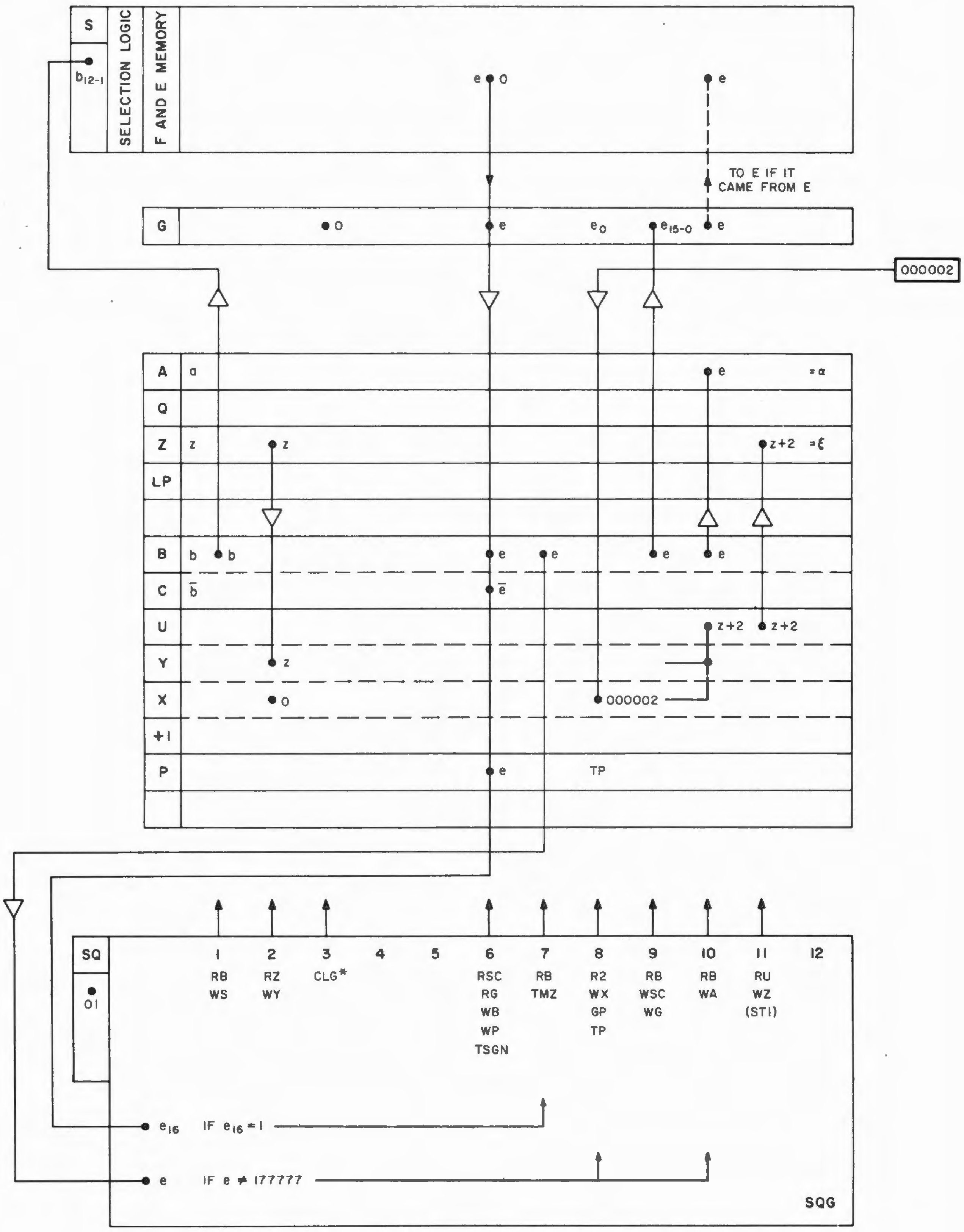

Figure 2-14. Subinstruction CCS0 (Example e < -0)

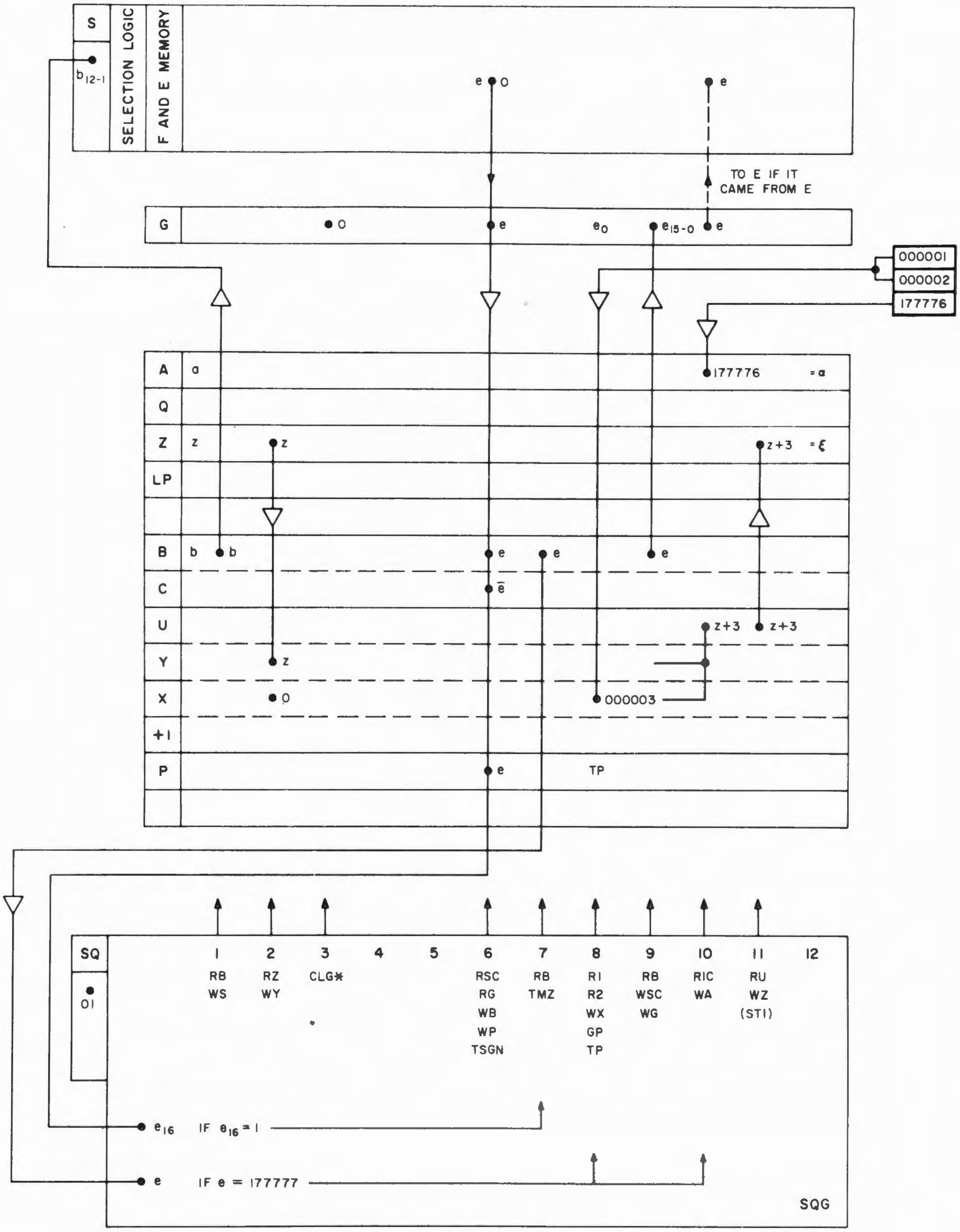

Figure 2-15. Subinstruction CCS0 (Example e = -0)

TABLE 2-3

CONTENT OF A AND Z AT END OF CCS0 AND CCS1

INITIAL CONDITION c(K) = e	α (CONTENT OF A AT END OF CCS0)	ζ (CONTENT OF Z AT END OF CCS0)	$\zeta + 1$ (CONTENT OF Z AT END OF CCS1)	$\overline{\alpha + 1}$ (CONTENT OF A AT END OF CCS1)
$e > +0$	$\bar{e}$	z	$z + 1$	$e - 1$
$e = +0$	177776	$z + 1$	$z + 2$	$+0$
$e < -0$	e	$z + 2$	$z + 3$	$\bar{e} - 1$
$e = -0$	177776	$z + 3$	$z + 4$	$+0$

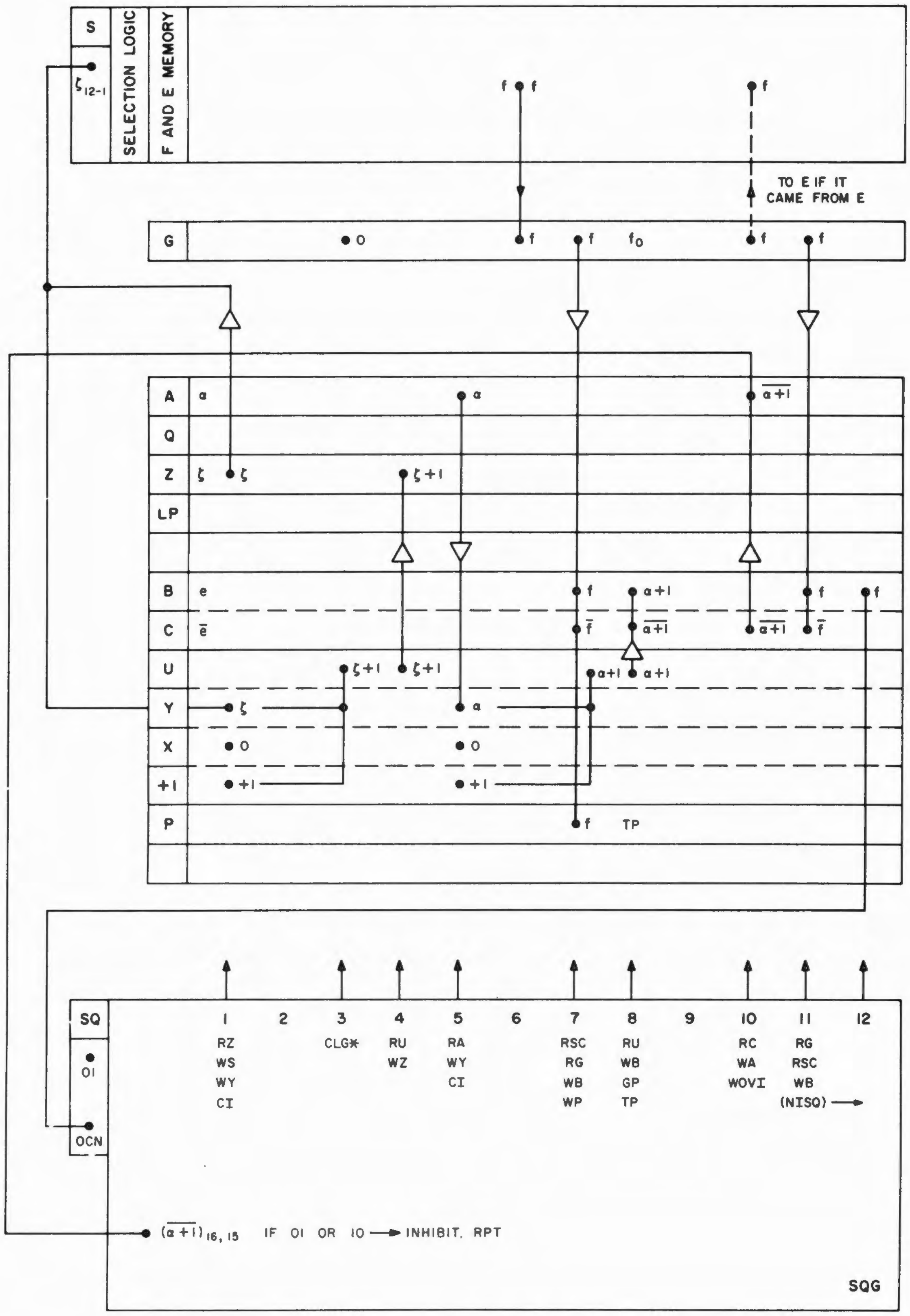

Figure 2-16. Subinstruction CCS1

the selection logic gates location 1051 for readout and write-in. At Action 2, the "next" address (z) is entered into Y and kept in U, and G is cleared at Action 3. At Action 6, control pulses RG, WB, and WP transfer the content of G to B and P. Remember that P now contains exactly the same word as G. The word in B contains the sign bit of G in bit positions 16 and 15, but does not contain the parity bit. Control pulse RSC has no effect unless a flip-flop register has been gated for readout. Control pulse TSGN enters the sign bit of e into the SQG. If $e_{16} = 0$, control pulses RC and TMZ are generated at Action 7 as shown in figures 2-12 and 2-13, transferring the entire word $\overline{e}$ into the SQG. If $\overline{e} \neq 177777$ ($\overline{e}$ is not minus zero, i.e., e not zero), then Actions 8 and 10 are generated as shown in figure 2-12. At Action 8, an alarm is caused in case of incorrect parity, and a new parity bit for word e is entered into G. At Action 9, word e is brought back to G, as usual. At Action 10, $\overline{e}$ is transferred to A, and z is returned to Z. Control pulse ST1 causes the SQG to execute-next subinstruction CCS1. (If K refers to a flip-flop register, Actions 6 and 9 have an effect similar to that of Actions 7 and 9 of figure 2-2.)

2-43. If (at Action 7) $\overline{e} = 177777$ ($\overline{e}$ is minus zero, i.e., e is zero), then Actions 8 and 10 are generated as shown in figure 2-13. The additional pulses R1 and WX write the value plus-one into X. The value plus-one is added to z and the sum appears at U 3 μsec later. At Action 10, the quantity minus-one is written into A. At Action 11, the quantity (z + 1) is transferred from U to Z, and subinstruction CCS1 follows after Action 12.

2-44. Returning to Action 6, if $e_{16} = 1$, control pulses RB and TMZ are generated at Action 7, as shown in figures 2-14 and 2-15, transferring the word e into the SQG. If $e \neq 177777$ (e is not minus zero), then Actions 8 and 10 are generated as shown in figure 2-14. The additional pulses R2 and WX write the value plus-two into X. The value plus-two is added to z and the sum appears at U 3 μsec later. At Action 10, word e is transferred to A. At

Action 11, the quantity (z + 2) is transferred from U to Z and subinstruction CCS1 follows after Action 12.

2-45. If (at Action 7) e = 177777 (e is minus zero), then Actions 8 and 10 are generated as shown in figure 2-15. Control pulses R1, R2, and WX write the value plus-three into X. The value plus-three is added to z and the sum appears at U 3 μsec later. At Action 10, the quantity minus-one is written into A. At Action 11, the quantity (z + 3) is transferred from U to Z and subinstruction CCS1 follows after Action 12.

2-46. Table 2-3 lists the various contents of A and Z at the end of subinstruction CCS0 as derived from figures 2-12 through 2-15. At the beginning of subinstruction CCS1 (figure 2-16) the content of A is called α, and the content of Z is called z. At Action 1 of CCS1, the "next" address (ζ) is transferred to S and the selection logic gates the location of the instruction to be executed next (f) for readout and write-in. The "next" address (ζ) is also entered into Y, incremented by one, and returned to Z. Table 2-3 also lists the final contents of A and Z. Register G is cleared at Action 3, as usual. At Action 5 the content of A is incremented by one and available at U within 3 μsec. At Action 7, next instruction f is entered into B and P, as usual. At Action 8, an alarm is caused in case of incorrect parity, and a new parity bit of f is entered into G, as usual. In addition, the quantity (α + 1) is entered into B, and $\overline{(\alpha + 1)}$ becomes available at C. At Action 10, quantity $\overline{(\alpha + 1)}$ is written into A. In case of overflow or underflow, a program interruption at the end of the current instruction is inhibited. At Action 11, next instruction f is entered finally into B and its order code (OCN) is transferred to register SQ, causing the SQG to execute-next instruction f.

2-47. EXTRA CODE INSTRUCTIONS

2-48. COMMON CHARACTERISTICS

2-49. Extra Code Instructions are very similar to Basic Instructions but cannot be written explicitly into a program. Extra Code Instructions are derived by indexing (modifying) Basic Instructions. If no modification of the relevant address, but only modification of the order code, is desired this is done by executing instruction NDX 5777 (mnemonic code EXTEND). The location 5777 contains the quantity 47777. For instance, if NDX 5777 is followed by AD 1043, then the derived Extra Code Instruction is 161043 + 147777 = 131043. The new order code to be entered into SQ is 13, as shown in table 1-1. Note that, at the end of subinstruction NDX1, bit position 16 of B contains a ONE, and bit position 15 a ZERO, indicating an underflow in B. As stated in paragraph 2-39, interrupt at the end of NDX1 is inhibited in case of underflow.

2-50. INSTRUCTION SU K (Subtract Data from K and Count on Overflow or Underflow, Order Code 6 with EXTEND Preceding)

2-51. SU K means: subtract the quantity located at K from the quantity contained in the accumulator (A). In case of overflow or underflow, increment or decrement the overflow counter (OVCTR). Instruction SU K is very similar to AD K, except that the subtraction is established by adding the complemented value located at K, rather than the original value, to the quantity in A. The entire operation SU K (for $0020 \leq K$) can be formulated as follows:

(1) Set $c(A) = b(A) + \overline{c}(K)$.

(2a) In case of overflow, set $c(OVCTR) = b(OVCTR) + 1$ by executing PINC.

(2b) In case of underflow, set $c(OVCTR) = b(OVCTR) - 1$ by executing MINC.

(3) Restore $c(K) = b(K)$ if $0024 \leq K \leq 1777$.

If $0020 \leq K \leq 0023$, then $c(K) = b^e(K)$.

(4) Execute-next the instruction located at $z = (L + 1)$.

The SU K instruction consists of two subinstructions; SU0 and STD2. Since an NDX and an SU instruction have to be executed to perform a subtraction, the time needed for a subtraction is four or five MCTs, as indicated in table 1-1.

2-52. Figure 2-17 illustrates the execution of SU 1043. Quantity e, located at 1031, is to be subtracted from the quantity (a) in A. The entire code of the instruction was entered into B during the execution of the preceding NDX instruction (B now contains b = 131043, as derived in paragraph 2-49), and order code 13 was entered into register SQ. Figure 2-17 is very similar to figure 2-9 except that quantity $\overline{e}$ is taken from C instead of e from B.

2-53. INSTRUCTION MP K (Multiply with Data at K, Order Code 4 with EXTEND Preceding)

2-54. MP K means: multiply the content of the accumulator (A) by the quantity located at K. The entire operation MP K (for $0020 \leq K$) can be formulated as follows:

(1) Set $c(A, LP) = b(A) \cdot c(K)$ where b(A) does not contain an overflow or underflow (OV) bit. Register A holds the high order product, LP the low order product. Fourteen value bits are stored in bit positions 14 through 1 of A, and fourteen in bit positions 14 through 1 of LP. The sign bits stored in bit positions 16 and 15 are identical for both registers.

(2) Restore $c(K) = b(K)$ if $0020 \leq K \leq 1777$.

(3) Execute-next the instruction located at $z = (L + 1)$.

Performing a multiplication requires the execution of Instruction NDX, subinstruction MP0 once, MP1 six times, and MP3 once.

2-55. Before discussing the execution of instruction MP K, the principle of multiplication applied should be explained. Figure 2-18 demonstrates the

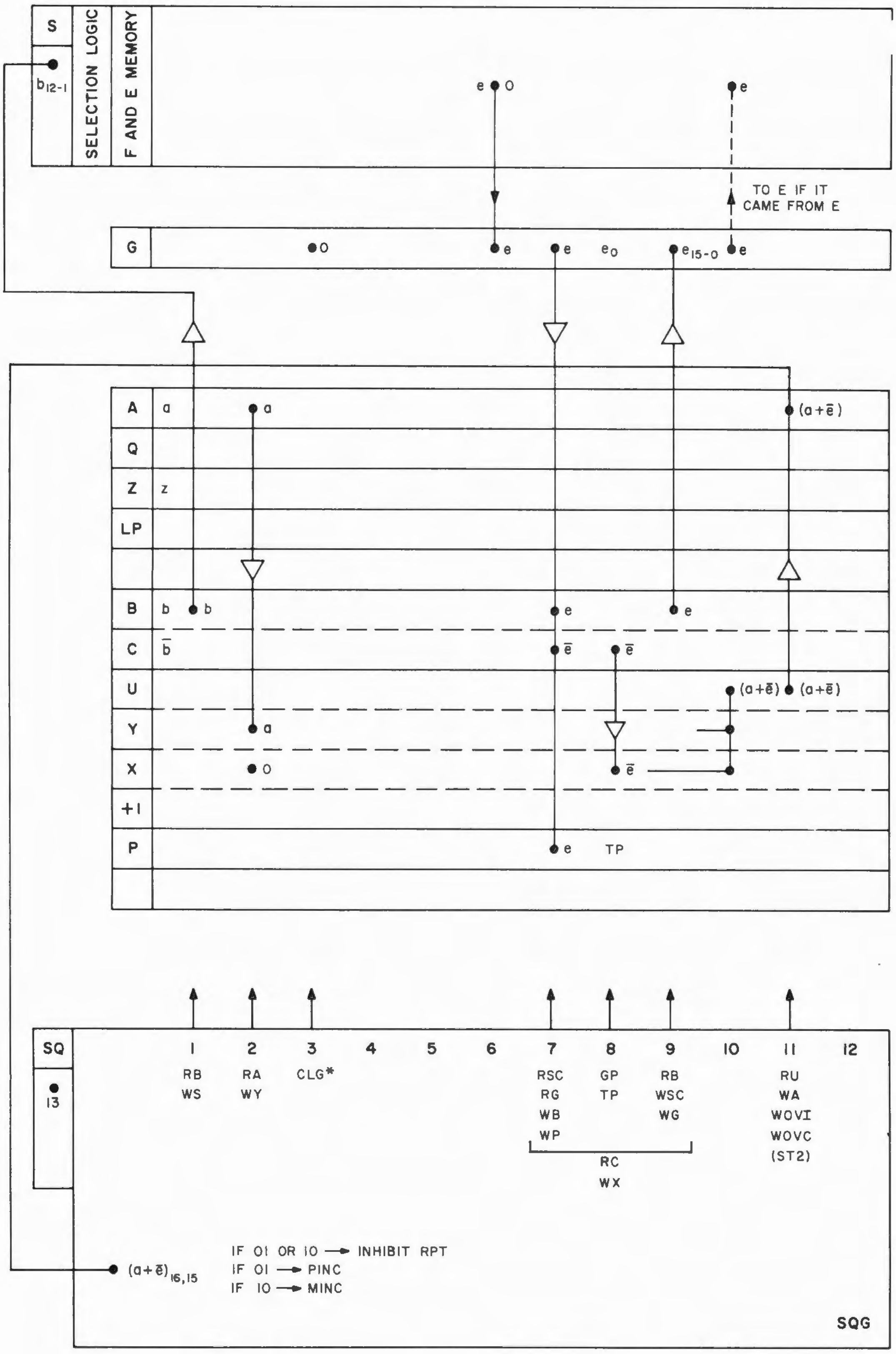

Figure 2-17. Subinstruction SU0

MANUAL OPERATION

$a = +1011$

$e = +1110$

$$e \cdot a = \underline{1110 \cdot 1011}$$

$$\begin{array}{r} 1110 \\ 1110 \\ 0000 \\ \underline{1110} \\ 10011010 \end{array}$$

MACHINE OPERATION

INSTANT	FUNCTION	REGISTERS Y,X,U,A						REGISTER LP				
		6	5	4	3	2	1	4	3	2	1	
4	$e \rightarrow Y$	0	0	1	1	1	0	X	X	X	X	
5	$o \rightarrow X$	0	0	0	0	0	0					
6	U	0	0	1	1	1	0					
8	$U \rightarrow A, LP_4$	0	0	0	1	1	1	0	X	X	X	1st SUBTOTAL
9	$A \rightarrow Y$	0	0	0	1	1	1					
10	$e \rightarrow X$	0	0	1	1	1	0					
11	U	0	1	0	1	0	1					
13,14	$U \rightarrow A, LP_4; \overrightarrow{LP_3}$	0	0	1	0	1	0	1	0	X	X	2nd SUBTOTAL
15	$A \rightarrow Y$	0	0	1	0	1	0					
16	$o \rightarrow X$	0	0	0	0	0	0					
17	U	0	0	1	0	1	0					
19,20	$U \rightarrow A, LP_4; \overrightarrow{LP_{3,2}}$	0	0	0	1	0	1	0	1	0	X	3rd SUBTOTAL
21	$A \rightarrow Y$	0	0	0	1	0	1					
22	$e \rightarrow X$	0	0	1	1	1	0					
23	U	0	1	0	0	1	1					
25,26	$U \rightarrow A, LP_4; \overrightarrow{LP_{3,2,1}}$	0	0	1	0	0	1	1	0	1	0	PRODUCT

Figure 2-18. Multiplication of Two Binary Numbers, Principle of Operation

multiplication of binary number e = +1110 with a = +1011, which is the same as multiplying a with e. First it is shown how the multiplication can be carried out manually, and then by a computer similar to the AGC. The procedure is basically the same for both approaches but with two differences. With manual operation, the quantity e is shifted left each time it is multiplied by a digit of a, and the partial products are then all added at once. With the machine, only two numbers may be added at a time, therefore it is necessary to compute subtotals of the partial products. In the example, registers Y, X, U, A, and LP each consist of six bit positions. At instant 4, quantity e is entered into register Y (the four value bits into positions 4 through 1, the sign into positions 5 and 6) because the lowest bit of multiplier a is a ONE. The quantity zero is fed into X at instant 5. At instant 6, the content of U (the sum of Y and X) is equal to e. At instant 8, the content of U is transferred to A and shifted one place to the right. Bit 1 of U is moved into position 4 of LP at the same time. The quantity in A and LP_4 (bit position 4 of LP) represents the first subtotal. Positions 3, 2, and 1 of LP may contain any information used for other purposes. Positions 5 and 6 of LP contain the same information as positions 5 and 6 of A and are not shown. At instant 9, the content of A is transferred to Y. Quantity e is entered into X at instant 10, because the second last bit of quantity a is a ONE. At instant 11, the sum is available in U. At instant 13, the content of LP_4 is moved to LP_3 (by shifting the whole content of LP one position to the right). The quantity in U is shifted and entered into A and LP_4 at instant 14. The quantity in A and $LP_{4,3}$ represents the second subtotal. During instants 15 through 20 no quantity e is added, because bit 3 of multiplier a is a ZERO. The third subtotal is contained in A and three bit positions of LP. During instants 21 through 26, quantity e is again added to the content of A because bit 4 of multiplier a is a ONE. The final product is contained in A and positions 4 through 1 of LP.

2-56. Figure 2-19 illustrates the method of multiplication in more detail, in particular how quantity a = +1011 is shifted in LP and used to decide whether

INSTANT	FUNCTION	REGISTERS Y,X,U,A						REGISTERS A,LP					
		6	5	4	3	2	1	6	5	4	3	2	1
1	c(A) = a							0	0	1	0	1	1
2	A → LP							1	1	–	1	0	1
3	LP → A							$\underline{1}$	1	0	1	0	1
4	e → Y	0	0	1	1	1	0						
5	o → X	0	0	0	0	0	0						
6	U	0	0	1	1	1	0						
7	→ LP							$\underline{1}$	1	–	0	1	0
8	U → A, LP_4	0	0	0	1	1	1			0			
9	A → Y	0	0	0	1	1	1						
10	e → X	0	0	1	1	1	0						
11	U	0	1	0	1	0	1						
12	LP → A							1	1	0	0	1	0
13	→ LP							$\underline{0}$	0	–	0	0	1
14	U → A, LP_4	0	0	1	0	1	0			1			
15	A → Y	0	0	1	0	1	0						
16	o → X	0	0	0	0	0	0						
17	U	0	0	1	0	1	0						
18	LP → A							0	0	1	0	0	1
19	→ LP							$\underline{1}$	1	–	1	0	0
20	U → A, LP_4	0	0	0	1	0	1			0			
21	A → Y	0	0	0	1	0	1						
22	e → X	0	0	1	1	1	0						
23	U	0	1	0	0	1	1						
24	LP → A							1	1	0	1	0	0
25	→ LP							0	0	–	0	1	0
26	U → A, LP_4	0	0	1	0	0	1			1			
	A, LP	0	0	1	0	0	1	0	0	1	0	1	0

Figure 2-19. Multiplication of Two Binary Numbers, Method of Operation

or not quantity e = +1110 is to be added. At instant 1, the accumulator (A) contains quantity a, which is transferred to LP and cycled one position to the right at instant 2. Note that bits 5 and 6 of quantity a get lost, that bit 1 of a is entered into bit positions 5 and 6 of LP, and that nothing is entered into bit position 4 of LP. At instant 3, the content of LP is transferred to A. Bit 6 contained in A is now used for decision-making as symbolized by an underline. Since bit 6 is a ONE, quantity e is entered into Y at instant 4. Instants 4, 5, and 6 are the same as described for the principles of multiplication. At instant 7, the content of A is again transferred to LP and cycled, as described for instant 2. Since bit position 6 of LP contains a ONE, the quantity e is entered into Y again at instant 10. Instants 8 through 11 are the same as described for the principles of multiplication. At instants 12 and 13, the content of LP is cycled once more. Since a ZERO is now in position 6, the quantity zero is entered into X at instant 16. At instant 22, quantity e is entered into X, because bit position 6 of LP contains a ONE at instant 19. At instants 24 and 25, the content of LP is cycled the last time. After instant 26, the final product is contained in registers A and LP. Note that sign bits 6 and 5 contained in A and LP are identical. The value bits are contained in bit positions 4 through 1 of A and LP. Having explained the principle of multiplication applied and the method of operation, the description of instruction MP K can be continued.

2-57. Figures 2-20 through 2-25 illustrate the execution of MP 1032. Quantity a, contained in the accumulator (A), is to be multiplied by quantity e, located at K = 1032. The entire code of the instruction was entered into B during the execution of the preceding NDX instruction (B now contains b = 111032), and order code 11 was entered into register SQ. At Action 1 of subinstruction MP0 (figures 2-20 through 2-23), the address contained in B is transferred to S, and the selection logic gates location 1032 for readout and write-in. At Action 2, register G is cleared, quantity a (the multiplier) is transferred to B, and the sign bit of a is entered into the SQG by means of TSGN. At Action 3,

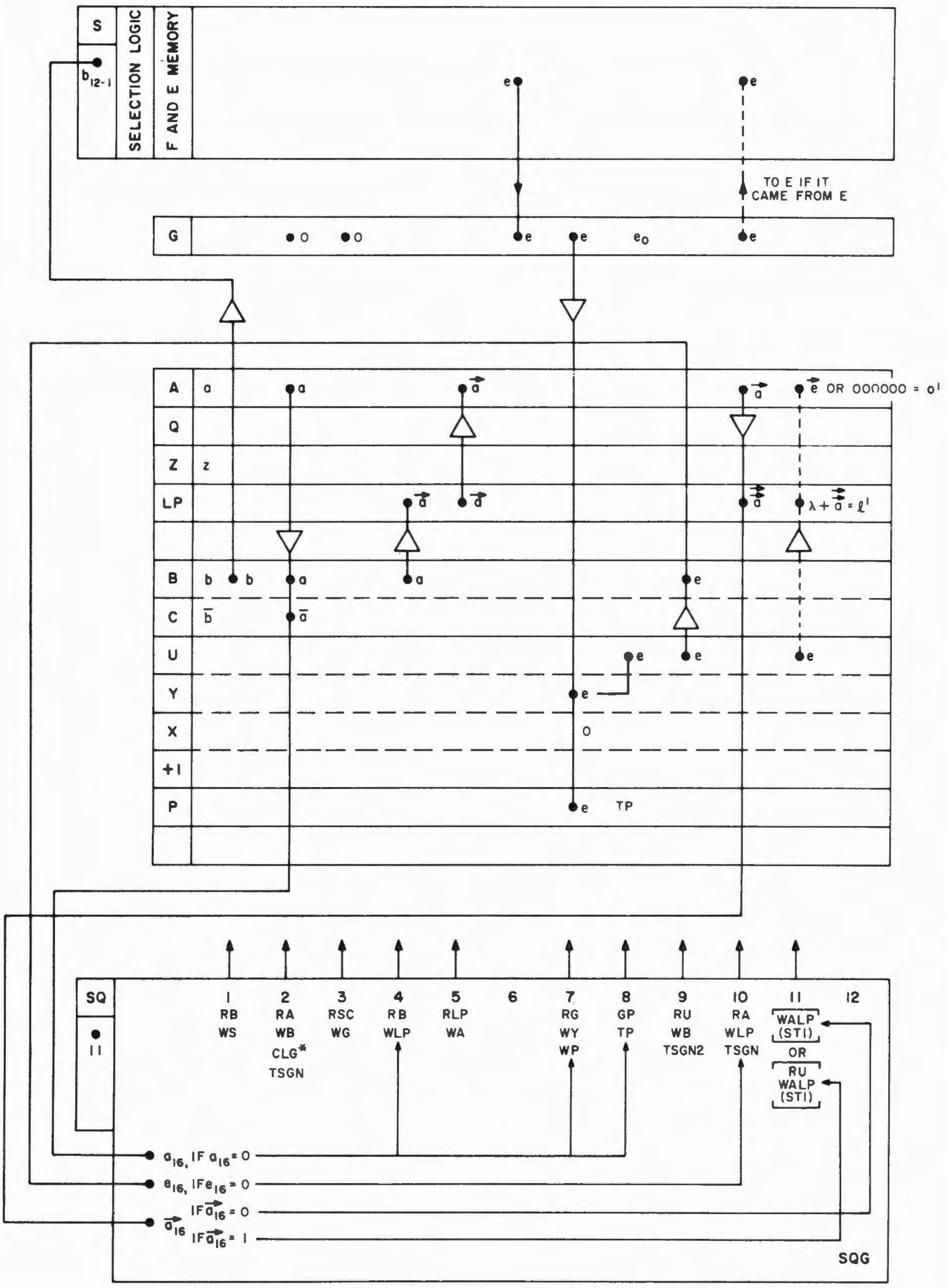

Figure 2-20. Subinstruction MP0 (a and e Positive)

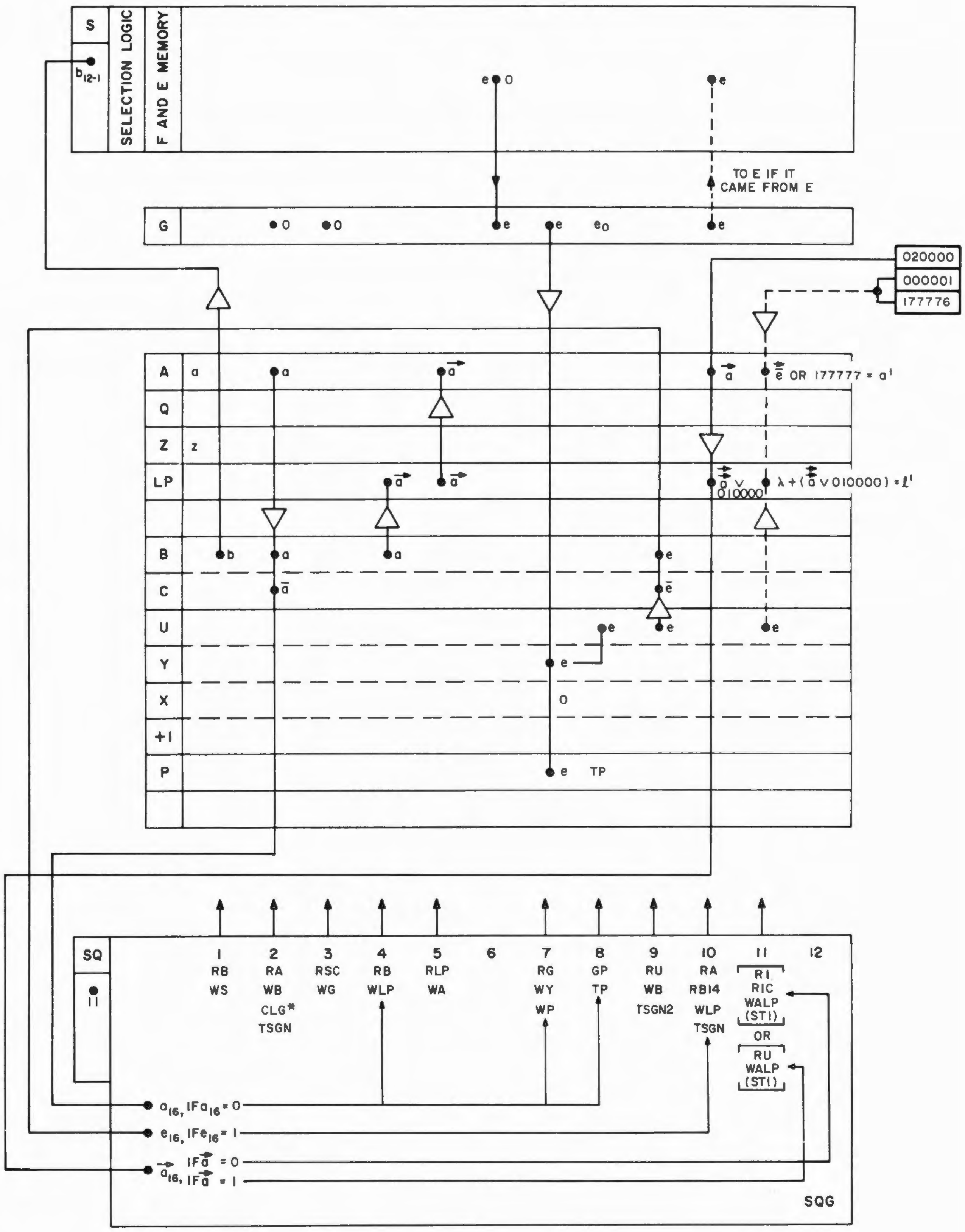

Figure 2-21. Subinstruction MP0 (a Positive and e Negative)

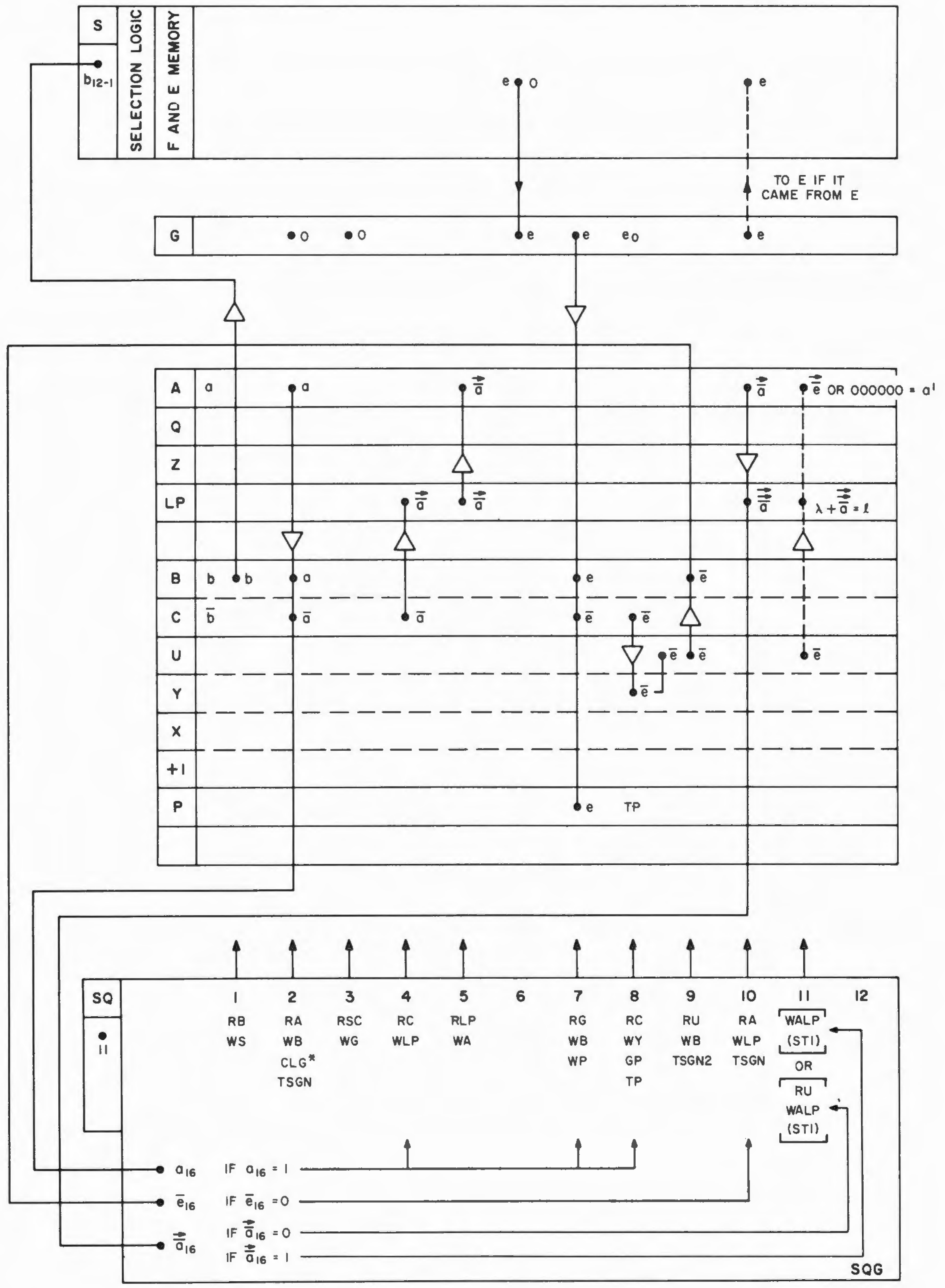

Figure 2-22. Subinstruction MP0 (a Negative and e Negative)

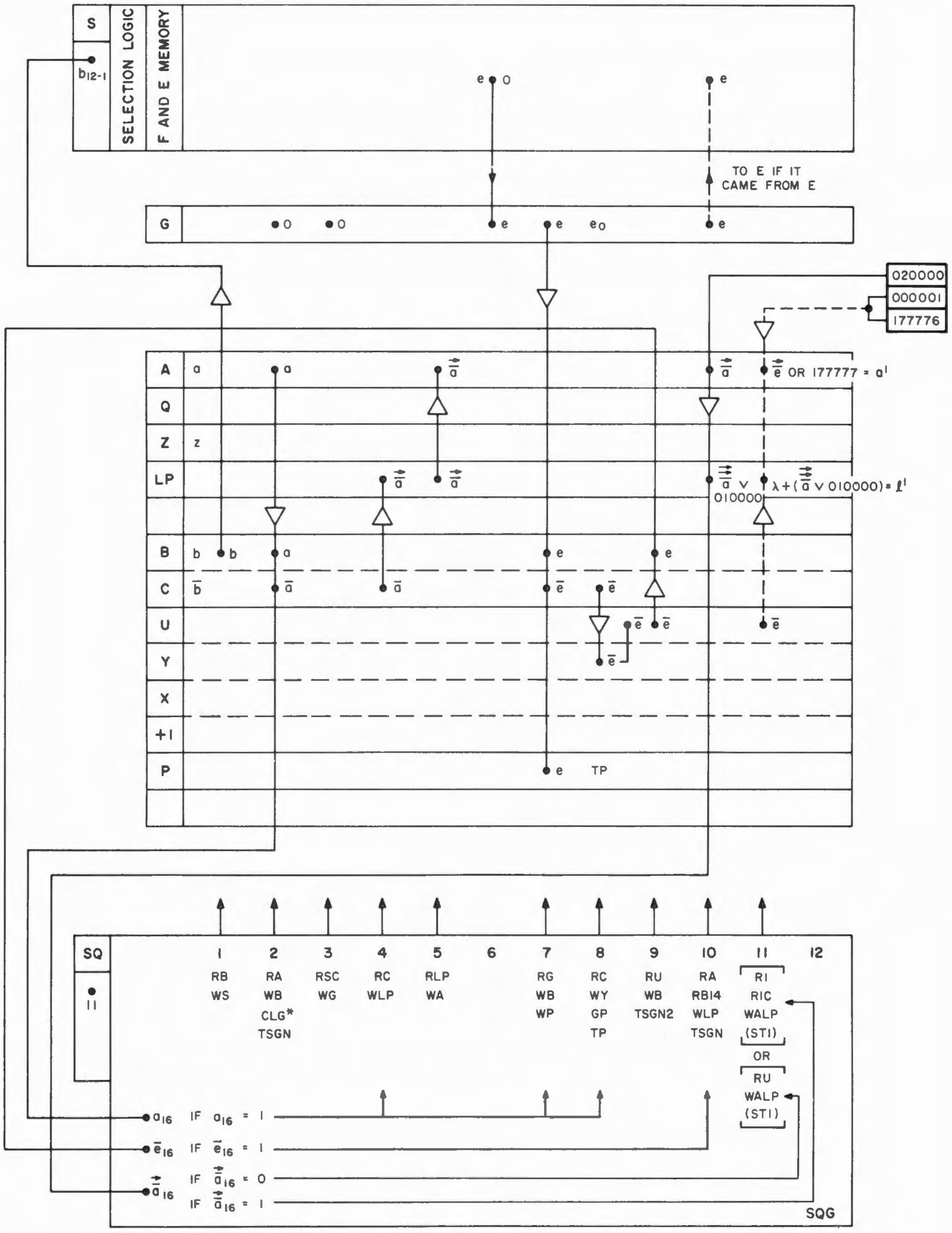
S
SELECTION LOGIC
F AND E MEMORY
b_{12-1}
TO E IF IT CAME FROM E
G
020000
000001
177776
A
Q
Z
LP
B
C
U
Y
X
+1
P
$\overrightarrow{e}$ OR 177777 = a'
$\overrightarrow{\overrightarrow{a}}$ ∨ 010000
λ+($\overrightarrow{\overrightarrow{a}}$ ∨ 010000) = ℓ'
TP
SQ
1 RB WS
2 RA WB CLG* TSGN
3 RSC WG
4 RC WLP
5 RLP WA
6
7 RG WB WP
8 RC WY GP TP
9 RU WB TSGN2
10 RA RB14 WLP TSGN
11 RI RIC WALP (STI) OR RU WALP (STI)
12
a_{16} IF a_{16} = 1
$\overline{e}_{16}$ IF $\overline{e}_{16}$ = 1
$\overrightarrow{a}_{16}$ IF $\overrightarrow{a}_{16}$ = 0
IF $\overrightarrow{a}_{16}$ = 1
SQG

Figure 2-23. Subinstruction MP0 (a Negative and e Positive)

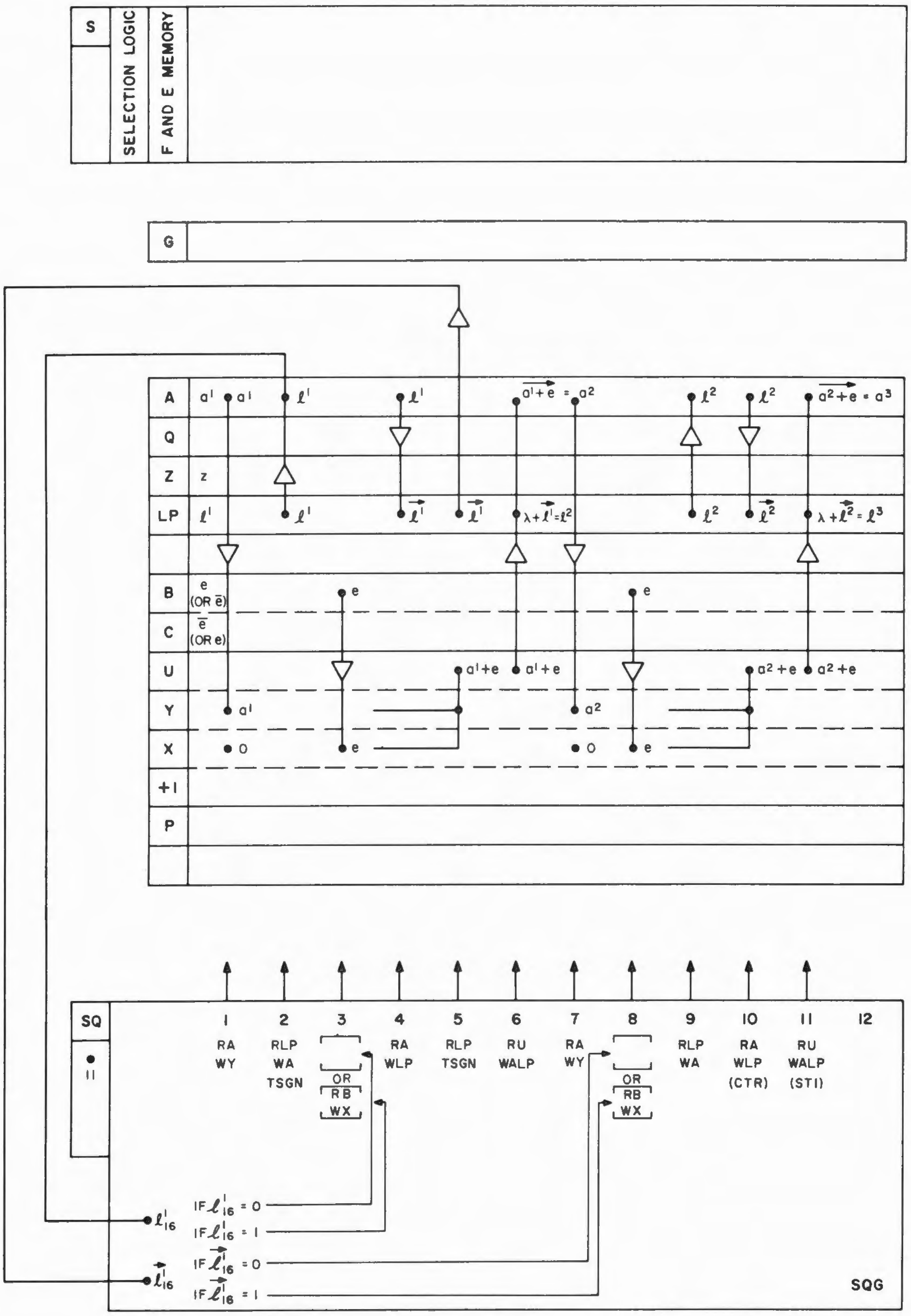

Figure 2-24. Subinstruction MP1

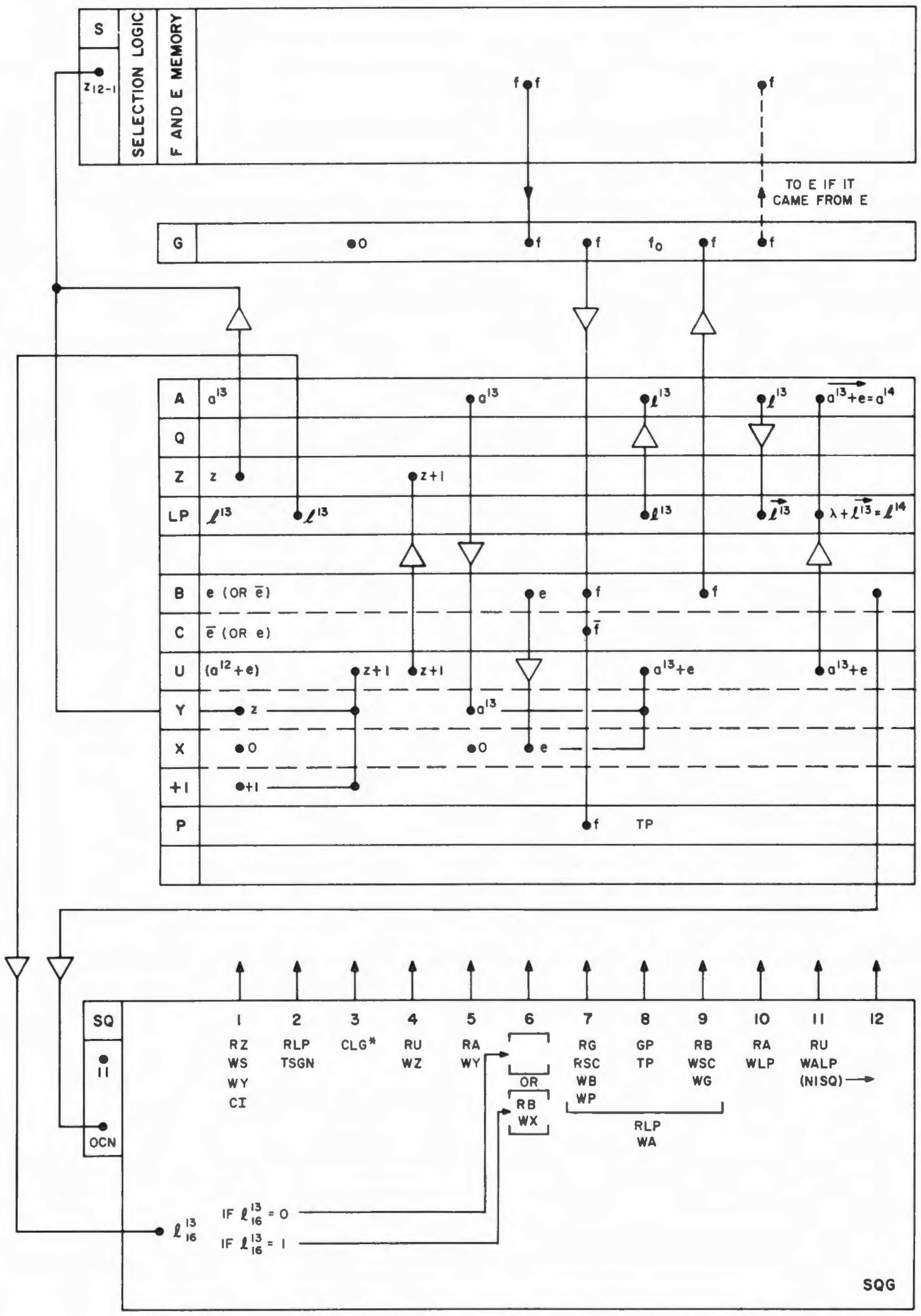

Figure 2-25. Subinstruction MP3

no transfer of data occurs unless the address in S is smaller than 0020. If $c(S) \leq 0017$, control pulse RSC reads the selected flip-flop register into G. This arrangement makes it possible to transfer quantity a to G and to square quantity a.

2-58. If the sign of quantity a was found (at Action 2) to be positive (figures 2-20 and 2-21), a is written into LP (and cycled one position to the right, as symbolized by $\overrightarrow{a}$) at Action 4. At Action 5, quantity $\overrightarrow{a}$ is transferred to A. At Action 7, quantity e is entered into Y and P. A new parity bit (e_o) is entered into G, and an alarm is caused in case of incorrect parity at Action 8. At Action 9, quantity e is transferred to B and the sign bit of e is entered into the SQG by means of TSGN2. If the sign of e is positive (figure 2-20), quantity $\overrightarrow{a}$ contained in A is transferred to LP (and cycled another position to the right as symbolized by $\overrightarrow{\overrightarrow{a}}$), at Action 10. Also at Action 10, bit 16 contained in A is entered into the SQG by means of another TSGN control pulse.

2-59. If the sign of e is negative at Action 9 (figure 2-21), quantity $\overrightarrow{a}$ contained in A is transferred to LP (and cycled a second time) at Action 10, and a ONE is entered into bit position 13 of LP at the same time. (This is equivalent to ORing a ONE into bit position 3 of LP at instant 7 of figure 2-19.) Also at Action 10, bit 16 contained in A is entered into the SQG. The ONE entered into bit position 13 of LP will be moved later several times to the right and, finally, will appear in bit positions 16 and 15 of LP. Since e is negative, bit positions 16 and 15 of A will also contain ONEs at the end of the multiplication. This can be proved by repeating the example in figure 2-19 with a negative quantity for e.

2-60. If the sign of quantity a was found (at Action 2) to be negative (figures 2-22 and 2-23), quantity $\overline{a}$ is written into LP (and cycled) at Action 4. At Action 5, quantity $\overrightarrow{\overline{a}}$ is transferred to A. At Action 7, quantity e is entered into B and P. A new parity bit (e_o) is entered into G, and an alarm is caused in case of incorrect parity, at Action 8. Furthermore, the quantity $\overline{e}$ is entered into Y and becomes available at U. At Action 9, quantity $\overline{e}$ is transferred to B and the sign bit of $\overline{e}$ is entered into the SQG by means of TSGN2. If the sign of $\overline{e}$ is positive (figure 2-22), i.e. if e is negative, quantity $\overset{\rightarrow}{\overline{a}}$ contained in A is transferred to LP (and cycled) at Action 10. Also at Action 10, bit 16 contained in A is entered into the SQG. The above operation (Actions 4 through 10) replaces negative quantity a by a positive quantity and negative quantity e by a positive quantity, after which the example in figure 2-22 is similar to the example in figure 2-20, and the final product is positive.

2-61. If the sign of $\overline{e}$ is negative at Action 9 (figure 2-23), i.e. if e is positive, quantity $\overset{\rightarrow}{\overline{a}}$ contained in A is transferred to LP (and cycled) and a ONE is entered into bit position 13 of LP, at Action 10. Also at Action 10, bit 16 contained in A is entered into the SQG. This operation replaces negative quantity a by a positive quantity and positive quantity e by a negative quantity, after which the example in figure 2-23 is similar to the example in figure 2-21, and the final product is negative.

2-62. At Action 11 of figure 2-20, the accumulator (A) is cleared if bit 16 of $\vec{a}$ is a ZERO. If bit 16 is a ONE, quantity e is shifted and entered into A and bit position 14 of LP. (Compare with instant 8 of figure 2-19.) The λ in figure 2-20 indicates that bit position 14 of LP has been filled. At Action 11 of figure 2-21, the quantity minus-zero (177777) is entered into A, if bit 16 of $\vec{a}$ is a ZERO. If bit 16 is a ONE, quantity $\overline{e}$ is shifted and entered into A and bit position 14 of LP. Action 11 of figure 2-22 is similar to Action 11 of figure 2-21, and Action 11 of figure 2-23 is similar to Action 11 of figure 2-21. In all four figures the content of A at Action 12 is renamed a^1, and the content of LP is called l^1. The upper index 1 means "containing parts of first subtotal". The first subtotal is contained in A and bit position 14 of LP and is symbolized as (a^1, l^1_{14}). Control pulse ST1 causes the SQG to execute-next subinstruction MP1. Table 2-4 shows the conditions of registers A, L, B, P, and Z after the execution of MP0.

2-63. Figure 2-24 illustrates the first execution of subinstruction MP1. Registers A and LP contain parts of the first subtotal before Action 1. The next address (z) is still contained in Z. In the case that MP1 is executed after MP0 of the example in figure 2-20 or 2-22, the buffer (B) contains the quantity e before Action 1, as shown in table 2-4 and figure 2-24. In the case that MP1 follows MP0 of the example in figure 2-21 or 2-23, B contains $\overline{e}$ instead of e. At Action 1 of MP1, quantity a^1 is entered into Y. At Action 2, quantity 1^1 is transferred to A, and bit 16 of 1 is entered into the SQG. If bit 16 is a ZERO, no operation takes place at Action 3. If bit 16 is a ONE, quantity e is entered into X (as at instant 10 of figure 2-19) and added to a^1. The quantity (a^1) or the sum (a^1 + e) is transferred to A and bit position 14 of LP at Action 6. (Compare with instant 14 of figure 2-19.) After Action 6 the second subtotal (a^2, 1^2_{14-13}) is contained in A and bit positions 14 and 13 of LP. At Action 7, quantity a^2 is entered into Y. At Action 4, the quantity 1^1 is transferred to LP (and cycled), and bit 16 of quantity 1^1 is entered into the SQG at Action 5. If bit 16 is a ZERO, no operation takes place at Action 8. (At instant 16 of figure 2-19 the quantity zero was entered into X.) If bit 16 is a ONE, quantity e is entered into X and added to a^2. At Actions 9 and 10, quantity 1^2 is cycled and the content of the MPCTR (multiply counter, located in the SQG) is decremented from six to five by means of control pulse CTR. The quantity (a^2) or the sum (a^2 + e) is transferred to A and bit position 14 of LP at Action 11. (Compare with instant 20 of figure 2-19.) After Action 11, the third subtotal (a^3, 1^3_{14-12}) is contained in A and bit positions 14 through 12 of LP.

2-64. Control pulse ST1 (Action 11 of figure 2-24) causes the SQG to execute next subinstruction MP1 again, if the multiply counter (MPCTR) contains a five, a four, a three, a two, or a one. Therefore, MP1 is executed six times. The fifth subtotal (a^5, 1^5_{14-10}) is established after the second MP1. The seventh subtotal (a^7, 1^7_{14-8}) is established after the third MP1. The ninth subtotal (a^9, 1^9_{14-6}) is established after the fourth MP1. The eleventh subtotal (a^{11}, 1^{11}_{14-4}) is established after the fifth MP1, and the thirteenth subtotal

TABLE 2-4

CONTENTS OF REGISTERS AT END OF MP0

INITIAL CONDITIONS	c(A)	c(LP)	c(B)	c(Z)
a POSITIVE e POSITIVE	000000 OR $\overrightarrow{e}$	$\lambda + \overrightarrow{\overrightarrow{a}}$	e	z
a POSITIVE e NEGATIVE	177777 OR $\overrightarrow{e}$	$\lambda \vee 10000 + \overrightarrow{\overrightarrow{a}}$	e	z
a NEGATIVE e NEGATIVE	000000 OR $\overrightarrow{\overline{e}}$	$\lambda + \overrightarrow{\overrightarrow{\overline{a}}}$	$\overline{e}$	z
a NEGATIVE e POSITIVE	177777 OR $\overrightarrow{\overline{e}}$	$\lambda \vee 100000 + \overrightarrow{\overrightarrow{\overline{a}}}$	$\overline{e}$	z

$(a^{13}, 1^{13}_{14-2})$ is established after the sixth MP1. If the MPCTR contains the number zero, control pulse ST1 causes the SQG to execute-next subinstruction MP3.

2-65. Figure 2-25 illustrates the execution of subinstruction MP3. Before Action 1, register A contains quantity a^{13}, LP contains 1^{13}, B still contains e (or $\bar{e}$), U contains $(a^{12} + e)$ or $(a^{12} + 0)$, Z still contains "next" address z, and order code 11 is still contained in register SQ. At Action 1 of MP3, "next" address z is entered into S and Y, and is incremented by one in the Adder. The selection logic gates the location of next instruction f for readout and write-in. At Action 2, bit 16 of 1^{13} is entered into the SQG. At Action 3, register G is cleared, as usual. At Action 4, quantity (z + 1) is transferred to Z. At Action 5, quantity a^{13} is entered into Y. If bit 16 of 1^{13} at Action 2 is a ZERO, no operation takes place at Action 6. If bit 16 is a ONE, quantity e is added to a^{13}. Actions 7 through 9 are very similar to the STMIC; entering, testing, and restoring f. The additional control pulses, RLP and WA, transfer 1^{13} to A. At Action 10, quantity 1^{13} is transferred to LP (and cycled). At Action 11, quantity a^{13} or $(a^{13} + e)$ is shifted and entered into A and bit position 14 of LP as described previously. The final product $(a^{14}, 1^{14}_{14-1})$ is now established. Control pulse NISQ enters the order code (OCN) of the instruction to be executed-next into the SQG at Action 12.

2-66. INSTRUCTION DV K (Divide by Data at K, Order Code 5 with EXTEND Preceding)

2-67. DV K means: divide the content of the accumulator (A) by the quantity located at K. K may be any legal address. The entire operation DV K (for $0020 \leq K$) can be formulated as follows:

(1) Set $c(A) = b(A) \div c(K)$ for $|b(A)| < |c(K)|$ where b(A) does not contain an overflow or underflow (OV) bit.
Set $c(Q) = \overline{|R|}$ (complemented quantity of the absolute value of the remainder).

Set c(LP) > 0, if quotient is positive.

Set c(LP) < 0, if quotient is negative.

If $|b(A)| = |c(K)|$, then $|c(A)| = 37777$ and $c(Q) = -|c(K)|$

If $|b(A)| > |c(K)|$, then $|c(A)| = 37777$ and c(Q) is meaningless.

(2) Restore c(K) = b(K), if $K \leq 1777$, except for K = 0000, 0001, or 0003.

(3) Execute-next the instruction located at z = (L + 1).

Performing a division requires the execution of Instruction NDX, subinstruction DV0 once, DV1 fourteen times, and STD2 once. Instruction DV K is the only Regular Instruction (except TC) which uses register Q. Therefore, any program portion following a TC instruction, and including a DV instruction, must preserve the return address (normally contained in Q) somewhere else.

2-68. Before discussing the execution of instruction DV K, the principle of division applied should be explained. Figure 2-26 demonstrates the division of binary quantity a = +1011 by e = +1110. Approach 1 shows how the division can be carried out manually in a way commonly used. Since a < e, a binary point has to be set into the quotient first. Then the dividend (a) is rewritten, and a ZERO is added as a new lowest order bit. If the new number (absolute value, not quantity) formed (10110) is larger than the number e, e is subtracted from the new number (thereby establishing the first remainder, 100. 0) and a ONE is set into the quotient. A ZERO is added as a new lower order bit position of the first remainder to again form a new number (10000). Since the new number is larger than e, e is subtracted again and a second ONE is set into the quotient. Quantity 00. 10 is the second remainder. Adding a ZERO to the second remainder leaves a new number (00100) still smaller than e. For this reason, a ZERO is set into the quotient and quantity 0. 100 becomes the third remainder. A ZERO is added to the third remainder, but again the new number (01000) is smaller than e. Therefore, a second ZERO is entered into the

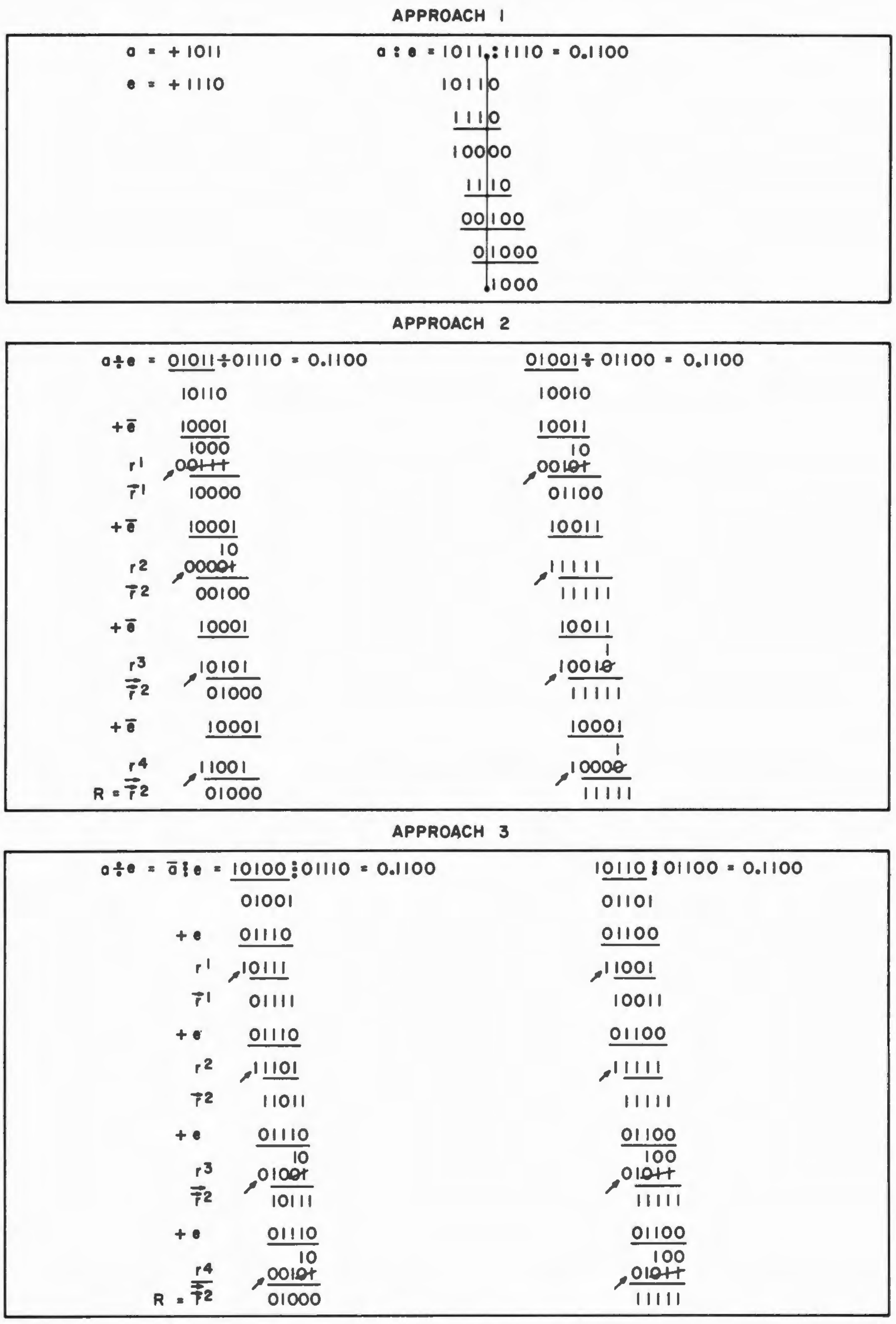

Figure 2-26. Division of Binary Numbers, Principle of Operation

quotient and quantity 0.1000 becomes the final remainder. Expressed in decimal numbers, the resulting quotient is 3/4 and the remainder is 1/2.

2-69. Approach 2 of figure 2-26 illustrates an operation which can be performed more easily by a computer than by the mental process of comparing the divisor with the various remainders. Sign bit ZERO has been added as bit 5 for dividend a and divisor e. Starting with the divide operation ($a \div e$), the dividend is cycled one position to the left which is the same as adding a ZERO as a new lowest order bit. The complement of the divisor ($\overline{e}$) is added to the new number thereby establishing the first remainder (r^1). The crossed-out numbers indicate the end around carry operation. If bit 5 of r^1 is a ZERO (indicating that r^1 is still positive), as in the example, a ONE is entered into the quotient. Then r^1 is cycled, $\overline{e}$ is added again, and another ONE is entered into the quotient because the second remainder (r^2) also contains a ZERO in bit position 5. At the next instant, r^2 is cycled, and $\overline{e}$ is added again. The third remainder (r^3) contains a ONE in bit position 5, indicating that the last remainder was too small for a correct subtraction. Consequently, a ZERO is entered into the quotient and r^2 (the last correct remainder) is cycled again. Number $\overline{e}$ is added once more, and remainder r^4 is also incorrect. A second ONE is set into the quotient and r^2 shifted twice ($\overset{\Rightarrow}{r^2}$) is taken as the final remainder (R). Setting a ONE into the quotient, if bit 5 of the last remainder contains a ZERO, or setting a ZERO, if bit 5 contains a ONE, leads to an error whenever a remainder happens to be a minus zero (11111). This is demonstrated by the example 01001 $\div$ 01100, where the second remainder is minus zero, but a ONE has to be entered into the quotient since the remainder is not smaller than zero. A remainder is incorrect only if it is smaller than zero (either plus zero or minus zero).

2-70. Approach 3 of figure 2-26 is free of the possibility for errors described in approach 2. Instead of dividing a by e, the complement of a is used for cycling, and e is added to the various remainders. This operation is indicated by three points (⋮). Starting with the operation $\bar{a}$ ⋮ e, quantity $\bar{a}$ is cycled and e is added. Because remainder r^1 contains a ONE, a ONE has to be entered into the quotient. Whenever a remainder contains a ZERO (as for r^3 and r^4) a ZERO is to be set into the quotient. The second example of approach 3 proves that this approach is also correct in case a remainder becomes minus zero. Approach 3 needs only the highest order bit of a remainder to decide a bit of the quotient and is relatively simple to instrument. Approach 3 is applied for the AGC. Although the AGC uses two sign bits (SQ and US) instead of one, this does not matter, as can be proved by performing 110100 ⋮ 001110.

2-71. Figure 2-27 illustrates the method of operation for the divide instruction and is similar to figures 2-18 and 2-19. At instant 1, the complement of dividend a is entered into Q. After instant 2, a positive quantity is stored in LP for later use if the quotient is to be positive, or a negative quantity is stored in LP if the quotient is to be negative. At instant 3, a ONE is entered in bit position 1 of B. The content of B will be cycled to the left several times during the operation. When the ONE entered at instant 3 moves into the highest bit position, this indicates that the divide operation has been completed. At instants 4 and 5, the quantity contained in Q is cycled one position to the left, returned to Q and entered into Y. At the same time, a ONE (minus sign) is entered into bit position 6 of Q and Y to correct the cycle operation. (The CYL operation of the computer transfers the ZERO contained in bit position 4 to positions 5 and 6. The ONE entered into position 6 by a special control pulse has the same effect as transferring the ONE at position 5 to position 6.) At instant 6, divisor e is entered into X and added to the quantity which was entered into Y. The sum is available at U at instant 7. Bit position 6 of U contains a ONE, therefore, as described for approach 3, the first remainder is correct, and the actions taken next are as shown for instants 8 and 9. The first remainder is transferred from U to Q for later use. The content of B is cycled to the left one place, and a ZERO is written into the quotient, which is

INSTANT	FUNCTION	REGISTERS Q,Y,X,U						REGISTERS LP, B						
		6	5	4	3	2	1	6	5	4	3	2	1	
1	$\overline{a} \rightarrow Q$	1	1	0	1	0	0							
2	$00010 \rightarrow LP$							0	0	0	0	0	1	
3	$00001 \rightarrow B$							0	0	0	0	0	1	
4	$100000\ V\overleftarrow{Q} \rightarrow Q$	1	0	1	0	0	1							
5	$\rightarrow Y$	1	0	1	0	0	1							
6	$e \rightarrow X$	0	0	1	1	1	0							
7	U	$\underline{1}$	1	0	1	1	1							r^1
8	$U \rightarrow Q$	1	1	0	1	1	1							
9	$\overleftarrow{B} \rightarrow B$							$\underline{0}$	0	0	0	1	0	
10	$100000\ V\overleftarrow{Q} \rightarrow Q$	1	0	1	1	1	1							
11	$\rightarrow Y$	1	0	1	1	1	1							
12	$e \rightarrow X$	0	0	1	1	1	0							
13	U	$\underline{1}$	1	1	1	0	1							r^2
14	$U \rightarrow Q$	1	1	1	1	0	1							
15	$\overleftarrow{B} \rightarrow B$							$\underline{0}$	0	0	1	0	0	
16	$100000\ V\overleftarrow{Q} \rightarrow Q$	1	1	1	0	1	1							
17	$\rightarrow Y$	1	1	1	0	1	1							
18	$e \rightarrow X$	0	0	1	1	1	0							
19	U	$\underline{0}$	0	1	0	1	0							r^3
20	$\overleftarrow{100000\ VB} \rightarrow B$							$\underline{0}$	0	1	0	0	1	
21	$100000\ V\overleftarrow{Q} \rightarrow Q$	1	1	0	1	1	1							
22	$\rightarrow Y$	1	1	0	1	1	1							
23	$e \rightarrow X$	0	0	1	1	1	0							
24	U	$\underline{0}$	0	0	1	1	0							r^4
25	$\overleftarrow{100000\ VB} \rightarrow B$							$\underline{1}$	1	0	0	1	1	$\overline{\text{QUOTIENT}}$
26	$\overline{B} \rightarrow A$							0	0	1	1	0	0	QUOTIENT

NOTES: ALL NUMBERS ARE WRITTEN IN BINARY FORM
V MEANS OR
_ MEANS USED FOR TEST

Figure 2-27. Division of Binary Numbers, Method of Operation

stored in complemented form in the bit positions following the ONE set into B at instant 3. Instants 10 through 15 are a repetition of instants 4 through 9. Bit position 6 of U contained a ONE at instant 13, which indicates that the second remainder is also correct. Instants 16 through 19 are identical in their actions to instants 10 through 13 and 4 through 7, but bit 16 of U now contains a ZERO. The ZERO indicates that the third remainder is incorrect. For this reason, the content of Q (the last remainder cycled) is not replaced by the new remainder and a ONE is entered into B as its content is cycled. Instants 21 through 25 are a repetition of instants 16 through 20 because bit 6 of the fourth remainder is also a ZERO. As the content of B is cycled the fourth time, the ONE set at instant 3 now moves into bit positions 5 and 6 and this indicates that the divide operation is complete. At instant 26, the content of B is complemented to become the final quotient and is transferred to A. Q contains the complemented quantity of the absolute value of the final remainder. For the given example, the final remainder as contained in Q is 1000, but becomes 0.1000 when setting the binary point. Having explained the principle of division applied and the method of operation, the description of instruction DV K can be continued.

2-72. Figures 2-28 through 2-33 illustrate the execution of DV 1125. Quantity a contained in the accumulator (A) is to be divided by quantity e located at 1125. The entire code of the instruction was entered into B during the execution of the preceding NDX instruction (B now contains b = 121125), and order code 12 was entered into SQ. At Action 1 of subinstruction DV0 (figures 2-28 through 2-31), the address contained in B is transferred to S and the selection logic gates location 1125 for readout and write-in. At Action 2, register G is cleared, dividend a is transferred to B, and the sign bit of a is entered into the SQG by means of pulse TSGN. At Action 3, no transfer takes place unless the address in S is smaller than 0020. If $c(S) \leq 0017$, control pulse RSC reads the selected flip-flop register into G.

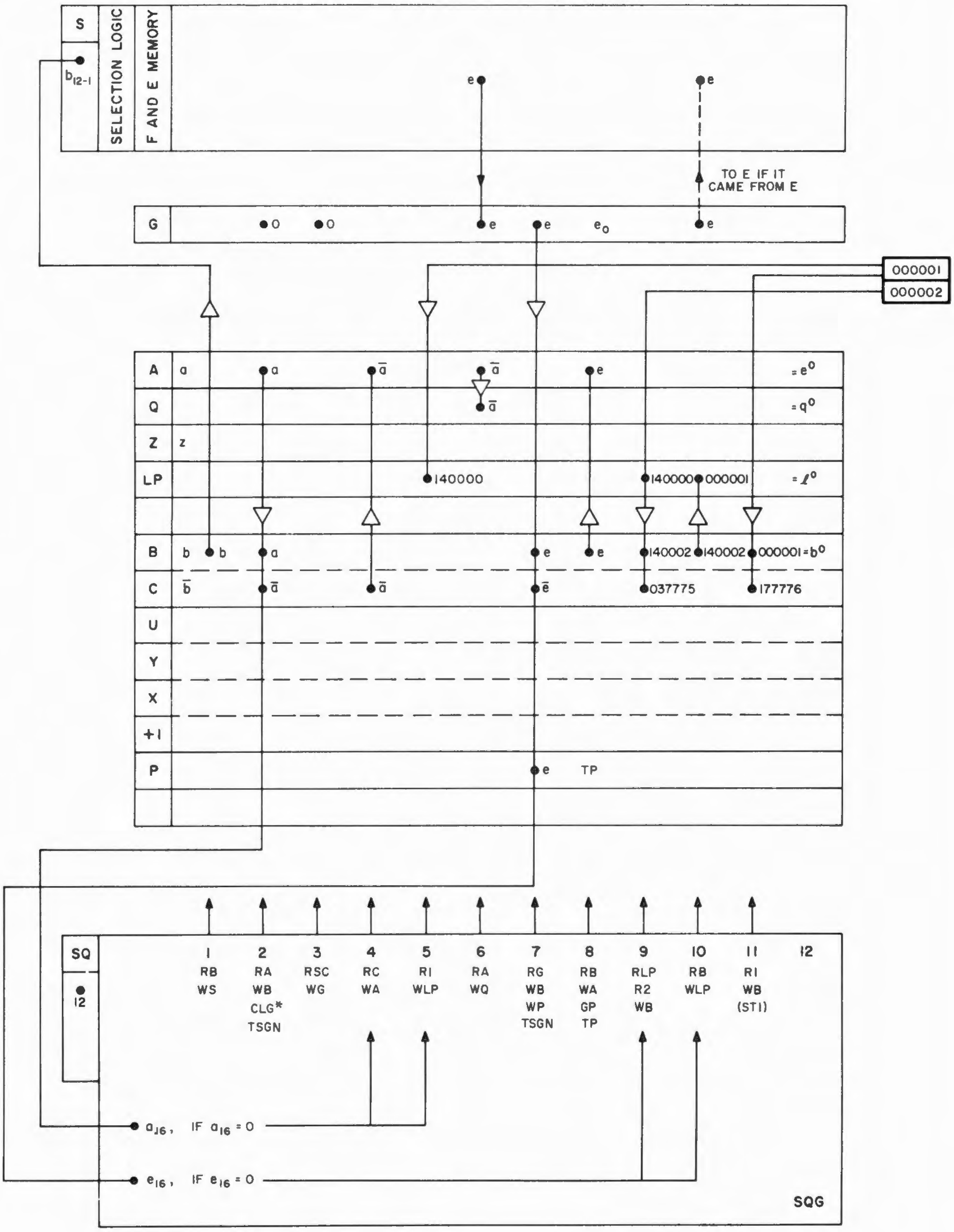

Figure 2-28. Subinstruction DV0 (a and e Positive)

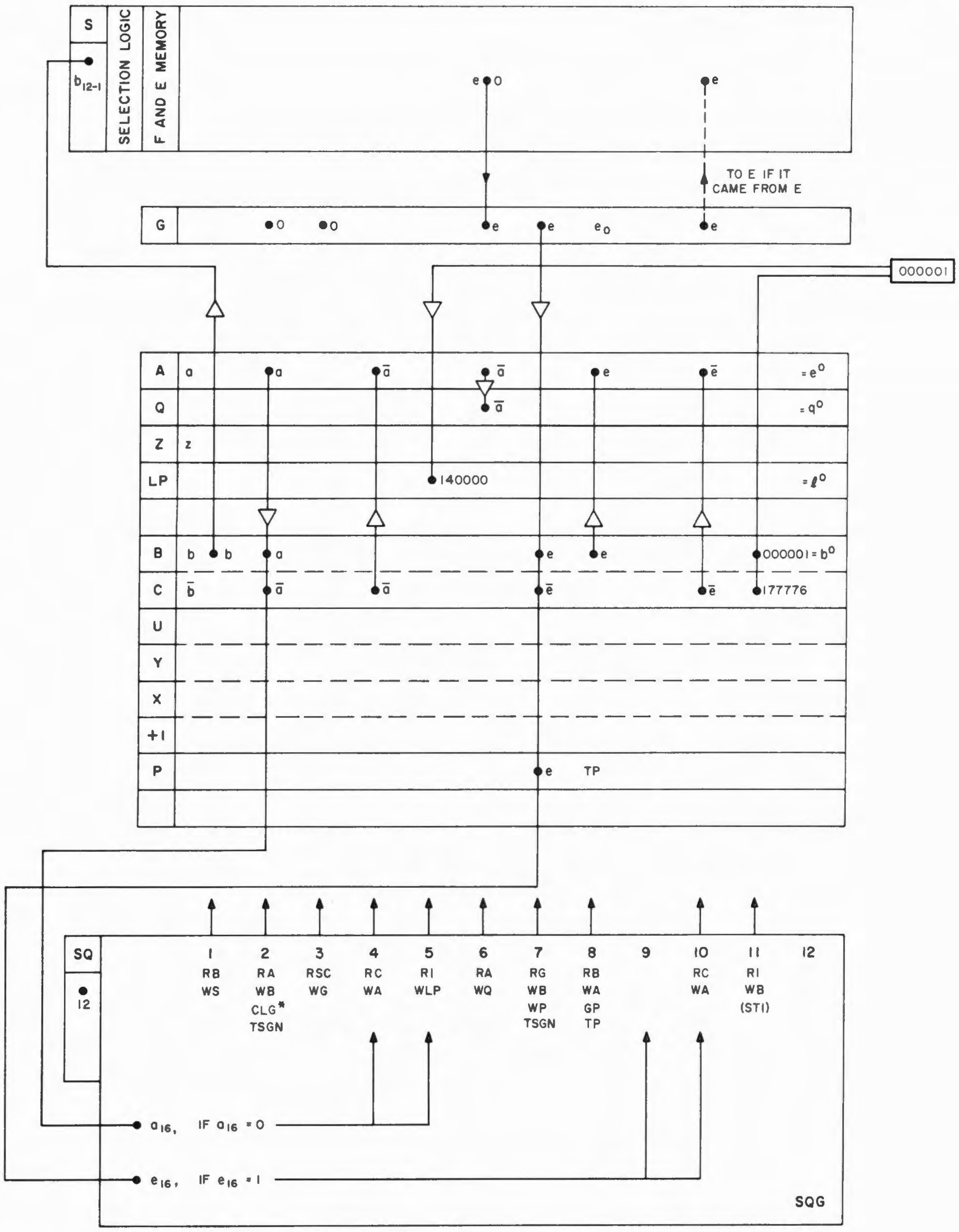

Figure 2-29. Subinstruction DV0 (a Positive and e Negative)

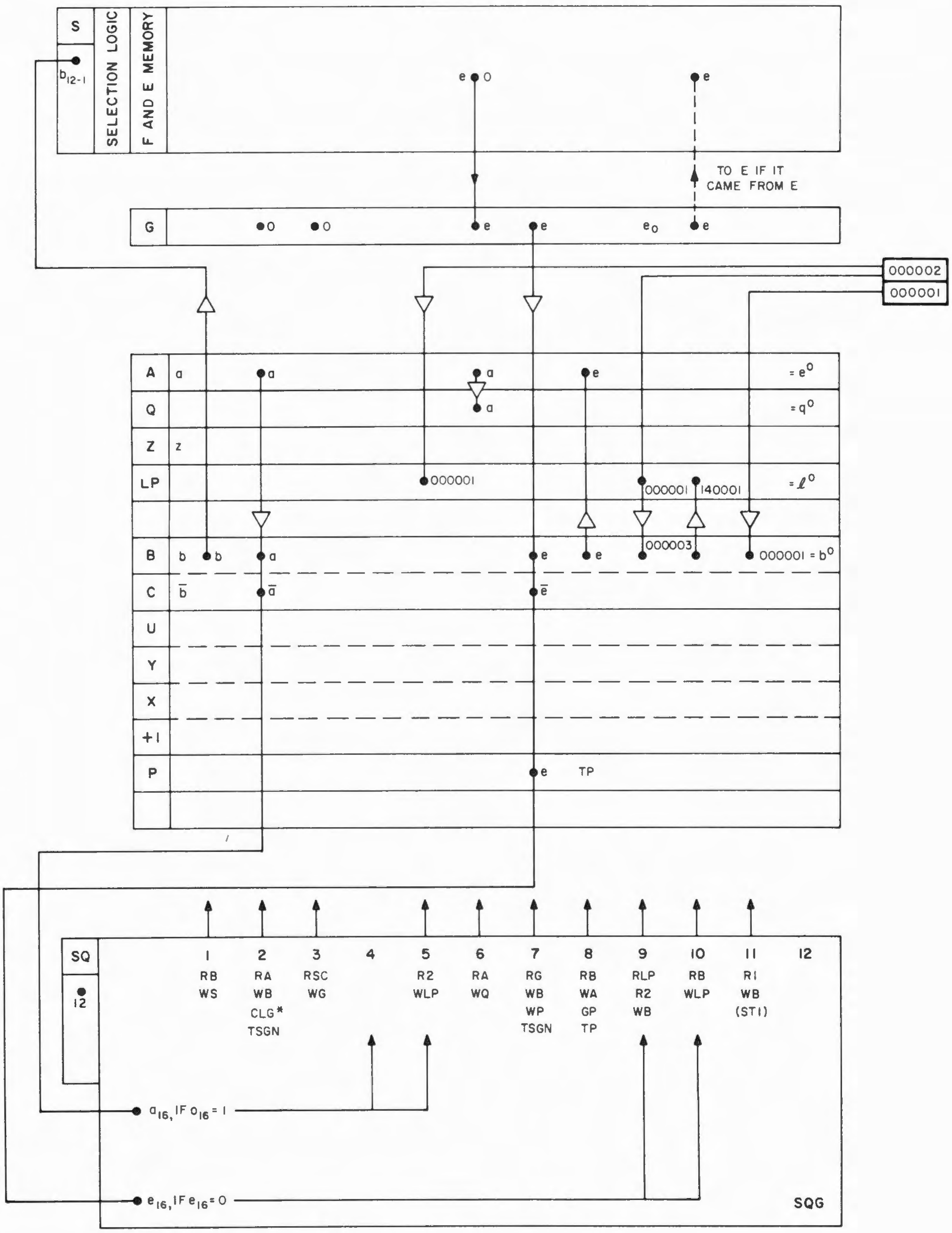

Figure 2-30. Subinstruction DVO (a Negative and e Positive)

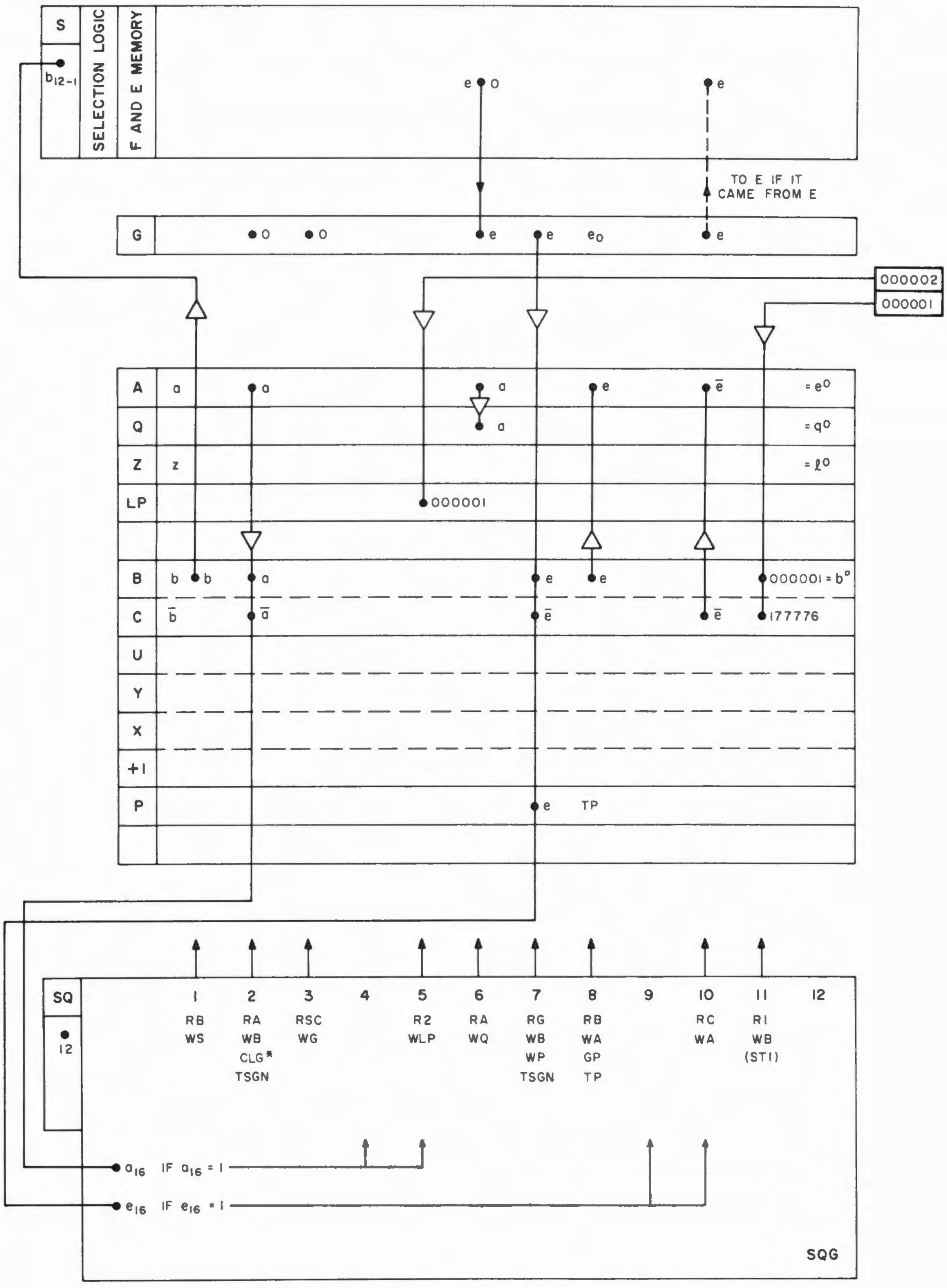

Figure 2-31. Subinstruction DV0 (a and e Negative)

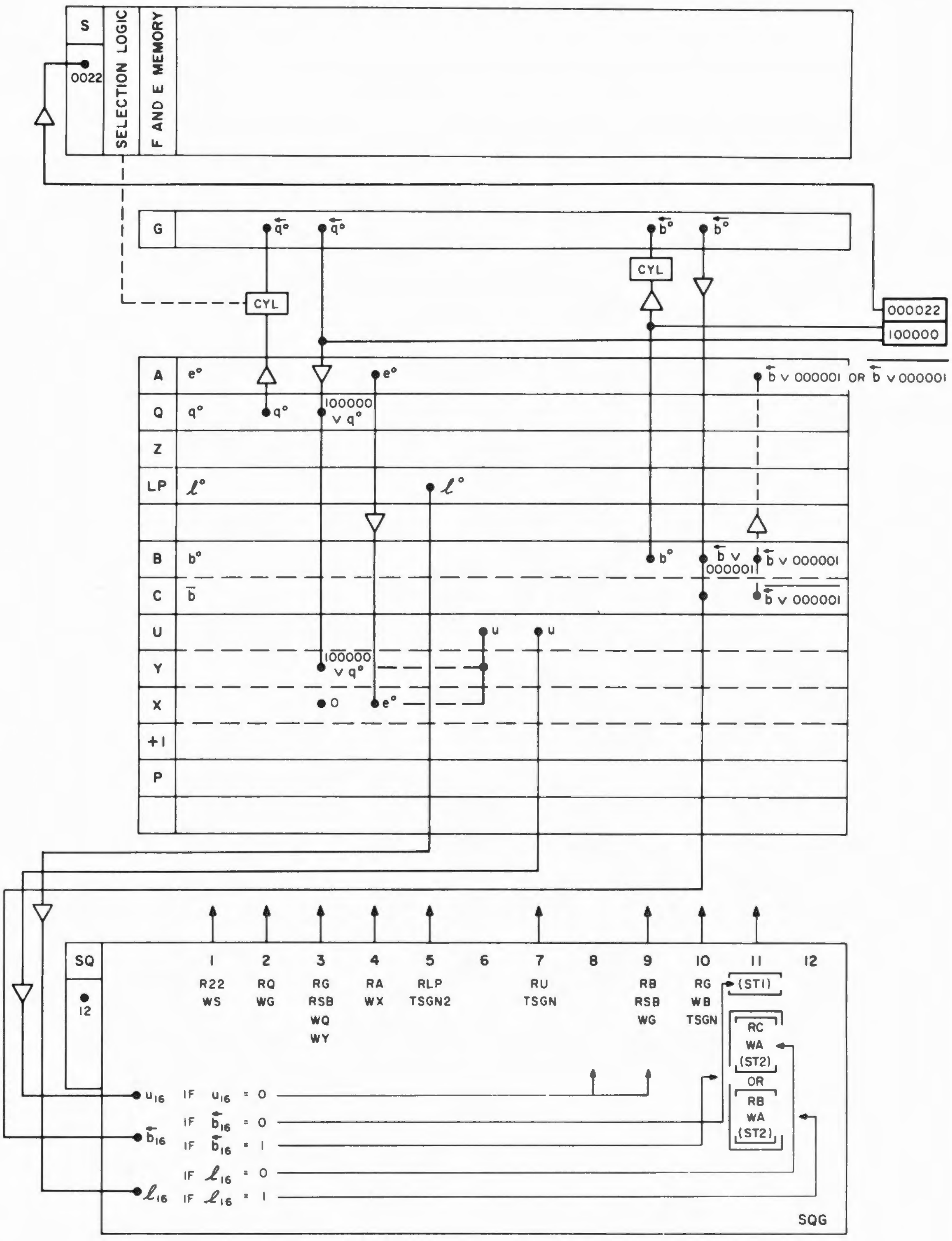

Figure 2-32. Subinstruction DV1 (Incorrect Remainder)

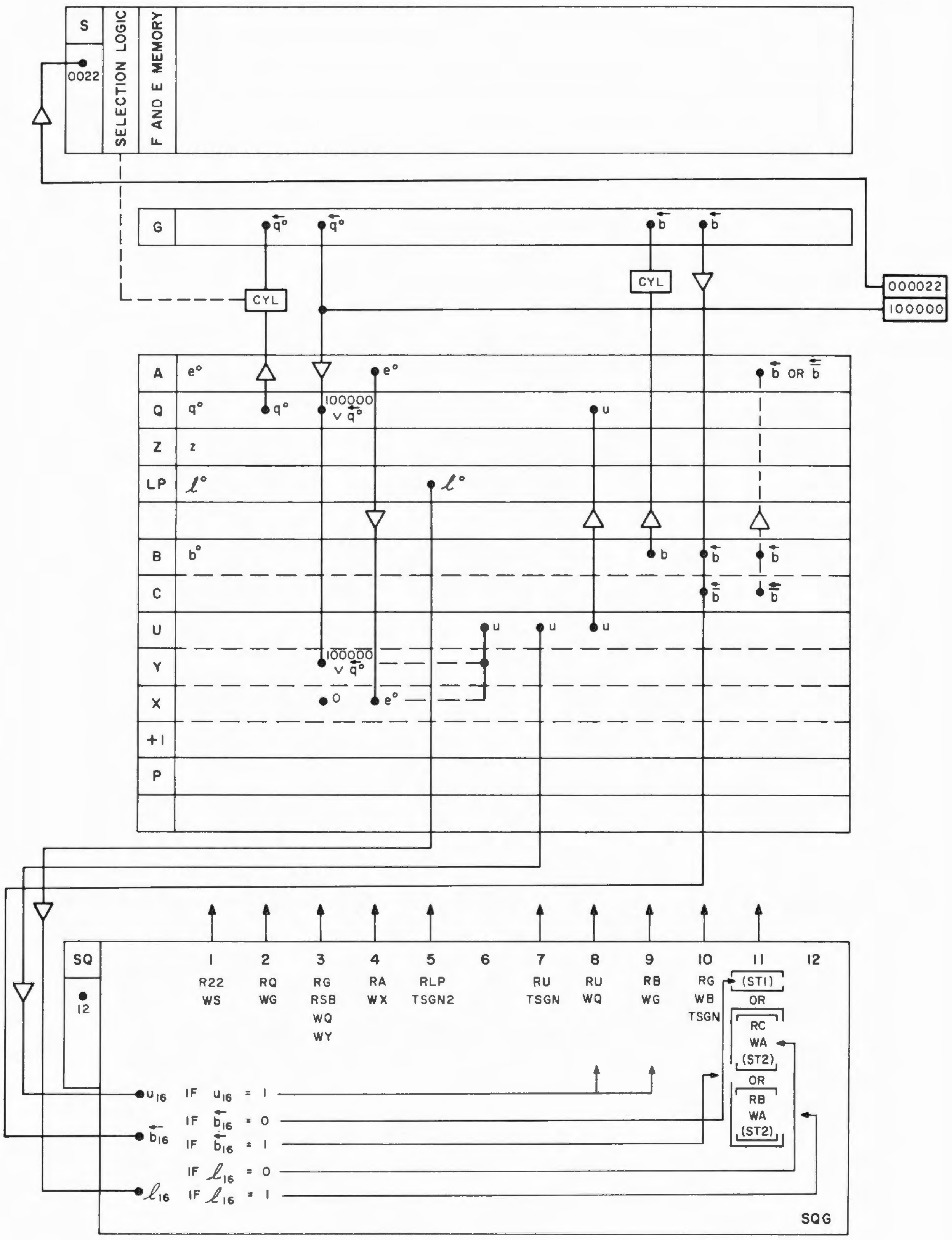

Figure 2-33. Subinstruction DV1 (Correct Remainder)

2-73. If the sign of dividend a was found (at Action 2) to be positive (figures 2-28 and 2-29), quantity $\bar{a}$ is written from C into A at Action 4, and transferred to Q at Action 6. At Action 5, the quantity 000001 is transferred (and cycled) into LP and becomes 140000. At Action 7, divisor e is entered into B and P, and bit 16 of e is entered into the SQG. At Action 8, the divisor is transferred to A, an alarm caused in case of incorrect parity, and a new parity bit of e is entered into G. If the sign of divisor e was found (at Action 7) to be positive (figure 2-28), the quantity located at LP and the quantity 00002 are written simultaneously (OR'd) into B which contains 140002 after Action 9. The quantity 140002 is transferred to LP at Action 10 and becomes 000001, indicating that the quotient has to be positive. (Compare with instant 2 of figure 2-27.) At Action 11, a ONE is entered into bit position 1 of B (as in figure 2-27) and used for shift counting. Control pulse ST1 causes the SQG to execute-next subinstruction DV1.

2-74. If the sign of divisor e was found (at Action 7) to be negative (figure 2-29), no operation is performed at Action 9, and at Action 10, quantity $\bar{e}$ is transferred from C to A. LP still contains 140000, indicating that the quotient is to be negative. Action 11 of figure 2-29 is identical to Action 11 of figure 2-28.

2-75. If the sign of dividend a was found (at Action 2) to be negative (figures 2-30 and 2-31), a is still in A at Action 4, and the quantity 000001 is written into LP at Action 5. At Action 6, dividend a is transferred to Q. The effects of Actions 7 and 8 in figures 2-28 through 2-31 are identical. If the sign of the divisor was found (at Action 7) to be positive (figure 2-30), the quantity located at LP and the quantity 00002 are written simultaneously into B, which contains 000003 after Action 9. The quantity 000003 is transferred to LP at Action 10 and becomes 140001, indicating that the quotient has to be negative. The effect of Action 11 is identical for figures 2-28 through 2-31.

2-76. If the sign of divisor e was found (at Action 7) to be negative (figure 2-31), the effect of Actions 9 and 10 is identical to that indicated in figure 2-29. LP still contains 000001 indicating that the quotient has to be positive. The contents of registers A, Q, LP, and B after the execution of DV0 are renamed as listed in table 2-5.

2-77. Figures 2-32 and 2-33 illustrate the first execution of subinstruction DV1. At Action 1, address 0022 is entered into register S and the selection logic sets the switch in front of the G register (figure 1-8) such that any word transferred from the write amplifiers to the G register is cycled one position to the left, as indicated by $\boxed{\text{CYL}}$ and the dotted line. At Action 2, quantity q^o is cycled one position to the left and entered into G. At Action 3, the content of G is transferred to Q and Y simultaneously with a ONE which is entered (OR'd) into position 16 of Q and Y (similar to instants 4 and 5 of figure 2-27). At Action 4, quantity e^o is entered into X (as at instant 6 of figure 2-27) and the sum $u = (100000 \vee q^o) + e^o$ becomes available at U. At Action 5, bit 16 of quantity 1 is entered into the SQG for later use. At Action 7, bit 16 of quantity u is entered into the SQG. If bit 16 of u is a ZERO (figure 2-32), no operation is performed at Action 8. At Actions 9 and 10, quantities b^o and 100000 are cycled simultaneously (OR'd) and entered into B (as at instants 20 and 25 of figure 2-27), and the new bit 16 ($\overleftarrow{b}_{16}$) is also entered into the SQG. If bit 16 of u is a ONE (figure 2-33) quantity u is transferred to Q (as at instants 8 and 14 of figure 2-27). At Actions 9 and 10, only quantity b^o is cycled and its new bit 16 entered into the SQG. Action 11 depends on bit 16 of b (entered into the SQG at Action 10) and on bit 16 of 1 (entered at Action 5). If $\overleftarrow{b}_{16} = 0$, indicating that the division operation has not been completed yet (as at instants 9, 15, and 20 of figure 2-27), control pulse ST1 is generated in order to execute-next subinstruction DV1 again. The contents of registers Q and B after the first execution of DV1 may be called q^1 and b^1; after the second execution q^2 and b^2, etc. After fourteen executions of DV1, quantity b^{14} will contain a ONE in bit position 16 (indicating that the division operation has been completed as at instant 25) and control pulse ST2 causes the SQG to execute-next

TABLE 2-5

CONTENTS OF REGISTERS AT END OF DV0

INITIAL CONDITIONS	$c(A) = e^\circ$	$c(Q) = q^\circ$	$c(LP) = \ell^\circ$	$c(B) = b^\circ$	c(Z)
a POSITIVE e POSITIVE	e	$\overline{a}$	000001	000001	z
a POSITIVE e NEGATIVE	$\overline{e}$	$\overline{a}$	140000	000001	z
a NEGATIVE e POSITIVE	e	a	140001	000001	z
a NEGATIVE e NEGATIVE	$\overline{e}$	a	000001	000001	z

subinstruction STD2. As DV1 is executed the final time the quotient has to be transferred from C to A if $\lambda_{16}^{14} = 0$, or from B to A if $\lambda_{16}^{14} = 1$. Subinstruction STD2, executed right after the final DV1 subinstruction, increments "next" address z by one and initiates the execution of the subsequent instruction.

2-78. SPECIAL CASES OF REGULAR INSTRUCTIONS

2-79. GENERAL

2-80. Paragraphs 2-6 through 2-77 deal with normal cases, i.e. instructions in which relevant address K refers to a location in the F or E memory. Instructions in which address K refers to flip-flop register A, Q, Z, or LP are discussed in the following paragraphs together with other special cases of NDX K instructions. For selection of the addressed flip-flop register and the operation of subinstruction STD2, see figure 2-2.

2-81. SPECIAL CASES OF BASIC INSTRUCTIONS

2-82. The special cases of instruction TC K are:

(1) TC A = XAQ which means: execute the instructions contained in A.

If c(A) = 000000, this is a program trap.

If A contains a TC K instruction, the transfer of control will be executed.

If A contains an instruction other than TC K, the instruction contained in Q will be executed after the instruction contained in A.

(2) TC Q = RETURN which means: return program control to the instruction entered into Q at the time the last TC K instruction was executed. Normally, Q contains a TC K instruction as explained in paragraph 2-17. (After the execction of this TC K instruction Q contains TC Z.) If Q

contains an instruction other than TC K, then TC Q is not a useful operation.

(3) TC Z which is not a useful operation.

(4) TC LP which transfers control to LP, $c(LP) = b^e(LP)$.

2-83. The special cases of instruction XCH K are:

(1) XCH A = NOOP: the instruction to be executed next is taken from $(L + 1)$, but $c(Q)$ is preserved. TC $(L + 1)$ is faster but changes $c(Q)$.

(2) XCH Z which transfers program control to the instruction which is located at the address contained in A. Similar to TS Z = TCAA.

(3) XCH Q which replaces the return address.

(4) XCH LP which results in $c(LP) = b^e(A)$.

2-84. The special cases of instruction CS K are:

(1) CS A = COM which means: complement contents of A, $c(A) = \overline{b}(A)$.

(2) CS Q which results in $c(A) = \overline{b}(Q)$ and $c(Q) = b(Q)$.

(3) CS Z which results in $c(A) = \overline{b}(Z)$ and $c(Z) = b(Z) + 1$.

(4) CS LP which results in $c(A) = \overline{b}(LP)$ and $c(LP) = b^e(LP)$.

2-85. The special cases of instruction TS K are:

(1) TS A = OVSK which means: overflow skip.

If A contains no overflow or underflow, the instruction at $z = L + 1$ is executed next. If A contains overflow or underflow, the instruction at $(L + 2)$ is executed next. TS A is very useful when used before executing an XCH instruction, in order to prevent loss of an overflow bit.

(2) TS Q which enters a new return address.

(3) TS Z = TCAA which means: transfer program control to the instruction which is located at the address stored in A. Q does not contain the same return address which would be entered by a TC K instruction. The first TC K instruction following a TCAA must not be a subroutine call, unless Z contains a TC K instruction. In this case, XAQ (TC A) is as fast and preferred, unless it is desired to save c(Q).

(4) TS LP which results in c(LP) = b^e(A).

2-86. The special cases of instruction MSK K are:

(1) MSK A = NOOP (No Operation).

(2) MSK Q which results in c(Q) = b(A) AND c(Q) and c(Q) = b(Q).

(3) MSK Z which results in c(A) = b(A) AND b(Z) and c(Z) = b(Z) + 1. This is not a useful operation.

(4) MSK LP which results in c(A) = b(A) AND b(LP) and c(LP) = b(LP). (No editing of b(LP) occurs.)

2-87. The special cases of instruction AD K are:

(1) AD A = DOUBLE which results in c(A) = 2b(A). In case of overflow or underflow, OVCTR is incremented or decremented.

(2) AD Q which results in c(A) = b(A) + c(Q) and c(Q) = b(Q).

(3) AD Z which results in c(A) = b(A) + b(Z) and c(Z) = b(Z) + 1.

(4) AD LP which results in c(A) = b(A) + b(LP) and c(LP) = b^e(LP).

2-88. The special cases of instruction NDX K are:

(1) NDX A which results in c(B) = c(z) + c(A) with z = L + 1; c(A) = b(A). Register B contains the instruction executed next.

(2) NDX Q which results in c(B) = c(z) + c(Q) with z = L + 1; c(Q) = b(Q).

(3) NDX Z which results in $c(B) = 2b(Z) + 1$ with $z = L + 1$; $c(Z) = b(Z) + 1$.

(4) NDX LP which results in $c(B) = c(z) + b(LP)$ with $z = L + 1$; $c(LP) = b^e(LP)$.

(5) NDX 5777 = EXTEND which is explained in paragraph 2-49.

(6) NDX 0025 = RESUME which is explained in paragraph 2-102.

(7) NDX 0017 = INHINT which is explained in paragraph 2-97

(8) NDX 0016 = RELINT which is explained in paragraph 2-97

2-89. The special cases of instruction CCS K are:

(1) CCS A which is very useful and very similar to CCS K as described in paragraph 2-40.

(2) CCS Q, and CCS LP which might be useful; CCS Z is not a useful operation.

2-90. SPECIAL CASES OF EXTRA CODE INSTRUCTIONS

2-91. The special cases of instruction SU K are:

SU A which puts 177777 into A but to do this takes four MCTs.

SU Q, SU Z, and SU LP which are similar to AD Q, AD Z, and AD LP except that complemented quantities are added.

2-92. The special cases of instruction MP K are:

MP A = SQUARE which results in $c(A, LP) = |b(A)|^2$.

MP Q which results in $c(A, LP) = b(A) \cdot c(Q)$ and $c(Q) = b(Q)$.

MP Z which results in $c(A, LP) = b(A) \cdot b(Z)$ and $c(Z) = b(Z) + 1$. This is not a useful operation.

MP LP which results in $c(A, LP) = b(A) \cdot c(LP)$ and $c(LP) = b(LP)$.

2-93. The special cases of instruction DV K are:

DV A which puts 037777 into A but to do this takes eighteen MCTs.

This is a useful test loop.

DV Q, which results in c(A) = (A) = b(A) ÷ b(Q) and c(Q) = R1.

DV Z, which results in c(A) = b(A) ÷ b(Z), c(Z) = b(Z) + 1.

This is not a useful operation.

DV LP, which results in c(A) = b(A) ÷ c(LP) and c(Q) + R1.

2-94. INVOLUNTARY INSTRUCTIONS

2-95. GENERAL

2-96. The Involuntary Instructions can be divided into two groups: Priority Program Instructions (RPT and RSM) and Counter Instructions (PINC, MINC, SHINC, and SHANC). The interruption of a current program section in favor of a program section of higher priority was discussed in paragraphs 15-47 through 15-49. There it was stated that certain inputs of the AGC are connected to the Interrupt Priority Control. At the time a signal arrives at one of these inputs (and no input signal of higher priority is present), the Interrupt Priority Control does two things: (1) it signals the SQG to execute-next instruction RPT, and (2) it provides address 2000, 2004, 2010, 2014, 2020, or 2024 (called RPT Transfer Routines, table 2-6) for initiating the execution of the requested program section. Execution of instruction RPT causes the last contents of registers Z and B to be transferred to register ZRUPT and BRUPT in the E memory (table 15-5) and program control to be transferred to one of the six Transfer Routines. No matter to which of the six routines control is transferred, the contents of A and Q are first transferred to register ARUPT and QRUPT in E memory. Thereafter program control is transferred (by a TC instruction) to routine T3RUPT, ERRUPT, T4RUPT, KEYRUPT, UPRUPT or DOWNRUPT (table 15-7). Each interrupting program section has the responsibility of returning the contents of register ARUPT and QRUPT to registers A and Q and of initiating instruction RSM. Execution of instruction RSM causes the contents of registers ZRUPT and BRUPT to be transferred to Z

TABLE 2-6

RPT and GO Transfer Routines [1]

Location	Content	Subroutine [2]	Order Code	Relevant Address
2000	5 0026 0	TIME3RPT	TS	ARUPT
2001	3 0001 0		XCH	Q
2002	3 0027 1		XCH	QRUPT
2003	0 2131 0		TC	T3RUPT
2004	5 0026 0	ERRORRPT	TS	ARUPT
2005	3 0001 0		XCH	Q
2006	3 0027 1		XCH	QRUPT
2007	0 2034 1		TC	ERRUPT
2010	5 0026 0	TIME4RPT	TS	ARUPT
2011	3 0001 0		XCH	Q
2012	3 0027 1		XCH	QRUPT
2013	0 2427 1		TC	T4RUPT
2014	5 0026 0	KEYMARPT	TS	ARUPT
2015	3 0001 0		XCH	Q
2016	3 0027 1		XCH	QRUPT
2017	0 2467 0		TC	KEYRUPT
2020	5 0026 0	UPLINRPT	TS	ARUPT
2021	3 0001 0		XCH	Q
2022	3 0027 1		XCH	QRUPT
2023	0 7300 0		TC	UPRUPT
2024	5 0026 0	DOWNLRPT	TS	ARUPT
2025	3 0000 1		XCH	Q
2026	3 0027 1		XCH	QRUPT
2027	0 2301 1		TC	DOWNRUPT
2030	2 0017 0	GO	INHINT	
2031	3 2066 0		CAF	PHASBANK
2032	5 0015 0		TS	BNK
2033	0 6774 0		TC	GOPROG
2034	0 2605 0	ERRUPT	TC	ALARM
2035	01101 0		OCT	01101
2036	0 2213 0		TC	NBRESUME

[1] As used in program SUNRSE33.

[2] Symbols not used in program listings.

and B and causes the execution of the interrupted program section to continue.

2-97. A program interruption (execution of RPT) may occur at the end of any instruction (not subinstruction) except when:

(1) An interrupting program is in progress.

(2) A program interruption has been inhibited by the execution of instruction INHINT (NDX 0017) and the inhibition has not been released by the execution of instruction RELINT (NDX 0016). By executing INHINT and RELINT, the selection logic gates the Program Interrupt Priority Control to prevent it from, or allow it to, send a command to the SQG.

(3) The next instruction to be executed is an Extra Code Instruction. In this case, register B contains a quantity with underflow, and control pulse WOVI, at Action 11 of subinstruction NDX1, prevents an interruption.

(4) The accumulator contains a quantity with overflow or underflow. In this case, control pulse WOVI is normally provided at Action 4, 10, or 11 to prevent program interruption.

2-98. Incrementing, decrementing, or shifting the content of a counter was discussed in paragraph 1-80, where it was stated that all incremental input signals of the AGC are first stored in the Priority Control. At the time the incremental data arrive (and if no incremental data of higher priority are present), the Counter Increment Priority Control does two things: (1) provides the address of the proper counter, and (2) it sends a signal to execute-next instruction PINC, MINC, SHINC, or SHANC. Dependent on the counter address provided, the increment or decrement is carried out with or without sign correction in case of overflow or underflow. Instructions PINC, MINC, SHINC, and SHANC can be executed after any Action 12.

2-99. INSTRUCTION RPT (Interrupt Program)

2-100. RPT means: transfer control to the interrupting program section and store enough information of the interrupted program section in order to be able to continue it later. The entire operation RPT can be formulated as follows:

(1) Set c(ZRUPT) = b(Z)
(2) Set c(BRUPT) = b(B)

} Remember that bit 15 of b(Z) and b(B) gets lost, and bit 16 is moved into position 15 of ZRUPT AND BRUPT, respectively.

(3) Execute-next the instruction located at the address provided by the Program Interrupt Priority Control.

(4) Inhibit interrupt until further notice. (This is accomplished by setting the Interrupt-in-Progress Flip-Flop in the SQG.)

(5) Reset Interrupt Priority Control.

The RPT instruction consists of three subinstructions: RPT1, RPT3, and STD2.

2-101. Figures 2-34 and 2-35 illustrate the execution of RPT. The current program section is to be interrupted and subroutine ERRUPT is to be executed immediately. At Action 1 of RPT1, address 0024 is entered into S, and the selection logic gates location 0024 for readout and write-in. At the same time, address 0024 is also entered into Y, and a one is added to it. At Action 3 register G is cleared as usual. At Action 9, the "next" address (z) of the program being interrupted is entered into G and transferred to location 0024 (ZRUPT) at Action 10. Also at Action 10, quantity 0025 is entered into Z. Control pulses ST1 and ST2 together cause the SQG to execute-next subinstruction RPT3. At Action 2, the address provided by the Priority Control (2004) is entered into Z. At Action 3, address 2004 is transferred to G and control pulse KRPT clears the Request Flip-Flop (in the Interrupt Priority Control) that initiated the program interrupt. At Action 9 the content of B is entered into G and transferred to

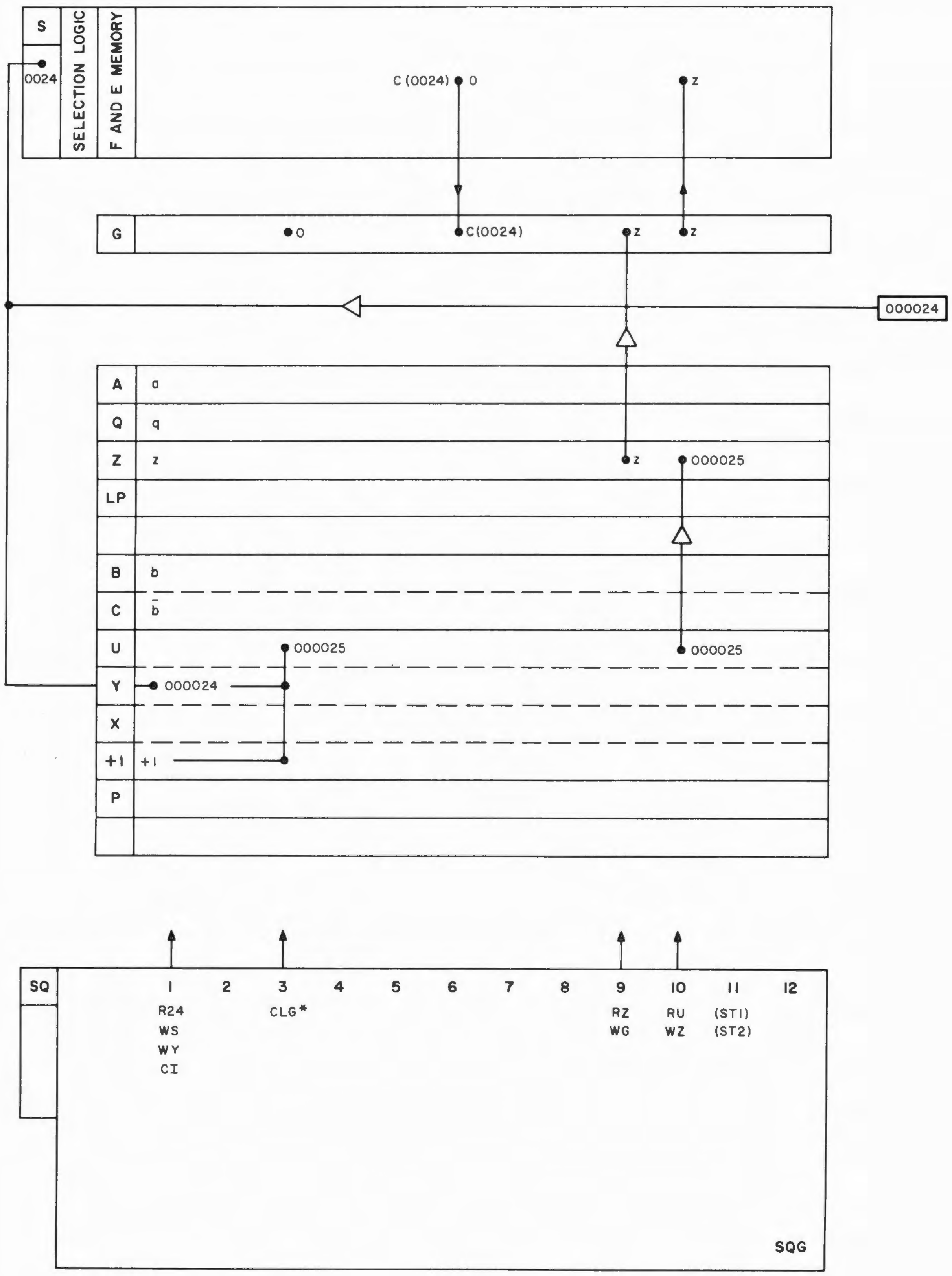

Figure 2-34. Subinstruction RPT1

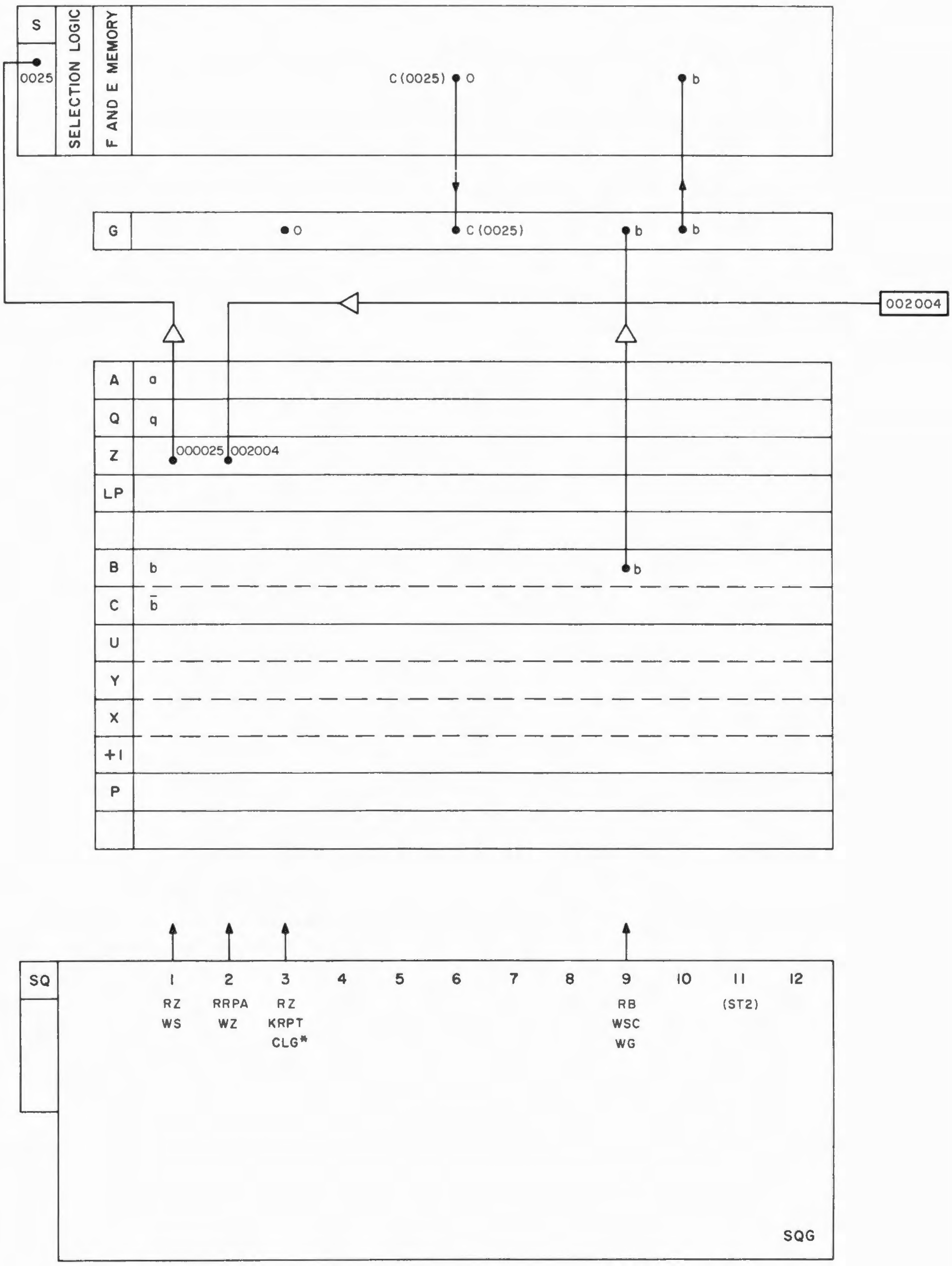

Figure 2-35. Subinstruction RPT 3

location 0025 at Action 10. Control pulse ST2 causes the SQG to execute-next subinstruction STD2, in order to increment address 2004, which is now the "next" address, and to initiate instruction TS ARUPT.

2-102. INSTRUCTION RSM (Resume Program, Order Code 2, K = 0025)

2-103. RSM means: resume the interrupted program section by entering into B and Z what was contained in B and Z at time of interruption. The entire operation RPT can be formulated as follows:

(1) Set c(B) = c(BRUPT).

(2) Set c(Z) = c(ZRUPT).

(3) Execute-next the instruction now contained in B.

(4) Release inhibition of program interruption. (This is accomplished by resetting the Interrupt-in-Progress Flip-Flop in the SQG.)

The RSM instruction consists of two subinstructions: NDX0 and RSM.

2-104. Figure 2-36 illustrates the execution of instruction RSM = NDX 0025. As subinstruction NDX0 is executed, the content of location 0025 (the instruction stored away by instruction RPT) is returned to B. At Action 10, control pulse TRSM causes the SQG to execute-next subinstruction RSM because c(S) = 0025. At Action 1 of RSM, address 0024 is entered into S and the selection logic gates location 0024, containing the "next" address (z) of the interrupted program, for readout and write-in. At Action 3, register G is cleared, as usual. At Action 7, address z is returned to Z. At Action 12, the order code (OC) is entered into register SQ and the execution of the instruction contained in B is initiated.

2-105. INSTRUCTION PINC (Increment Content of Addressed Counter)

2-106. PINC means: increment by one the quantity contained in that counter of which the address is provided by the Increment Priority Control. The entire

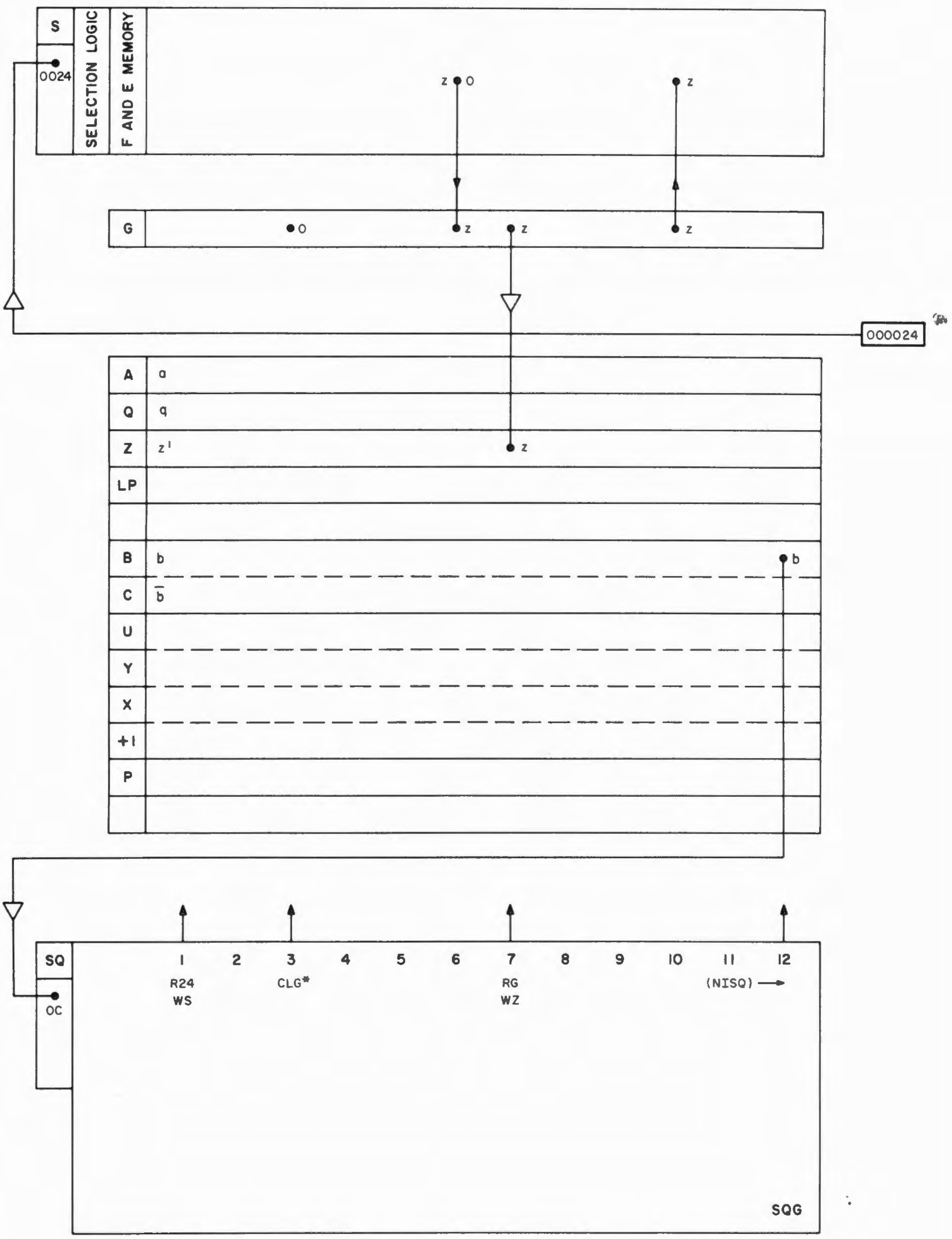

Figure 2-36. Subinstruction RSM

operation of PINC can be formulated as follows:

(1) Enter into S the address provided by the Increment Priority Control.

(2) Set c(CTR) = b(CTR) + 000001.

(3) In case of overflow, send signal to Priority Control (for operation carried out after overflow, see table 1-5) or reverse sign if CTR=41 or $0047 \leq CTR \leq 0056$.

(4) Reset Counter Priority Control.

The PINC instruction consists of only one subinstruction, PINC.

2-107. Figure 2-37 illustrates the execution of PINC which increments the content of counter 0044. At Action 1, address 0044, provided by the Increment Priority Control is entered into register S, and the selection logic gates location 0044 for readout and write-in. At Action 3, register G is cleared, as usual. At Action 4, quantity 000001 is entered into Y. At Action 6, quantity e of the addressed counter is transferred to X and P. At Action 7, an alarm is caused in case of incorrect parity. At Action 8, register P is cleared. At Action 9, register G is cleared again, and the sum (e + 1) is written into P. At Action 10, control pulse RU* reads the incremented quantity (e + 1) into the WA's, and control pulse WG* writes the contents of the WA's and the parity bit generated for quantity (e + 1) into register G. The complete word $(e + 1)_{15-0}$ is returned to location 0044 at Action 10. Control pulse WOVR transfers bits 16 and 15 of (e + 1) into the SQG for test of overflow or underflow and resets the Counter Priority Control. In case of overflow, a signal is sent to the priority inputs.

2-108. INSTRUCTION MINC (Decrement Content of Addressed Counter)

2-109. MINC means: decrement by one the quantity contained in that counter of which the address is provided by the Increment Priority Control. The entire operation MINC can be formulated as follows:

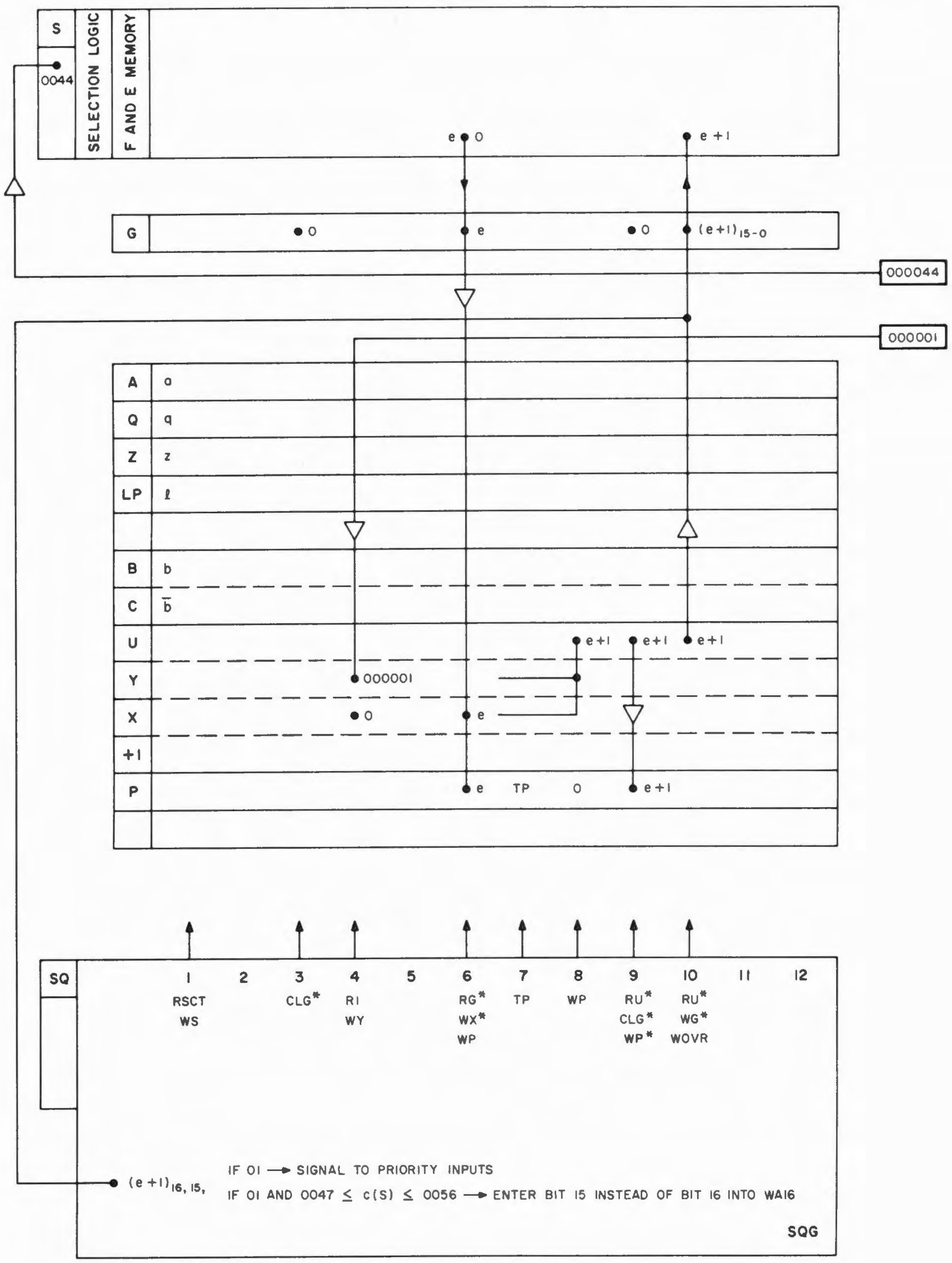

Figure 2-37. Subinstruction PINC

(1) Enter into S the address provided by the Increment Priority Control.

(2) Set c(CTR) = b(CTR) + 177776.

(3) In case of underflow, send signal to Priority Control (for operation carried out after underflow, see table 1-5) or reverse sign if CTR = 41 or $0047 \leq CTR \leq 0056$.

(4) Reset Counter Priority Control.

The MINC instruction consists of only one subinstruction, MINC.

2-110. Figure 2-38 illustrates the execution of MINC which decrements the content of counter 0044. Figure 2-38 is very similar to figure 2-37 except that control pulse R1 at Action 4 is replaced by control pulse R1C, which reads quantity 177776 into Y. The decremented quantity is (e - 1). In case of underflow of quantity (e - 1), a signal is sent to the Counter Priority Control.

2-111. INSTRUCTION SHINC (Shift Content of Addressed Counter)

2-112. SHINC means: shift one position to the left the quantity contained in that counter of which the address is provided by the Increment Priority Control. Instructions SHINC and SHANC are used to transform serial uplink and radar range data into parallel data. The entire operation SHINC can be formulated as follows:

(1) Enter into S the address provided by the Increment Priority Control

(2) Set c(CTR) = 2b(CTR) = b(CTR).

(3) If bit 15 of the quantity located at 0041 or 0056 is a ONE, send a signal to the Program Interrupt Control at the time the quantity is transferred into the Adder to initiate the UPRUPT program.

(4) In case of overflow, reverse bit 16 (as for addition of angular data) and prevent end-around carry.

(5) Reset Counter Priority Control.

The SHINC instruction consists of only one subinstruction, SHINC.

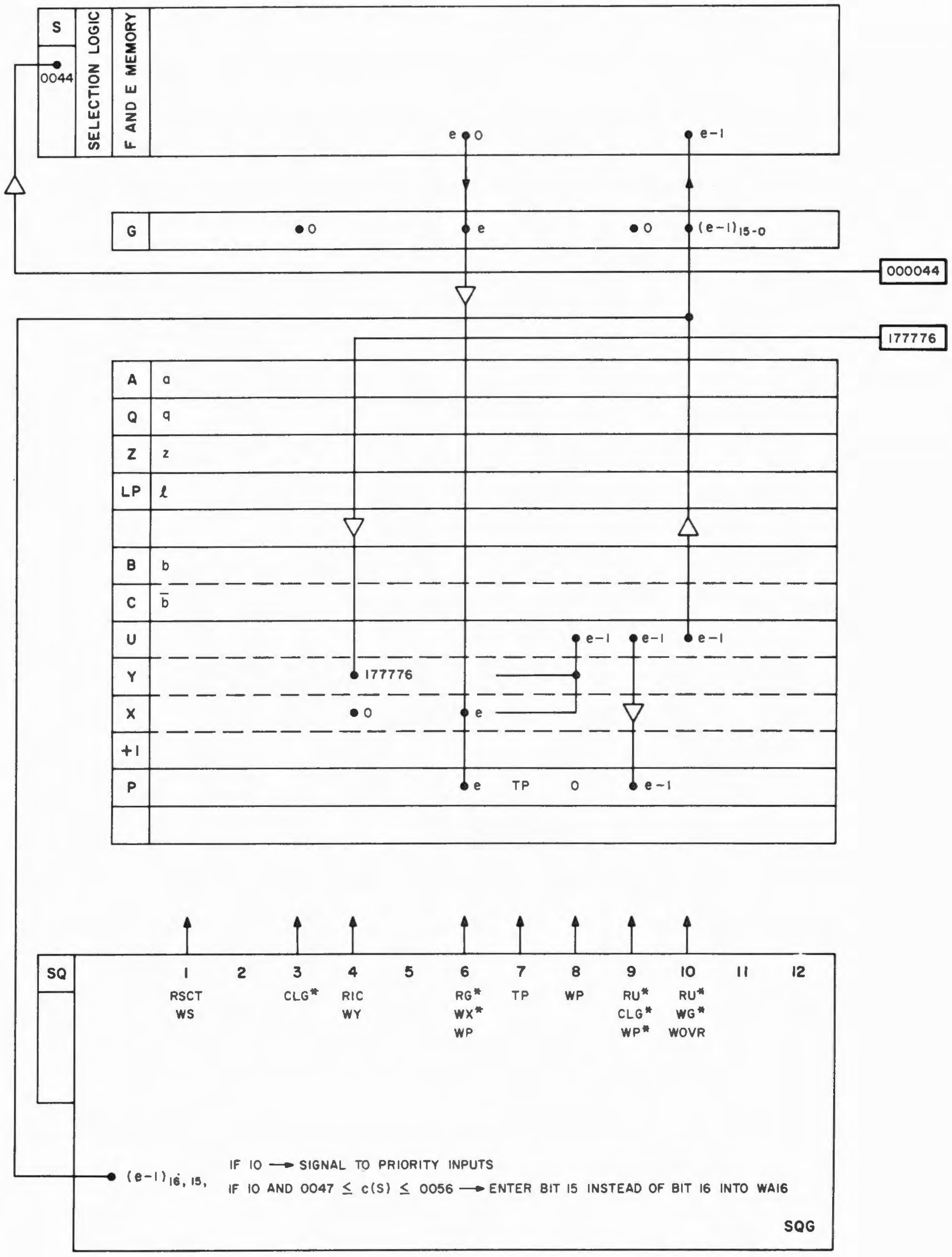

Figure 2-38. Subinstruction MINC

2-113. Figure 2-39 illustrates the execution of SHINC which shifts the content of location 0041 one position to the left. At Action 1, address 0041, provided by the Increment Priority Circuit, is entered into S and the selection logic gates location 0041 for readout and write-in. At Action 3, register G is cleared, as usual. At Action 4, registers X and Y are cleared. At Action 6, quantity e of the addressed counter is transferred to Y, X, and P and the sign bit is entered into the SQG. If bit position 16 of Y or X (i.e., bit position 15 of location 0041) contains a ONE, a signal is sent to the Program Priority Circuit to initiate the UPRUPT program. At Action 7, an alarm is caused in case of incorrect parity. At Action 8, register P is cleared. At Action 9, register G is cleared again and the sum (e + e) = 2e = $\overleftarrow{e}$ is written into P. At Action 10, control pulse RU* reads the shifted quantity ($\overleftarrow{e}$) into the WAs and control pulse WG* writes the contents of the WAs and the parity bit generated for quantity $\overleftarrow{e}$ into register G. Also at Action 10, the complete word $\overleftarrow{e}_{15-0}$ is returned to location 0041. Control pulse WOVR causes reversal of sign in case of overflow.

2-114. Figure 2-40 demonstrates how the completion of an uplink word is signalled. In the example shown, several bits of the uplink word have already been received and the flag bit (the first bit of a word received via uplink) has moved into bit position 14 of location 0041. The flag bit is used to indicate the completion of the uplink word. At instant 1, the flag bit ($\underline{1}$) is located at bit position 14 and bit positions 13 through 1 contain data. At instant 2, the quantity located at 0041 is entered into both Y and X, and U contains the sum with corrected sign (reversed sign) at instant 3. At instant 4, the shifted quantity is transferred from U to location 0041 and the flag bit is now contained in position 15. By shifting the content of location 0041 once more at instants 5, 6, and 7 the data moves into bit positions 15 through 3 and the flag bit gets lost.

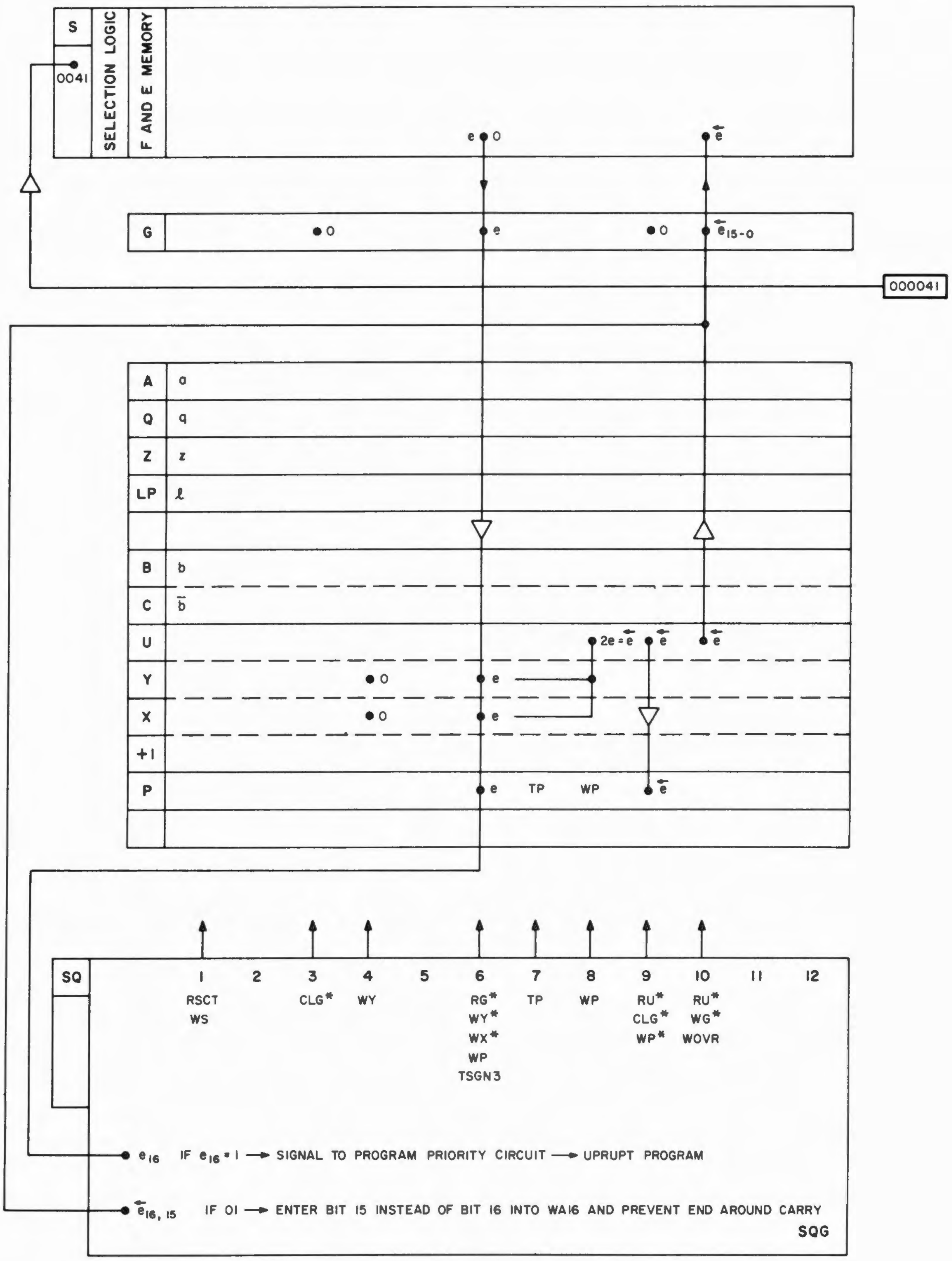
S
0041
SELECTION LOGIC
F AND E MEMORY
e 0
e
G
0
e
0
$\overleftarrow{e}_{15-0}$
000041
A a
Q q
Z z
LP ℓ
B b
C $\bar{b}$
U
2e = $\overleftarrow{e}$
$\overleftarrow{e}$
Y 0 e
X 0 e
+1
P e TP WP $\overleftarrow{e}$
SQ
1 RSCT WS
2
3 CLG*
4 WY
5
6 RG* WY* WX* WP TSGN3
7 TP
8 WP
9 RU* CLG* WP*
10 RU* WG* WOVR
11
12
e_{16} IF e_{16} = 1 → SIGNAL TO PROGRAM PRIORITY CIRCUIT → UPRUPT PROGRAM
$\overleftarrow{e}_{16,15}$ IF 01 → ENTER BIT 15 INSTEAD OF BIT 16 INTO WA16 AND PREVENT END AROUND CARRY
SQG

Figure 2-39. Subinstruction SHINC

INSTANT	FUNCTION	LOCATION 0041															
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0041		0	$\underline{1}$	1	0	1	1	0	1	1	1	0	1	1	1	1
2	0041 → Y, X	0	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1
	NORMAL SUM	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	0
3	U	1	1	1	0	1	1	0	1	1	1	0	1	1	1	1	0
4	0041		$\underline{1}$	1	0	1	1	0	1	1	1	0	1	1	1	1	0
5	0041 → Y, X	$\underline{1}$	$\underline{1}$	1	0	1	1	0	1	1	1	0	1	1	1	1	0
	NORMAL SUM	1	1	0	1	1	0	1	1	1	0	1	1	1	1	0	1
6	U	1	1	0	1	1	0	1	1	1	0	1	1	1	1	0	0
7	0041		1	0	1	1	0	1	1	1	0	1	1	1	1	0	0

Figure 2-40. Completion of an Uplink Word

2-115. INSTRUCTION SHANC (Shift Content of Addressed Counter and Add One).

2-116. SHANC means: shift one position to the left the quantity contained in that counter of which the address is provided by the Increment Priority Control, and add one to it. Instructions SHINC and SHANC are used to transform serial uplink and radar range data into parallel data for use by the computer. The entire operation SHANC can be formulated as follows:

(1) Enter into S the address provided by the Increment Priority Control.

(2) Set c(CTR) = 2b(CTR) + 1 = $\overleftarrow{b}$(CTR) + 1.

(3) If bit position 15 of location 0041 or 0056 contains a ONE, send a signal to the Program Interrupt Control to initiate the UPRUPT program.

(4) In case of overflow or underflow, reverse bit 16 (as for addition of angular data) and prevent end-around carry.

The SHANC instruction consists of only one subinstruction, SHANC.

2-117. Figure 2-41 illustrates the execution of SHANC which shifts the content of location 0041 one position to the left and enters a ONE into the new bit position 1. Figure 2-41 is very similar to figure 2-39 except that control pulse CI is added to Action 7 to add a one to the shifted quantity.

2-118. <u>MISCELLANEOUS INSTRUCTIONS</u>

2-119. GENERAL

2-120. Miscellaneous instructions can be divided into two groups: Start Instructions (GO and TCSA) and Display and Load Instructions (OINC and LINC).

2-121. INSTRUCTION GO (Computer GO)

2-122. GO means: get computer operating by executing-first the instruction

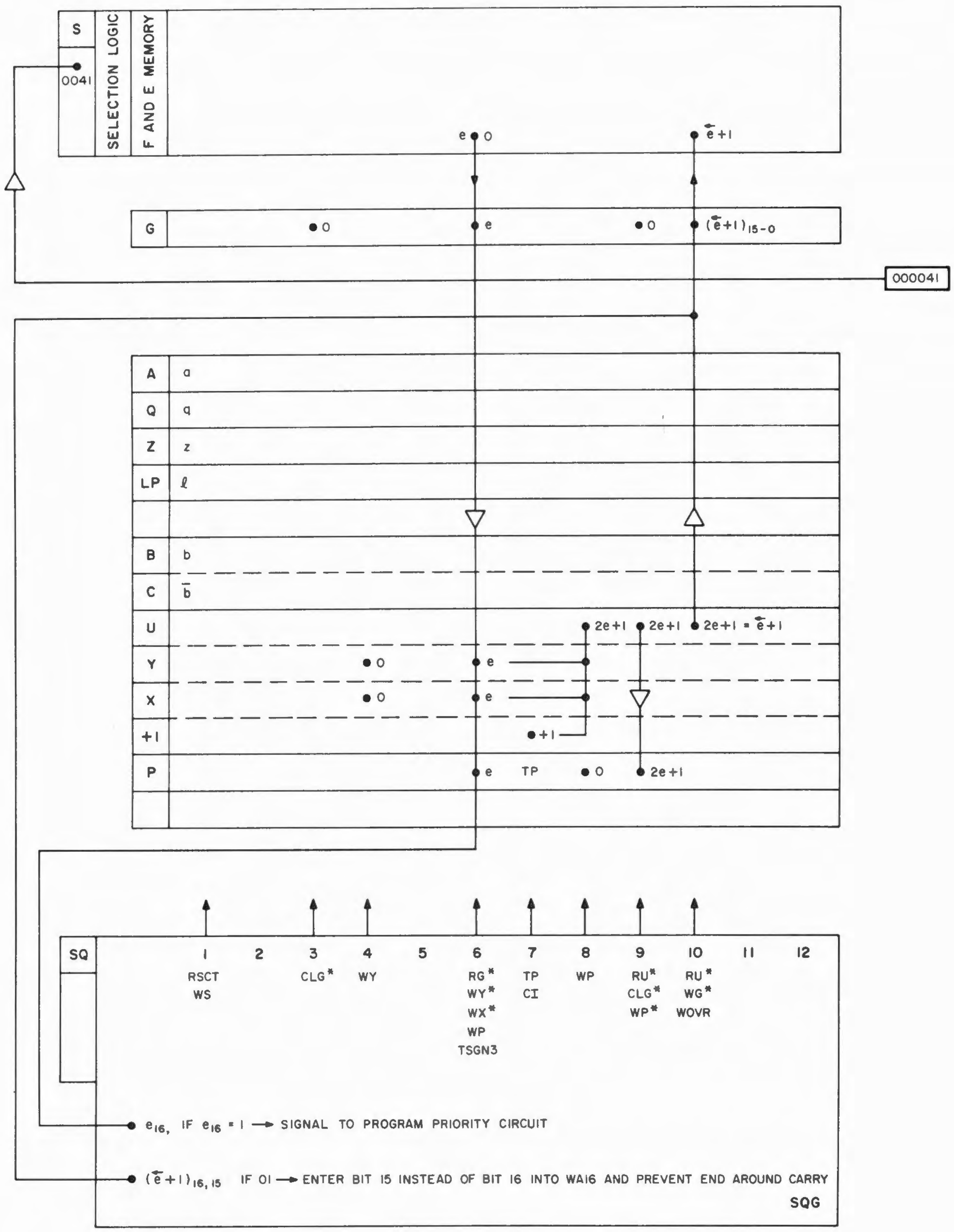
S
0041
SELECTION LOGIC
F AND E MEMORY
e
0
$\overleftarrow{e}+1$
G
0
e
0
$(\overleftarrow{e}+1)_{15-0}$
000041
A a
Q q
Z z
LP ℓ
B b
C $\overline{b}$
U
2e+1
2e+1
2e+1 = $\overleftarrow{e}+1$
Y
0
e
X
0
e
+1
+1
P
e
TP
0
2e+1
SQ
1 RSCT WS
2
3 CLG*
4 WY
5
6 RG* WY* WX* WP TSGN3
7 TP CI
8 WP
9 RU* CLG* WP*
10 RU* WG* WOVR
11
12
e_{16}, IF e_{16} = 1 → SIGNAL TO PROGRAM PRIORITY CIRCUIT
$(\overleftarrow{e}+1)_{16,15}$ IF 01 → ENTER BIT 15 INSTEAD OF BIT 16 INTO WA16 AND PREVENT END AROUND CARRY
SQG

Figure 2-41. Subinstruction SHANC

situated at location 02030 (table 2-6). Instruction GO is identical with instruction TC (figure 2-3) except that control pulse RB at Action 1 is replaced by RSTRT (Read STRT). Pulse RSTRT transfers start address 02030 into register S, and the selection logic gates location 02030 start for readout. After the execution of GO the instruction stored at location 02030 is contained in B and its order code is in register SQ.

2-123. INSTRUCTION TCSA (Start at Specified Address)

2-124. TCSA means: start the execution of a program by executing-first the instruction located at SA where SA might be any Specified Address entered into the computer by means of the Computer Test Set. Instruction TCSA is identical with GO except that, at Action 1, RSA replaces RB instead of RSTRT.

2-125. INSTRUCTION OINC (Display Content of Addressed Location)

2-126. OINC stands for zero increment and means: read and display the content of the addressed location. Instruction OINC is used in conjunction with the Computer Test Set. By punching an address into the keyboard and pressing a READ button the content of the addressed location is displayed. Figure 2-42 illustrates the execution of OINC for display of data e located at address 1.

2-127. INSTRUCTION LINC (Load Addressed Location)

2-128. LINC stands for load increment and means: write into the addressed location the data punched into the keyboard. Instruction LINC is also used in conjunction with the Computer Test Set. By punching in both an address and data into the keyboard, then pressing a LOAD button, the data will be entered into the location selected. Figure 2-43 illustrates the execution of LINC for data d to be written into location 1.

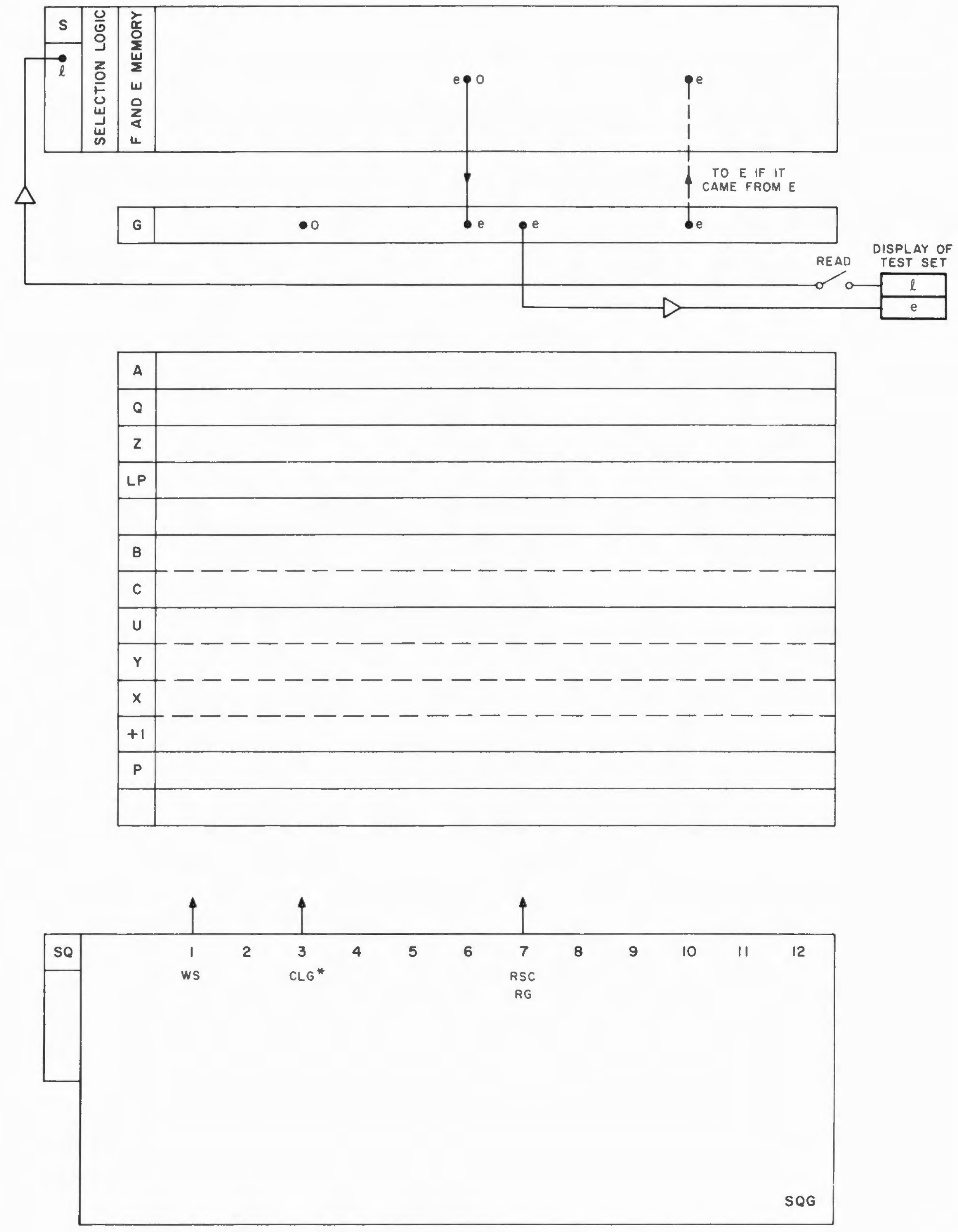

Figure 2-42. Subinstruction OINC

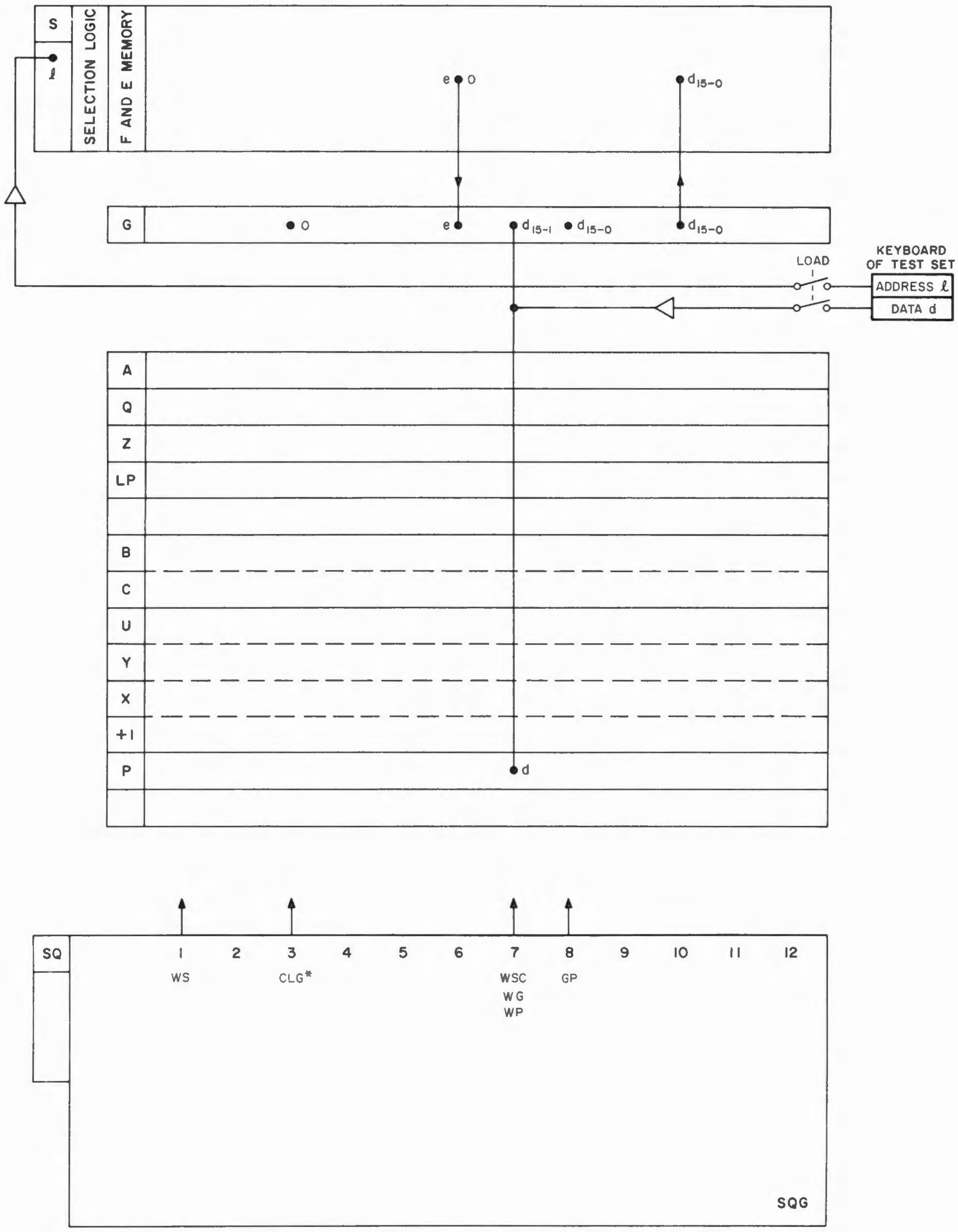

Figure 2-43. Subinstruction LINC