channel and main storage, it does not pass through the R register. Refer to the Field Engineering Manual of Instruction, IBM 2030 I/O Control, Form Y24-3362.)

Notice in Figure 1-8 that the MN-bus (really the M and N busses--eight information bits plus one parity bit for each bus) provides input paths to the MN-registers. The following table summarizes the address information source inputs to the MN-registers:

Source Register	Destination	Usual Immediate Source of
I	M	High order address bits for an instruction byte
J	N	Low order address bits for an instruction byte
U	M	High order address bits for a data byte
V	N	Low order address bits for a data byte
T	N	Address bits for certain auxiliary storage locations

Addresses are frequently obtained from instructions which are in main storage. Hence, there must be a path, during normal instruction processing, over which these addresses can be set into the U, V, I, J, or T registers. One path is from storage, to the R register, through the A-register inputs to ALU, through ALU to the Z bus, and from there to the appropriate register (Figure 1-8). This description is not meant to imply that every time a byte is sent from storage it follows the path just described into all registers. The microprogram specifies which registers are to take part in the operation, and, as already pointed out, the microprogram steps used depend upon the operation being performed.

During instruction processing, the G-register usually contains the instruction operation code. Hence, the values of the bit positions of this register indicate such items as instruction length and format.

Many other registers are used. However, how a register is used is mainly dependent upon the operation performed. The following table summarizes the usual functions of some important registers in the data flow (Figure 1-8):

Register	Usual Function
I	Instruction address (high-order bits)
J	Instruction address (low-order bits)
U	Data address (high-order bits)
V	Data address (low-order bits)
L	Data length
T	Auxiliary storage address
D	General purpose data register
R	Storage data register
S	Status (CPU)
G	Instruction operation code
H	Priority status register
Q	Storage-Protection key in PSW (High 4 bits); Storage-Protection key of block of storage just used (low 4-bits)
C	Interval Timer Count
F	External Interrupt: Interval Six direct-control interruptions (bits 2 through 7).

The W and X registers hold information that is used to address ROS. A maximum of 13 bits are needed to address any ROS word. The W register holds the 5 high-order bits and the X register holds the 8 low-order bits. (Note that the W register has only five bit positions which are W3, W4, W5, W6, and W7.) In addition, each of these registers has a parity bit position.

The FW-FX and GW-GX registers are backup registers for ROS addresses. The FW-FX registers are used to retain the ROS address just held by the WX registers when certain multiplexor channel operations break into CPU instruction processing. The GW-GX registers provide backup for addresses in WX when selector channel one requires use of ROS (such as in chaining operations). A similar set of registers (HW-HX) is used during ROS operations for selector channel two. For detailed information about the WX registers, refer to Chapter 2 of this publication. Multiplexor and selector channel operations and register usage) are described in Field Engineering Manual of Instruction, IBM System/360 Model 30, 2030 I/O Control, Form Y24-3362.

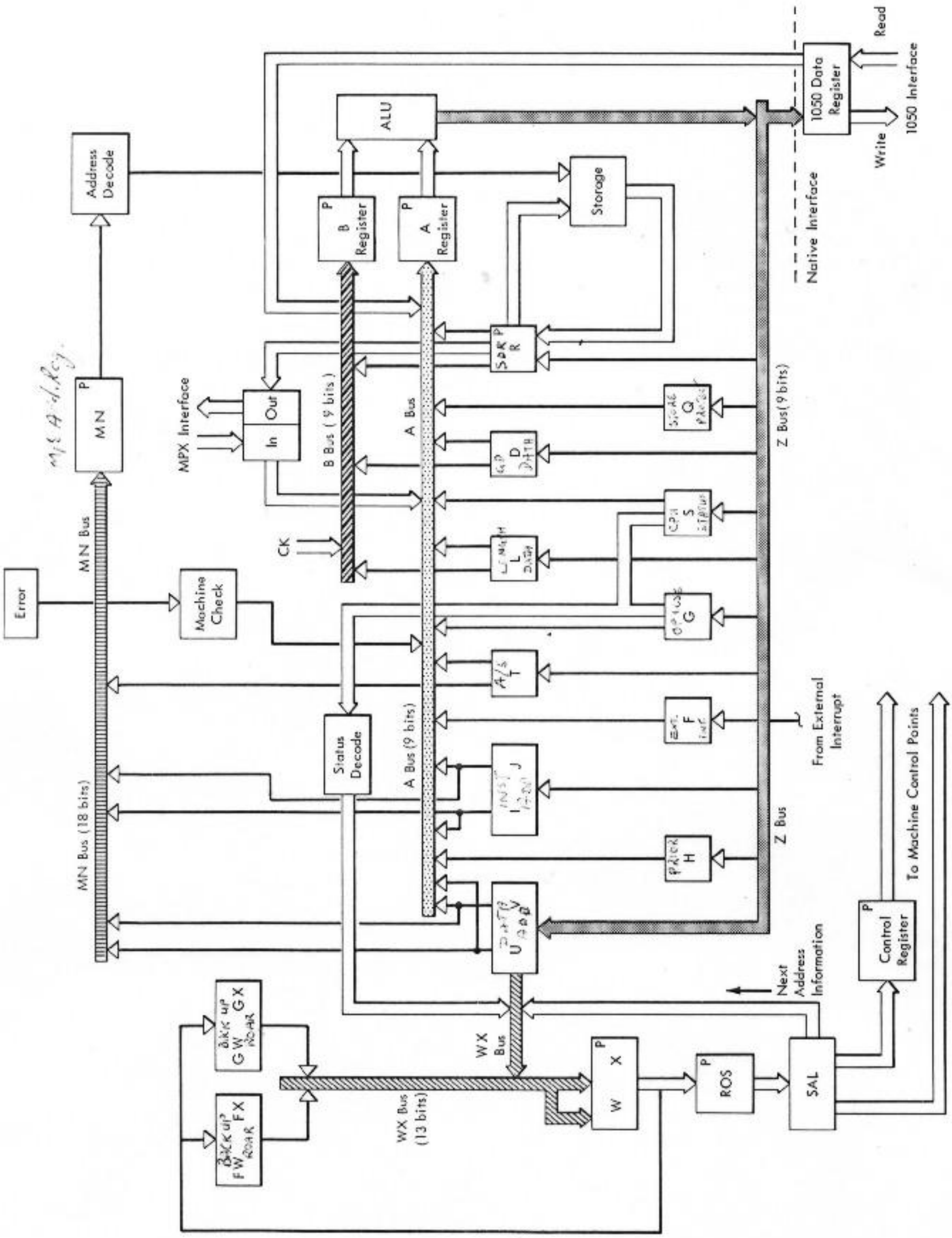


Figure 1-8. 2030 Basic Data Flow

Functional Units

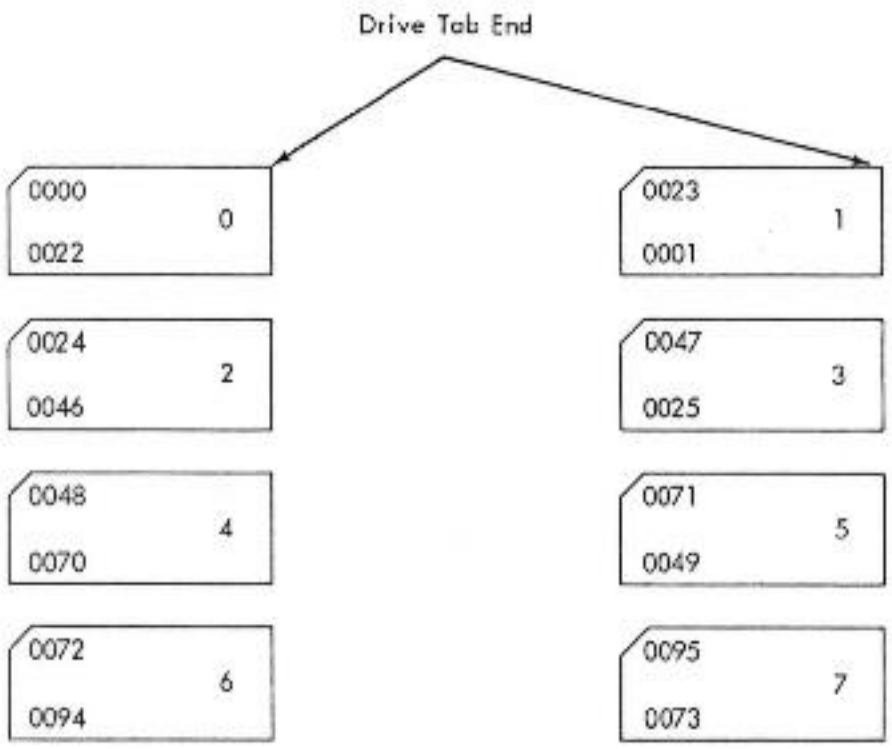


Figure 2-37. ROS Word Numbering

MICROPROGRAM

- The microprogram is used to control the function of the 2030.
- Each machine cycle is controlled by one microprogram word.
- A microprogram word is punched in a ROS card and becomes a ROS word.

In all computers it is necessary to have some method to perform a sequence of logical steps. The 2030 uses a microprogram. Within the microprogram, the microprogram word is the functional statement. The microprogram word is punched in a ROS card to form a ROS word.

A ROS word is selected by the decode of the address in ROAR (Read Only Address Register) and the ROS word contents are decoded to activate control points in the system. The ROS word consists of specific fields programmed to perform a logic statement. The activated word sends back part of the next address for ROS to ROAR. Coupled with branch control (machine status test), the partial address forms the complete address of the next ROS word. To read and understand the ROS word, we must know what the ROS word can contain and what format is used to write the word.

ROS Word Control Fields

- The ROS word used in the 2030 is 60 bits wide.
- The ROS word is divided into control fields.

The 60-bit ROS word is divided into control fields (Figure 2-38) and these fields can be separated into six broad groups:

1. Function control CA, CF, CB, CG, CC, CV, CD, CK
2. Main and auxiliary storage CM, CU
3. Branching and ROS address CN, CH, CL
4. Set and reset of status condition CS
5. Alternate AA, AS, AK
6. Parity for different sections of the control fields

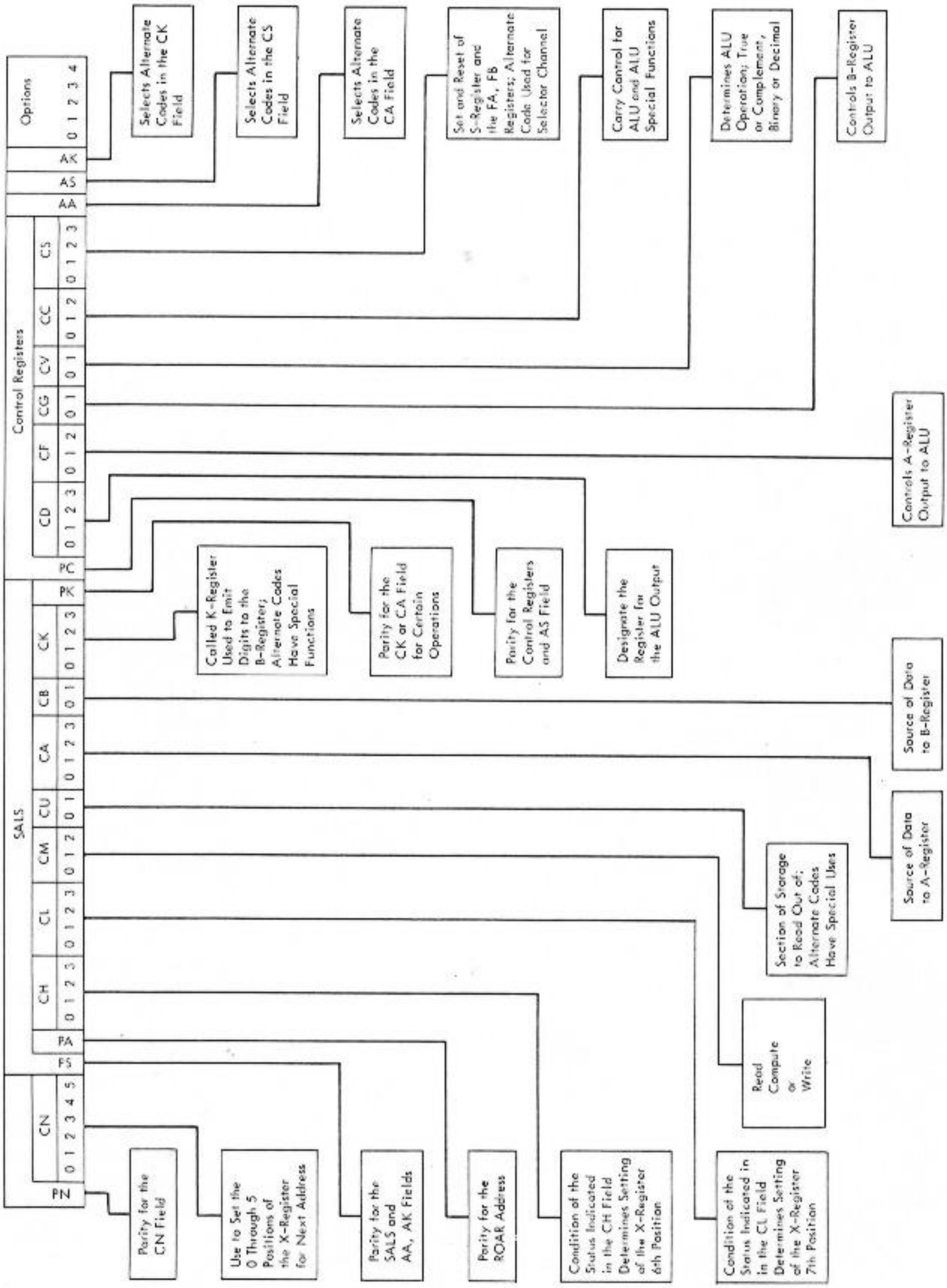


Figure 2-38. ROS Control Fields

Notice the control fields vary in numbers of bit positions. Example: the CU field is two bits wide and the CD field is four bits wide. If the field is two bits wide, we can set and decode four combinations: 0-00, 1-01, 2-10, 3-11. A three position field can be set and decoded in eight combinations, 0-000 through 7-111, and a 4-bit field has 16 combinations, 0-0000 through F-1111.

FUNCTION CONTROL. The function control fields (Figure 2-39) are used to control all data movement in the CPU and the ALU. ALL DATA MOVEMENT IS THROUGH THE ALU. The function control fields can be subdivided into four groups.

1. Source to the A-register and control of the A-register output to the ALU; CA, CF.
2. Source to the B-register and control of the B-register output to the ALU; CB, CK, CG.
3. Function and control of the ALU; CV, CC.
4. Destination of the ALU output; CD

Source to the A-register (CA): This 4-bit

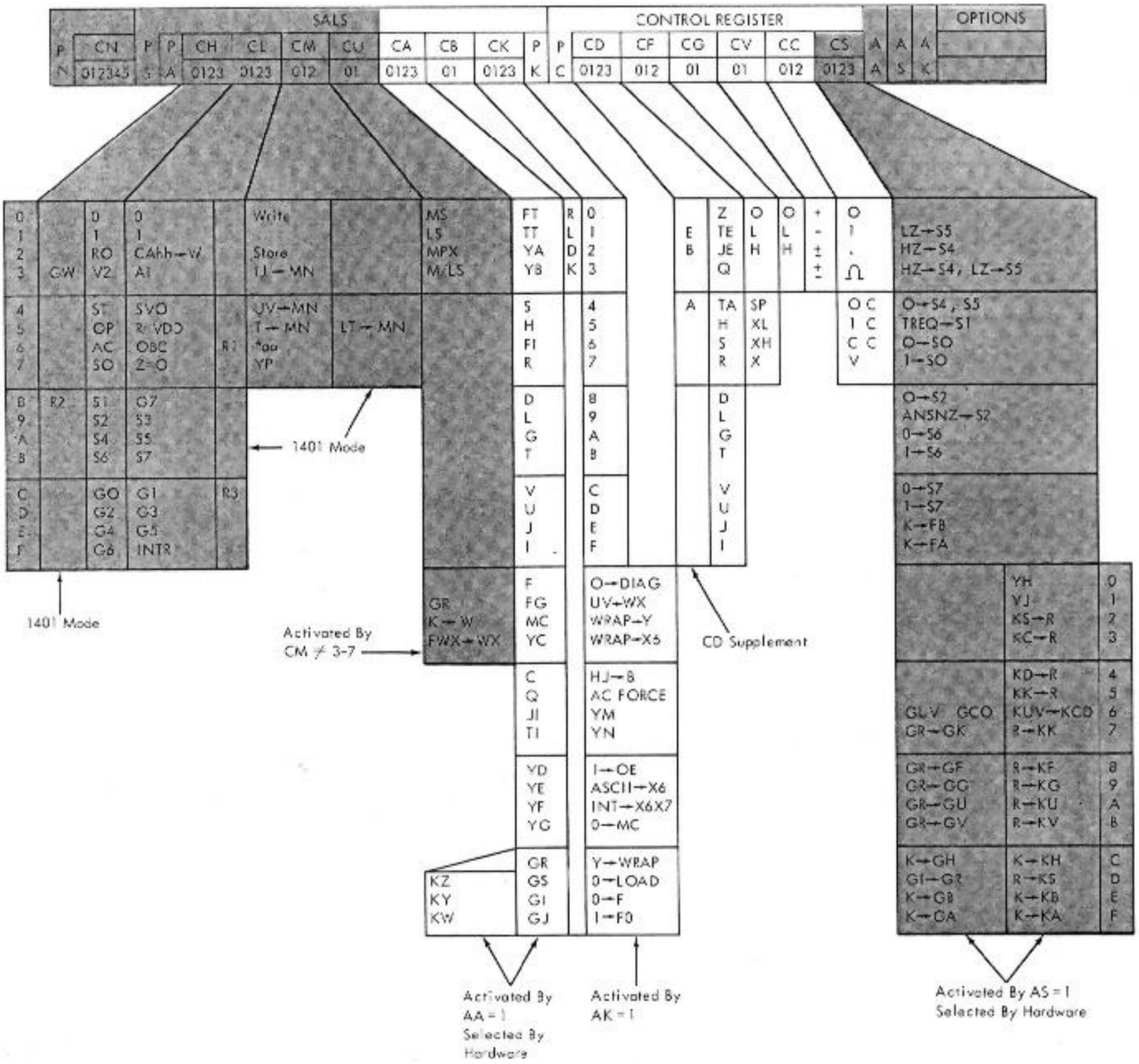


Figure 2-39. ROS Function Control Fields

field is decoded to select the data to be routed to the A-register. It can be decoded to 16 combinations, but by using the AA field (explained later), the CA field has 16 alternate sources to select. This makes 32 combinations for the A-register source.

Control of the A-register output (CF): This 3-bit field controls the method that the data from the A-register is presented to the ALU. The field is essentially bit significant. There are eight bits routed to the ALU from the A-register; we can block all of them, block the four high bits, block the four low bits, or allow all eight bits.

If we block any bits, zeros are routed to the ALU in place of the blocked bits. We can also cross the four low bits with the four high bits or cross and block four bits. Figure 2-40 shows: if the 2 bit is on, the four low bits are allowed, if the 1 bit is on the four high bits are allowed and if the 0 bit is on the high and low bits are crossed.

Cross the A-Register Four High Bits with the Four Low Bits	Block the A-Register Four Low Bits Replace with Four Zeros and Allow High Bits	Block the A-Register Four High Bits Replace with Four Zeros and Allow Low Bits	
Bit 0	Bit 1	Bit 2	
0	0	0	Block A-Register-Route Zeros to ALU
0	0	1	Block High Bits-Route 0000 and Low Bits
0	1	0	Block Low Bits-Route High Bits and 0000
0	1	1	Route A-Register to ALU
1	0	0	Conditional Machine Stop
1	0	1	Block High Bits-Route Low Bits and 0000
1	1	0	Block Low Bits-Route 0000 and High Bits
1	1	1	Route A-Register Crossing Low and High

Figure 2-40. CF Field Bit Significant

If both the 1 and 2 bits are off, the information in the A-register is blocked. Notice there are two possible conditions for this, all three bits off or just the 0-bit on. The condition of just the 0-bit on has been selected as the machine stop function since it did not serve any other useful purpose. The stop function is explained in greater detail later in this section.

Source to the B-register (CB): This 2-bit field is decoded to select the data to be routed to the B-register from either the R, L, D, or K register. The K-register is the CK field of the ROS word.

The K-register (CK): The K-register is also called the emit field or the constant field. This 4-bit field can be decoded to 16 combinations; there are 16 alternate combinations which are active when the AK field has a 1 bit.

The primary bit configuration can be used to emit a digit 0 through F. The same digit is presented to both the high and low four bits of the B-register. For example, the K-register has a 1 in it and the CB field decodes to route K-register to B-register, the 1 enters the high four bits and the low four bits giving us the number 11. By using the CG field, we can route to the ALU from the B-register the number 01, 10, 11, or 00. The W-register can be set from the CK field if desired.

The K-register can also be used to create an address to set in the N-register. This is explained in greater detail later in the section.

Control of the B-register output (CG): This 2-bit field controls how the data from the B-register is presented to the ALU. The operation is the same as the CF field except the B-register cannot be crossed. We can block the output and route eight zeros, or block either the high or low four bits and route zeros where the bits were blocked. The B-register output can be routed direct (both high and low four bits) to the ALU.

Control of ALU (CV): This 2-bit field decodes to select what type of arithmetic operation (true/complement and binary/decimal) is to be performed. The B-register input to the ALU is the true/complement side.

(CC): This 3-bit field decodes to control the carry-in and carry-out to the ALU and permits the setting of a carry-out into the carry latch. This field also decodes to control the AND, OR, and EXCLUSIVE OR function of the ALU.

MAIN AND AUXILIARY STORAGE CONTROL. The two ROS fields which control main and auxiliary storage are the CM and CU fields (Figure 2-41) and work in conjunction with each other. To understand the functions of the two fields, it is easier to explain the operation of the two fields together.

Functional Units

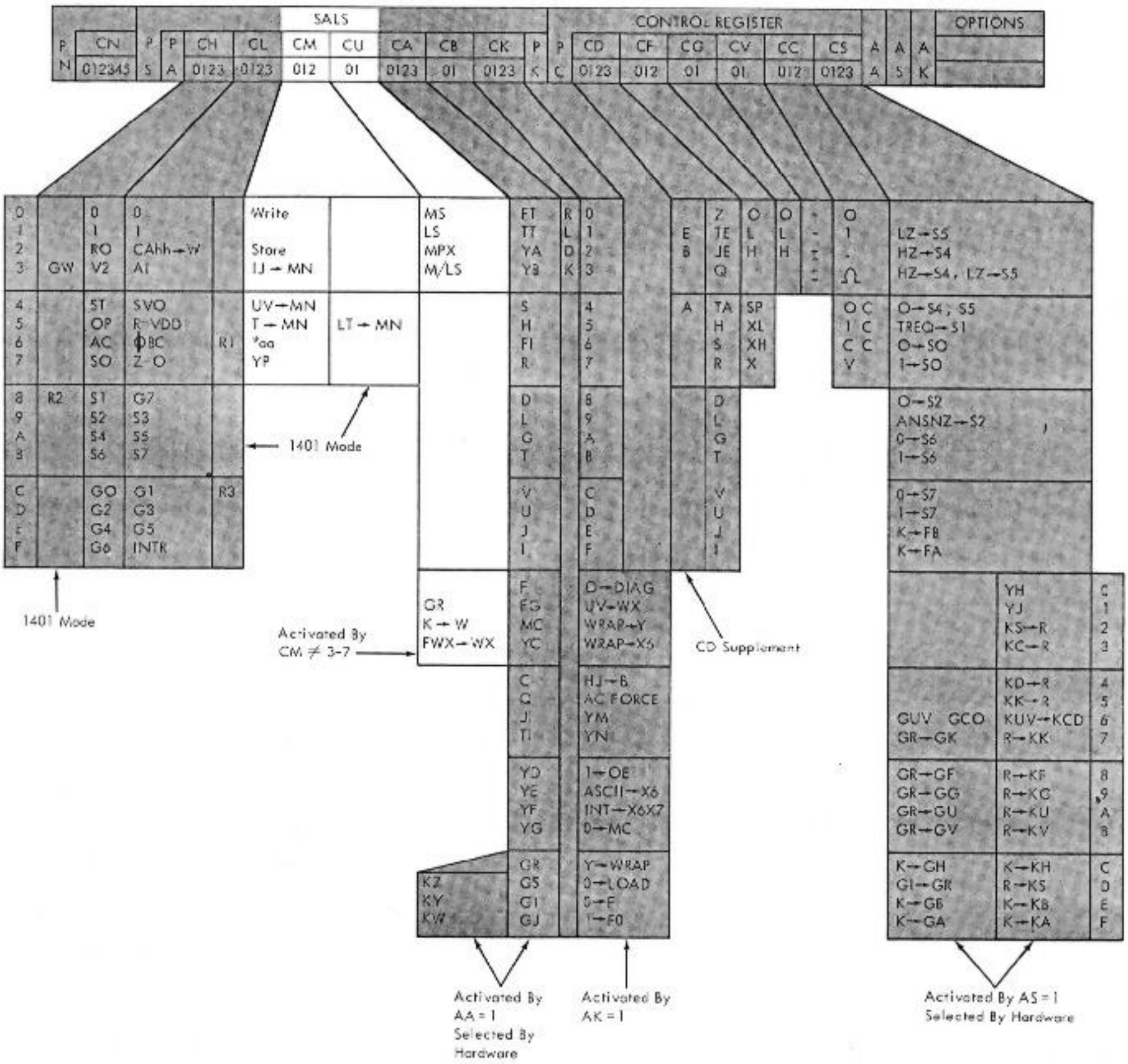


Figure 2-41. ROS Storage Control Fields

The 3-bit CM field decodes to select the type of operation - read-compute or write. The 2-bit CU field decodes to select what section of storage to operate in: main storage or auxiliary storage. Auxiliary storage includes local storage and the multiplexor storage blocks.

In the 2030, the four basic core storage cycles are:

Read, Write (R, W)

Read, Compute, Write (R, C, W)

Read, Store (R, S)

Read, Compute, Store (R, C, S)

Remember from the study of ROS hardware and timing, that the data from core storage is not ready for use until the beginning of the next ROS cycle. Therefore, if a read call is given, the next cycle must be a write, a store, or a compute cycle. Also, a write or store cycle should follow a read cycle within three ROS cycles. If this rule is not followed, it is possible to have an over-run condition of an I/O unit on the selector channel. Over-run is where

new data is ready but can not be accepted before more data is ready. There is an allow write latch on the 2030 which is used to recognize whether the last cycle was a read or a write. If a read is followed by a read, there will be a position in storage with all bits missing. This happens because the position read first had nothing written into it before its storage address was changed. If a write is followed by a write, the second write becomes a compute cycle because the allow write latch is off (set to allow a read cycle).

If the read cycle is followed by a write cycle, the data is set in the R-register and is routed to the core storage unit from the R-register during the write cycle. If the read cycle is followed by a store cycle, the output from core-storage is not used. Instead, new information is in the R-register at the end of the read cycle and is then written into core-storage during the store cycle.

If the read cycle is followed by a compute cycle, the output from core-storage during the read cycle is set into the R-register. During the next cycle, the information in the R-register may or may not be used in the computation. The next cycle is either a write or a store cycle and the R-register may contain the original information or the result of the computation. In any case, what is finally in the R-register is written in core-storage during the write or store cycle.

The core storage read-write control (CM): This 3-bit fields is decoded to determine if the cycle is a read, compute, or write cycle. A 0 or 2 decodes to a write cycle (2 is a store but brings up a write operation), a 1 is a compute cycle, while 3 through 7 are read cycles.

The section of core storage used (CU): This 2-bit field decodes to select which section of core storage is used during the read cycle. Write at the same address. The alternate decodes for the CU field are activated when writing by the CM field having a decode of 0, 1, or 2. The alternate decodes are explained later in this section.

Note: If the CU field is a 3 (M/LS), the operation must be checked further to see if main storage or local storage is to be used. This is done by checking the two high-order bits of the G-register which contain the op code during this time. If the two bits are 00, the op code format is RR and local storage is used. Any other combination of the two bits (01, 10, or 11) requires the use of main storage.

BRANCHING AND ROS ADDRESS. The complete ROS address is held in the W- and X- registers. The W- registers hold the five high-order positions of the ROS address and can be set by a ROS statement CAhh->W (detail on this ROS statement later) and the eight low-order positions of the ROS address are in the X- register. Normally the X- register is set from the CN, CH, and CL fields (Figure 2-42).

Functional Units

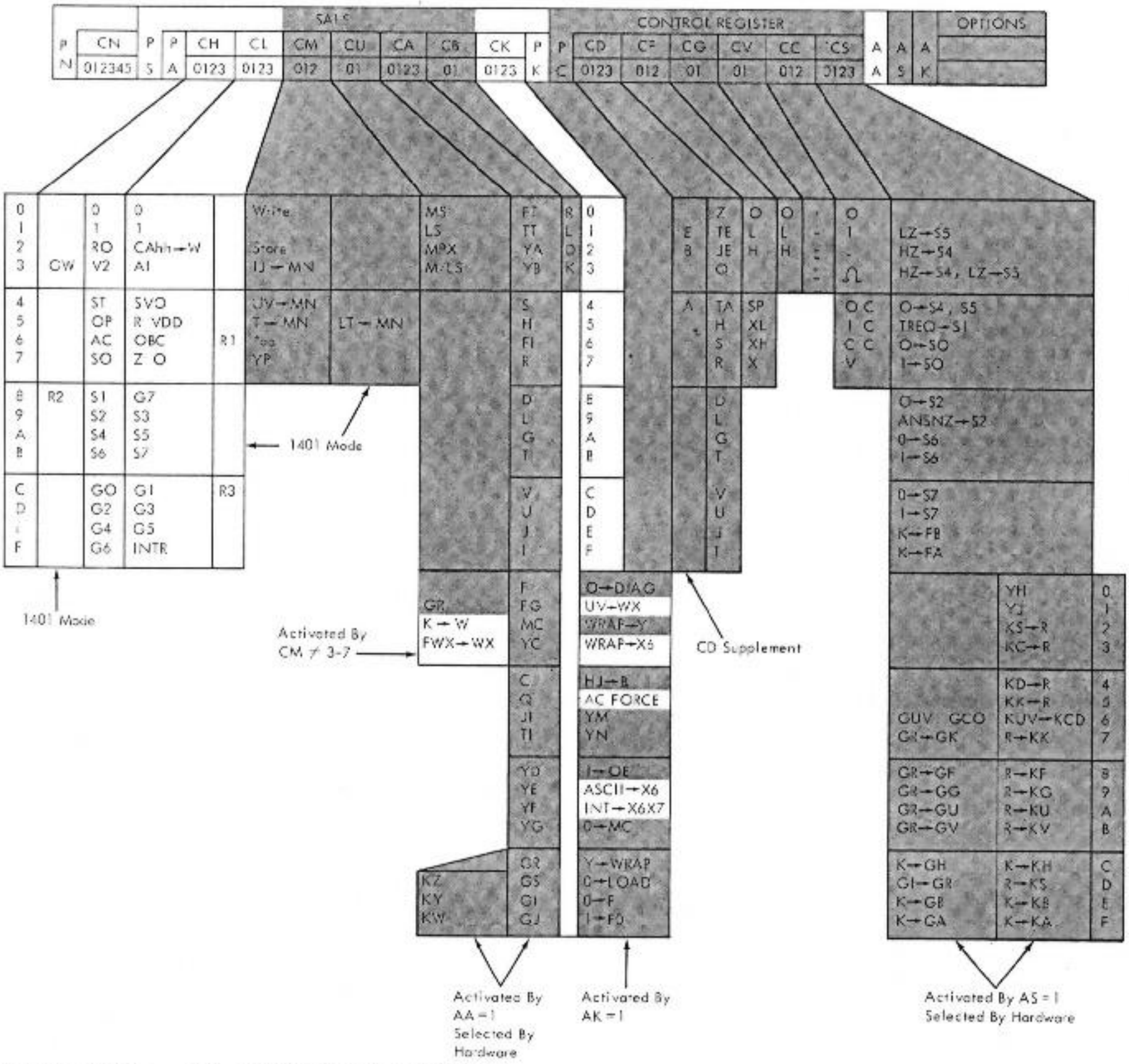


Figure 2-42. ROS Branch Control Fields

The 0 through 5 positions of the X register are set from the CN field while the 6 and 7 positions are set by decoding the CH and CL fields. If the condition of the CH field is satisfied, the 6th position of the X- register is set to the on condition and if the condition is not satisfied, the position is set to 0. The same operations for the 7 position are used except the CL field is decoded to determine the on or off condition.

STATUS SET AND RESET. Certain bit positions in the S-register are controlled by the CS field (Figure 2-43). The FB and FA latches for the multiplexor channel are also controlled by the CS field. The alternate codes of the CS field are used for the selector channel.

Functional Units

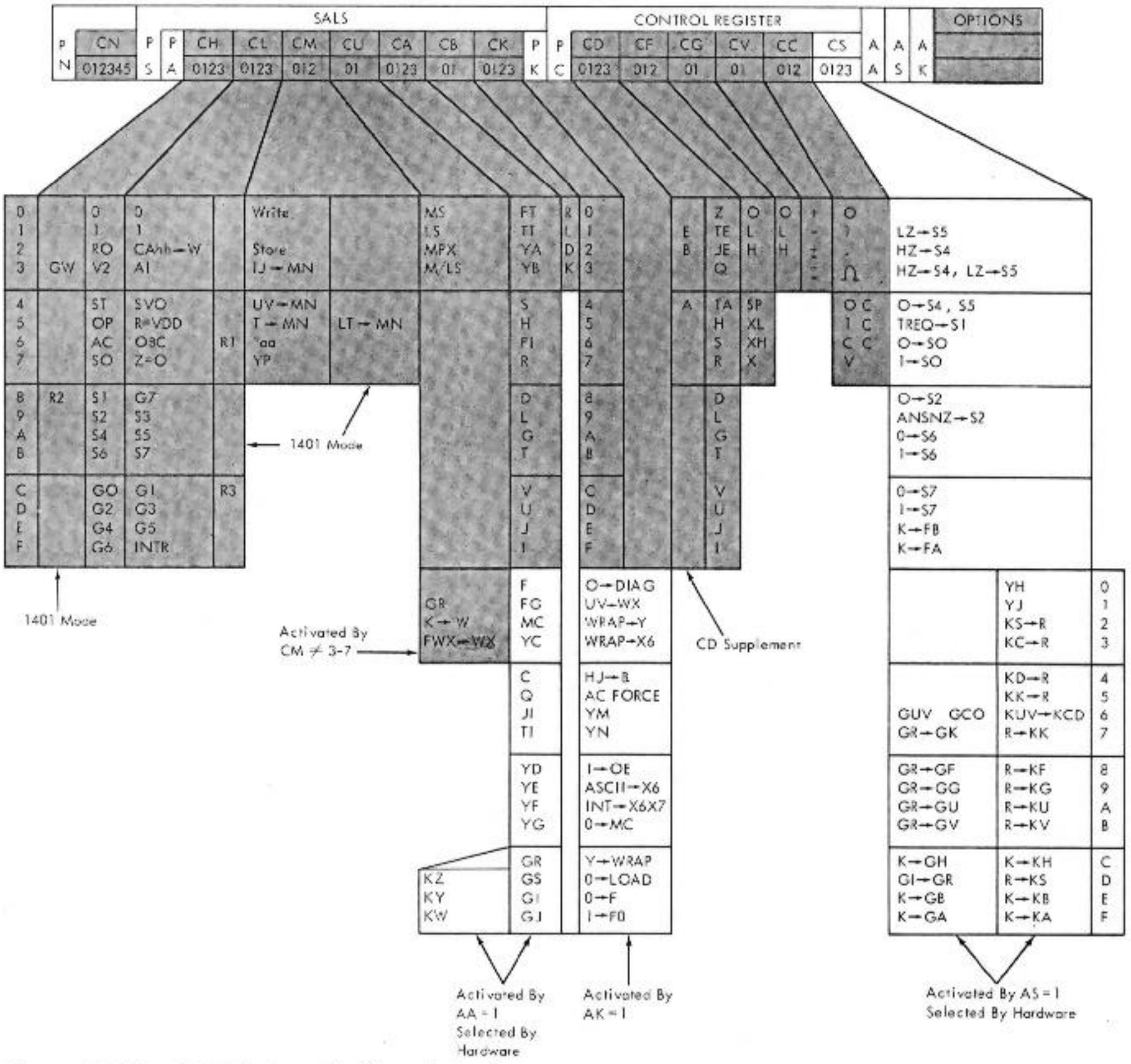


Figure 2-43. ROS Status Field and Parity

ALTERNATE DECODE. When the 1-bit AA field has a 1, the alternate codes for the CA field are used. If the 1-bit AS field has a 1, the alternate codes for the CS field are used. When the 1-bit AK field has a 1, the alternate CK codes are used (Figure 2-43).

When the 2030 is in 1401 compatibility mode the AA field needs a 1 in conjunction with the mnemonic CAhh->W, to set the ROS address.

CONTROL FIELD PARITY BITS. There are five parity bits associated with the control fields: PN, PS, PA, PK, and PC. (Figure 2-44) shows the fields and the parity bits used for each checking circuit.

Functional Units

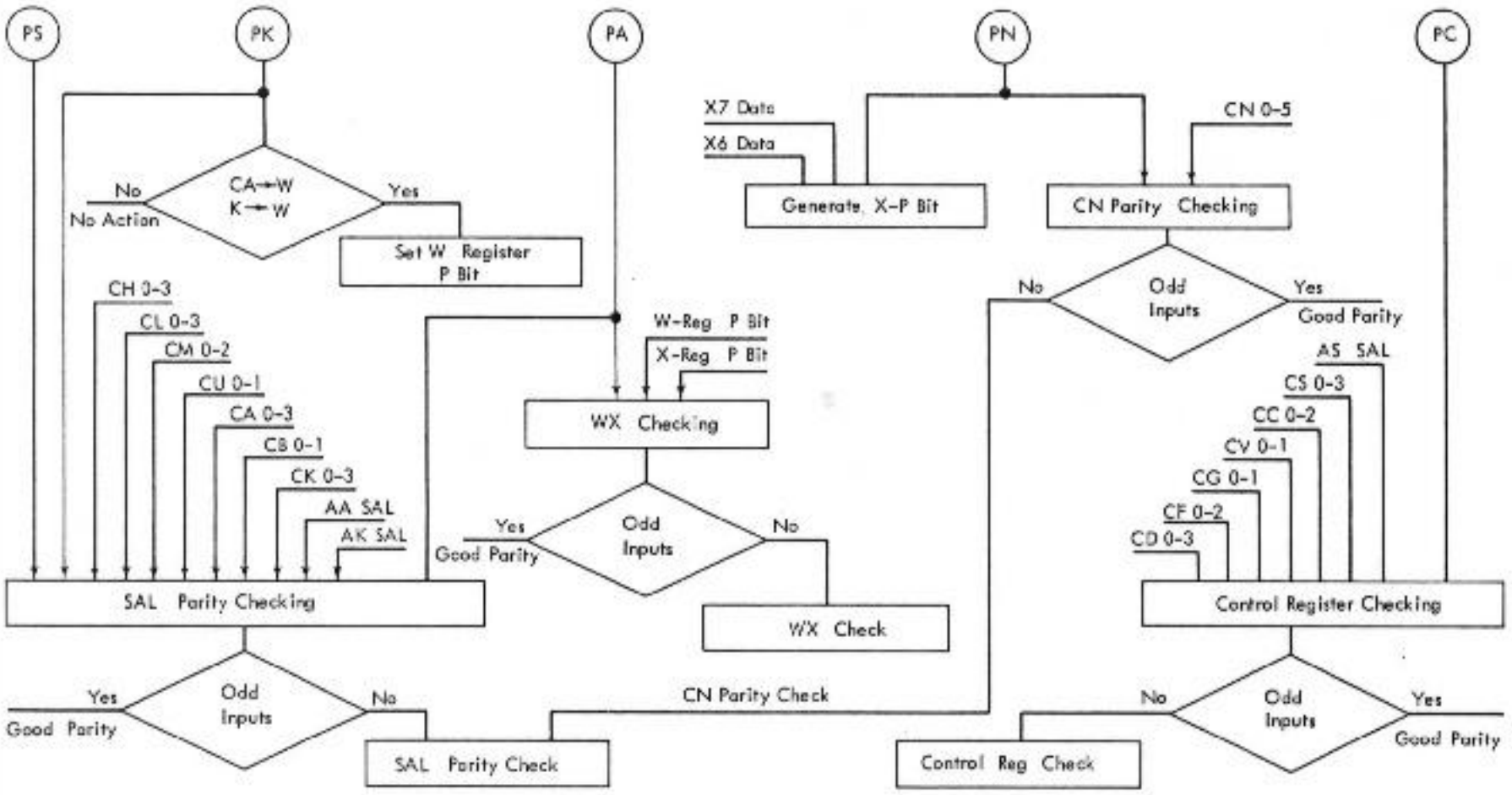


Figure 2-44. Parity Check Bits

The PN parity bit is used to maintain odd parity for the CN field. This bit is used with X6 and X7 bits to set X- register parity bit when CN is gated to the X- register. The PS parity bit is used to maintain odd parity for the AA, AK, CA, CB, CH, CK, CL, CM, and CU fields and the PA and PK bits.

The PA parity bit is used to maintain odd parity for the ROAR. As an example, if the address of the ROS word is 01BF (0000 0001 1011 1111), the PA bit must be a one to maintain odd parity.

The PK parity bit is used to maintain odd parity for either the CA or CK fields depending on the mnemonic used.

When the CK field is used as a constant in an arithmetic statement, the PK bit is not specified. In this case, the PK bit can be 0 or 1; usually 0. In the storage statement (*aa) or in a statement where K is used to change the W- register (K->W), the PK bit is used to provide odd parity on the W- register.

If the CA field is used to set the W- register (CAhh->W), the PK bit is used to maintain odd parity for the W- register.

The PC parity bit is used to maintain odd parity for the AS, CC, CD, CF, CG, CS, and CV fields.

Control Field Mnemonics

- Most of the control fields have from one to sixteen mnemonics.
- Some of the control fields have alternate mnemonics which are activated by the condition of another field.

By this time, we know the concepts of ROS, along with the names and functions of each ROS control field. Now we need to know how each control field is coded and how this coding is written in the microprogram so the microprogram can be read and punched in the ROS card.

Figure 2-45 shows the symbols used in the mnemonics and the meaning of the symbol. Figure 2-46 shows the mnemonics for each field and gives a brief description of the purpose each mnemonic serves.

Symbol New	Symbol Old	Definition	Example
+	+	True Add/Positive	A + B: B is Added (True) to A
-	-	Complement Add/Subtract, Negative	A - B: B is Complement Added to A
=		Equal	A = B: A is Equal to B
≠	≠	Unequal	A ≠ B: A is Unequal to B
←	=	Is Set Into	A←B: A is Set Into B (Destructive Read-In is Implied.)
·	·	Is ANDed With (Logical)	A . B: A is ANDed with B
,	,	AND (Non-logical)	A←B, C: A is Set Into B and C
∩	$	Is ORed With (logical)	A∩B←C: A is ORed with B and the Result is Set Into C
/	/	OR (Non-logical)	A / B←C: A or B is Set into C
⊻	⊻	Is Exclusive ORed With	A⊻B←C: A is Exclusive ORed with B and Set Into C
±		True or Complement Add/Positive or Negative	A ± B: B is True or Complement Added to A and the Result is Set into C
±	±	Binary Add Under T/C Control	A ± B←C: B is True or Complement Added to A and the Result is Set into C.
±	@	Decimal Add Under T/C Control	A@B←C: B is True or Complement Decimal Added to A and the Result is Set into C.
<	<	Is Less Than	A < B: A is Less than B
¬	¬	Not (Boolean - Used as the Not Function on CLD's)	
:	:	Is Compared to	A : B: A is Compared to B
()	()	Used for Normal English Punctuation or to Enclose an Expression Within a Statement	
*		Special. 2030 Uses the * for One Mnemonic.	*aa: Explained in Mnemonic Section.
?	?	Indeterminate Function (This Describes a function which is Hardware Controlled Rather than Under the Direct Control of the Micro Program.	A ? B←C: A and B are Logically Combined (Under Hardware Control) and the Result is Set Into C.

Figure 2-45. CLD Block Symbols

Field	Hex	Mnemonic	Old Form	Operation. Location of Field is in Reference to Automated CLD Box
0 - 5 CN	-	-	-	Shown in Hex on Right Side of Line 7 in the CLD Box. Sets Position 0 through 5 of the X-Register for Next Address.
0 - 3 CH Set 6th Position of X-Register	- 0 1 2 3 4 5 6 7 8 9 A B C D E F	- 0 1 R0 • VZ ST OP AC S0 • S1 S2 S4 S6 G0 G2 G4 G6	- 0 1 R0 V=00 STI OPI AC S0 S1 S2 S4 S6 G0 G2 G4 G6	Shown on Left Side of Line 7 in the CLD Box. Set X-6 to ZERO Set X-6 to ONE Set X-6 to the Condition of R-Register Position 0 Set X-6 to ONE, if the V-Register Positions 6 and 7 are ZERO Status in (I/O) OP in (I/O) Set X-6 to ONE: if there is a Carry Out of ALU Position 0 } Set X-6 to ONE, if the Tested Position of the S-or G-Register is Equel to ONE
0 - 3 CL Set 7th Position of X-Register	- 0 1 2 3 4 5 6 7 8 9 A B C D E F	- 0 1 • CAhh→W AI SVI • R=VDD • 1BC Z=0 G7 S3 S5 S7 • G1 G3 G5 INTR	- 0 1 W=CA AI SVI RVDD 1BC Z=0 G7 S3 S5 S7 G1 G3 G5 INTR	Shown on Left Side of Line 7 in the CLD Box-Example; CH, CL Set X-7 to ZERO Set X-7 to ONE Set Value of CA Field into W-Register, Set X-7 to ONE. hh is the Hex Value of the CA Field and AA Field Address in (I/O Address) Service in (I/O) Set X-7 to ONE if the R-Register Contains Valid Decimal Digits. Set X-7 to ONE if there is a Carry Out of ALU Position One. Set X-7 to ONE if the Z-Bus (Bits 0-7) is ZERO } Set X-7 to ONE, if the Tested Position of the S-or G-Register is Equal to ONE Test for any Interrupt, Set X-7 to ONE if ther is a Interrupt.
0 - 2 CM Storage Control	- 0 1 2 3 4 5 6 7	- WRITE STORE IJ →MN UV →MN • T →MN • *aa YP	- WRITE STORE IJ UV T K GUV	Shown on Left Side of Line 4 in the CLD Box Write the Data in the R-Register into the Storage Position Addressed by the M-and N-Registers No Mnemonic-Compute Cycle, Storage not Used. Write NEW R-Register Data into the Storage Position Addressed by the M-and N-Registers Set the M-and N-Registers to the Address in the I-and J-Registers and Read from Storage at that Address Set the M-and N-Registers to the Address in the U-and V-Registers and Read from Storage at that Address Set the N-Registers to the Address in the T-Register and Read from Storage at that Address Set the N-Register using the CK Field (Note 1) Dummy Symbol-No Action or Can be Used in a Diagnostic Area. (Old From was a Selector Channel Code).
0 - 1 CU Storage Selection	- 0 1 2 3	- MS LS MPX M/LS	- MEM CPU UCW M,C	Shown on Right Side of Line 4 in the CLD Box Addressing MAIN Storage Addressing Auxiliary Storage-LOCAL Store Section Addressing Auxiliary Storage-Multiplexor UCW Section Addressing MAIN Storage or LOCAL Store Section, Depending on the OP Code-RR Format Selects LS In 1400 Mode this Selects the Local Storage Area for NPL Area
0 - 1 Alternate CU	- 0 1 2 3	- GR K →W FWX →WX	- Use GR W=K WX=FWX	Shown on Right Side of Line 4 in the CLD Box No Action Use the GR-Register in the Selector Channel in Place of the R-Register for Storage Input and Output Set the W-Register to the Hex Valve of the CK Field Set the W-and X-Registers to the Address in the Multiplexor Back-Up Registers (FW and FX)
0 - 3 CA A-Register Source Control	- 0 1 2 3 4 5 6 7 8 9 A B C D E F	- FT TT YA YB S H FI R D L G T V U J I	- FT TT S H FI R D L G T V U J I	Shown on Left Side of Line 3 in the CLD Box Multiplexor Channel Tags in 1050 Tags In Dummy Symbol-No Action or Can be Used in a Diagnostic Area Dummy Symbol-No Action or Can be Used in a Diagnostic Area Gate the S-Register to the A-Register Via the A-Bus Gate the H-Register to the A-Register Via the A-Bus Multiplexor Channel Bus In } Gate the - Register to the A-Register Via the A-Bus

Figure 2-46. Mnemonics, Sheet 1

Field	Hex	Mnemonic		Old Form	Operation
0-3	-	-	-	-	Shown on Line 4 of the CLD Box
Alternate	0		F	F	Gate the F-Register to the A-Register Via A-Bus (External Interrupts).
CA	1		FG	FG	Gate the F-and G-Switches to the A- Register Via the A-Bus
Activated	2		MC	MC	Gate the Machine Check Register to the A-Register Via the A-Bus
by "AA"=1	3		YC		Dummy Symbol-No Action or Can be Used in the Diagnostic Area
	4		C	C	Gate the C-Register to the A-Register Via the A-Bus (Interval Timer)
	5		Q	Q	Gate the Q-Register to the A-Register Via the A-Bus (Protect Storage)
	6		JI	JI	Direct Data Channel Bus In
	7		TI	TI	1050 Bus In
	8		YD		} Dummy Symbols - No Action or Can be Used in the Diagnostic Area
	9		YE		
	A		YF		
	B		YG		
	C		GR	GR	Gate the GR-Register (Selector Channel) to A-Register Via A-Bus
	D	KZ	GS	GS	Gate the GS-Register (Selector Channel) to A-Register Via A-Bus
	E	KY	GT	GT	Gate the GT-Register (Selector Channel) to A-Register Via A-Bus
	F	KW	GJ	GJ	Gate the GJ-Register (Selector Channel) to A-Register Via A-Bus
0-1	-	-		-	Shown on Line 3 of the CLD Box
CB	0	R		R	Gate the R-Register to the B-Register Via the B-Bus
B-Register	1	L		L	Gate the L-Register to the B-Register Via the B-Bus
Source	2	D		D	Gate the D-Register to the B-Register Via the B-Bus
Control	3	K		K	Gate Hex Value of the CK Field to the B-Register Via the B-Bus
	-	-		-	
0-3	-	- - - -		- - - -	Shown on Line 2 of the CLD Box
CK	0	0 0 0 0		0 0 0 0	
Emit Value	1	0 0 0 1		0 0 0 1	
	2	0 0 1 0		0 0 1 0	
	3	0 0 1 1		0 0 1 1	
	4	0 1 0 0		0 1 0 0	
	5	0 1 0 1		0 1 0 1	
	6	0 1 1 0		0 1 1 0	
	7	0 1 1 1		0 1 1 1	
	8	1 0 0 0		1 0 0 0	} Binary Bit Form of the Hex Number is Routed to the Selected Area When Requested.
	9	1 0 0 1		1 0 0 1	
	A	1 0 1 0		1 0 1 0	
	B	1 0 1 1		1 0 1 1	
	C	1 1 0 0		1 1 0 0	
	D	1 1 0 1		1 1 0 1	
	E	1 1 1 0		1 1 1 0	
	F	1 1 1 1		1 1 1 1	
0-3	-	-		-	Shown on Left Side of Line 6 of the CLD Box
Alternate	0	O→DIAG		RESET DIAG	Reset the Diagnostic Latch
CK	1	UV→WX		WX = UV	Gate the U-and V-Register to the W-and X-Registers Via the WX-Bus
Activated	2	• WRAP→Y		RESTORE WRAP	Gate the Wrap Buffer Latch to the Wrap Latch
by "AK"=1	3	• WRAP→X6		TEST WRAP	Set X6 to ZERO if Wrap Latch is On
	4	HJ→B		HJ	Gate the H-and J-Switches to the B-Register Via the B-Bus
	5	• AC FORCE		AC FORCE	Set X-Register to ZERO, if a ALU Carry Occurred in Previous Cycle
	6	YM			Dummy Symbol-No Action or Used in Diagnostic Area. Old Mnemonic was Reset 1050 Line Latch
	7	YN			Dummy Symbol-No Action or Used in Diagnostic Area. Old Mnemonic was Set 1050 Line Latch
	8	• 1→OE		OE = 1	Force an ALU Check (Note 3)
	9	• ASCII→X6		TEST ASCII	Set X-6 to ZERO if the ASCII Latch is On.
	A	• INT→X6, X7		TEST INT	Set X-6 and X-7 per Stacked Interrupts (Note 4)
	B	0→MC		MC = 0	Set Machine Check Register to All ZEROS
	C	Y→WRAP		STORE WRAP	Gate the Wrap Latch to the Wrap Buffer Latch
	D	0→LOAD		LOAD	Reset the LOAD, ODD/EVEN, and INTRODUCE ALU CHECK Latches
	E	• 0→F		F = 0	Reset the F-Register to All ONES. Note: The Reset Condition of the F-Register is All ONES
	F	• 1→F0		F0 = 1	Set the F-Register Position 0 to ZERO
0-3	-	-		-	Shown on Line 3 of the CLD Box
CD	0	Z		Z	To Show That the Z-Bus is the Only Place the Output of the ALU is Routed
Destination	1	• TE		TE	1050 Bus Out (Exit)
of ALU	2	• JE		JE	Direct Data Channel Bus Out (Exit). Set JE-Register from D-Register, Z-Bus not Used.
Output	3	Q		Q	Gate the Output of the ALU to the Q-Register Via the Z-Bus
	4	• TA		TA	1050 Tags Out[3]
	5	H		H	} Gate the Output of the ALU to the ____Register Via the Z-Bus.
	6	S		S	
	7	R		R	
	E	D		D	
	9	L		L	
	A	G		G	
	B	T		T	
	C	V		V	
	D	U		U	
	E	J		J	
	F	I		I	

Figure 2-46. Mnemonics, Sheet 2

Field	Hex	Mnemonic	Old Form	Operation
0-2 CF A-Register to ALU	-	-	-	Shown on Line 3 of the CLD Box
	0	0	0	Block A-Register Exit to the ALU. Route All ZEROS to ALU Entry for the A-Register.
	1	L	L	Block High 4 Bits of A-Register. Route 4-ZEROS and Bits 4-7 of A-Register to the ALU
	2	H	H	Block Low 4 Bits of A-Register. Route 4-ZEROS and Bits 0-3 of A-Register to the ALU
	3			Gate the Entire A-Register to the ALU
	4	SP	STOP	Conditional Machine Stop (Note 5)
	5	XL	XL	Block A-Register Bits 4-7 Exit. Gate Register Bits 0-3 to ALU Entry Bits 4-7 and 4-ZEROS to Bits 0-3.
	6	XH	XH	Block A-Register Bits 0-3 Exit. Gate A-Register Bits 4-7 to ALU Entry Bits 0-3 and 4-ZEROS to Bits 4-7.
	7	X	X	Gate A-Register Bits 0-3 to ALU Entry Bits 4-7 and Gate A-Register Bits 4-7 to ALU Entry Bits 0-3.
0-1 CG B-Register to ALU	-	-	-	Shown on Line 3 of the CLD Box
	0	0	0	Block B-Register Exit to the ALU. Route All ZEROS to the ALU Entry for the B-Register.
	1	L	L	Block High 4-Bits of the B-Register. Route 4-ZEROS and Bits 4-7 of the B-Register to the ALU.
	2	H	H	Block Low 4 Bits of the B-Registers. Route 4-ZEROS and Bits 0-3 of the B-Register to the ALU.
	3			Gate the Entire B-Register to the ALU.
0-1 CV Arithmetic Functions	-	-	-	Shown on Line 3 of the CLD Box
	0	+	+	True Add B-Register Data
	1	-	-	Complement Add B-Register Data
	2	±	±	Binary Add or Subtract Depending on the Status of S0
	3	±	ⓓ	Decimal Add or Subtract Depending on the Status of S0
0-2 CC Arithmetic Controls	-	-	-	Shown on Line 3 of the CLD Box
	0	0	0	Block Carry
	1	1	1	Insert Carry
	2	*	*	AND Function-Check to See if Same Bits are Set to ONE in both the A-and B-Register Using the ALU
	3	⌐┘	$	OR Function-Check to See if Either Bit in the Same Position of the A-and B-Register is Set to ONE.
	4	0C	CO	No Carryin. Set S3 to ONE if A Carryout Occurs
	5	1C	C1	Insert A Carryin and Set S3 to ONE if a Carryout Occurs.
	6	CC	CC	Allow Carryin from Carry Latch and Set S3 to ONE if a Carryout Occurs.
	7	⌐┘	⌐┘	Exclusive OR Function-Check to See If Only the A-or B-Register has the Same Bit Position Set to ONE.
0-3 CS Status Conditions	-	-	-	Shown on Line 5 of the CLD Box
	0			No Action
	1	LZ→S5	S5=LZ	Set S5 to ONE if Bits 4-7 of the Z-Bus Are 0. Reset S5 if Non-Zero (Set S6 to ZERO).
	2	HZ→S4	S4=LZ	Set S4 to ONE if Bits 0-3 of the Z-Bus are 0. Reset S4 if Non-Zero (Set S5 to ZERO).
	3	HZ→S4, LZ→S5	S4, S5=HZ, LZ	Combines the Conditions of CE Field Mnemonics LZ→S5 and HZ→S4.
	4	0→S4, S5	S4, S5=0	Set S4 and S5 to ZERO.
	5	TREQ→S1	S1=TREQ	Set S1 to ONE if a 1050 Request has Occurred. Set S1 to ZERO if no 1050 Request.
	6	0→S0	S0=0	Set S0 to ZERO.
	7	1→S0	S0=1	Set S0 to ONE.
	8	0→S2	S2=0	Set S2 to ZERO.
	9	ANSNZ→S2	S2=ANSNZ	Set S2 to ONE if the Output from the ALU is Non-Zero. (Note 6).
	A	0→S6	S6=0	Set S6 to ZERO.
	B	1→S6	S6=1	Set S6 to ONE.
	C	0→S7	S7=0	Set S7 to ZERO.
	D	1→S7	S7=1	Set S7 to ONE.
	E	K→FB	FB=K	Multiplexor Channel Tags Out.
	F	K→FA	FA=K	Multiplexor Channel Tags Out.
0-3 Alternate CS Selector Channel Activated by *AS*=1 Selected by Hardware	-	-	-	Shown on Line 5 of the CLD Box.
	0	YH	-	- } Dummy Symbols
	1	YJ	-	-
	2	KS→R	-	- } Used for Selector Channel Operations These are Selector Channel Registers (Note 2)
	3	KC→R	-	-
	4	KD→R	-	-
	5	KK→R	-	-
	6	KUV→KCD	GUV→GCD	GCD=GUV
	7	R→KK	GR→GK	GK=GR
	8	R→KF	GR→GF	GF=GR
	9	R→KG	GR→GG	GG=GR
	A	R→KU	GR→GU	GU=GR
	B	R→KV	GR→GV	GV=GR
	C	K→KH	K→GH	GH=K
	D	R→KS	GI→GR	GR=GI
	E	K→KB	K→GB	GB=K Selector Channel Tags Out.
	F	K→KA	K→GA	GA=K Selector Channel Tags Out.

Note 1
N-Register Set as Follows:
N0-Forced to ONE
N1-Forced to ZERO
N2-CN 08it
N3-CK 08it
N4-Forced to ONE
N5-CK 18it
N6-CK 28it
N7-CK 38it

Note 2
These Mnemonic Depends on the Channel Requested. Mnemonics may be of Three Types for one Hex Number, Example-Alternate CA Field. Hex E can be KY, GT, or HT

Note 3
Used on Diagnostics to Force Parity

Note 4

	X6	X7
Timer/External	0	0
Channel 1	1	0
Channel 2	0	1
Multiplexor Channel	1	1

Note 5
Micro Program Stop or Process Loop Stop

Note 6
In Diagnostic Mode, If the "Malfunction Trap Latch" is Set, this Mnemonic will Cause Machine Stop.

Figure 2-46. Mnemonics, Sheet 3

Some of the mnemonics need a more complete explanation than given in the chart. There is a • next to the new mnemonic having a more detailed description. The column to the left of the mnemonic contains the hex number punched in the bit positions for that control field. As an example, the CH field would be punched 1010(A) for a mnemonic S4.

CAhh->W: The value in the CA field is gated to the W-register positions 4-7 and the AA field is gated to position 3. Example: to change from ROS address 01XX to 08XX, the mnemonic CA08->W is used. Also, position 7 of the X-register is set to 1 for the next address. Parity for the W-register is maintained by using the PK bit.

Note: If this mnemonic is used in the 1401 compatibility mode, the AA 1-bit field is set to 1. This is routed to the 3rd position of the W-register to select the second ROS module or if set to 0, selects the first ROS module. Also when this mnemonic is used, the add statement normally has an A entry of 0 (CF = 000).

R = VDD: Each half of the R-register is checked for a valid decimal digit (0-9). Set X7 to a 1 if both digits are valid decimal digits.

K->FA: The CK field is used to set and reset latches in the multiplexor FA-register, singly or in combination. The value of the CK field is shown on the E line of the new CLD box which is explained later. The PK bit is necessary and it's condition is also specified on the E line. If the CK field value is 3 and PK is 1, the E line will have K = 0011,1.

K->FB: The CK field and the PK bit is used to set and reset some of the multiplexor FB-register latches. This mnemonic also provides a gate for the set and reset of other latches.

Y->WRAP: An address overflow (memory wrap) may arise on a 64K core storage unit. Additional circuitry is needed to detect the error which occurs when there is a carryout of the high-order position of the I- or U-register as a result of up-dating the address.

If the I- and J-registers are used to set the M- and N-registers to address core storage, a memory wrap condition sets a wrap latch. Certain routines, which may be needed during the decode of an SS instruction, require the condition of this latch to be retained. The mnemonic Y->WRAP gates the status of the wrap latch to another latch called the wrap-buffer-latch. It is retained there until the WRAP->Y mnemonic is used.

WRAP->Y: When it becomes necessary to determine if there had been a wrap earlier, the mnemonic WRAP->Y gates the status of the wrap-buffer latch to the wrap latch. This mnemonic is also used to reset the wrap latch in some routines.

WRAP->X6: To test the status of the wrap latch for branching, the mnemonic WRAP->X6 is used. If the wrap latch is on, a 00 or 01 branch is taken. However, it if is off, the X6 portion of the branch is still controlled by the CH field. Note: A wrap condition can also occur on an 8K, 16K or 32K machine. This is detected by testing one of the three high-order positions of the M-register to see if it is set to 1. The position tested depends on the size of core-storage the machine has. This does not use the wrap circuits used on 65K machines.

AC Force: This alternate CK mnemonic causes all positions of the X-register to be set to zeros if there was a carryout of the ALU during the previous ROS word. The information in the CN, CH, and CL fields is blocked. With the X-register equal to zero, the microprogram is branched to 00 of the block addressed by the W-register decode.

1->OE: This mnemonic is used in diagnostic testing. The first time this mnemonic is used in a routine, bad parity is forced by blocking the + L Z bus 0 line and + L Z bus 4 line. The next time this mnemonic is used in the routine, an ALU check is forced by forcing all the minus SUM lines and the minus carry 0-bit line to a plus L levels (Figure 2-47).

The odd-even-control latch is turned on (EVEN) by the decoded

line 1->OE, T2 and the introduce ALU check latch set off. A circuit to gate the + L Z bus 0 line and the + L Z bus 4 line requires that the odd-even latch be ODD. The introduce-ALU-check latch is turned on at T1 when the expression 1->OE is used again. This latch blocks the -L SUM lines and the -L carry 0-bit line so the lines are all plus and an ALU check is forced. The odd-even-control control latch is turned off three ways: machine reset, reset load line (which is developed when the mnemonic 0->IPL is used), or at T2 time when the introduce-ALU-check latch is on.

ASCII->X6: This mnemonic tests the ASCII latch to see if the ASCII latch is on. If the latch is on, the 6th position of the X-register is set to 0. If the latch is off, the 6th position of the X-register is set by the CH field conditions.

H

When the H-register is specified by the CD field coding of 5 (0101), the priority-reset-control latch is set on. This latch ANDed with T3 time, turns the priority latch off so priorities may be recognized.

S.P. or STOP

This CF field mnemonic causes the stop latch to be turned on at T4 time (Figure 2-48). The output from the stop latch feeds two circuits. If the J-register is not specified by the CA field, one circuit causes a microprogram stop line to be active. This line stops the CPU clock by blocking the clock start circuit. If the J-register is specified and the process stop latch is on, a process-loop-stop line is made active when the stop latch comes on. The process-loop-stop line allows the CPU clock to run until all ROS share requests or multiplexor share requests have been honored. The CPU clock is stopped by turning off the clock-start latch. The process-loop stop line blocks the set of the W- and X-registers so the microprogram returns to the address of the STOP word after execution of any ROS or multiplexor share request.

*

This mnemonic addresses a byte in local storage. The expression on the S line of the CLD box to read addressable byte 27 from local storage is *BB LS. Figure 2-49 shows how the N-register is set to BB so byte 27 can be addressed. The CK field must be a B (1011). Since this is a constant, the CK field is specified on line E of the CLD box.

VZ: When operating in 1401 mode, this mnemonic appears as GW and is used to allow a branch on a group mark word mark combination.

S1: When operating in 1401 mode, this mnemonic appears as R2 and is used to allow a branch on the condition of the bit 2 in the R-register.

1BC: When operating in 1401 mode, this mnemonic appears as R1 and is used to allow a branch on the condition of the bit 1 in the R-register.

G1: When operating in 1401 mode, this mnemonic appears as R3 and is used to allow a branch on the condition of the bit 3 in the R-register.

T->MN: When operating in 1401 mode, this mnemonic appears as LT->MN and is used to gate the L- and T-registers to the M- and N-registers. This performs the functions of the A-star in the 1401.

0->F: The F-register is used to recognize external interrupts. This mnemonic resets the F-register so the output lines of the F-register are plus, preventing any external interrupts from being recognized until the F-register is set from an external device. Since the F-register is reset so all output lines are plus, we say it is reset to ONE's.

1->F0: Since the F-register is reset to ONE's, this mnemonic set the 0-position of the F-register to ZERO.

Special Statements

The following special statements are used in diagnostic programming:

0V±0->Z: This expression brings up the control lines to check positions

4 and 5 of the R-register. If bit 4 is a one, the ASCII latch is set. If bit 5 is a zero, the suppress malfunction trap latch is set. Decimal mode is specified on line 2 of the CLD block.

J ±0->Z: The wait latch is set on and the ROS word that contains this statement is continually executed until an interrupt occurs. Decimal mode is specified on line 2 of the CLD block.

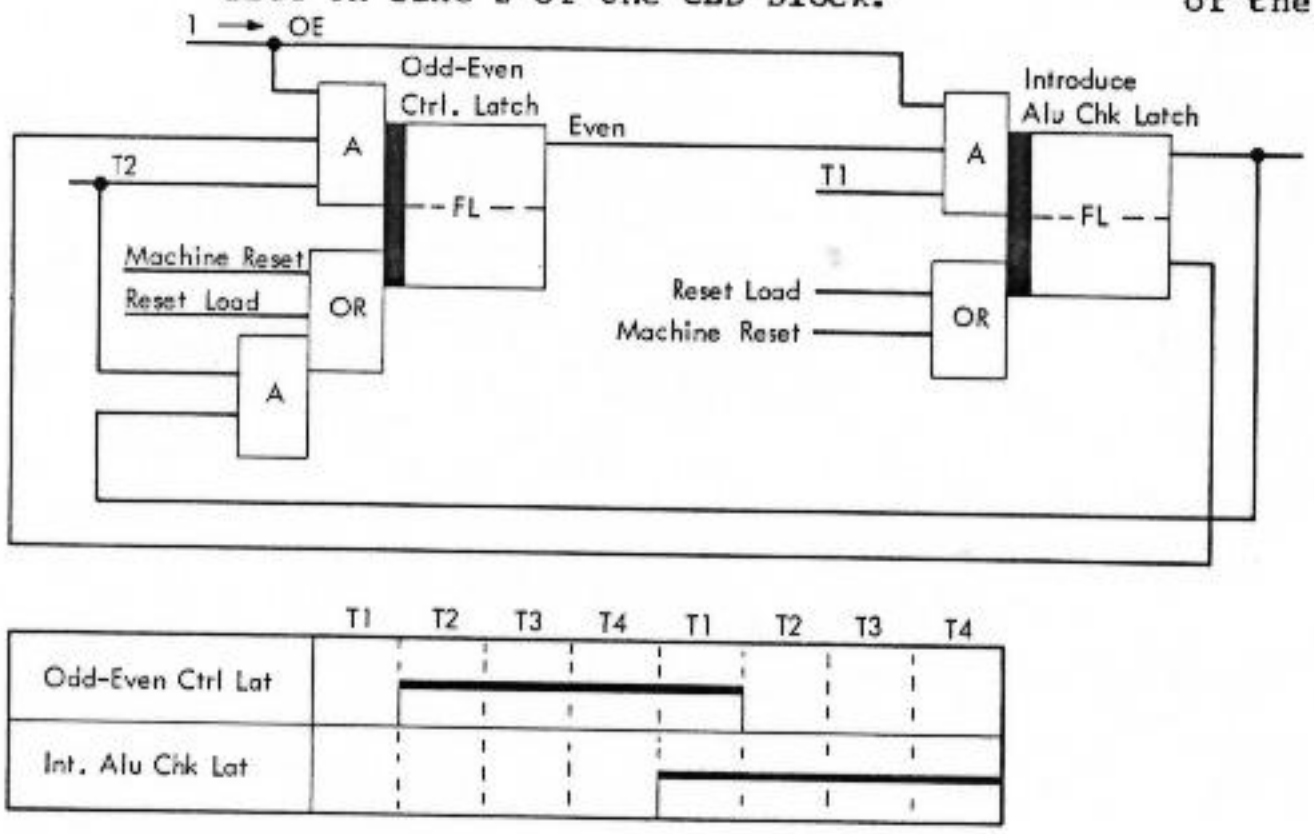


Figure 2-47. 1 -> OE Control

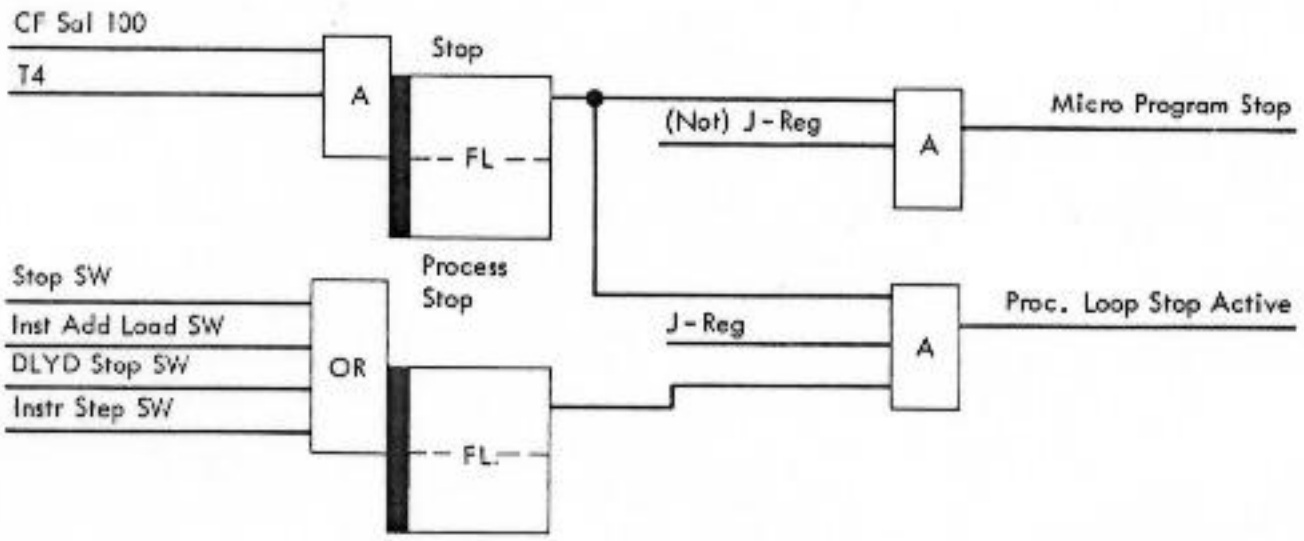


Figure 2-48. Stop Mnemonic

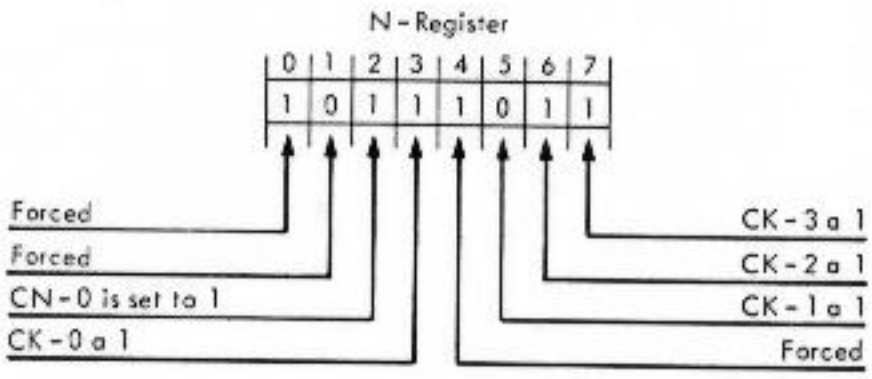


Figure 2-49. N-Register Set from *aa

CYCLE 0104

This is the last cycle of the alternate program. During this cycle, the control-register latches are good for address 0104. At T1 time, it is necessary to set ROAR with the next address to be executed. Since this is the last step of the alternate routine, the address where the main program was when the break-in occurred is needed. Therefore, the mnemonic FWX->WX causes the backup ROAR to be gated to ROAR.

MPX ROS LATCH OR ROAR RESTORE BUFFER LATCH SX

This latch is turned on at T2 time when a statement in the microprogram specifies that ROAR must be changed using the backup ROAR. In our example, the statement is in word 0104, FWX->WX. One purpose of this latch is to allow the ROAR Restore Latch to be set at T4 time of the cycle in which the ROS word at 0001 is read out.

CYCLE 0001

The control-register latches are good for address 0001 during this cycle. Even though we have addressed this ROS word before, this is the first time that it is executed. The first time that 0001 was addressed, the control registers were not set. At T1 of this cycle, the branch portion of address 0002 is set up by using the backup X6 and X7 latches.

ROAR RESTORE LATCH OR ROAR RESTORE LATCH SX

This latch provides a gate to set X6 and X7 positions of ROAR from X6 and X7 buffer latches.

CYCLE 0002

This is the normal execution of the ROS word at address 0002.

BINARY ADD

- The binary add instruction is in the RR format with an op-code of 1A.
- The second byte of the two-byte instruction contains the addresses of two of the general purpose registers located in local storage.
- The sum is stored in the first of the two general registers specified.
- The value in the second register remains intact.
- The adding routine loops while progressing across the register values.
- After adding the last bits, the operation moves into a set condition routine to indicate overflow and sign conditions.

You have see the many parts that, put together, make up the microprogram. To tie these pieces together, let's work our way through a microprogram for a fixed point binary add.

The instruction for a binary add is written in RR format. The Op code for Fixed-Point Binary Add is 1A in hexadecimal. RR format, if you will remember, is two bytes in length. The first byte is the Op code. The second byte of the instruction consists of two general purpose register addresses in hexadecimal.

In the example you will be working through, assume that the data in general purpose register 5 must be added to the data in general purpose register 7. The instruction to accomplish this becomes

1 A 7 5

0001 1010 0111 0101.

The first byte is the Op code 1A. The last byte represents the addresses of the two registers.

Let's briefly review the addressing of a general purpose register. A register contains four bytes of data. Since only one byte of data is addressable at a time, the N-register address must be constructed by

the microprogram. As an example: To address the units position of general purpose register 7, the N-register must be set to 0111 0111. The first four bits specify the register to use. The last four bits specify a particular byte of the register.

Before starting into the program itself, you should realize some functions that must be performed by the microprogram. The program must:

1. Read the instruction, analyze the format, determine the Op code.
2. Construct addresses to set the N-register starting with the units byte of each register.
3. Add four bytes of data from register 5 to the data in register 7.
4. Check for overflow conditions after the data has been added.
5. Set the condition register to indicate the status of the resultant answer (greater than, equal to, less than zero).
6. On overflow conditions, test program masks to determine if the condition should be ignored or not.
7. Branch to I-cycles, or to another microprogram if an overflow is unmasked.

The CAS sheets as written by a microprogrammer might appear as shown in Figures 3-3, 3-6, and 3-7. The description of each ROS word that is used to execute the instruction will be made in reference to the actual address which appears in the upper right corner of each block. These facts will be assumed before starting the example.

1. The instruction is

1	A	7	5
0001	1010	0111	0101

2. The address of the instruction is in the IJ registers.
3. The data in register 5 is:

Byte 0	Byte 1	Byte 2	Byte 3
00000000	00000000	00000000	01011101

4. The data in register 7 is:

Byte 0	Byte 1	Byte 2	Byte 3
00000000	00000000	00000000	10011001

5. The L and S registers are zero.

Objective: The answer in register 7 as a result of the addition should be: 00000000 00000000 00000000 11110110. Using Figure 3-3 let's determine how the first function, the reading and decoding of the instruction, is accomplished. The first ROS word to be executed is at address 0100 at figure location A2. Had it been necessary to change the instruction counter or test for interrupts, a ROS word at address 0101, 0102, or 0103 would have been executed.

ADDRESS 0100 (FIGURE 3-3): The expression IJ->MN MS on the S line brings up control lines to read the first byte of the instruction from core storage. The address in the I- and J-registers is set into the MN register. Main storage is specified by MS on this line. Once core has been addressed to read out the first byte, the address in IJ can be updated for reading the next byte of the instruction. To be more explicit, only the J-register need be increased by 1 because all instructions start at an even address.

Assume that byte 01FE in main storage is to be addressed by the I- and J-registers. The I- and J-registers will then contain the address

I	J
00000001	11111110.

To address byte 01FF it is only necessary to add the value of 1 to the J-register. The registers now contain:

I	J
00000001	11111111.

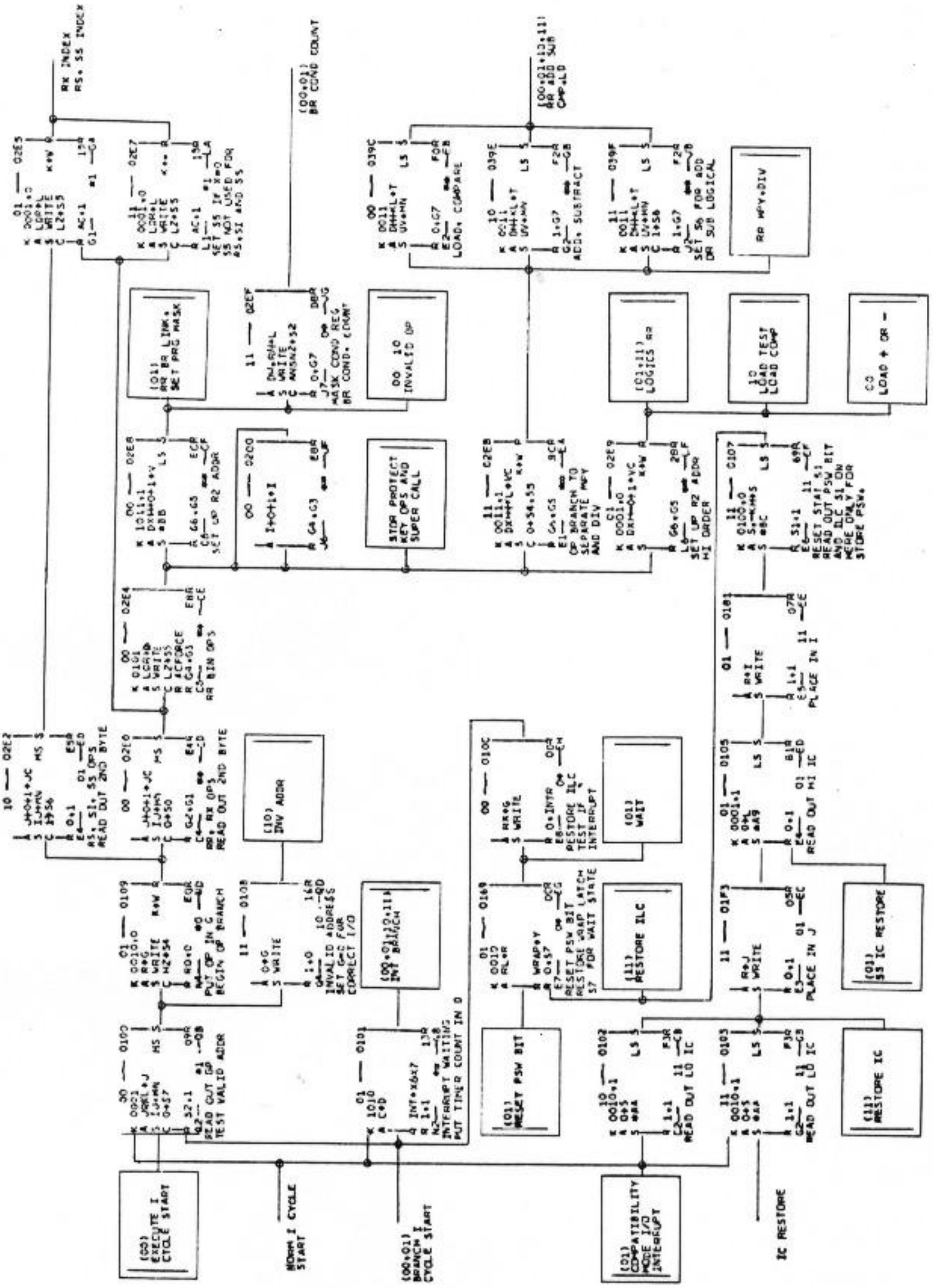


Figure 3-3. I-cycle Start (Binary Add)

Should the value of 1 be added to the J-register again, the resultant address in I and J is 00000001 00000000, or 0100. Thus, if the value 1 is added to the J-register when it is odd, it is necessary to take into account the possibility of a carryout which might affect the I-register address. Ther can be no carryout when adding 1 to the J-register address when the J-address is an even number.

The statement in the ROS word to update the J-register is J+KL->J, where K has a value of one. The A-register input of ALU is the data in the J-register, and 0001 0001 is set into the B-register from the K-field. The A-register is gated directly to the ALU. Only the lower half of the B-register is gated to the ALU. The inputs are OR'ed, and the result is a 1 bit in the low-order position of the ALU, which is then gated to the J-register.

The expression on the C line is 0->S7. This statement brings up control lines to set position 7 of the S register to 0. The function performed by this statement has little bearing on our operation. It is used in an indexing routine for RX format. This brings up an important point. In any ROS word, a statement such as 0->S7 may be used that seems to have no relation to what is being done. However, it may be used further in the microprogram and should not be ignored.

The expression on the R line is S2,1. Remember that when the box format was discussed, this line was used for branching. If you look at the output line from this box, you will see that there are two ROS words that may be executed next. They are the ROS words at addresses 0109 or 010B. The expression S2,1 must somehow control a decision circuit. The convenient place to make this decision is the ROAR address itself. The two low-order positions of ROAR, X6 and X7, are controlled for branching purposes. To see how this is done, first convert the addresses of the two ROS words to binary.

```
                                        XX
                                        67
Address 0109 in binary is 0001 0000 1001
Address 010B in binary is 0001 0000 1011
```

On the R line, the left portion of the expression controls the X6 position of ROAR. To carry this one step further, the S2 portion of the expression will determine the status of X6. For this example, position 2 of the S register is zero, so the X6 position of ROAR will be set to 0. The 1 in the expression (S2,1) forces the X7 position of ROAR to a one. A 01 branch is taken to address 0109. Notice that on the top line of the box for address 0109 you see 01. These are the two low-order bits of the actual address. Had position 2 of the S register been set, X6 would be set to a one and a 1,1 branch would be executed to address 010B.

ADDRESS 0109: The first byte which was read and set into the R-register is transferred to the G-register by the expression R->G. The G-register is interrogated later in the program to determine the Op code. The data movement from the R-register to the G-register is through ALU. The output of ALU feeds the Z bus.

On line C, the expression, HZ->S4, brings up control lines that check the four high bits of the Z bus for zero. Position 4 of the S-register is set to a one if the high bits are zero. Since the data on the Z bus is:

```
          1     A

       0001  1010
```

S4 is not set. The S4 bit is interrogated in a Branch and Link routine and has no bearing on our example.

Because core readout is destructive, the information in the R-register is returned to core by the statement, WRITE.

The expression on the S line is K->W. The W-register, remember, controls the high-order positions of the ROAR address. The CK control field (K) value sets the W-register to the value shown on the K line of this block.

On the lower R line, an R0,0 branch is executed. Since R0 is off in our example, the branch is to address 02E0. This branch determines that the RR or RX format will be used.

All the ROS words until now have been at addresses 01XX. When the second high-order position of the ROS word address changes value, the W-register must be set to a new value. Since we are at address 01XX and must step to 02XX, the expression K->W is used. Notice, the K line specifies the binary value of 2. The high six bits of E0 came from the CN field of address 0109. X6 and X7 controlled the two low bits.

ADDRESS 02E0: The next byte of the instruction is read from core by the expression IJ->MN MS. This is the byte that contains the addresses of the two general purpose registers.

Once the MN registers have been set, the J-register is again updated by the expres-

sion J + 0 + 1->JC. Notice that a new element has been added to the arithmetic statement. The C to the right of the arrow allows a carryout (if there is one) to set the third position of the S-register. If no carryout results, the S3 position is set to zero. This is necessary because, should a carryout result, the I-register address portion must also be updated. At this point there is a carryout, and S3 is set to a one.

The C line of the ROS word causes the control lines to set S0 to zero. The 0 position of the S-register is a control for true or complement add when the arithmetic operation is undetermined (‡). If S0 is zero, the arithmetic operation is a true add. If S0 is a 1, the operation is complement.

The G-register positions 2 and 1 are interrogated by the expression G2, G1. The data in the G-register is:

1 A

0001 1010

Finding that G1 = 0, and knowing that G0 = 0, we find that this Op code must be in RR format. RX Op codes begin with 01, RS with 10, and SS with 11.

Because G2 and G1 are both zero, a 00 branch is taken to address 02E4.

ADDRESS 02E4: The data in the R-register is again returned to core by the WRITE expression.

The arithmetic expression LΩR->D will OR the data in the L- and R-registers and transfer the resultant answer to the D-register. The symbol for the OR function is the omega. The L-register is always zero on entering I-phase except for the EXECUTE Op code. Since the L-register is zero and the R-register contains the second byte of the instruction, the D register is set 01110101.

The low-order four bits of the Z bus are checked for a zero condition by the expression LZ->S5. S5 is set to a 1 if the data on the low portion of the Z bus is zero. Because the data on the low portion of the Z bus is 0101, S5 is not set to one.

A test is made on other positions of the G-register to further decode the instruction. Looking at Figure 3-4, we see that by checking G4 and G3 our Op code must now be add, subtract, multiply, or divide. The G4 and G3 positions should set X6 and X7 to 11, the low order bits of address 02EB, but the AC-force condition overrides this and forces a branch to address 0200.

ADDRESS 0200: The only function performed by this word is I+0+1->I. This function updates the I register at the times when there is an address carry from the J-register. After updating the I-register, the microprogram again branches on the G4 and G3 positions to address 02EB.

At this time, check the data in the registers. The D-register contains 0111 0101, which is the specification for registers 7 and 5. The G-register contains 0001 1010, the Op code. The S3 bit is set to a 1, all other positions of the S register are still zero.

ADDRESS 02EB: Before any data from the general purpose registers can be added, the microprogram must set up the address of each register. The address for the low-order byte of general purpose register 5 is set up by the expression DXH + KL->VC. See Figure 3-5. To address the low-order byte, the N-register must be set to:

Reg 5 Byte 3
0101 0011.

In the expression DXH + KL->VC, consider the DXH portion first. The A-register input of ALU is set with the data from the D-register. The data in the A-register is now:

High Low
0111 0101.

Next, the output from the A-register is crossed (X) so that the data is:

High Low
0101 0111

The data is further controlled by the H. The H specifies that only the high portion of the data is to be used as A source data. The A source data to ALU then becomes, 0101 0000.

The B source input to ALU is controlled by the KL portion of the expression. K represents a value in the CK ROS control field. The constant is 3 and is shown in binary form on the K line of this CAS block. The B-register is set with the data 0011 0011. Only the low portion (L) of the B-register data is gated to ALU. The B source data is 0000 0011.

		RR				RX				RS		INV		SS			
		FIXED		FLT. PT.		FIXED		FLT. PT.						HI OPS		LO OPS	
	BITS	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	↓ / →	0123															
	4567	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000		Load Positive	Load Positive	Load Positive	Half Store	Store	Store D	Store S	Set Sys Mask	Store Multiple						
1	0001		Load Negative	Load Negative	Load Negative	Load Address					Test Under Mask				Move Numeric		Move With Offset
2	0010		Load & Test	Load & Test	Load & Test	Store Char.				Load PSW	Move Char.				Move		Pack
3	0011		Load Complement	Load Complement	Load Complement	Insert Char.				Diagnose					Move Zone		Unpack
4	0100	Set Prog Mask	AND	Halve	Halve	Execute	AND			Test I/O	AND				AND		
5	0101	Branch & Link	Compare Logical			Branch & Link	Compare Logical			Accept	Compare Logical				Compare Logical		
6	0110	Branch on Count	OR			Branch on Count	OR			Branch X HI	OR				OR		
7	0111	Branch on Cond	XOR			Branch on Cond	XOR			Branch X LO-EQ	XOR				XOR		
8	1000	Set Tag	Load	Load	Load	Half Load	Load	Load D	Load S	Shift RT S L	Load Multiple						Zero and Add
9	1001	Insert Tag	Compare	Compare	Compare	Half Compare	Compare	Compare	Compare	Shift Left S L							Compare
A	1010	Monitor Call	Add	Add N D	Add N S	Half Add	Add	Add N D	Add N S	Shift RT S A							Add
B	1011		Subtract	Sub N D	Sub N S	Half Sub	Sub	Sub N D	Sub N S	Shift Left S A							Subtract
C	1100		Multiply	Mult D	Mult S	Half Multiply	Mult	Mult D	Mult S	Shift RT D L	Start I-O				Translate		Multiply
D	1101		Divide	Divide D	Divide S		Divide	Divide D	Divide S	Shift Left D L	Test I-O				Translate and Test		Divide
E	1110		Add	Add U D	Add U S	Convert to Dec	Add Logical	Add U D	Add U S	Shift RT D A	Halt I-O				Edit		
F	1111		Subtract Logical	Sub U D	Sub U S	Convert to Bin	Subtract Logical	Sub U D	Sub U S	Shift Left D A	Test Channel				Edit and Mark		

Figure 3-4. Op Codes

The result of adding the B source data to the A source data is set into the V-register.

```
  A source data    = 0101  0000
+ B source data    = 0000  0011
-----------------------------
                     Reg 5 Byte 3
V-register data  = 0101  0011
```

The C-line of the block insures that the S-register S4 and S5 positions are blank (0) before proceeding.

The S-line in this block has the statement K->W. Since the K-field contains a three, the next address to be used will be 03XX.

The R-line is a branch on G6 and G5. This branch further breaks down the Op code and for this operation, the branch is to address 039E indicating an add or subtract Op.

ADDRESS 039E: The arithmetic statement DH+KL->T sets up the units address of register 7 in the T-register. Again consider the first portion of this expression, DH.

The A-register is set with the data in the D-register:

```
High  Low
0111  0101.
```

Only the high portion (H) is presented to ALU. A-source data is therefore 0111 0000. Again, the expression KL brings up the control lines to use the CK field constant of 3. The B source data is 0000 0011 because only the low portion (L) is gated to ALU.

```
  A source data    = 0111  0000
+ B source data    = 0000  0011
-----------------------------
                     Reg 7 Byte 3
T-register data  = 0111  0011
```

The expression UV->MN LS addresses core to read out the first byte from general purpose register 5. The data read out is 01011101. Local storage, rather than main storage, is specified by the LS portion of the expression.

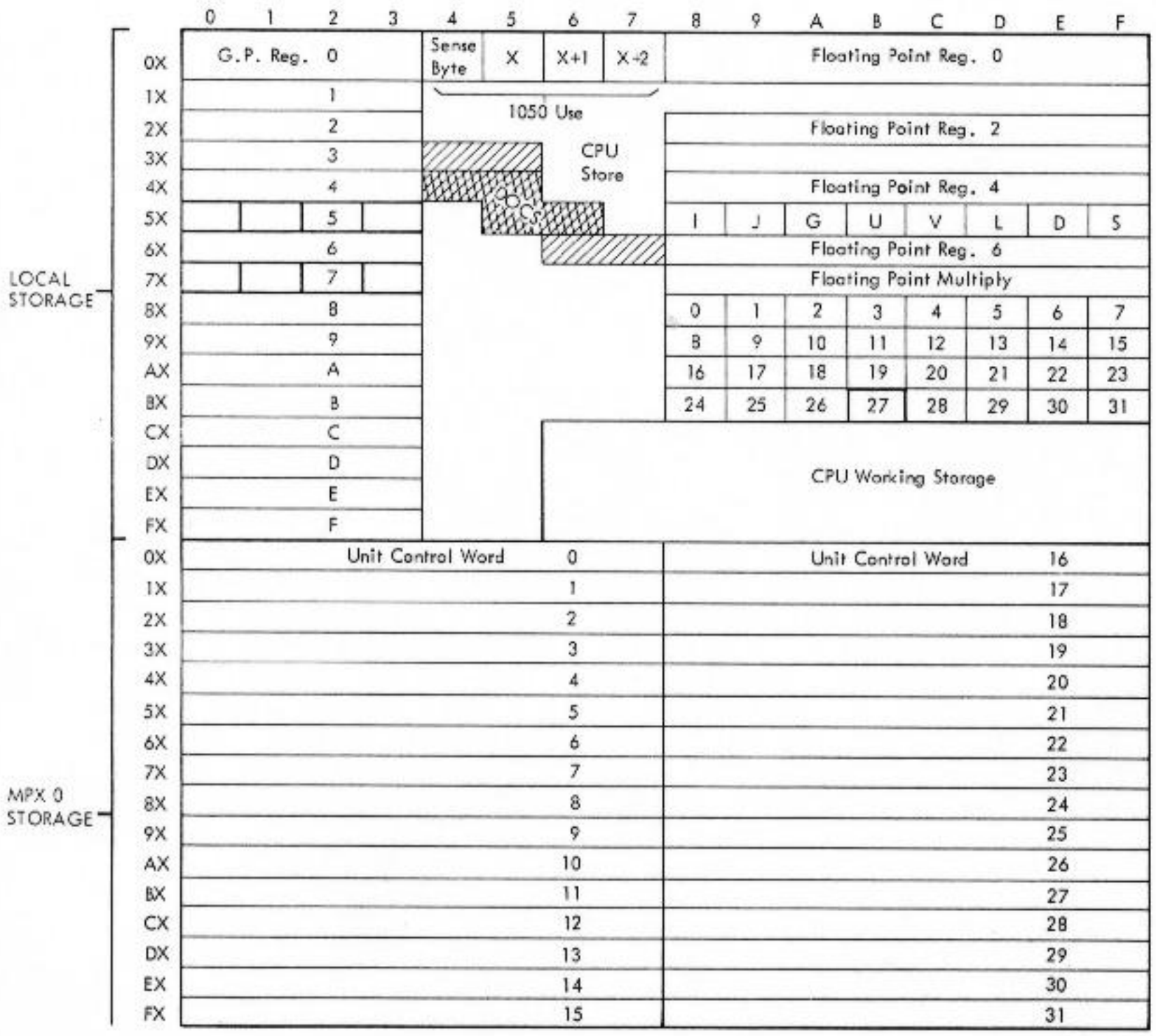


Figure 3-5. Auxiliary Storage Map

The last bit of the Op code is checked by the expression 1, G7. Since G7 is a zero, a branch is executed to address 03F2. The microprogram has fully decoded the G-register data to determine that the Op code must be a Fixed-Point Binary Add in RR Format. While this was being done, we have been setting up register addresses and even read our first byte of data from register 5.

ADDRESS 03F2 (FIGURE 3-6): The first byte of data from general purpose register 5 must be stored before reading any data from register 7. This is done by the expression R->DC. The information in the D-register is no longer needed and it is replaced by the data from register 5. The first byte of data is regenerated by the expression WRITE. The "C" following the "D" will preserve a carry out of the ALU in S3 if there is one. Here there cannot be a carry so the result is a reset of S3 for future use.

The branch condition on the R-line forces a branch to address 03A3.

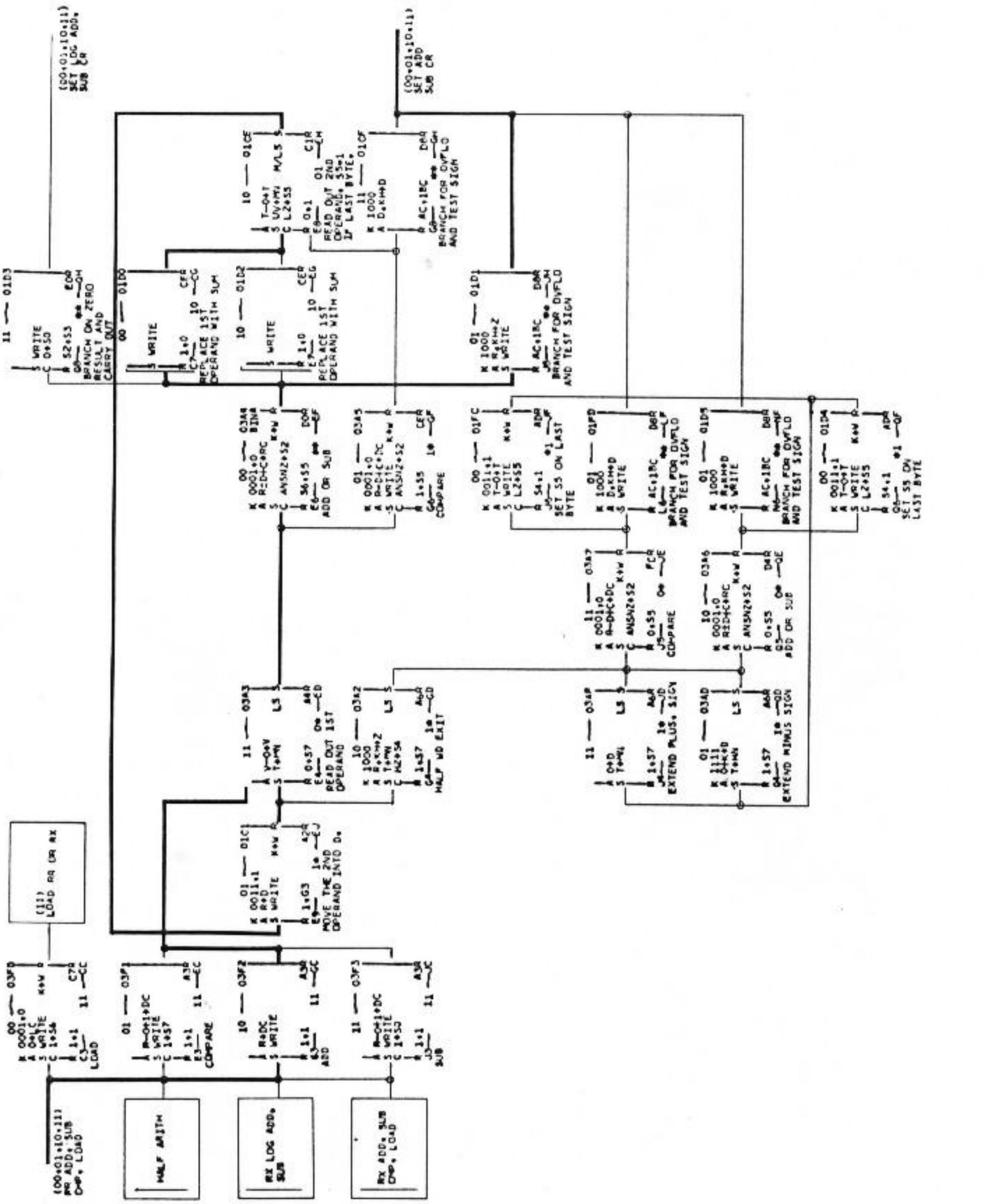


Figure 3-6. Binary Add

ADDRESS 03A3: The first byte of data from register 7 is read from core by the expression T->MN LS. The N-register is set by the data in the T-register. LS defines the core area addressed as local storage.

As this is being done, the expression V-0->V causes the value of one to be subtracted from the V-register. The V-register contains the address of byte 3 and must be changed before the next byte of data for this register is read. The example shows how ONE is subtracted by the expression V-0.

```
                        Reg 5 Byte 3
  V-register data = 0101  0011
  minus 0         = 1111  1111
  ----------------------------
                        Reg 5 Byte 2
V-register result = 0101  0010
```

As you can see, some artithmetic statements should be worked out in detail. If not, the wrong impression might be assumed from just reading the statement.

A 0,0 branch to address 03A4 is taken because S7 is still a zero.

ADDRESS 03A4: The first byte of data from register 5 and register 7 is added together by the expression R±D+C->RC. The C to the left of the arrow is a conditional carry insert. If the third position of the S-register is set to a 1, then a carry is inserted. The C to the right of the arrow allows a carry out that may result from the addition of the R and D data to set S3. The arithmetic operation, ±, is determined to be an add because the S0 position of the S-register is not set to a 1. Had it been the Subtract Op code, S0 would have been set to a 1 at ROS word address 03F3. S0 is the true or complement control position of the S-register. We know that it is a binary add rather than a decimal add because binary is specified on the K-line of the CAS block. The result of the addition is:

```
  D-register      = +01011101
  Reg 7 byte 3    = +10011001
  ---------------------------
R-register result =  11110110
```

The expression ANSNZ->S2 sets S2 to a one because the Z bus has on it the data 11110110. ANSNZ means Answer Non Zero. S2 is tested further in the program to determine whether our answer is plus, minus, or zero.

Since positions 5 and 6 of the S-register are zero, a 0,0 branch is executed to address 01D0. Because we are again changing the second high digit of our address, the expression K-->W is used. This time the value of K is 1 as shown on the K-line.

ADDRESS 01D0: The sum of the first byte from each register is regenerated (WRITE). The data in byte 3 of register 7 is now 11110110. An unconditional 1,0 branch is executed to address 01CE.

At this time, again review the location of the data and addresses in the registers.

1. The V-register contains Reg 5 Byte 2
   0101 0010
2. The T-register contains Reg 7 Byte 3
   0111 0011
3. Register 7 byte 3 data is 1111 0110.
4. The S2 position of the S-register is set to one.
5. The G-register contains 0001 1010

ADDRESS 01CE: The second byte of data from register 5 is read by the expression, UV->MN M/LS. M/LS can be either main core (M) or local storage (LS). This portion of the expression further checks the G- register. Since the G-register determines that the Op code is in RR format, only the control lines for local storage are brought up. The second byte of data read from register 5 is 000000000.

While register 5, byte 2 is read, there is no reason why the address of the next byte from register 7 cannot be set up. This is done by T-0->T, which subtracts one from the data in the T-register. The resultant answer in the T-register is:

```
Reg 7 byte 2
0111  0010.
```

The expression LZ->S5 does not set S5 to a one at this time. LZ is a check for zero on the four lower bits on the Z bus as a result of the arithmetic statement T-0->T. These four lower bits will not be zero until the last address of register 7 is obtained.

```
Reg 7 Byte 0
0111  0000.
```

A 0, 1 branch is taken to address 01C1.

ADDRESS 01C1: The byte of data just read is regenerated (WRITE). This data is also stored in the D-register, R->D. The D-register now contains 000000000, or byte 2 of register 7.

A 1,1 branch is executed to address 03A3. Position 3 of the G-register is a 1 because the Op code stored there is Add.

ADDRESS 03A3: Entering this address for the second time starts a loop in the microprogram. The loop continues until the four bytes of data from the two registers are added together.

The second byte from register 7 is read, (T->N LS).

The V-register address is changed to:

Reg 5 Byte 1
0101 0001.

A 0, 0 branch is executed to address 03A4 because S7 is still zero.

ADDRESS 03A4: The second byte of data from both registers is added and the result is stored in the R-register (R±D+C->RC). The result of this second addition is 000000000. S2 is not set to a zero (ANSNZ->S2) even though there is nothing on the Z bus because the S-register is not made up of polarity hold latches. It takes a definite reset expression to clear an S-register position to zero (0->S0).

S6 and S5 are again tested to determine the branch set up. Neither position has been set to one, therefore, the 0, 0 branch is again taken to address 01D0. K->W sets the W-register of ROAR to the value of one because of the high-order address change.

ADDRESS 01D0: The second byte of the added data is regenerated (WRITE). A 1, 0 branch is taken to address 01CE.

ADDRESS 01CE: Byte 1 of general purpose register 5 is addressed (UV->MN M/LS). The address for byte 1 of register 7 is set up (T-0->T).

S5 is still not set to a one because the data on the Z bus is:

Reg 7 Byte 1
0111 0001

Take the 0,1 branch to 01C1.

ADDRESS 01C1: Byte 1 from register 5 is stored in the D-register (R->D). It is also regenerated (WRITE). K->W is again used for the address change. Take the 1,1 branch to address 03A3.

ADDRESS 03A3: Byte 1 from register 7 is read (T->N LS). The address for general purpose register 5, byte 0 is obtained (V-0->V). Branch 0,0 to address 03A4 because S7 is still zero.

ADDRESS 03A4: Add byte-1 data from both registers (R±D+C->RC). S2 is still set to 1 and cannot be reset by the expression ANSNZ->S2. S6 and S5 are still zero. Branch to Address 01D0.

ADDRESS 01D0: Regenerate (WRITE) the sum to core. Branch 1,0 to address 01CE.

ADDRESS 01CE: Read the last byte of data from register 5 (UV->MN M/LS).

Change the address in the T-register (T-0->T).

	Reg 7	Byte 1
Old T-register address	= 0111	0001
minus 0	= 1111	1111
	Reg 7	Byte 0
New T-register address	= 0111	0000

The information on the Z bus as a result of the arithmetic statement is 0111 0000. The low-order four bits are 0000. The C-line of the box has the expression LZ->S5. S5 is now set to a 1 because the low position of the Z bus is zero (LZ). Advance to address 01C1.

ADDRESS 01C1: Store the last byte of data that came from register 5 (R->D). Regenerate this data (WRITE). Control the address change (K->W). Again check G3, set up a 1,1 branch to address 03A3.

ADDRESS 03A3: Address core and read the last byte of data from register 7 (T->N LS). Subtract one from the data in the V-register. This address, 0101 1111, is invalid for register 5 but it will not be used as we are in this loop for the last time. Check S7, which is still zero, and branch 0,0 to address 03A4.

ADDRESS 03A4: Add the last byte of data from both registers and store the result in the R-register (R±D+C->RC).

S2 is still set and cannot be reset by the expression ANSNZ->S2. K->W sets up an address change. Positions 6 and 5 of the S-register are tested. S5 had been set to a 1, therefore, a 0, 1 branch is taken to address 01D1.

ADDRESS 01D1: The last byte is stored in core (WRITE). Register 7 now contains the answer: 00000000 00000000 00000000 11110110. The data in the R-register for the last sum is 00000000. In the expression R • KH->Z, this data in the R-register is ANDed (•) with the K source high (H) and gated to the Z bus. The value of K on the K-line is eight (1000).

```
R data                  = 00000000
Constant                = 10000000
-------------------------------
ANDed result            = 00000000
```

On the branch line, R, you see the two mnemonics AC and 1BC. AC (ALU carry) brings up control lines to test for a carryout condition of ALU as a result of the arithmetic expression executed in the previous ROS word (Address 03A4, expression R+D+C->RC). 1BC (one-bit carry) brings up the control lines to test for a carry into the highest position of ALU as a result of the previous arithmetic statement. To show the positions of ALU effected, assume this data:

```
A-register  =  0100        0000

B-register  =  1100        0000
-------------------------------
               11 carries

ALU output  =  0000        0000

AC
1BC
```

The previous expression R+D+C->RC added the last two bytes of data from both registers. Both bytes of data were zero, therefore the output from ALU was zero. We had neither an ALU carry nor a 1-bit carry. A 0,0 branch is taken to address 01D8 which is in Figure 3-7.

The AC and 1BC mnemonics test to determine overflow conditions. An overflow condition would have caused branching to either address 01D9 or 01DA.

ADDRESS 01D8: The expression * BB LS addresses a byte of local storage. See Figure 3-5. This byte contains the condition code and program mask bits. Certain bits are set according to whether the answer is equal to, greater, or less than zero, plus overflow conditions.

Program masks are checked further in the program.

To address a byte in local storage, the N-register format is:

```
N0  N1  N2   N3  N4  N5  N6  N7
1,  0,  CN0, K0, 1,  K1, K2, K3.
```

The N0 and N1 positions are set 1, 0 respectively. The 2 position of the N-register is set by the CN ROS control field, 0 bit position. Position N3 is set by the CK ROS control field 0 position. N4 is set to a 1 unconditionally. N5, N6, and N7 are set by the remaining positions of the CK field. Figure 3-5 shows that the coordinates BB address byte 27. BB in binary is:

```
 B     B
1011  1011
```

Match this with the address format, and you see that the CK field must be coded 1011. This is the value that appears on the K line of the CAS block.

```
Format = 1, 0, CN0, K0, 1, K1, K2, K3,
Value BB = 1011  1011
```

The control line expression 0->S0 sets S0 to zero in case we had been in a complement operation.

The branch tests are S2 and Z = 0.

S2 was set to a 1 because we had significant data. The expression Z = 0 checks the Z bus for a zero as a result of the previous expression on the arithmetic line (R•KH->Z). Had the resultant answer been minus, the expression R•KH->Z would have provided a 1-bit output for the highest position of ALU. Because our answer is positive, the Z bus is zero and a 1,1 branch is executed to address 01EB. Had our answer been minus, the Z = 0 expression would have set X7 to a 0. A 1,0 branch would have taken us to address 01EA.

ADD, SUBT, CMP	BITS STORED IN AR HIGH
=, ZERO	1000
LO, <0	0100
HI, >0	0010
OVERFLOW	0001

N SET = 1, 0, CND, K0, K1, K2, K3

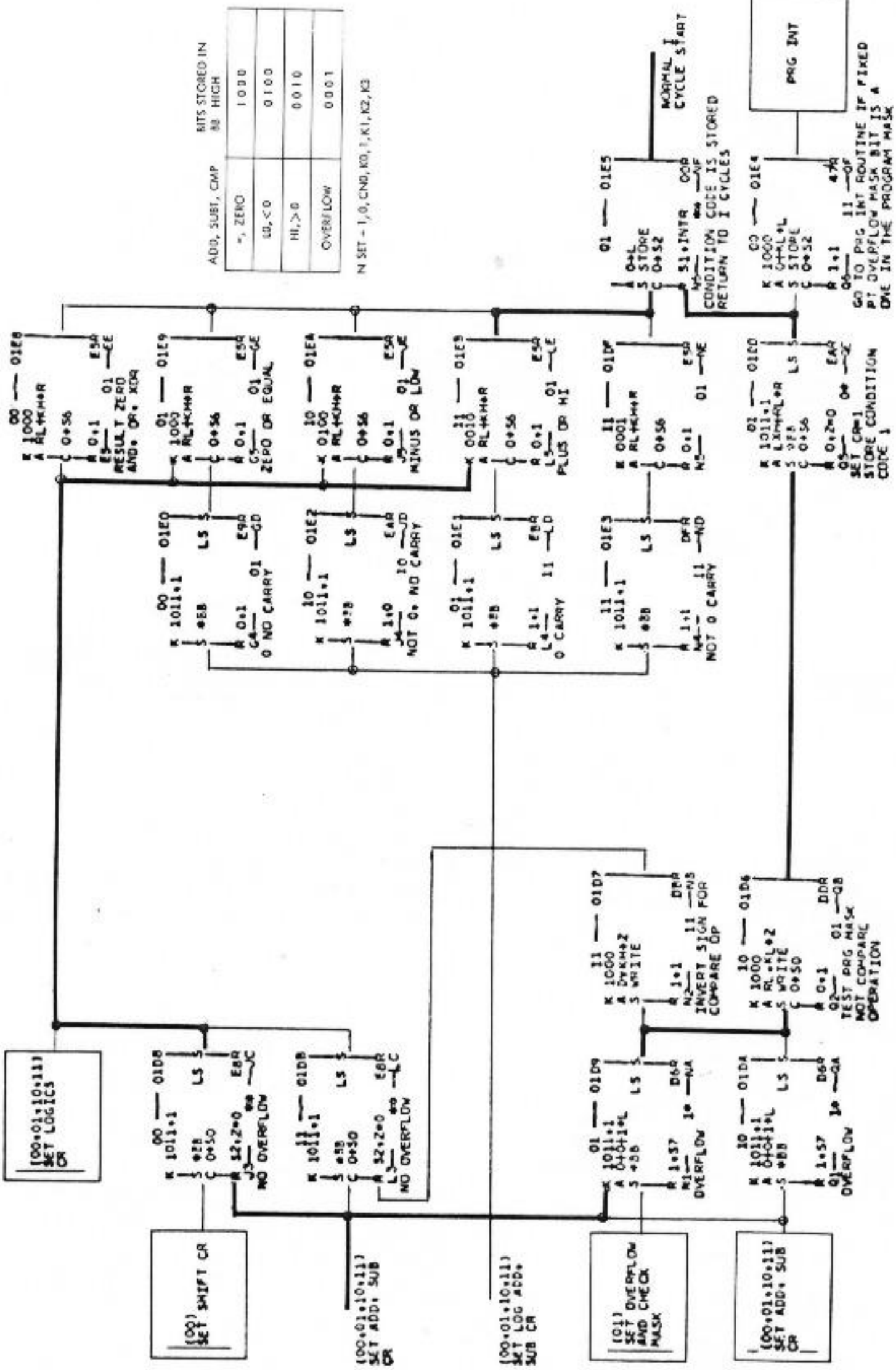


Figure 3-7. Set Condition Register

ADDRESS 01EB: The byte just read from core contains information pertaining to condition codes and program masks. The high 4-bit positions are for the condition code settings according to the answer of the problem. The expression RL + KH->R presents the data previously read to the A-register. A constant (K) sets the B-register to the value of 2 as specified by line K. The low portion (L) of the data in the A-register is used as the A source for ALU. The high position (H) of the B-register data is used as the B source for ALU. The data is then:

A source	0000	xxxx x are the program mask bits
B source	0010	0000
R-register set	0010	xxxx

The four high bits (0010) when returned to core signify that our resultant answer was greater than zero. Note the table in Figure 3-7.

The expression 0->S6 sets S6 to a zero.

A 0,1 branch is taken to address 01E5.

ADDRESS 01E5: The mnemonic, STORE, returns to core the information that is in the R-register. This consists of a new condition code, and the original program mask bits.

The L-register is set to zero by the expression 0->L.

Position S2 of the S-register is set to zero (0->S2).

The branch mnemonics test S1 and interrupt. S1 is not set. If there is an interrupt, a 0, 1 branch is taken to address 0101, Figure 3-3. If no interrupt exists, a 0,0 branch is executed to address 0100, Figure 3-3. Address 0100 is the ROS word where this operation began. Address 0101 is the beginning of a microprogram to test the interrupt and determine what it is (selector channel, multiplex, etc.).

This completes the Add operation. However, to carry the microprogram one step further, assume that the result of adding the two registers produced an overflow condition. An overflow makes it necessary to set a different condition code before returning to I-cycles (see chart insert on Figure 3-7). The problem program can, at some later time, branch on the condition code set.

Start at address 01D9 in Figure 3-7.

ADDRESS 01D9: The local storage byte is read out by *BB LS. The L-register is set to 00000001 (0+0+1->L).

Since S7 is not set to a 1 a 1,0 branch is taken to address 01D6.

ADDRESS 01D6: The data just read consists of the condition code and program mask bits. It is regenerated to core (WRITE). The arithmetic expression RL•KL->Z tests the program mask bits by allowing or preventing a bit on the Z bus. Assume the data in R is xxxx 0yyy. The x positions are those for the condition code. The 0 means that this position is not set. The y positions are the remaining program mask bits.

The data for this expression is:

A source data (RL) =	0000	0000
B source data (KL) =	0000	1000
ANDed ALU output =	0000	0000

Position zero of the S-register is set to zero (0->S0).

ADDRESS 01DD: Again, the same byte of information is read by the expression: *BB LS.

The condition code is set by the expression LXH+RL->R. The data in the L-register sets the A-register. The A-register now contains:

High	Low
0000	0001

This is crossed (X)

High	Low
0001	0000

and only the high (H) portion is used for the A source to ALU. The data in the B-register is xxxx 0yyy. Only the low portion is presented to ALU (RL). The result of the addition becomes:

A source =	0001	0000
B source =	0000	0yyy
R-reg set =	0001	0yyy

The C-line statement, 0->S6, sets position 6 of the S-register to a zero.

The branch conditions are 0 and Z = 0. Z = 0 brings up control lines to check the Z bus as a result of the arithmetic statement executed in the previous ROS word. This is how the program mask condition is checked. Our output was 0000 0000 as a result of the expression RL • KL->Z. Therefore, Z = 0 sets X7 to a 1. A 0,1 branch is taken to address 01E5 because the overflow was masked off.

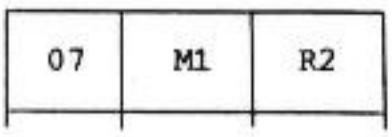
ADDRESS 01E5: The data in the R-register is returned to core (STORE). The four highest bits are the new condition code:

High	Low
0001	0yyy

0001 is the coding for an overflow. The L-register is set 0000 0000 by the expression 0->L. The S2 position is set to zero and the branch conditions are S1 set to zero and no pending interrupt. This branches the microprogram to address 0100 on Figure 3-3.

BRANCH ON CONDITION (RR FORMAT)

- The updated instruction address is replaced by the branch address (R2) if the bit that is on in the condition code is also on in the mask.
- The M1 field is used as a four-bit mask.
- Normal instruction sequencing proceeds with the updated instruction address if the condition is not met.



The branch will be successful whenever the condition code has a corresponding mask bit of one. The four bits of the mask correspond with the condition code as follows:

Condition Code Bit	Op Code Bit (Mask)
0	8
1	9
2	10
3	11

Description of a Branch-on condition instruction, RR format, with reference directly to the microprogram follows. Assume the previous instruction was a fixed point add which set bit 3 of the condition code, indicating an overflow. A Branch-on condition is issued with bit 11 of the Mask field at one.

Starting at ROS address 0100 Figure 3-8, the Branch-on condition Op code is read out. The Op code is placed in the G-register at ROS address 0109.

ADDRESS 02E0: The second byte of the instruction is read out and a branch on G-register bits G2 and G1 is taken. This determines that the operation is in RR format.

The second byte is placed in the D-register and S5 is set to 0 if the R2 field is not zero. If the R2 field had been zero, S5 would have been set to one and a no-branch condition would have resulted at address 02D9 (Figure 3-9).

ADDRESS 02E8: The R2 portion of the instruction is placed in the V-register in the event the branch address must be fetched. The condition code is read from local storage. A branch to 02EF is made by testing G-register bits 6 and 5.

ADDRESS 02EF: The mask portion of the instruction is ANDed with the condition code and the result is placed in the L-register. S2 is set to one if the result was not zero. A branch is made on G-register bit 7 which results in going to address 02D9 (Figure 3-9) .