IBM®

Field Engineering

Maintenance Diagrams

Restricted Distribution

This manual is intended for internal use only and may not be used by other than IBM personnel without IBM's written permission.

2091 Processing Unit -- Volume 4

I, FXP, FLP Operations

PREFACE

This is one of five volumes of the IBM 2091 Processing Unit, Field Engineering Maintenance Diagrams Manual (FEMDM). The organization of the FEMDM, the general content of each volume, and the form numbers of the five volumes are:

Title	Contents
Volume 1 - Diagnostic Techniques, ECAD's (Form Y22-6671)	DIAGNOSTIC TECHNIQUES Diagrams 1-1 to 1-XX ERROR CONDITIONS Diagrams 2-1 to 2-XX
Volume 2 - Data Flow; I, FXP, FLP, Functional Units (Form Y22-6672)	DATA FLOW Diagram 3-1 System Data Flow 3-2 I Unit Data Flow 3-3 Fixed Point Data Flow 3-4 Floating Point Data Flow 3-5 MSCE Data Flow 3-6 PSCE Data Flow 3-7 MC Data Flow FUNCTIONAL UNITS Diagrams 4-1 to 4-XX I Unit 4-100 to 4-1XX Fixed Point Unit 4-200 to 4-2XX Floating Point Unit
Volume 3 - MSCE, PSCE, MC, Functional Units (Form Y22-6673)	FUNCTIONAL UNITS (Cont'd) 4-300 to 4-3XX MSCE 4-400 to 4-4XX PSCE 4-500 to 4-5XX MC
Volume 4 - I, FXP, FLP Operations (Form Y22-6674)	OPERATIONS 5-1 to 5-XX I Unit 5-100 to 5-1XX Fixed Point 5-200 to 5-2XX Floating Point
Volume 5 - MSCE, PSCE, MC Operations; Power (Form Y22-6675)	OPERATIONS (Cont'd) 5-300 to 5-3XX MSCE Operations 5-400 to 5-4XX PSCE Operations 5-500 to 5-5XX MC Operations POWER SUPPLIES 6-1 to 6-XX

Diagrams contained in this manual are referenced from the seven 2091 Field Engineering Theory of Operation Manuals (FETOM's). References to FEMDM diagrams take the form "Diagram 5-103"; references to figures in a FETOM take the form "Figure 3-22." The seven 2091 FETOM's are:

IBM 2091 Processing Unit, FE Theory of Operation Manuals:

System Introduction, Instruction Processor,
Form Y22-6622

Power Supplies and Control,
Form Y22-6623

Console and Maintenance Features,
Form Y22-6624

Fixed Point Execution Element,
Form Y22-6625

Main Storage Control Element,
Form Y22-6626

Peripheral Storage Control Element,
Form Y22-6627

Floating Point Execution Element,
Form Y22-6628

Other FE Manuals containing information pertinent to the 2091 are:

2091 Processing Unit, FE Maintenance Manual, Form Y22-6659

2091 Processing Unit, 2395 Processor Storage, FE Installation Manual, Form Y22-6634

Advanced Solid Logic Technology Packaging, Tools, Wiring Change and Repair Procedures, FE Theory-Maintenance Manual, Form Y22-6620

Solid Logic Technology, Packaging, Tools, Wiring Change Procedure, FE Theory of Operation Manual, Form Y22-2800

Solid Logic Technology, Component Circuits, FE Theory of Operation Manual, Form Z22-2798*

Solid Logic Technology, Power Supplies, FE Theory of Operation Manual, Form Y22-2799

* Available to authorized IBM employees only.

First Edition (September 1967)
Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or FE Supplements.

This manual has been prepared by the IBM System Development Division, Product Publications, Dept. B95, PO Box 390, Poughkeepsie, N.Y. 12602. A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be sent to the above address.

ABBREVIATIONS

A	AND
AC	Address Check
Acc	Access; Accumulator
Acpt	Accept
Acptng	Accepting
Addr	Address
Adr	Address
Adv	Advance
AE	Address Exception
ALD	Automated Logic Diagram
Altr	Alteration
Amt	Amount
AOC	Array Out Counter
AR	Amplifier
Arg	Argument
Arg Wd	Argument Doubleword
AS	Accept Stack
ASLT	Advanced Solid Logic Technology
ATI	Auxiliary Tape Input
Avail	Available
B	Bit
BAB	Byte Address Buffer
BAC	Buffer Address Counter
BAL	(Mnemonic)Branch and Link (RX)
BALR	(Mnemonic) Branch and Link (RR)
BAR	Byte Address Register
BB	Bank Bit
BC	(Mnemonic) Branch on Condition; Bus Control
BCR	(Mnemonic) Branch on Condition (RR)
BCQT	Branch on Condition Quick Trigger
BCT	(Mnemonic) Branch on Count (RX)
BCTR	(Mnemonic) Branch on Count (RR)
BCU	Bus Control Unit
BCUNCONT	Unconditional Branch Trigger
Bd	Board
Bdy	Boundary
Bfr	Buffer
BIA	Branch In Array
BIAT	Back in Array Trigger
BOM	Basic Operating Memory
Br	Branch
BRT	Branch Trigger
BSM	Basic Storage Module
Bsy	Busy
BXH	(Mnemonic) Branch on Index High (RS)
BXLE	(Mnemonic) Branch on Index Low or Equal (RS)
BXQT	Branch on Index Quick Trigger
BZ	Busy
BZTP	Busy-to-Priority
BZTPSCE	Busy-to-PSCE
BZTR	Busy-to-Request
CAB	Channel Address Bus
CAR	Console Address Register;
CAR	Channel Address Register
CAW	Channel Address Word
C BACKL8 T	Condition Back Less than Eight Trigger
C BIA T	Conditional Back in Array Trigger
CBR	Console Buffer Register
CC	Command Counter; Condition Code
CCC	Common Channel Control
CCW	Channel Command Word
CD	Chain Data
CDB	Common Data Bus
CDBI	Console Data Bus In
CDBO	Console Data Bus Out
Ch	Channel
Chan	Channel
Ch Fr	Channel Frame
Chk	Check
Ck	Check
Chn	Chain
CIn	Carry In
CLC	(Mnemonic) Compare Logical (SS)
Clk	Clock
CM	Conditional Mode; Console Mode; Cripple Mode
Cncl	Cancel
Cndl	Conditional
Cnt	Count
CO	Conditional Op
Comp	Compare; Comparator
Cond	Condition
COut	Carry Out
CPA	Carry Propagate Adder
CPC	Cyclic Program Counter
CPE	Central Processing Element
Cpr	Computer
CPU	Central Processing Unit
C QUICK T	Conditional Quick Trigger
CR	Control Register
Crip	Cripple
CSA	Carry Save Adder
CSW	Channel Status Word
Ctl	Control
Ctr	Counter
Ctrl	Control
CV	Converter
CVB	(Mnemonic) Convert to Binary (RX)
CVD	(Mnemonic) Convert to Decimal (RF)
CXR	Console Auxiliary Register
D	Displacement
Dbl	Double
DC	Data Check; Display Check
Dcd	Decode
Dcdr	Decoder
Des	Designation
Det	Detection; Detector
DG	Display Gate
Diag	Diagnose
DIG	Data Ingate
Disp	Displacement
Dist	Distributor
Div	Divide
Dly	Delay, Delayed
Dlyd	Delayed
DM	Diagnostic Monitor
DOG	Data Outgate
DPC	Display Parity Check
Dsbl	Disable
Dt	Data
DW	Doubleword
DWC	Doubleword Counter
DWCR	Doubleword Count Register
EBA	Ending Byte Address
EBAR	Ending Byte Address Register
EBCDIC	Extended Binary Coded Decimal Interchange Code
EC	Engineering Change
ECAD	Error Check Analysis Diagram
ED	(Mnemonic) Edit (SS)
EDMK	(Mnemonic) Edit and Mark (SS)
EMS	Extended Main Storage (Same as LCS)
Eq	Equals
Err	Error
EX	(Mnemonic) Execute (RX)
Excpn	Exception
Exce	Execute
Exp	Exponent
FAU	Floating Point Add Unit
FE	Field Engineering
FEMDM	Field Engineering Maintenance Diagram Manual
FETOM	Field Engineering Theory of Operation Manual
FIFO	First-In, First-Out
Fir	First
FIWADFO	First-In-With-Available-Data, First-Out
FIWAMFO	First-In-With-Available-Memory, First-Out
FLA	Floating Point Area
FLB	Floating Point Buffer
FLBB	Floating Point Buffer Bus
Fld	Field
FLEU	Floating Point Execution Unit
FLIU	Floating Point Instruction Unit
FLOS	Floating Point Op Stack
FLP	Floating Point
FLR	Floating Point Register
FLRB	Floating Point Register Bus
FLU	Floating Point Unit
FMDU	Floating Point Multiply/Divide Unit
FP	Fetch Protect
FPA	Floating Point Area
Frm	Frame
Frac	Fraction
FS	False Start
FSB	Fixed Store Bus
Fth	Fetch
Fwd	Forward; Forwarding
FXA	Fixed Point/VFL Area
FXB	Fixed Point Buffer
FXEU	Fixed Point Execution Unit
FXIU	Fixed Point Instruction Unit
FXOS	Fixed Point Op Stack
FXP	Fixed Point

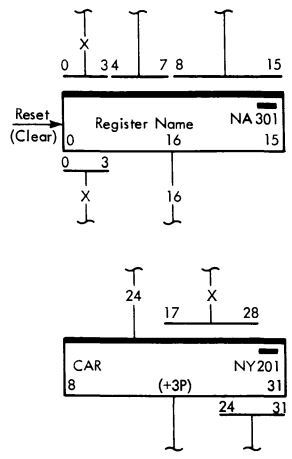
Gen	General; Generate
GP Acpt	General Purpose Register Accept
GPR	General Purpose Register
Gr	Group
Gt	Gate
Gtd	Gated
GWFCDB	Go When Full Common Data Bus
GWFFLBB	Go When Full Floating Buffer Bus
HIO	(Mnemonic) Halt I/O (SI)
HPMS	High Performance Main Storage
HOD	High Order Digit
HS	Half Sum
HSB	High Speed Bus
HW	Halfword
I	Instruction
I-Box	Instruction Processor
IC	(Mnemonic)Insert Character (RX); Instruction Count
IDR	Immediate Data Register
IF	Instruction Fetch
IFT	Instruction Fetch Trigger
Ig	Ingate
ILC	Instruction Length Code
IMRT	Instruction from Memory Request Trigger
Incr	Increment
Ind	Indication; Indicator
Inh	Inhibit
Init	Initialize
In Pri	Input Priority
Insn	Instruction
Int	Internal
Intr	Interrupt
Inv	Invalid
I/O	Input/Output
IOC	I/O Channel
IPL	Initial Program Load
IR	Instruction Register
IRCTR	Instruction Register Counter
ISK	(Mnemonic) Insert Storage Key (RR)
ISR	Instruction Sink Register
IWC	Indicator Word Counter
K	Thousand
L	Operand Length
LA	(Mnemonic) Load Address (RX)
Last	Last Trigger
LB	Lower Bound; Loop Block
LBCTR	Lower Bound Counter
L Cnt	Length Count
LCS	Large Capacity Storage (Same as EMS)
Ld	Load
LM	(Mnemonic) Load Multiple (RS)
LO	Low Order
LOD	Low Order Digit
LPSW	(Mnemonic) Load PSW (SI)
LSN	Load Multiple (LM), Store Multiple (STM), and NOXCM Instructions
Lth	Latch
MA	Multi-Access
MAC	Multi-Access Code
MALS	Multi-Access Link Suppressed
Man	Manual
MAR	Memory Address Register (Same as SAR)
MAT	Multi-Access Trigger
MC	Maintenance Console; Megacycle; Marginal Checking
Mcand	Multiplicand
MCW	Maintenance Control Word
M/D	Multiply/Divide
MDR	Memory Data Register (Same as SDR)
Mem	Memory
MG	Motor Generator; Multiple Gate
MOP	Multiple Operation
Mplr	Multiplier
Mple	Multiple
Mod	Modifier
Mply	Muliply
MS	Main Storage (Same as MWS)
MSB	Medium Speed Bus
MSC	Monolithic Storage Cell
MSCE	Main Storage Control Element
MSM	Main Storage Module
MTBF	Mean Time Between Failures
Mul Dec	Multiplier Decoder
MVC	(Mnemonic) Move (SS)
MVN	(Mnemonic) Move Numerics (SS)
MVO	(Mnemonic) Move with Offset (SS)
MVZ	(Mnemonic) Move Zones (SS)
MWS	Main Working Storage (Same as MS)

N	Inverter
NC	Mnemonic AND (SS)
Neg	Negative
NIAT	New Instruction Address Trigger
No.	Number
N Op	No Operation
Norm	Normalize
NOXCM	(Mnemonic) NC (AND) OC (OR) XC (Exclusive OR) CLC (Compare Logical) MVC (Move) MVZ (Move Zone) MVN (Move Numeric)
Ns	Nanosecond
NSI	Next Sequential Instruction
NUBAT	New Upper Bound Address Trigger
OC	(Mnemonic) OR (SS)
Oflo	Overflow
Og	Outgate
Olap	Overlap
Op	Operation
Opnd	Operand
OR	Outring (Line Name Only)
Out Pri	Output Priority
Ord	Order
Osc	Oscillator
Ovrd	Override
Ovrlp	Overlap
P	Parity; Position; Priority
PA	Propagate Adder
PACK	(Mnemonic) Pack (SS)
Par	Parity
PAR	Position Address Register
PAW	Position Address Word
PC	Parity Check
PDU	Power Distribution Unit
PG	Parity Generate
PH	Polarity Hold
PK	Protection Key
PM	Program Mask; Protect Memory (Same as PS)
Pos	Position; Positive
PPE	Peripheral Processor Element
PPln	Pipeline
Prec	Precision
Pred	Predict
Pri	Primary; Priority
Prob	Problem
Prog	Program
Prop	Propagate
Prot	Protect; Protection
PS	Protect Storage (Same as PM); Power Supply
PSCE	Peripheral Storage Control Element
PSW	Program Status Word
Ptrn	Pattern
Pty	Parity
PUMO	(Mnemonic) PACK (Pack) UNPK (Unpack) MVO (Move with Offset)
PV	Protection Violation
Pwd	Powered
Q	Queue
Qx	Queue (any number)
R	Ready
Rd	Read
RDD	(Mnemonic) Read Direct (SI)
Rdy	Ready
Rec	Record
Reg	Register
Rel	Release
Req	Request
Res	Reset; Residue
Resd	Reserved
Resp	Response
Rgen	Regenerate; Regeneration
RI	Read In
R/L	Remote/Local
RO	Read Out
RR	(Instruction Format) Both Operands from GPR's
RS	Request Stack; (Instruction Format) One Operand from a GPR, the Other from Storage
Rslt	Result
Rsrvtn	Reservation
Rt	Right
Rtn	Return
RUA	Register Unavailable for Address Generation
RUM	Register Unavailable for Modification

RX	(Instruction Format) One Operand from a GPR, the Other from an Indexed Storage Location
S0	State Zero
SA	Sink Address
SAA	Storage Address Alteration
SAB	Storage Address Bus
SAP	Storage Address Protection
SAR	Storage Address Register (Same as MAR); Store Address Register
SB	Sink Address Bus
SBI	Storage Bus In
SBO	Storage Bus Out
SC	Single-Cycle; Storage Channel; Sequence Complete
Sc	Source
SDB	Storage Data Buffer
SDE	Storage Distribution Element
SDR	Storage Data Register (Same as MAR)
Sel	Select
SERR	CPE Status Recording Program
SEVA	Systems Evaluation Program
S/F	Store/Fetch
Sh	Shift
Shftr	Shifter
SI	(Instruction Format) One Operand from Storage, the Other Is Immediate
SIAT	Store Into Array Trigger
SIIS	Store into Instruction Stream
SIO	(Mnemonic) Start I/O (SI)
SIT	Store Interlock Trigger
SK	Storage Key
Sk	Sink
S/L	Short/Long Precision
SLA	(Mnemonic) Shift Left Single (RS)
SLCB	Save Loop Close B Register
SLC	Save Loop Close
SLCIR	Save Loop Close Instruction Register
SLDA	(Mnemonic) Shift Left Double (RS)
SLCX	Save Loop Close - X Register
SLI	Suppress-Length-Indication
SLT	Save Loop Target; Solid Logic Technology
SM	Storage Module
SMAL	Suppress Multi-Access Link
Sng	Single
SO	Storage Operand
SP	Storage Protect; Single Pulse
SPAD	Select Parity and Display Counter
SPAR	Storage Protect Address Register
SPC	Storage Protect Check
SPF	Storage Protect Feature
SPM	(Mnemonic) Set Program Mask (RR); Storage Protect Memory
SP91	Protect Storage for System/360 Model 91
Sr	Source
SRA	(Mnemonic) Shift Right Single (RS)
SRDA	(Mnemonic) Shift Right Double (RS)
SS	Snapshot Register; Storage-to-Storage; Stepping Switch
S/S	Source/Sink
SSC	Selector Subchannel
SSK	(Mnemonic) Set Storage Key (RR)
SSM	(Mnemonic) Set System Mask (SI)
ST	(Mnemonic) Store (RX)
Stat	Station
STC	(Mnemonic) Store Character (RX)
Stg	Stage; Storage
Stk	Stack
Sto	Store; Storage
STOOP	Storage Operation
Stor	Store; Storage
Stp	Stop
STR	Source Tag Register
Sup	Suppress
SVC	(Mnemonic) Supervisor Call (RR)
SVIR	Save Instruction Register
SVR1	Save R1 Register
Sw	Switch; Switch Enabled
SW	Single Word
Syl	Syllable
Sync	Synchronize
Sys	System
T	Time
TAT	Time Address Trigger
Tbl Wd	Table Word
TCH	(Mnemonic) Test Channel (SI)
T/CT	True/Complement Trigger
TD	Time Delay
Temp	Temporary
TERMT	Terminate Trigger
TFMT	Temporary Fetch Made Trigger
Tgr	Trigger
TI	Terminate Immediate
TIO	(Mnemonic) Test I/O (SI)
TM	(Mnemonic) Test under Mask (SI)
Tof	Turn Off
Ton	Turn on
Tot	Total
TR	(Mnemonic) Translate (SS)
Trans	Transpose
Trnsps	Transpose
TRT	(Mnemonic) Translate and Test (SS)
TS	(Mnemonic) Test and Set (SI); Timing Stack
T&S	Test and Set
U1	Unit 1
U2	Unit 2
UB	Upper Bound
UABI	Unit Address Bus In
UABO	Unit Address Bus Out
UBCTR	Upper Bound Counter
UCC	Unit Communications Control
Ucndl	Unconditional
Uncond	Uncondition
UNPK	(Mnemonic) Unpack (SS)
Val	Valid
Var	Variable
VFL	Variable Field Length
VFLEU	Variable Field Length Execution Unit
Viol	Violate; Violation
WAM	With Available Memory
WC	Word Counter
Wd	Word
Wd Bdy	Word Boundary
WR	Working Register
WRD	(Mnemonic) Write Direct (SI)
XC	(Mnemonic) Exclusive OR (SS)
Xec	Execute
XOR	Exclusive OR
ZAP	(Mnemonic) Zero and Add (SS)
ZET	Zero Test Unit
1A2	SAR 1 Loaded after SAR 2
1B2	RS 1 Loaded before RS 2
1B3	RS 1 Loaded before RS 3
1B4	RS 1 Loaded before RS 4
1C2	SAR 1 Address Compares with SAR 2 Address
2A3	SAR 2 Loaded after SAR 3
2B3	RS 2 Loaded before RS 3
2B4	RS 2 Loaded before RS 4
2C3	SAR 2 Address Compares with SAR 3 Address
3A1	SAR 3 Loaded after SAR 1
3B4	RS 3 Loaded before RS 4
3C1	SAR 3 Address Compares with SAR 1 Address

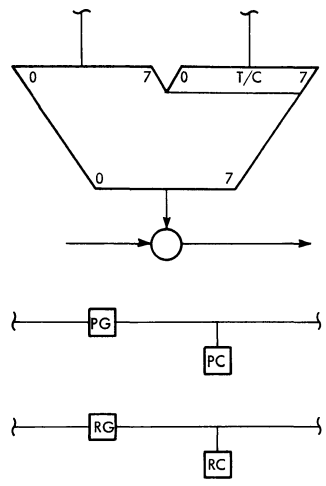
LEGEND

1. Data Flow Diagrams



Register, Counter

Heavy line indicates input side of a functional unit that can store information. A partial transfer of contents is shown by numbered input/output lines. An input or output line connected directly to the register denotes a complete transfer of contents. An X placed in an input or an output line means that a gating condition is required to activate the transfer path. A number in a transfer path denotes the number of lines. A bar in the upper right corner means that the status of register positions is shown in indicator lights. An ALD page reference is given under the indicator bar. The bottom line within a register gives either the register position numbers or a single number indicating register size. Where register position numbers are given, a symbol such as (+3P) indicates that the register also contains 3 parity bits.



Adder, Incrementer

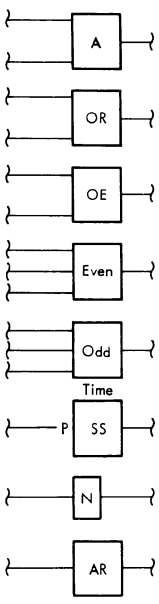
Notches across top divide logical inputs. T/C indicates that the associated input can be entered in either true or complement form.

Functional or Switching OR

Parity Generate, Parity Check

Residue Generate, Residue Check

2. Positive Logic Diagrams



AND
The output is active only when all inputs are active.

OR
The output is active if one or more inputs is active.

Exclusive OR
The output is active only when one input is a active and the other is not.

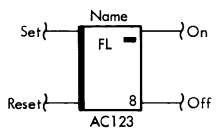
Even
The output is active only when an even number of inputs is active.

Odd
The output is active only when an odd number of inputs is active.

Singleshot, Time Delay, Oscillator

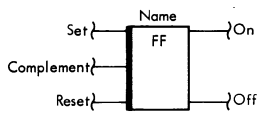
Inverter, or Negator
The output is active only when the input is not active.

Amplifier, Signal Mode Converter



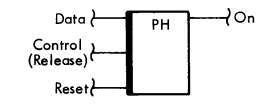
Flip Latch

Input side is denoted by thick line. Bar means that an indicator is connected to this latch. The number in the lower right corner means that this symbol represents 8 actual flip latches in the machine. An ALD page reference may be given below the block. A set input activates the on output and deactivates the off output; a reset input activates the off output and deactivates the on output. The device holds its outputs between active inputs. Simultaneous set and reset inputs activate both on and off outputs until one input is deactivated.



Flip-Flop

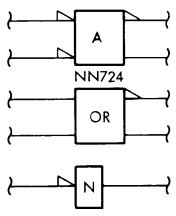
Same as flip latch except that an active complement input causes the device to switch to the opposite state. Also, on and off outputs can never be active simultaneously: Simultaneous set and reset inputs are equivalent to a complement input.



Polarity Hold

The PH output follows the data input when the control (release) input is activated. Between control inputs, the PH holds the previously sampled state of the data line. The reset input (when used) deactivates the output.

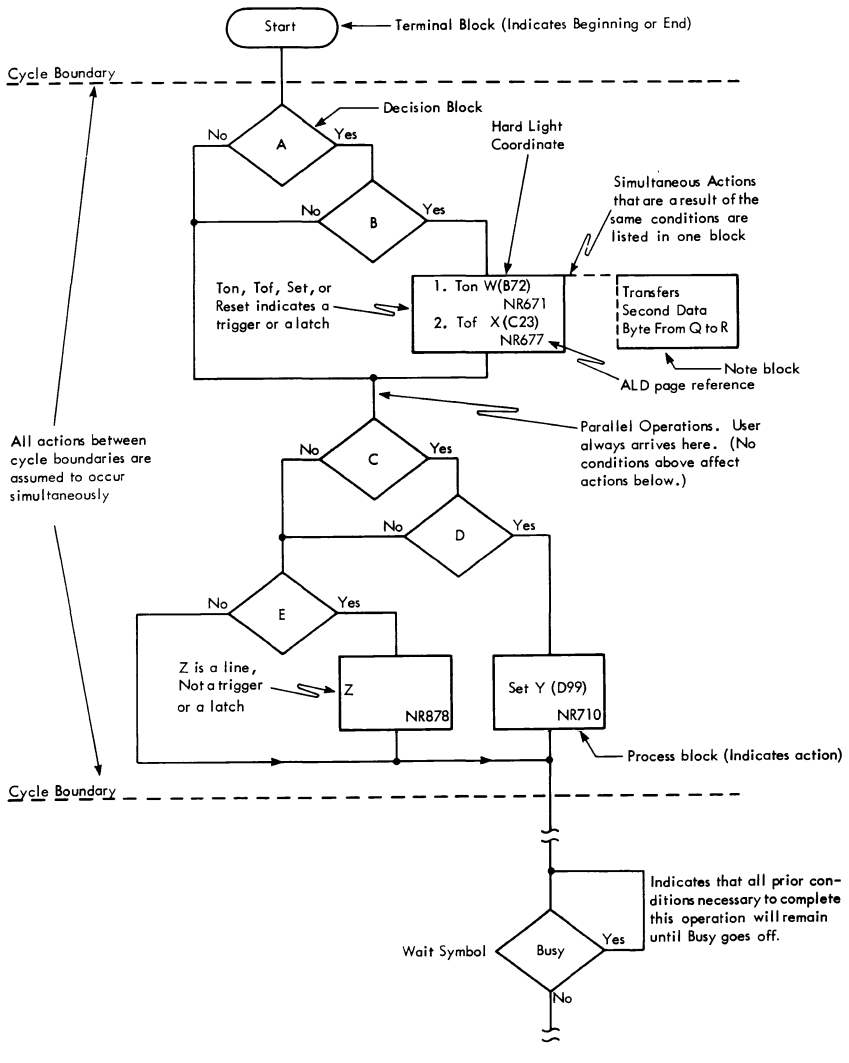
3. Simplified Logic Diagrams



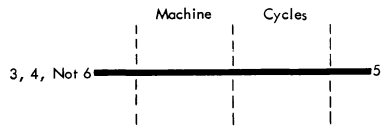
Input wedges mean that the more negative line level is required to activate the circuit; output wedges mean that the more negative line level is present when the circuit is activated. Lack of wedges indicate the more positive level. Blocks may have more than one output line. All line titles are preceded by + or - to indicate line level.

Note: Additional SLD symbology used only on ECAD's is shown in Volume 1.

4. Flowcharts

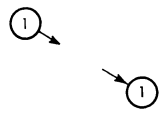


5. Timing Charts



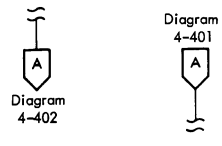
Heavy bar indicates active state. Numbers at beginning and end of the bar identify the signal(s) (also on the same chart) that activate and deactivate this line. "Not" preceding a number means that the deactive signal conditions this line.

6. General



On-Page Connector

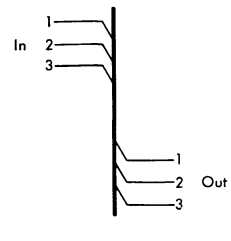
Indicates connection between two points on the same diagram. Arrow leaving symbol points to symbol with the same number.



Off-Page Connector

Indicates connection between two points located on separate pages. Where the connection is between two pages of a multipage diagram, a reference such as "Sheet 2" is given instead of a diagram number.

Text Reference Point (Reference from FETOM)



Multiple Line Transfer

Objectives:

1. Test for a possible instruction fetch every machine cycle.
2. Try to fetch four doublewords of instructions ahead of the word being processed.
3. Increase the priority of instruction fetch over operand fetches whe IR gets closer than 4 to UB.

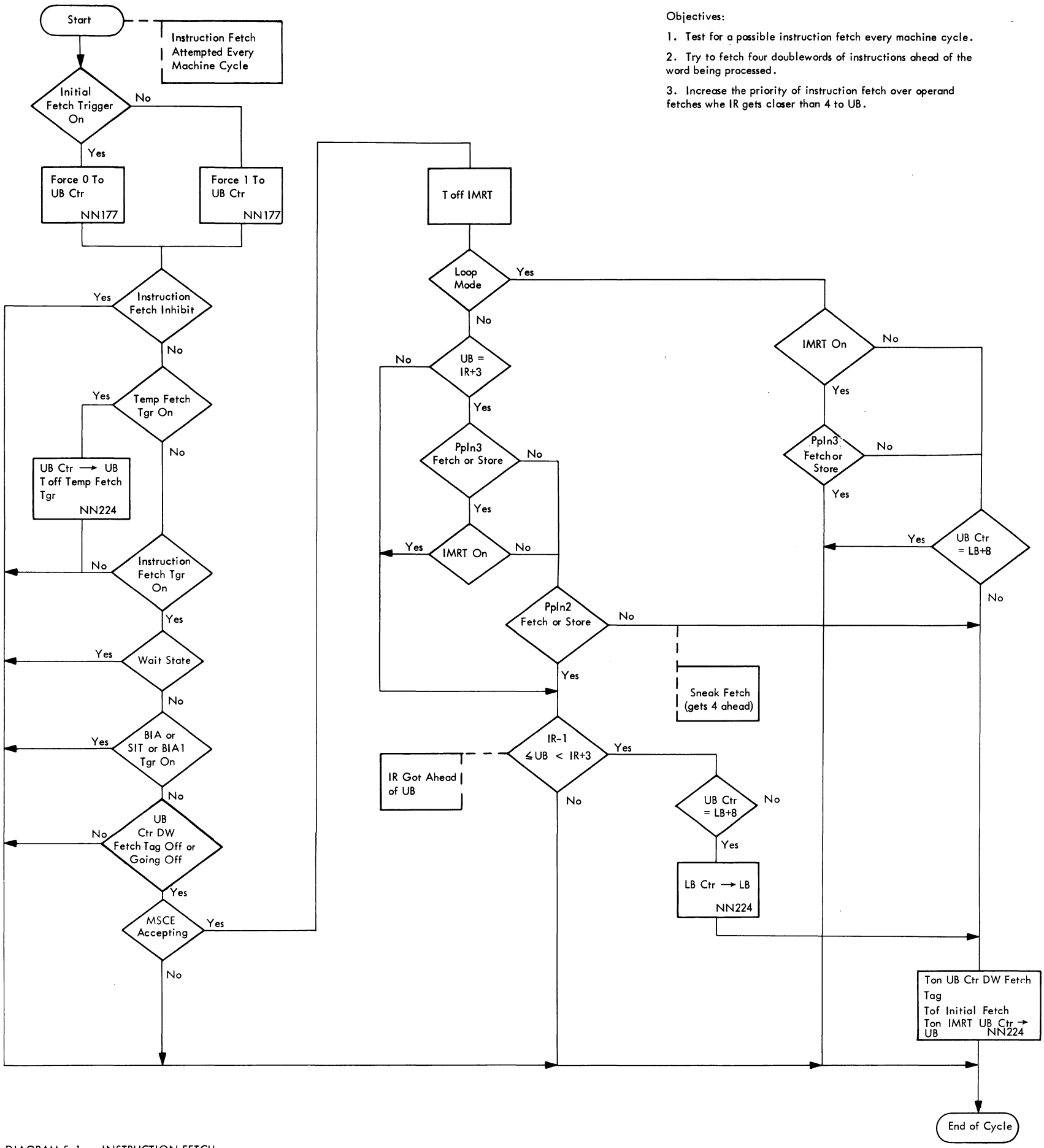


DIAGRAM 5-1. INSTRUCTION FETCH

Objectives:

1. Direct an instruction doubleword to the correct destination when returning from storage.
2. Step AOC when the op reg is filled.
3. Ingate the proper instruction syllables into the op reg.
4. Control fetch tag, protect tag, and valid trigger turn on and turn off.

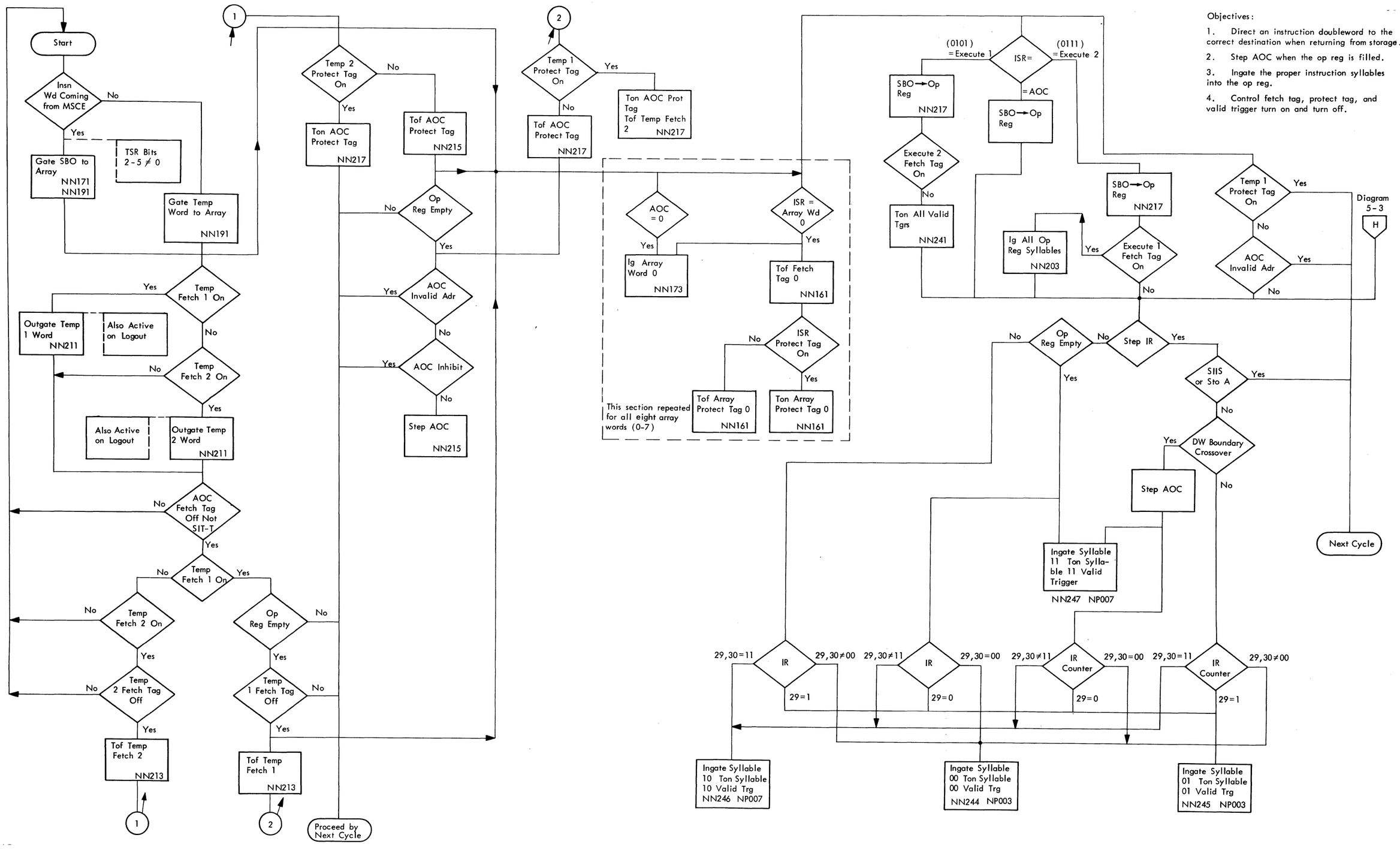


DIAGRAM 5-2. INSTRUCTION FETCH RETURN AND OP REGISTER INGATE CONTROLS

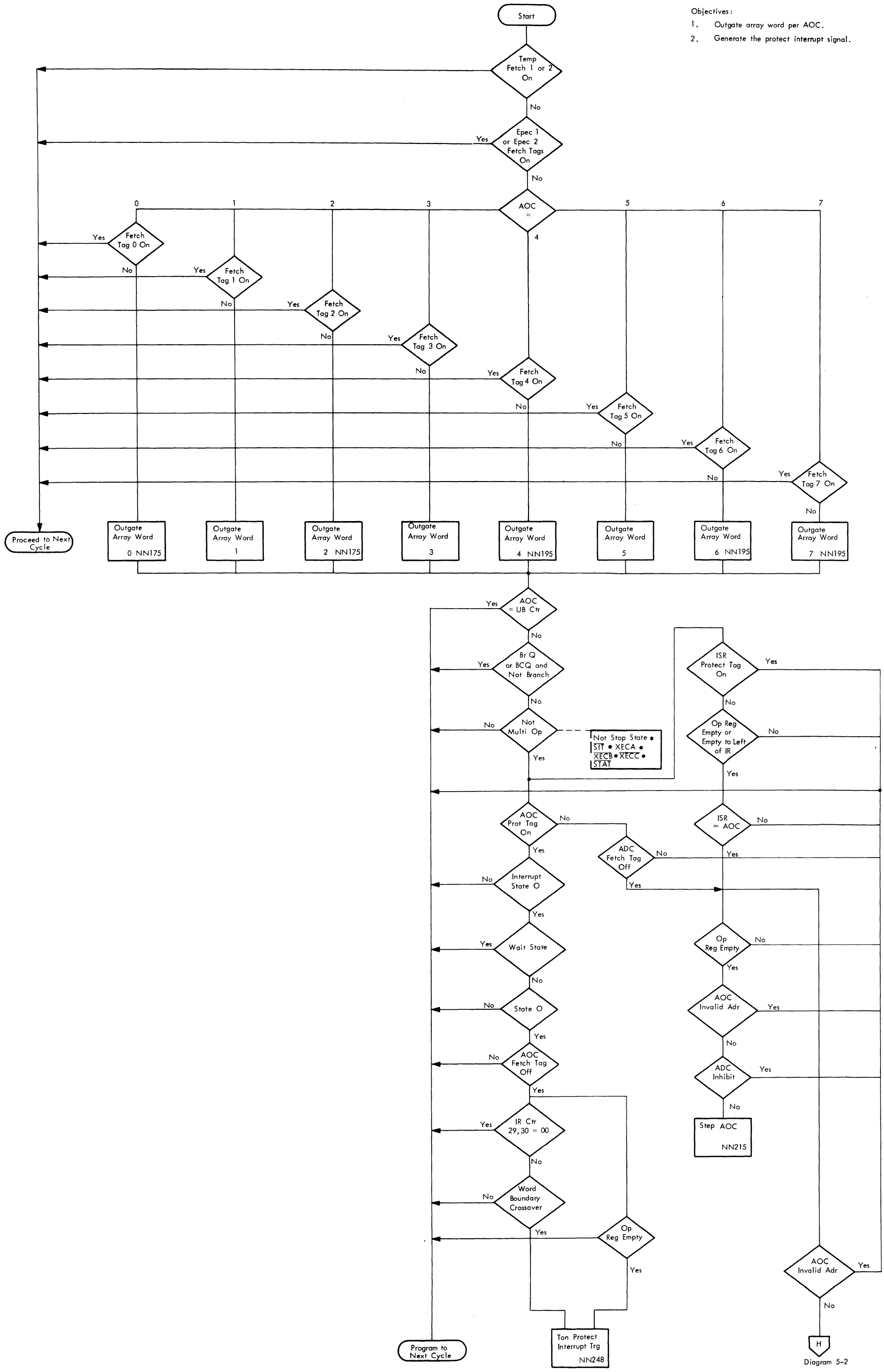
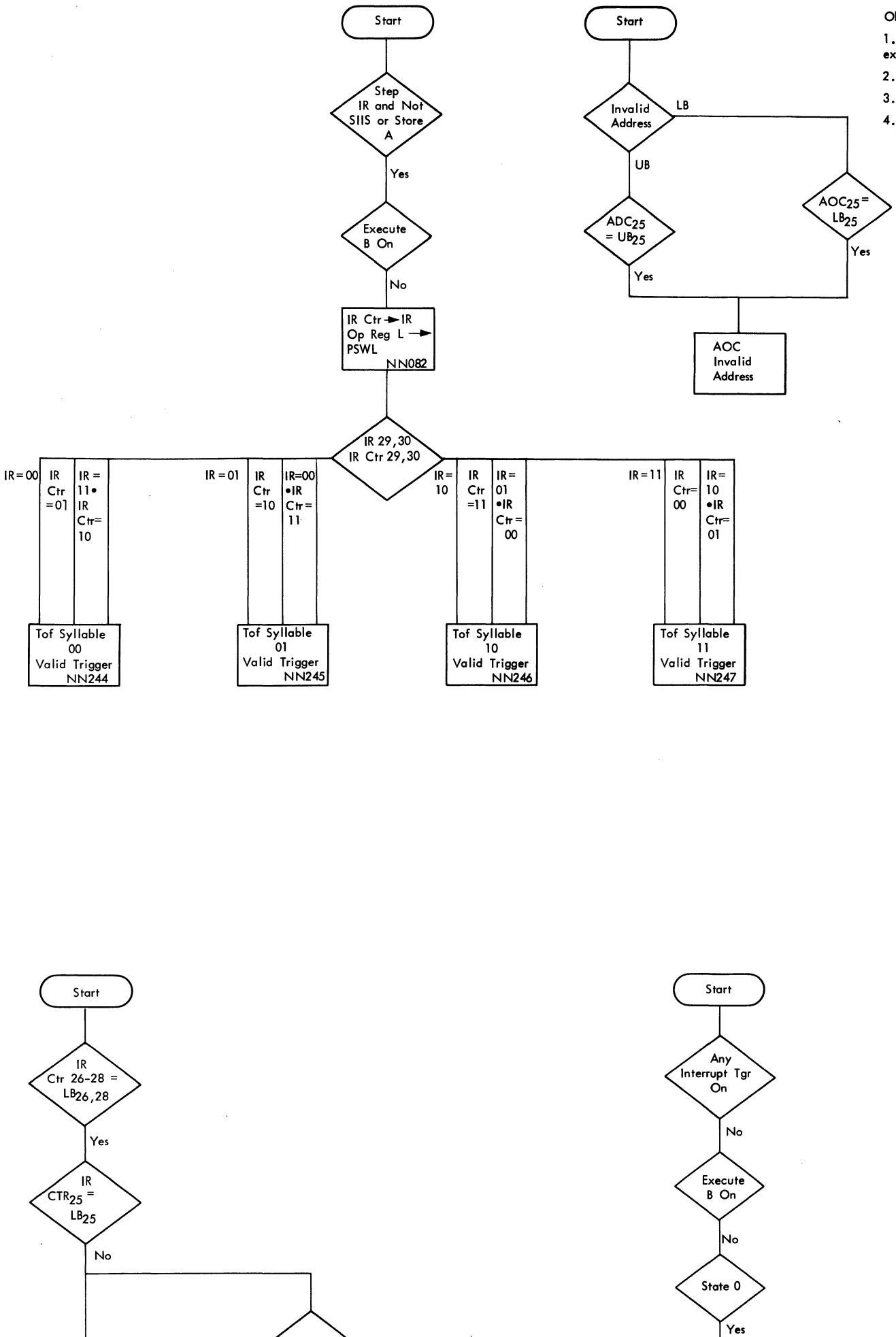


DIAGRAM 5-3. ARRAY WORD OUTGATE FETCH PROTECT INTERRUPT



Objectives:

1. Control the turn off of the op reg valid triggers by examining bits 29 and 30 of IR and IR counter.
2. Test for AOC invalid address.
3. Test for the turn off loop mode conditions.
4. Test for invalid instruction address.

DIAGRAM 5-4. AOC INVALID ADDRESS, VALID TRIGGER TURN OFF, AND TURN OFF LOOP MODE

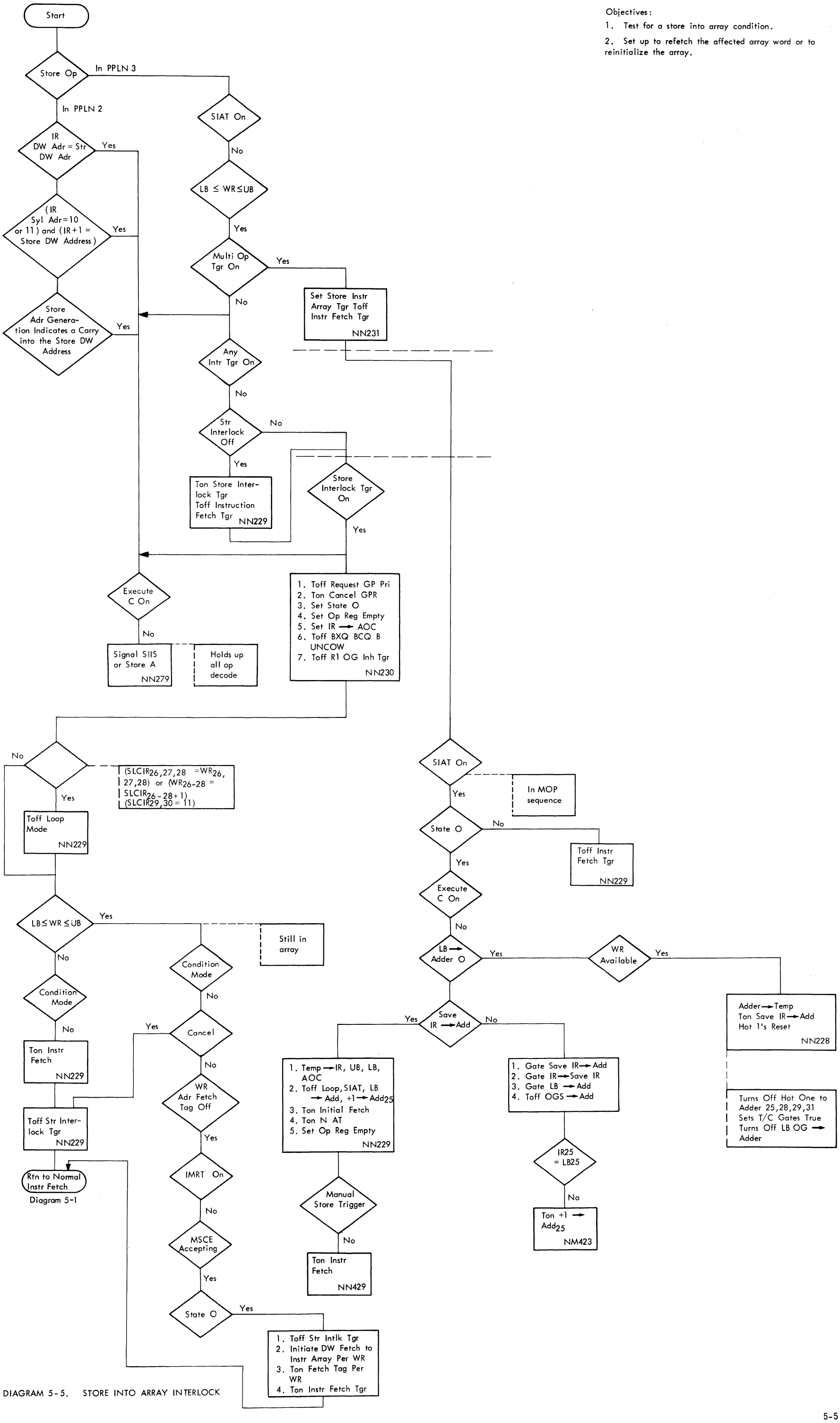


DIAGRAM 5-5. STORE INTO ARRAY INTERLOCK

Objectives:

1. Test for a store into array condition.
2. Set up to refetch the affected array word or to reinitialize the array.

Objectives:

1. Generate pipeline ready and working register available signals.
2. Control fetch and store tags in pipeline 2.
3. Set up controls for condition mode.

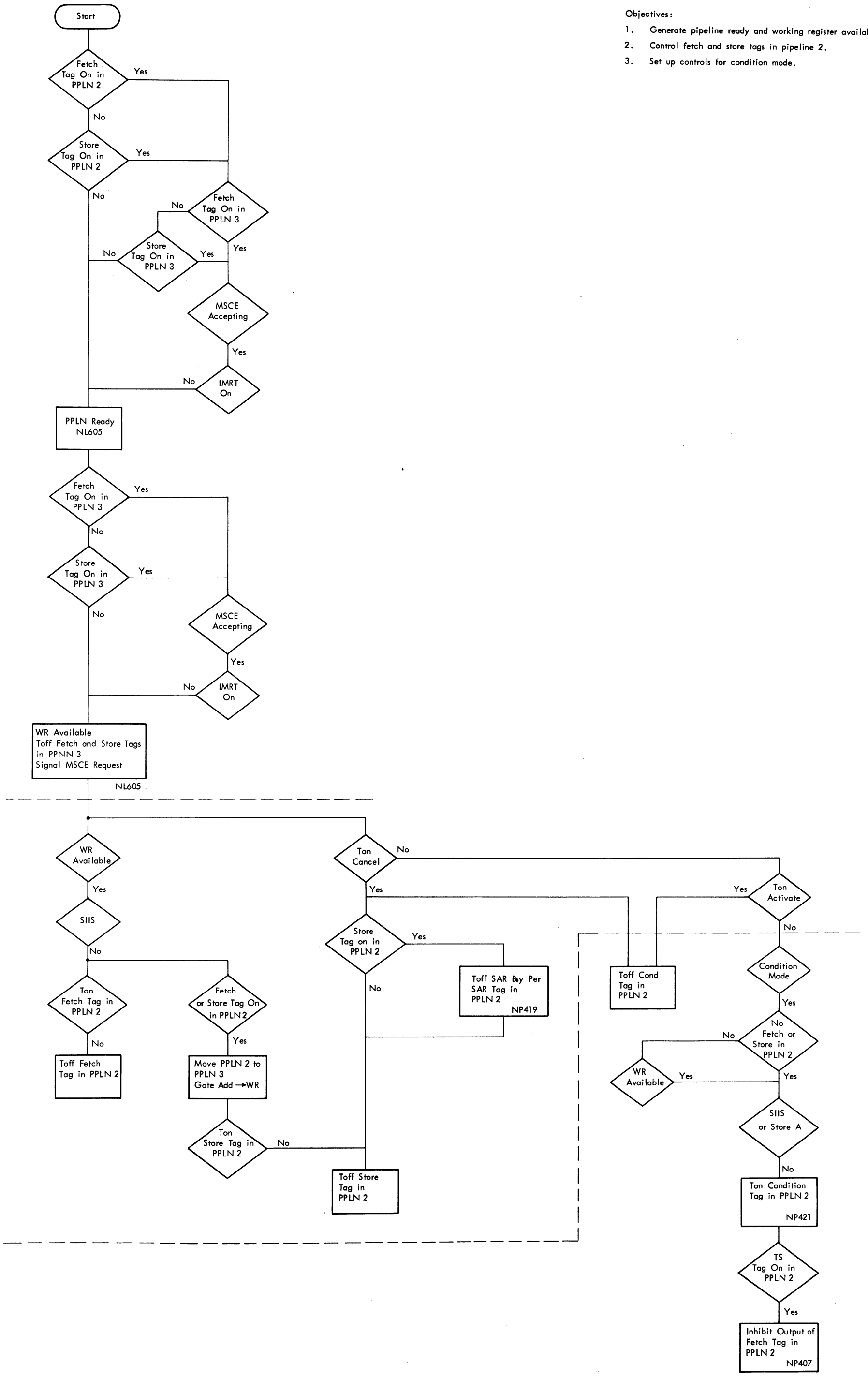


DIAGRAM 5-6. PIPELINE 2 AND 3 CONTROL

Objectives:

1. To check basic interlocks during decode cycle (state 0)

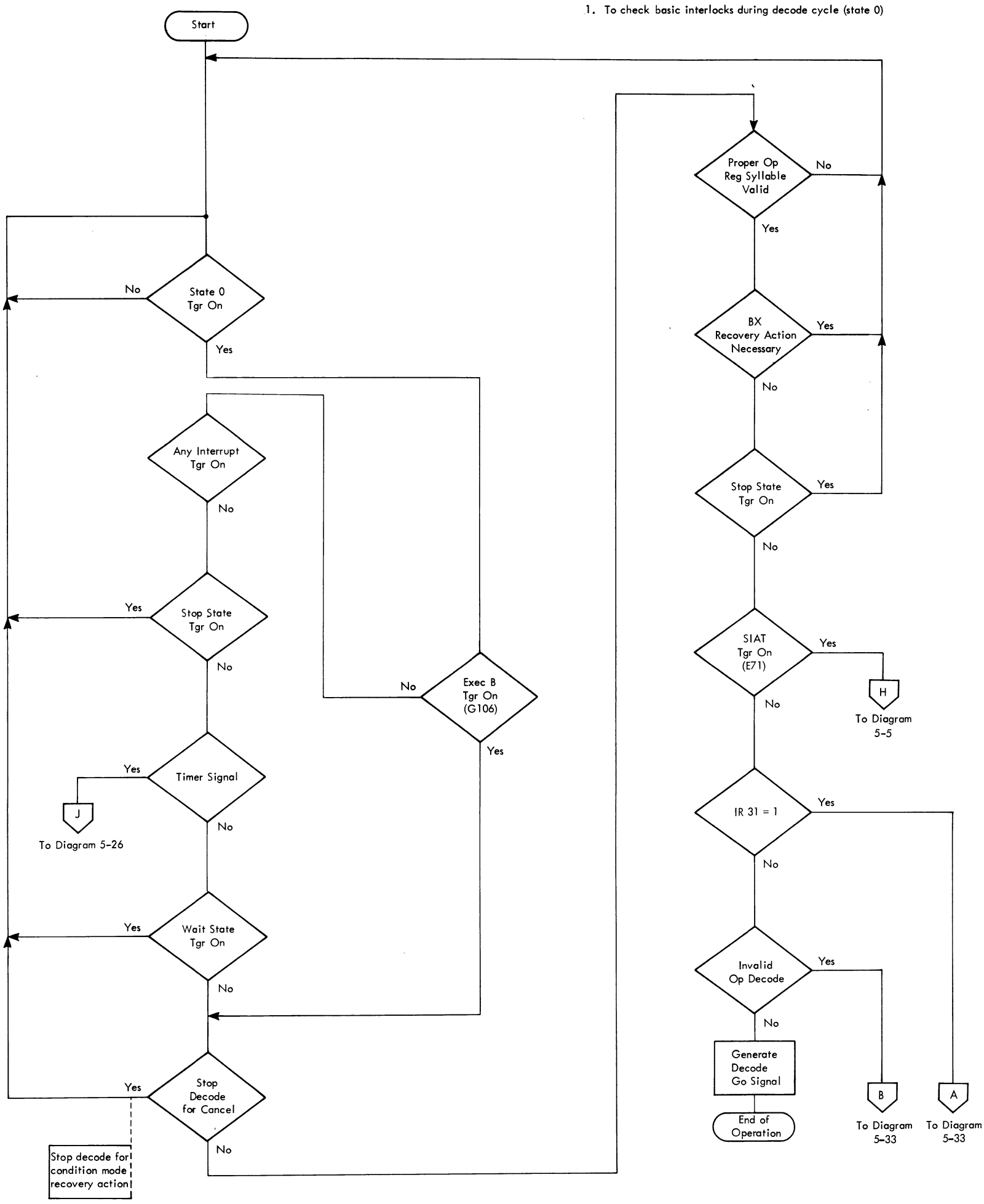


DIAGRAM 5-7. DECODE CYCLE (STATE 0) BASIC INTERLOCK CHECK

Objectives:

1. To decode and issue a floating point instruction.

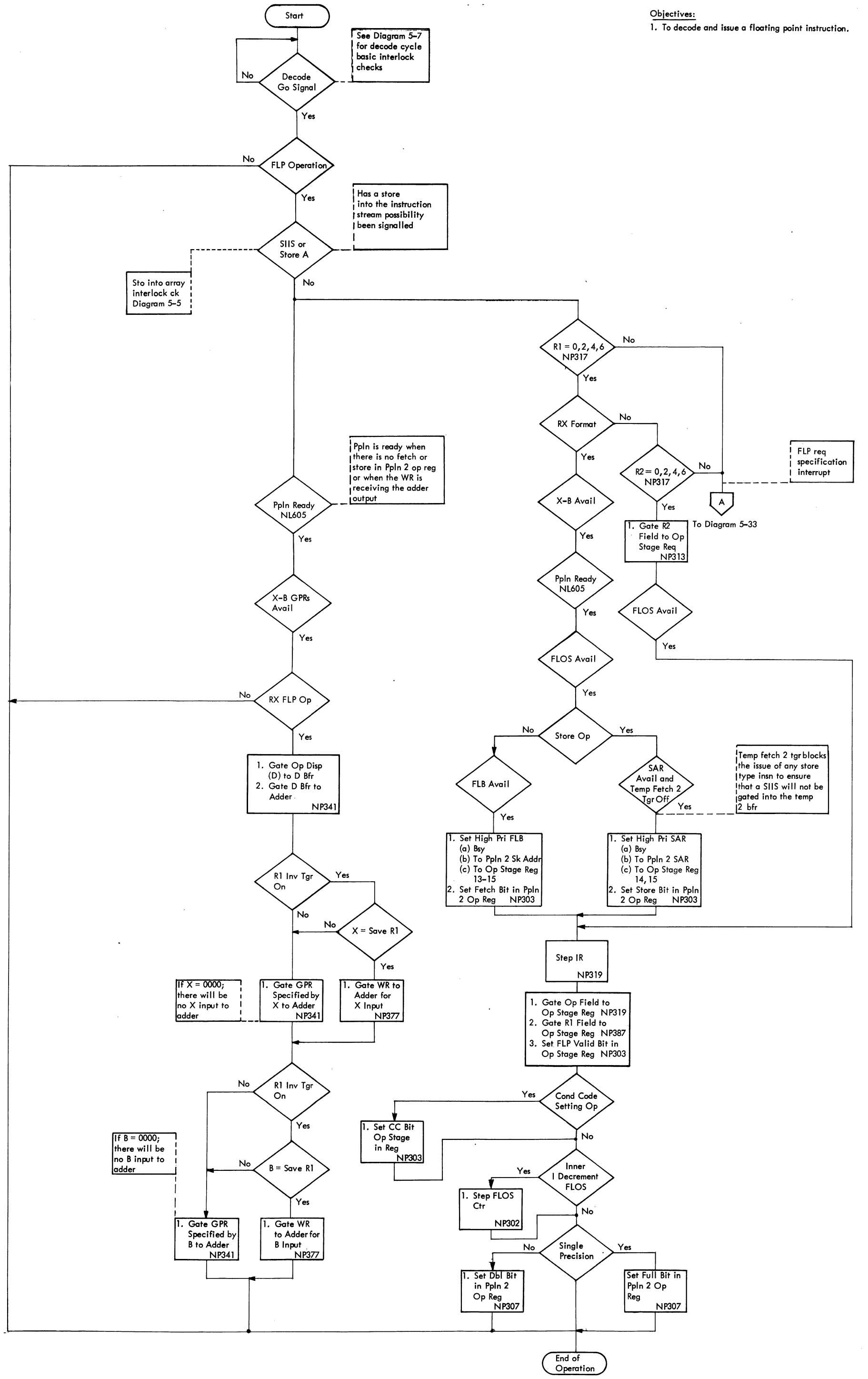


DIAGRAM 5-8. FLOATING POINT ISSUE SEQUENCE

Objectives:

1. To decode and issue a FXP full word operation.

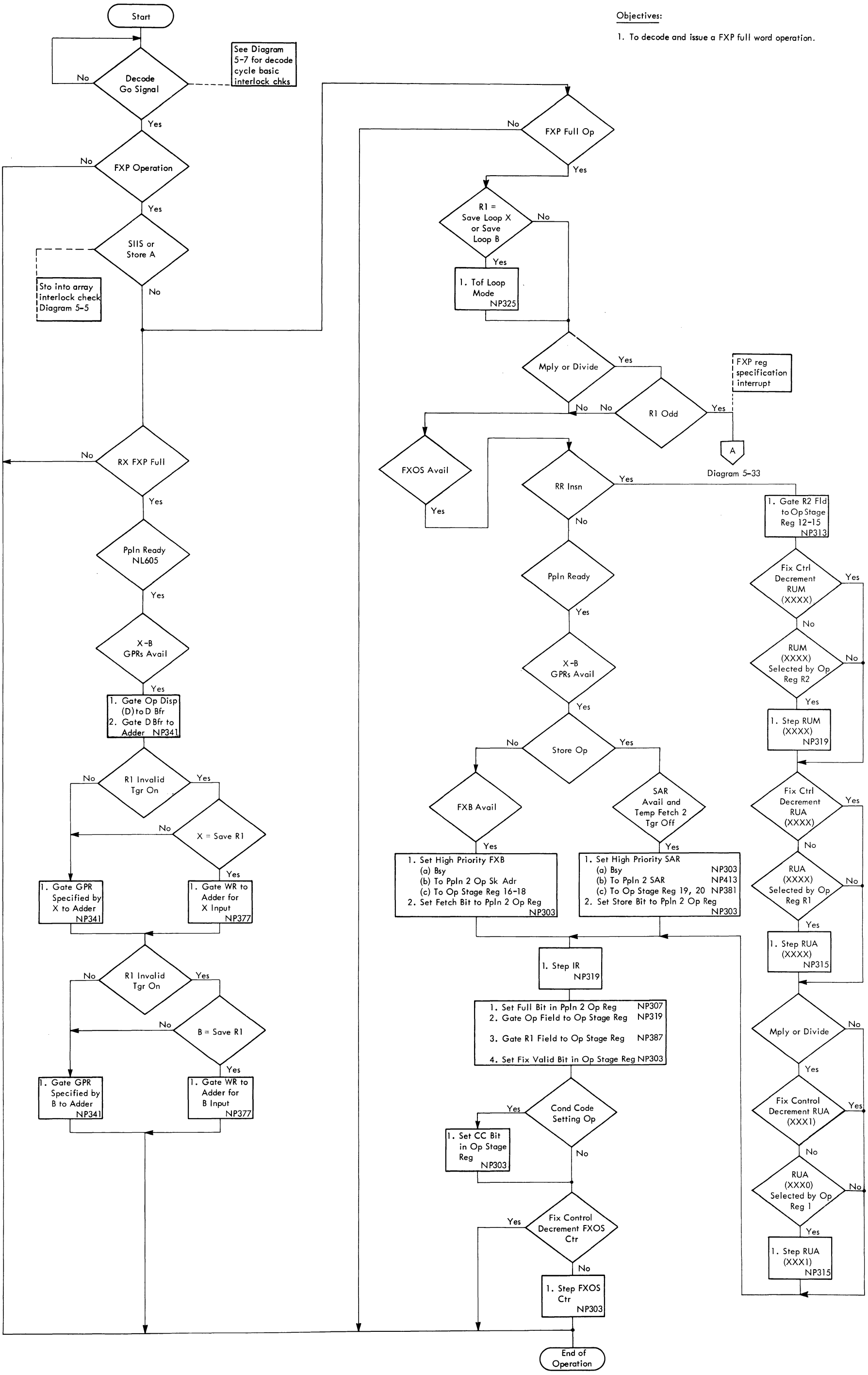


DIAGRAM 5-9. FIX POINT FULL WORD ISSUE SEQUENCE

Objectives:

1. To decode and issue a fix point halfword instruction.
2. To decode and issue a insert and store character instruction.

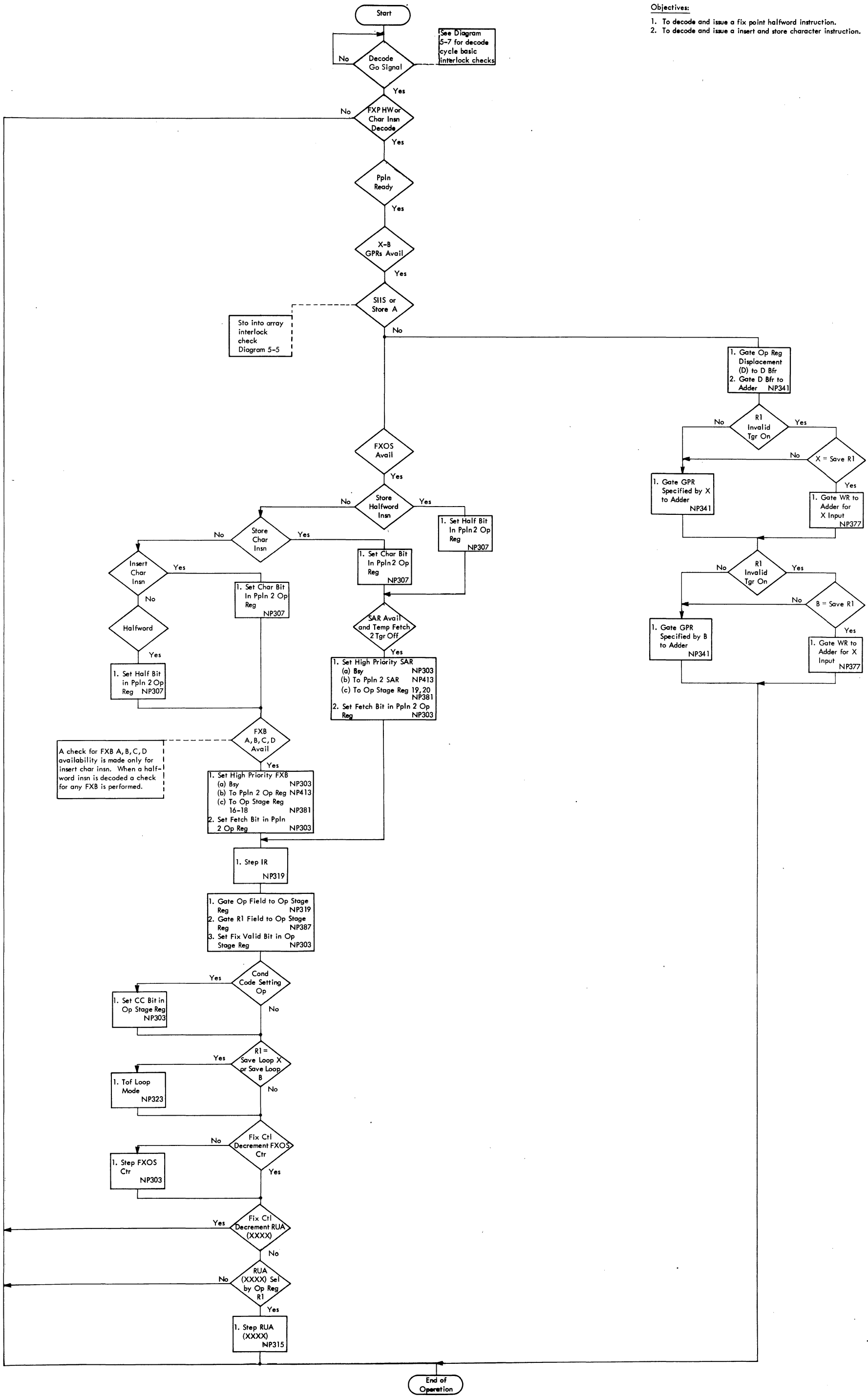


DIAGRAM 5-10. FIX POINT HALFWORD AND INSERT AND STORE CHARACTER ISSUE SEQUENCE

Objectives:

1. Decode and process a BC instruction.
2. Check all interlocks which may cancel the execution of this instruction.
3. Establish condition mode if the condition code is not immediately valid.
4. Establish an unconditional branch when the M1 (four bit mask) field in the BC instruction format contains all ones.
5. If conditions permit, establish loop mode.

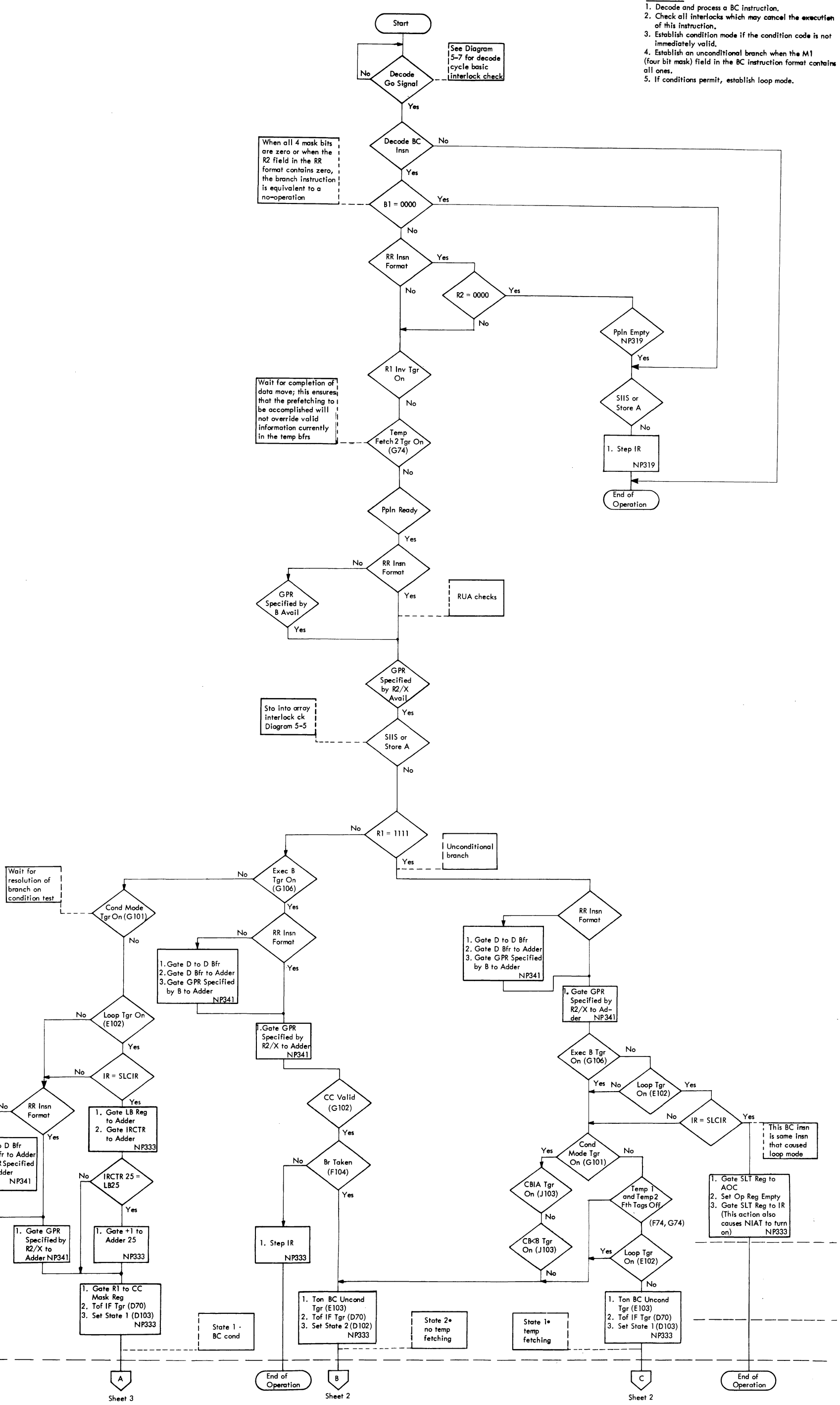


DIAGRAM 5-11. BRANCH ON CONDITION SEQUENCE (SHEET 1 OF 5)

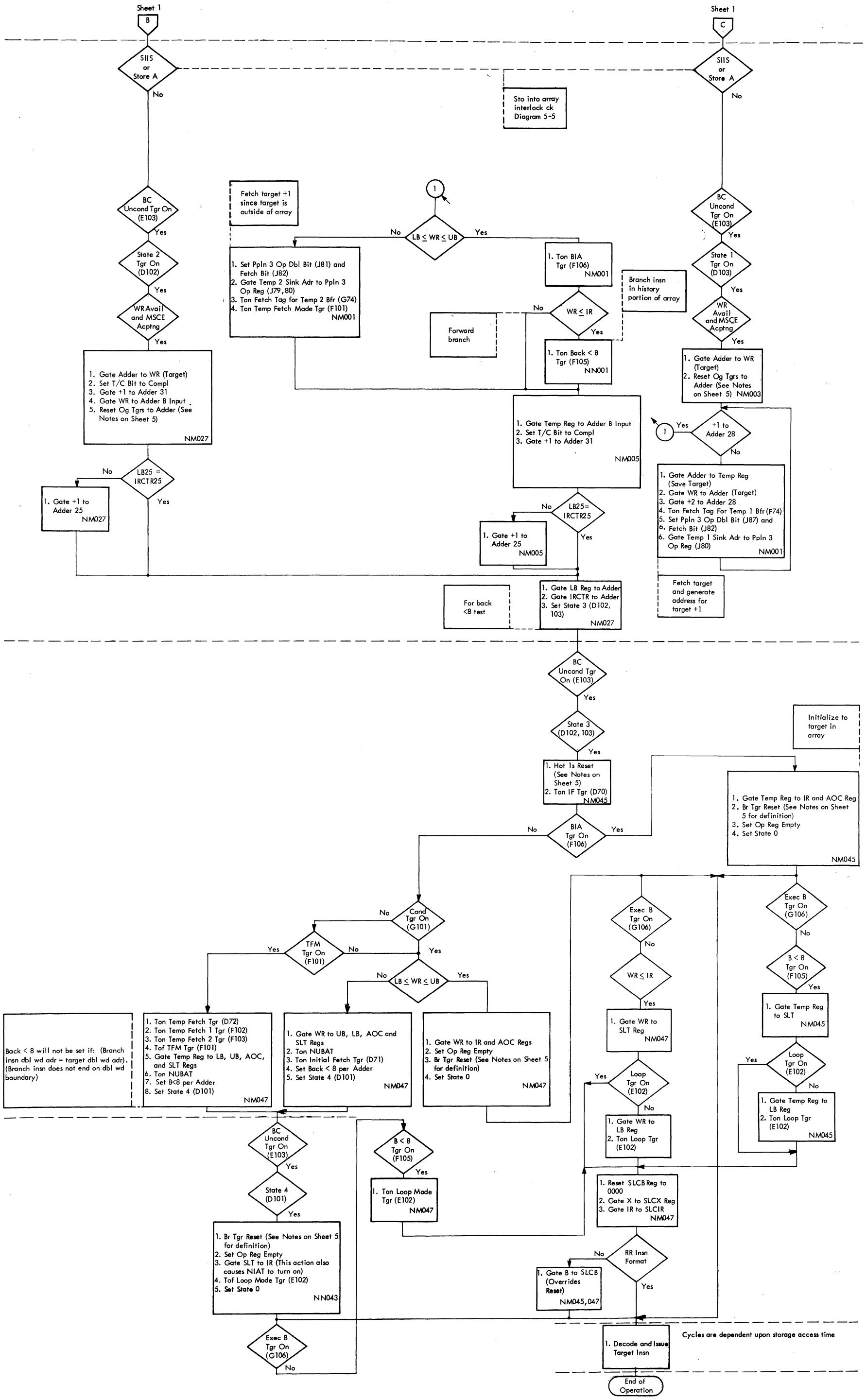


DIAGRAM 5-11. BRANCH ON CONDITION SEQUENCE (SHEET 2 OF 5)

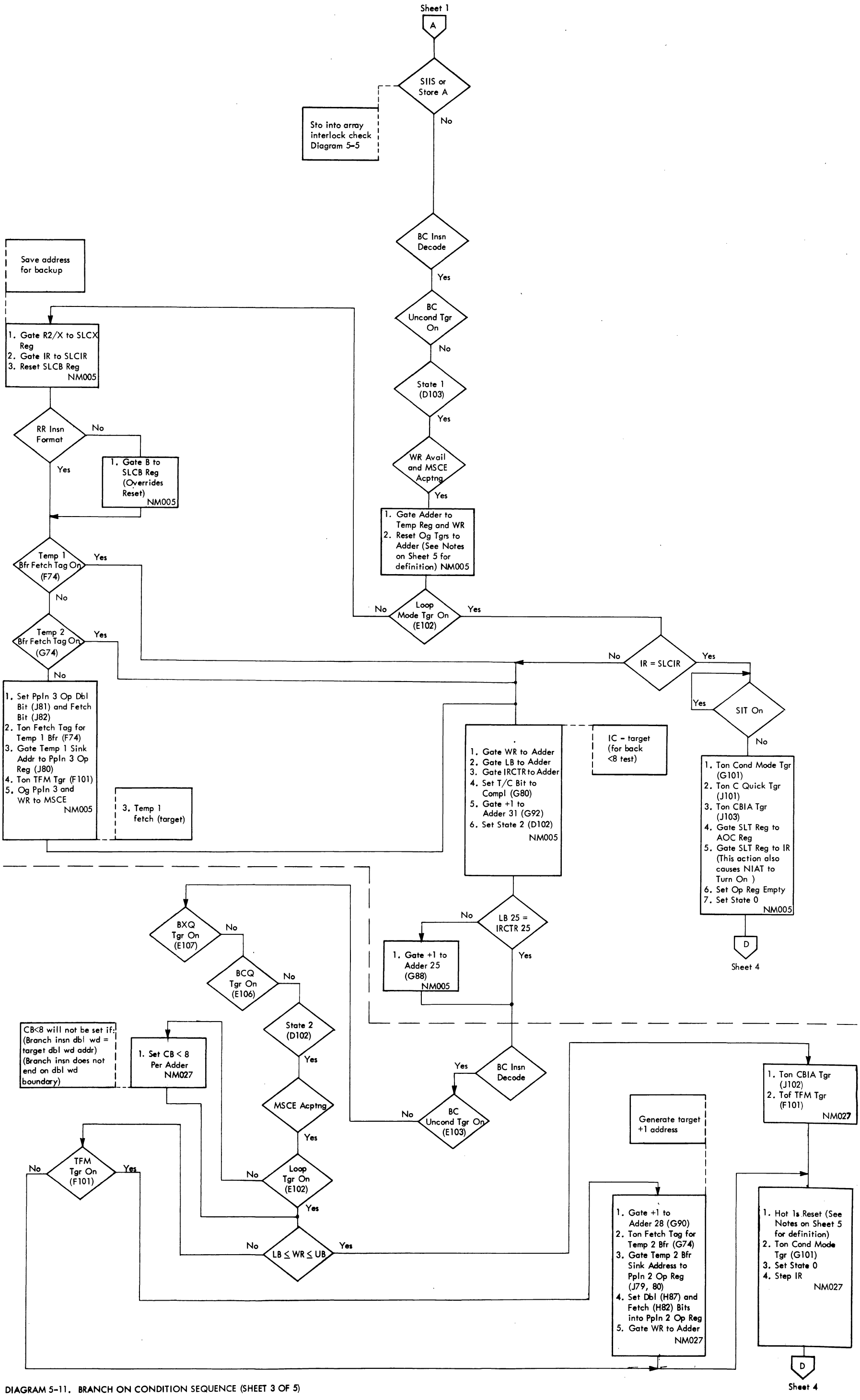


DIAGRAM 5-11. BRANCH ON CONDITION SEQUENCE (SHEET 3 OF 5)

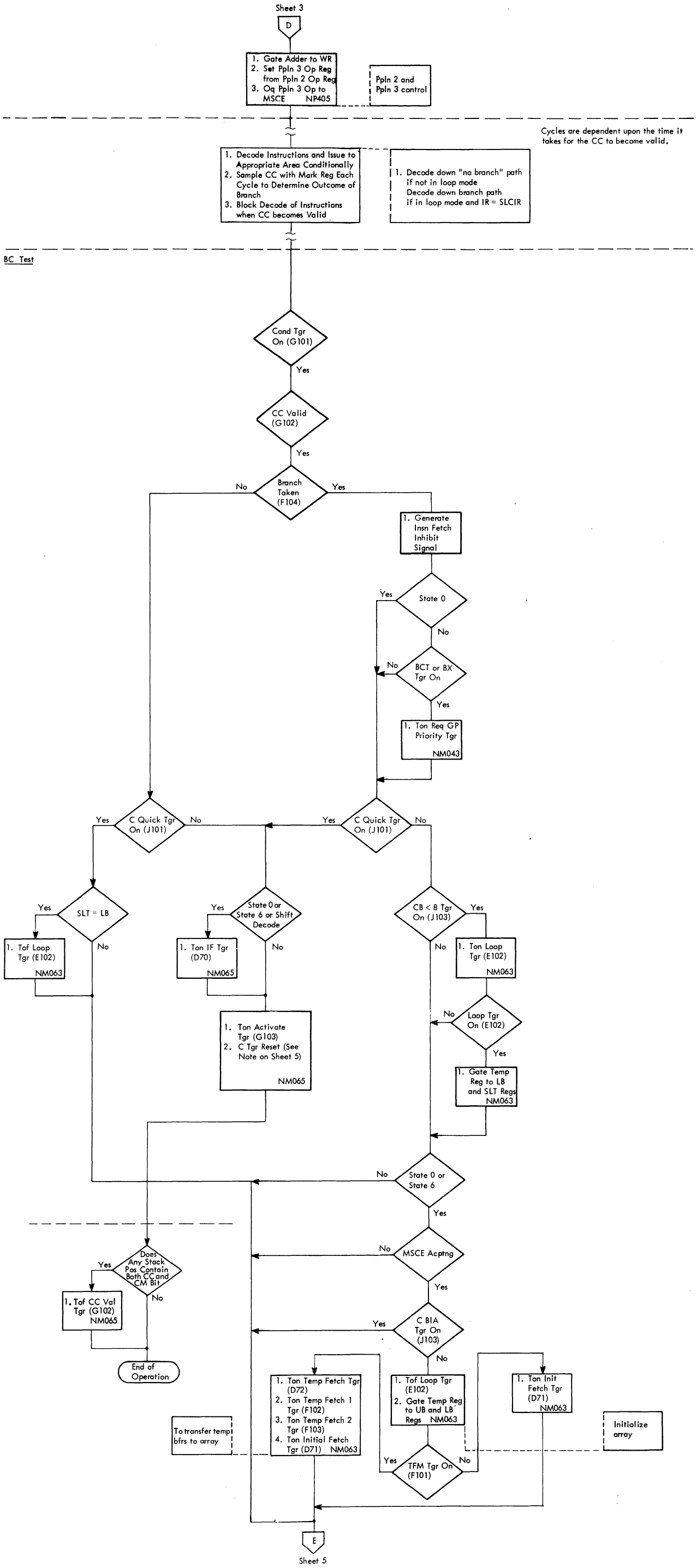
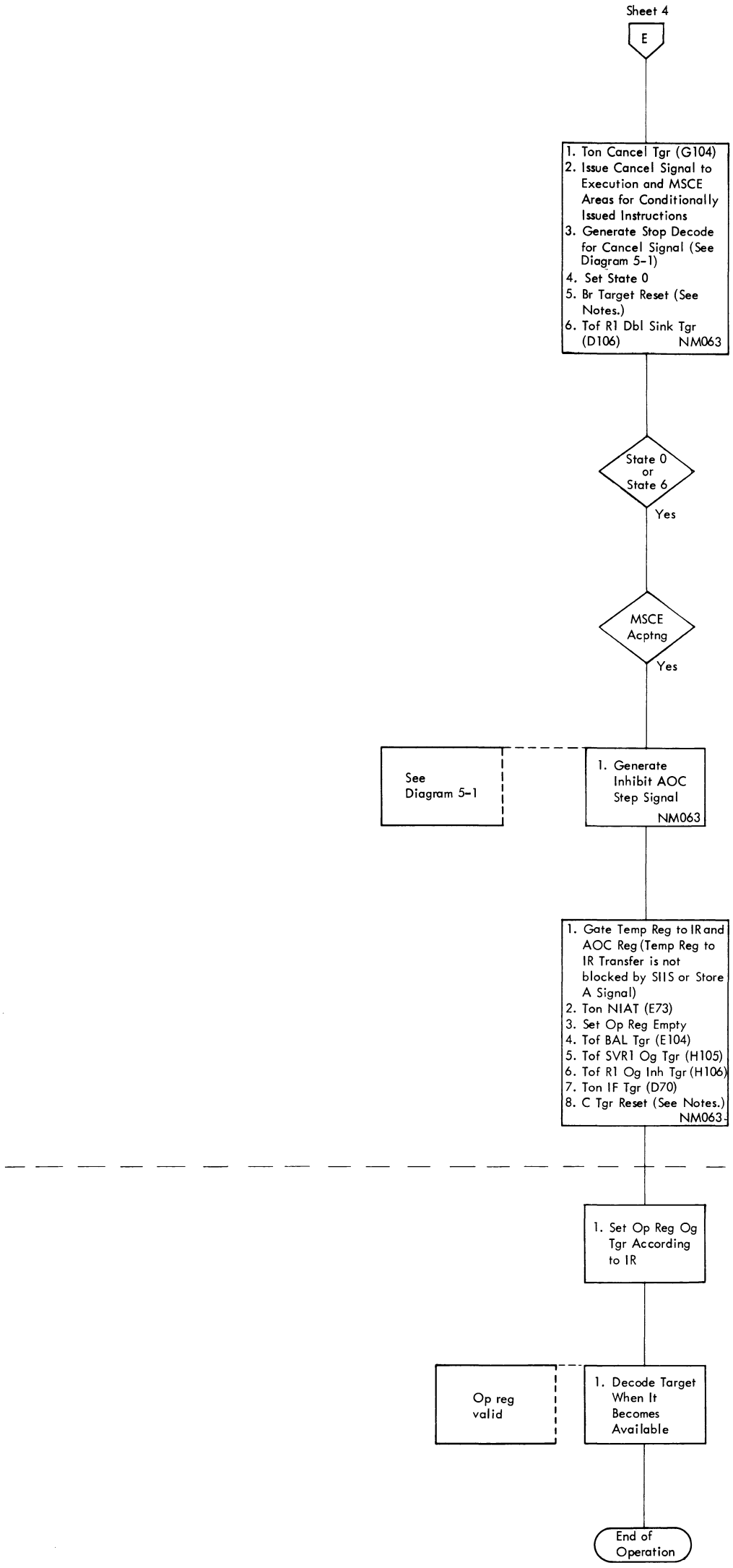


DIAGRAM 5-11. BRANCH ON CONDITION SEQUENCE (SHEET 4 OF 5)



Notes:

1. Reset Og to Adder = GPR Reset and Hot 1s Reset
2. GPR Reset = (a) Tof R1 Og Tgrs
   (b) Tof X/R2; B; D; Og Tgrs
   (c) Tof WR Og Tgrs
   (d) Tof Temp Reg Og Tgrs
3. Hot is Reset = (a) Set T/C Tgr to T
   (b) Tof +1 to Adder 31; 29; 28; TAT; and 25
   (c) Tof SVIR Og Tgr
   (d) Tof LB Og Tgr
   (e) Tof IRCTR Og Tgr
4. Br Tgr Reset = (a) Tof Br Tgr
   (b) Tof BIA Tgr
   (c) Tof BIA 1 Tgr
   (d) Tof B<8 Tgr
   (e) Tof Exec Tgr
   (f) Tof BCUNCON Tgr
5. C Tgr Reset = (a) Tof Cond Tgr
   (b) Tof CBIA Tgr
   (c) Tof CB<8 Tgr
   (d) Tof C Quick Tgr
   (e) Tof TFM Tgr

DIAGRAM 5-11. BRANCH ON CONDITION SEQUENCE (SHEET 5 OF 5)

Objectives:

1. Decode and process a BCT instruction.
2. Check all interlocks which may cancel the execution of this instruction.
3. Condition mode must be removed before processing of the BCT instruction can be completed.
4. The BCT instruction can either break, establish, or retain a loop mode of operation.

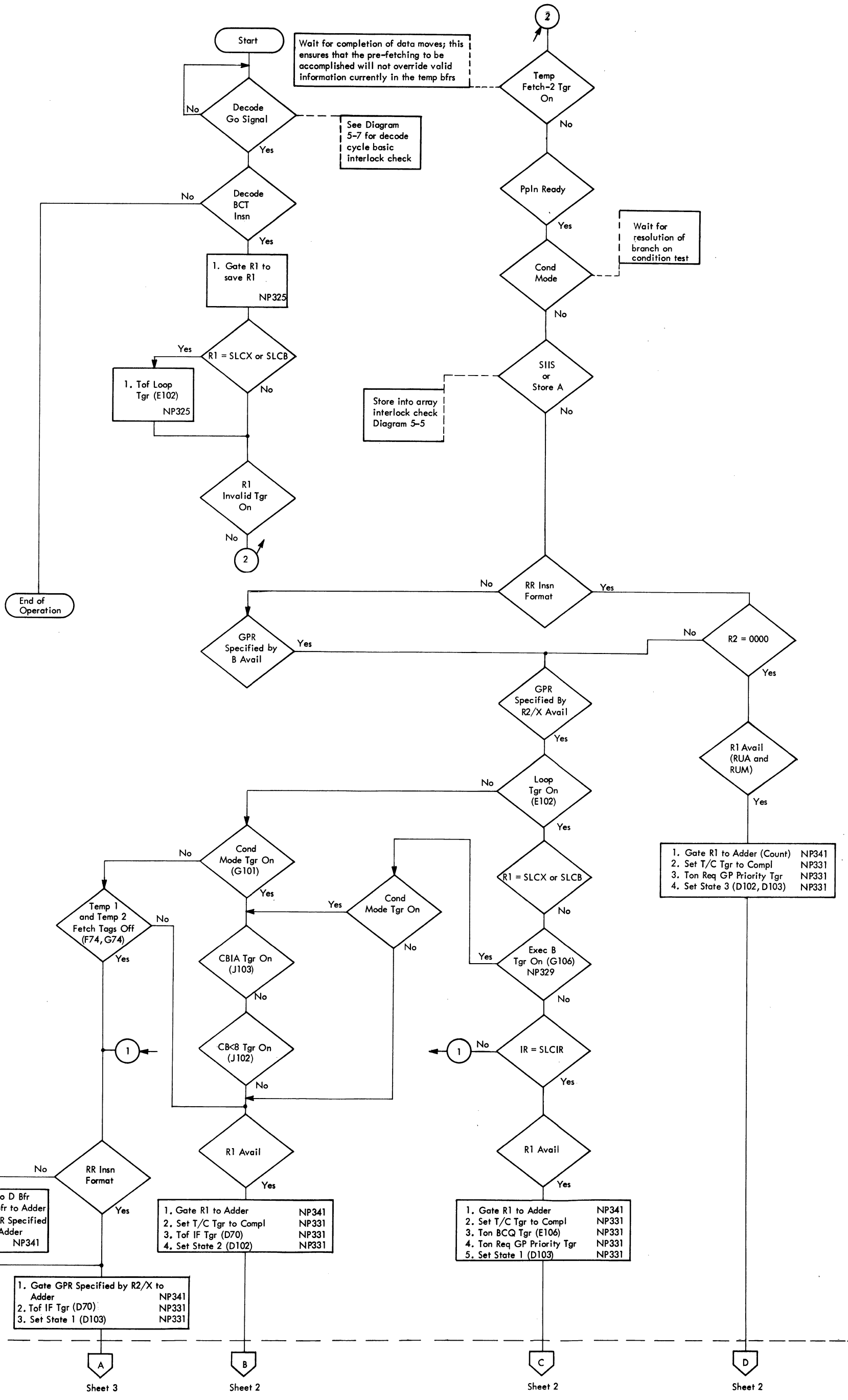


DIAGRAM 5-12. BRANCH ON COUNT SEQUENCE (SHEET 1 OF 5)

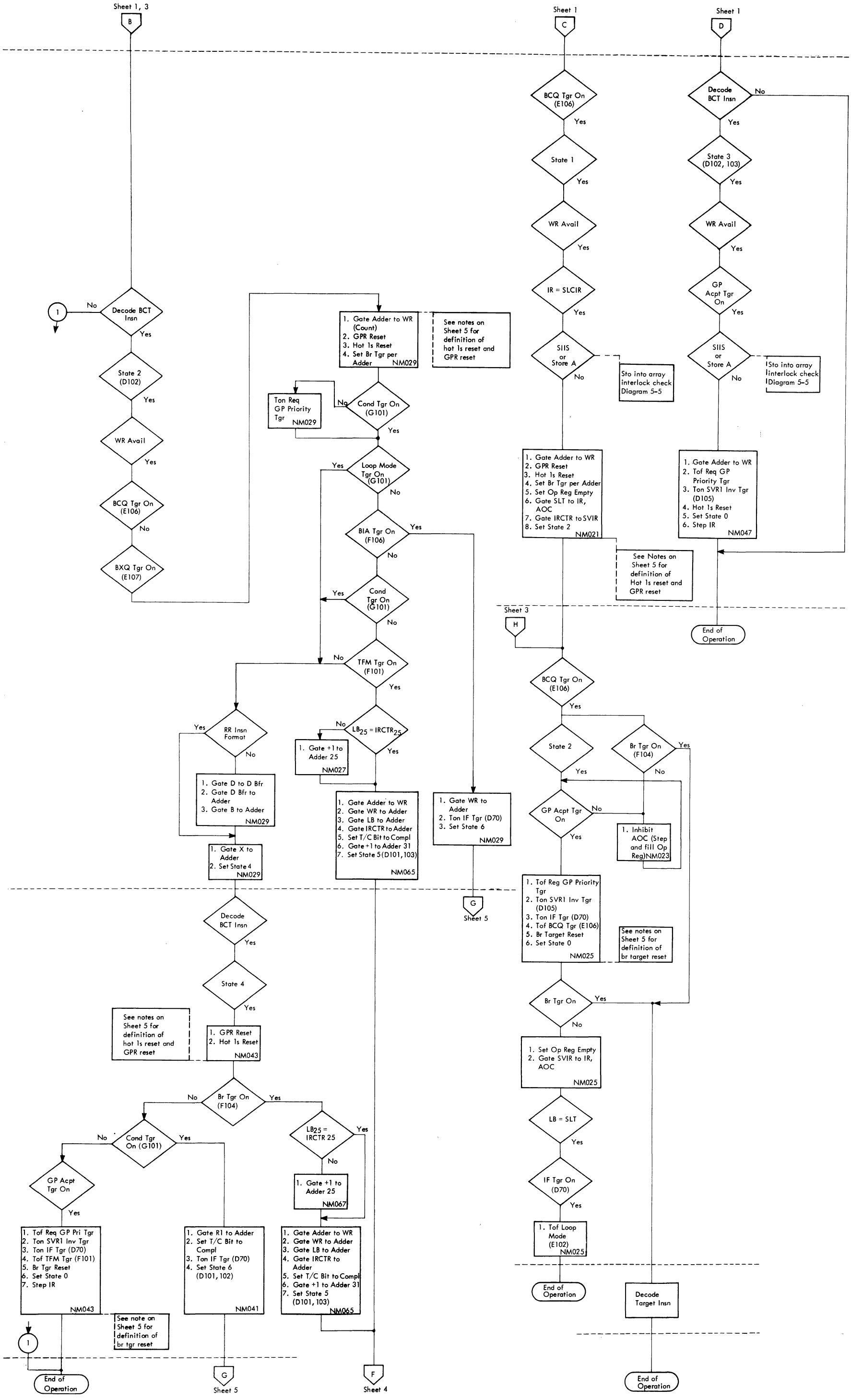


DIAGRAM 5-12. BRANCH ON COUNT SEQUENCE (SHEET 2 OF 5)

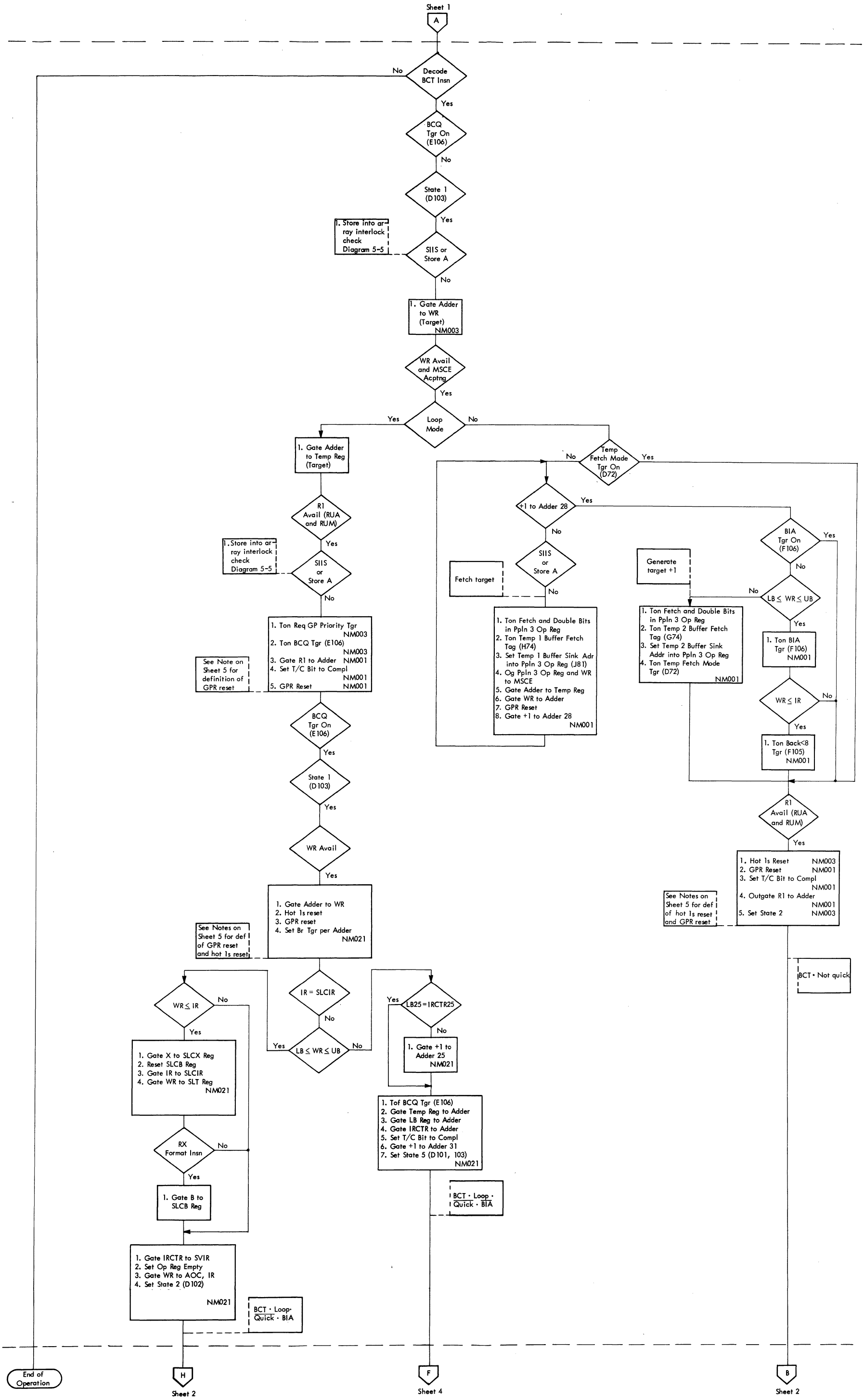


DIAGRAM 5-12. BRANCH ON COUNT SEQUENCE (SHEET 3 OF 5)

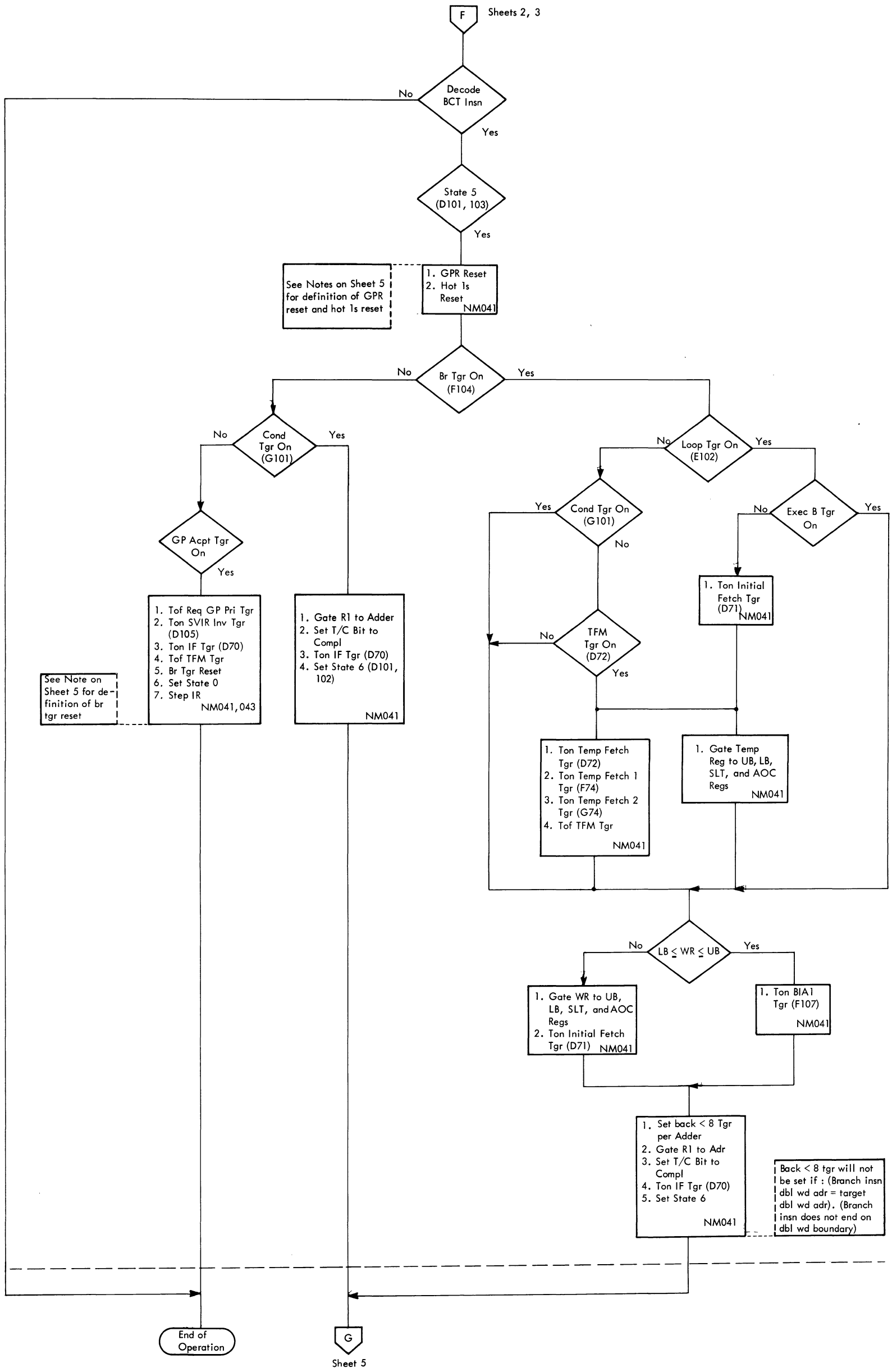
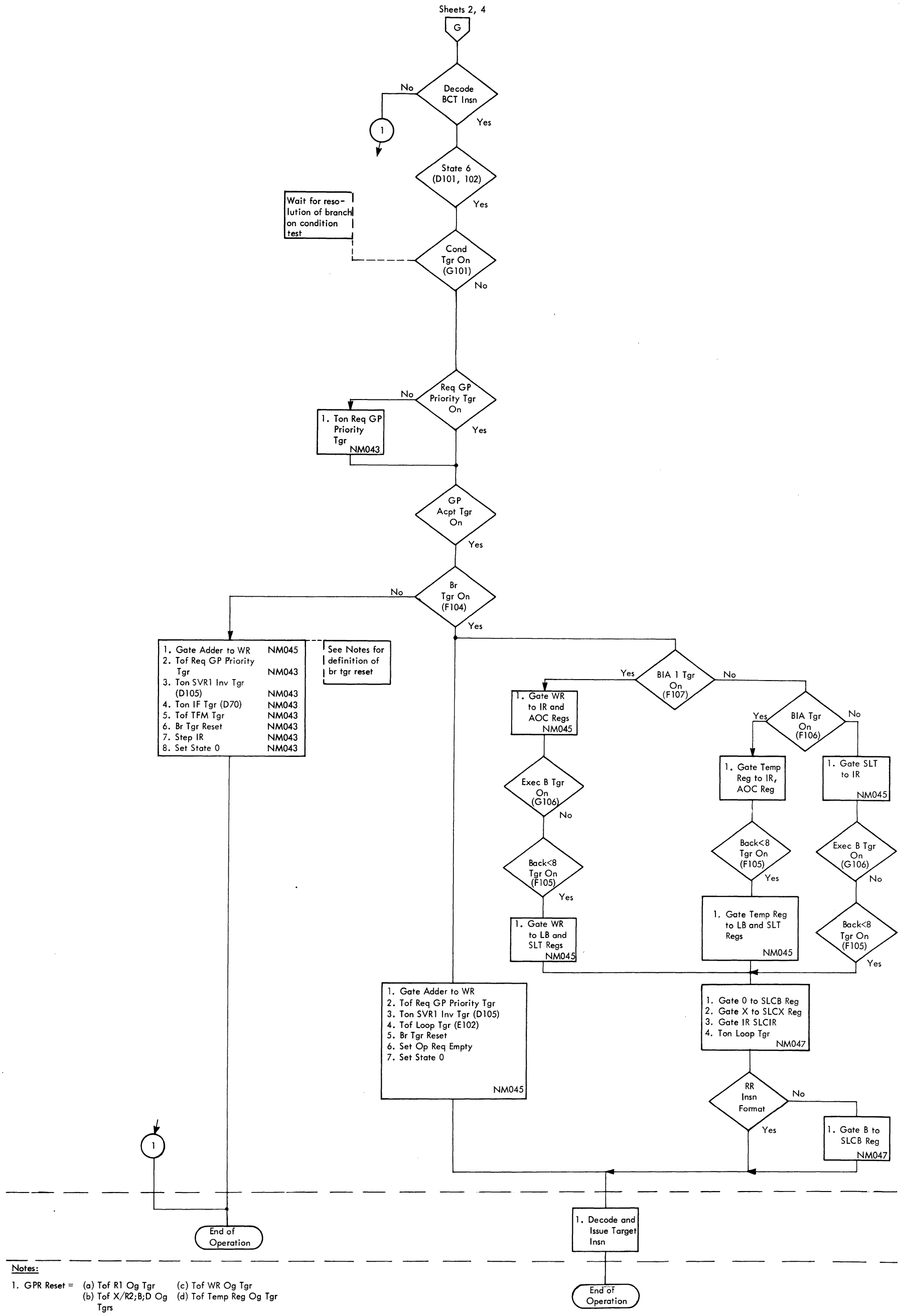


DIAGRAM 5-12. BRANCH ON COUNT SEQUENCE (SHEET 4 OF 5)



Notes:

1. GPR Reset = (a) Tof R1 Og Tgr (c) Tof WR Og Tgr
   (b) Tof X/R2;B;D Og Tgrs (d) Tof Temp Reg Og Tgr
2. Hot 1s Reset = (a) Set T/C Tgr to T (d) Tof IRCTR Og Tgr
   (b) Tof SVIR Og Tgrs (e) Tof +1 to Adder 31,29,28,TAT,25
   (c) Tof LB Og Tgrs
3. Br Tgr Reset = (a) Tof Br Tgr (d) Tof Back<8 Tgr
   (b) Tof BIA Tgr (e) Tof Exec B Tgr
   (c) Tof BIA1 Tgr (f) Tof BCUNCONT
4. Minimum time from request to accept is 2 cycles.
5. CP request off and priority granted causes fixed point execution unit to gate WR into GPR.

DIAGRAM 5-12. BRANCH AND COUNT SEQUENCE (SHEET 5 OF 5)

Objectives:

1. Decode and process a BX instruction.
2. Check all interlocks which may cancel the execution of this instruction.
3. Condition mode must be removed before processing of the BX instruction can be completed.
4. The BX instruction can either break, establish, or retain a loop mode of operation.

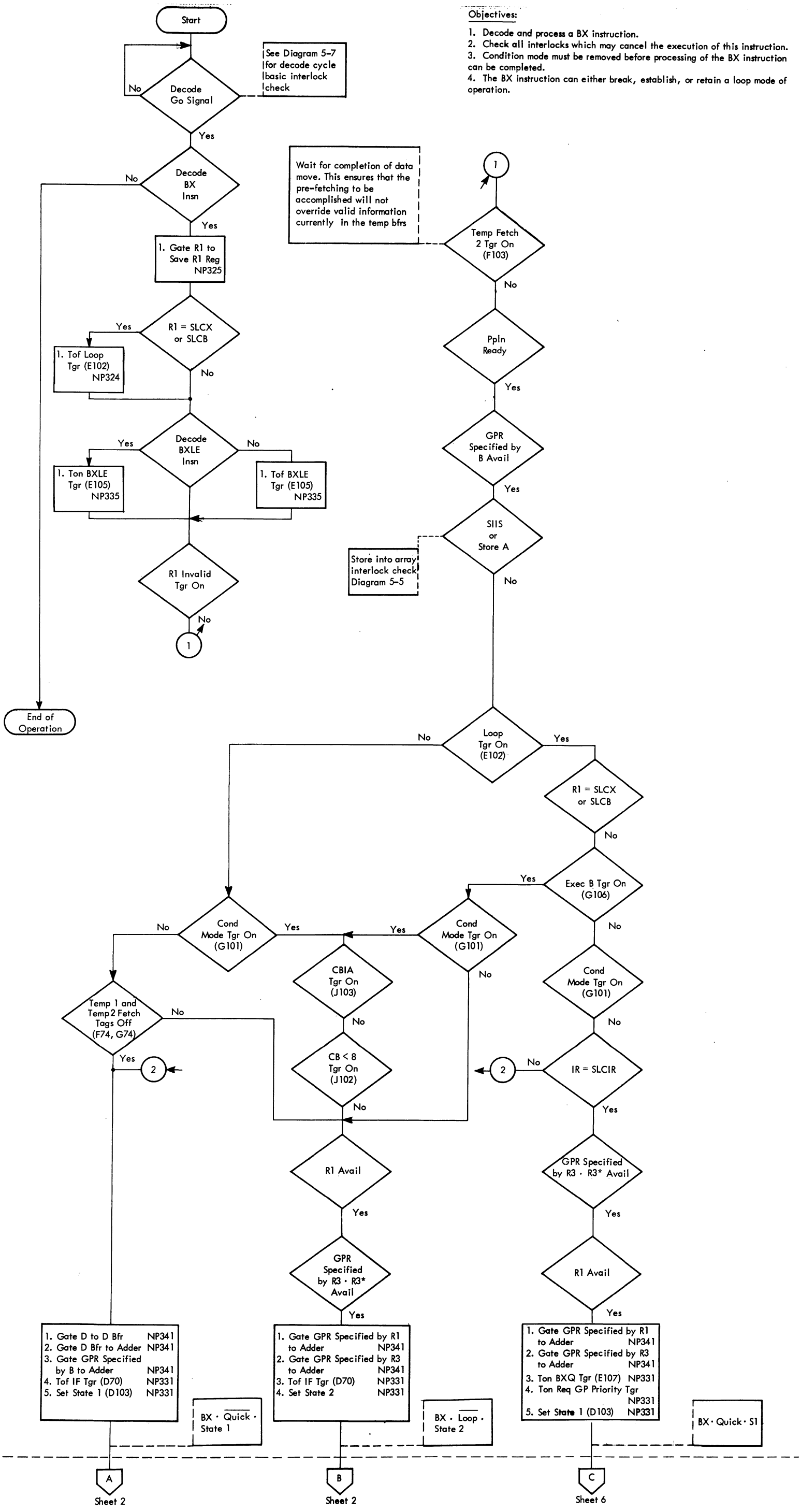


DIAGRAM 5-13. BRANCH ON INDEX SEQUENCE (SHEET 1 OF 6)

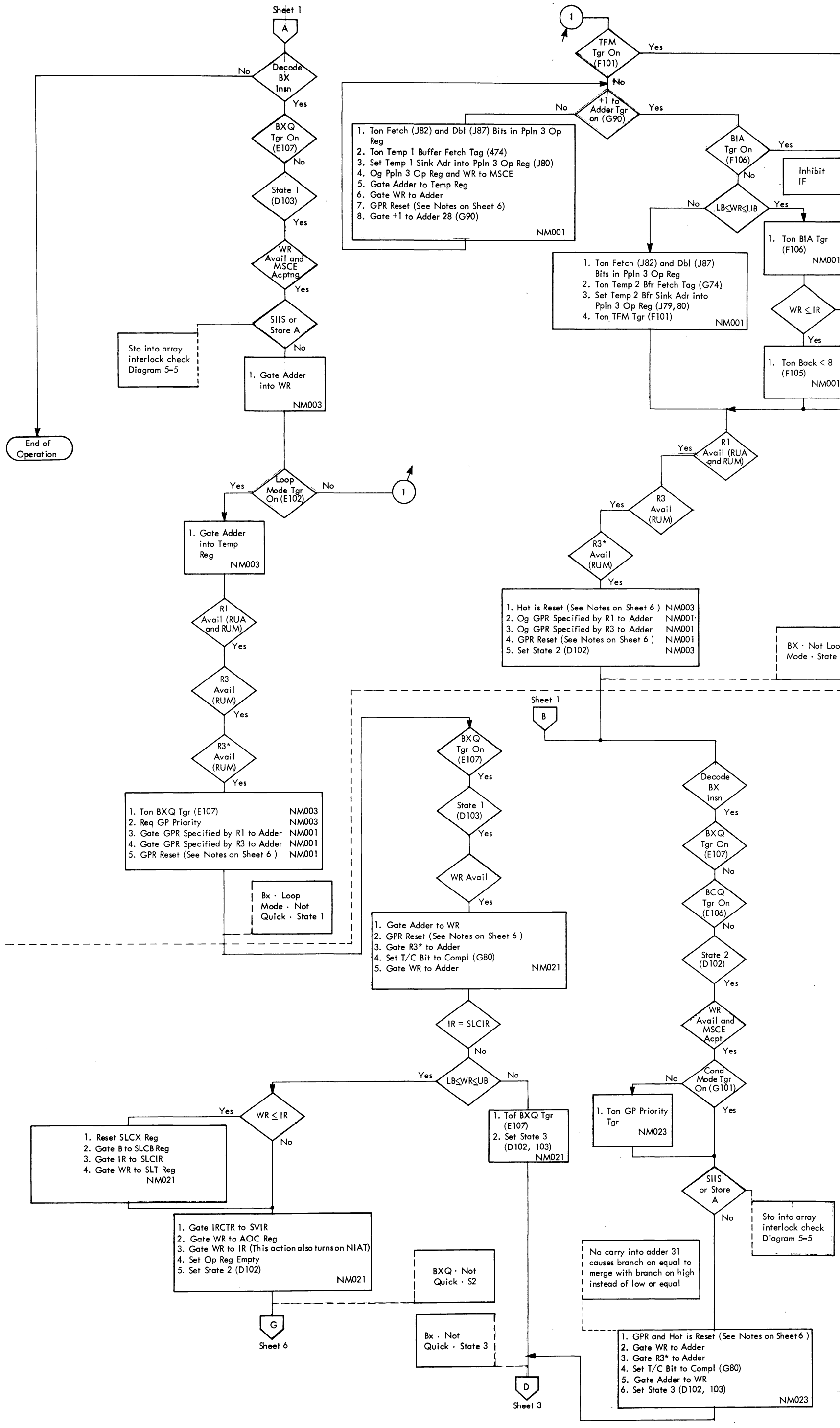


DIAGRAM 5-13. BRANCH ON INDEX SEQUENCE (SHEET 2 OF 6)

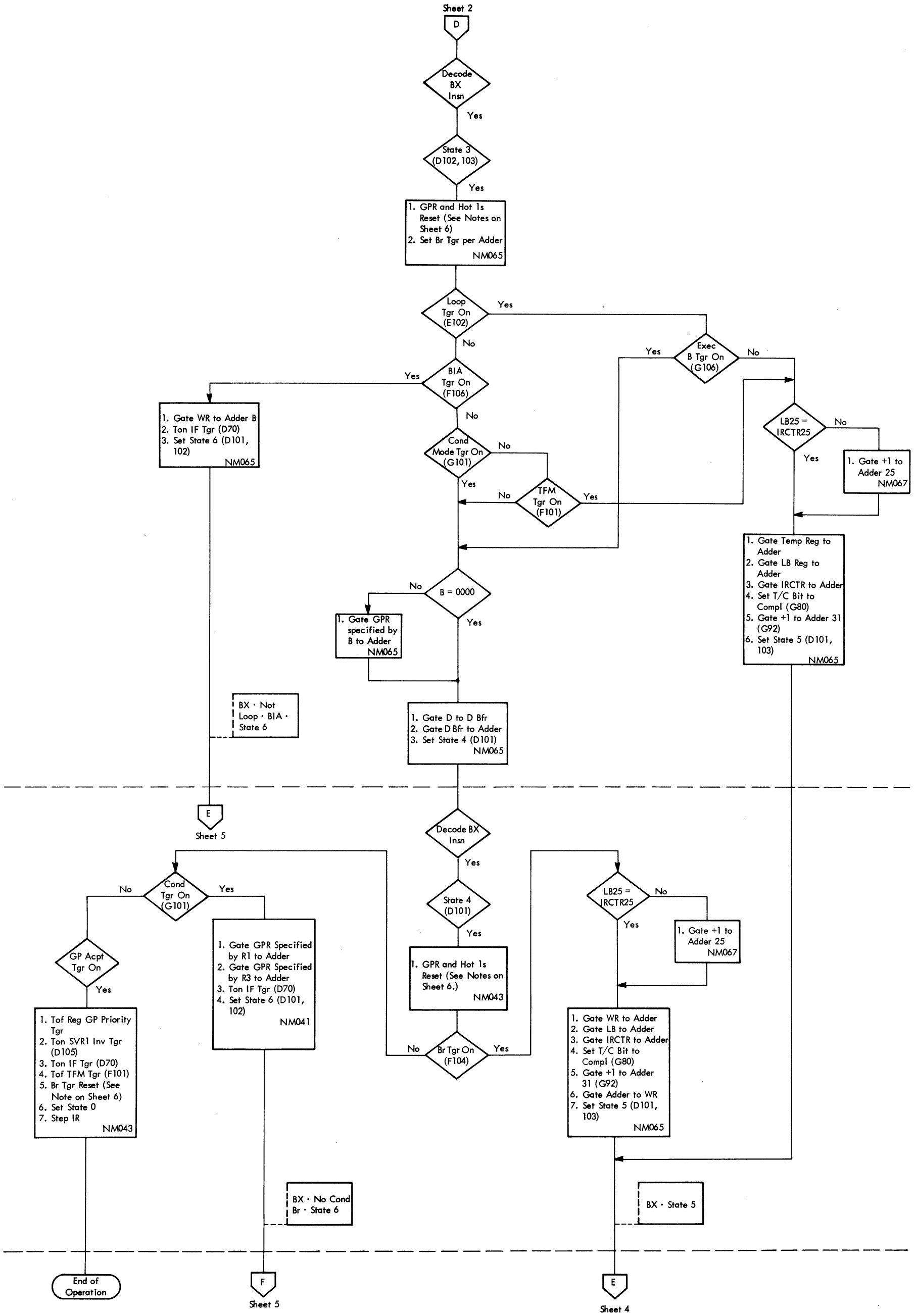


DIAGRAM 5-13. BRANCH ON INDEX SEQUENCE (SHEET 3 OF 6)

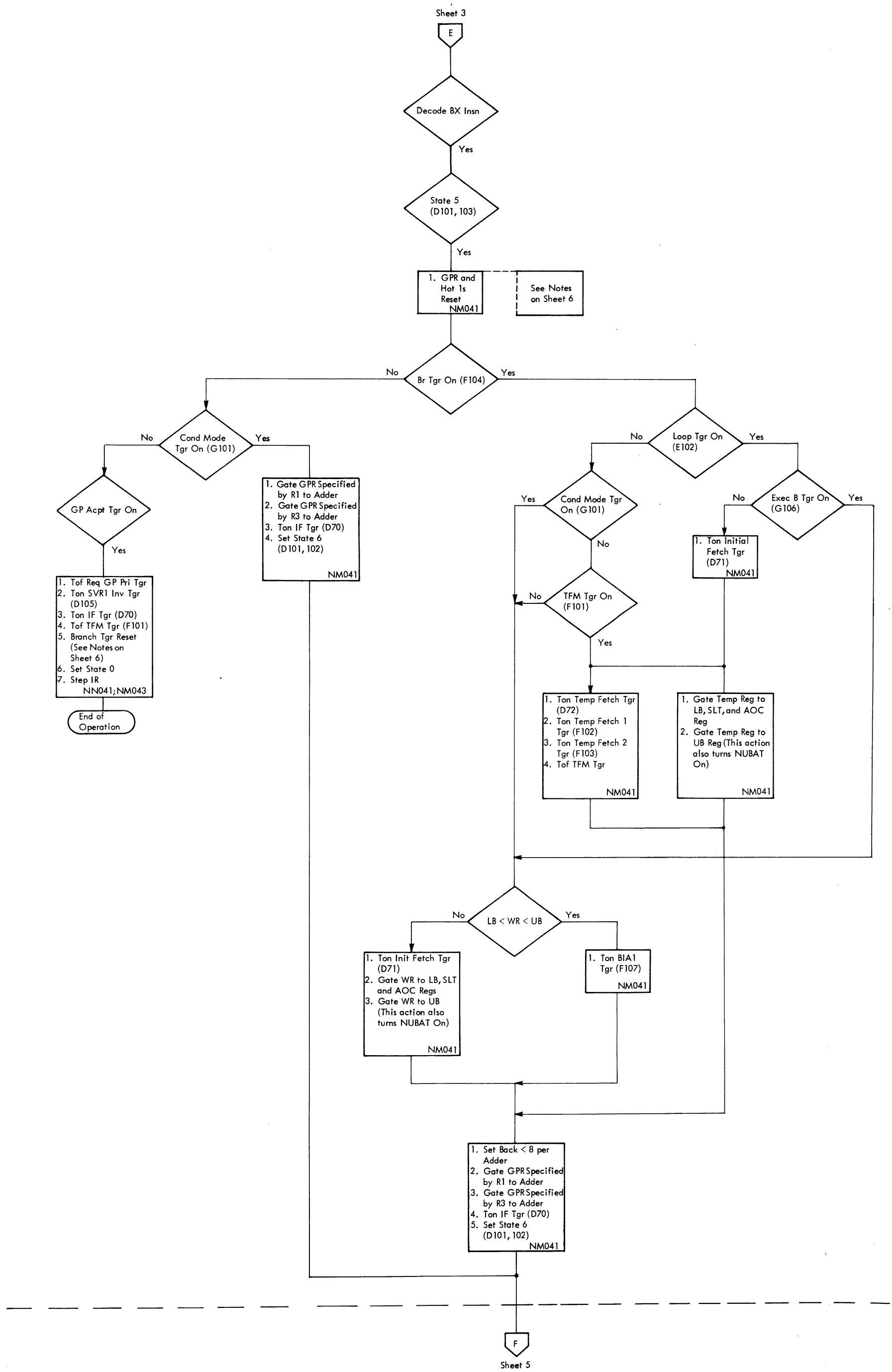


DIAGRAM 5-13. BRANCH ON INDEX SEQUENCE (SHEET 4 OF 6)

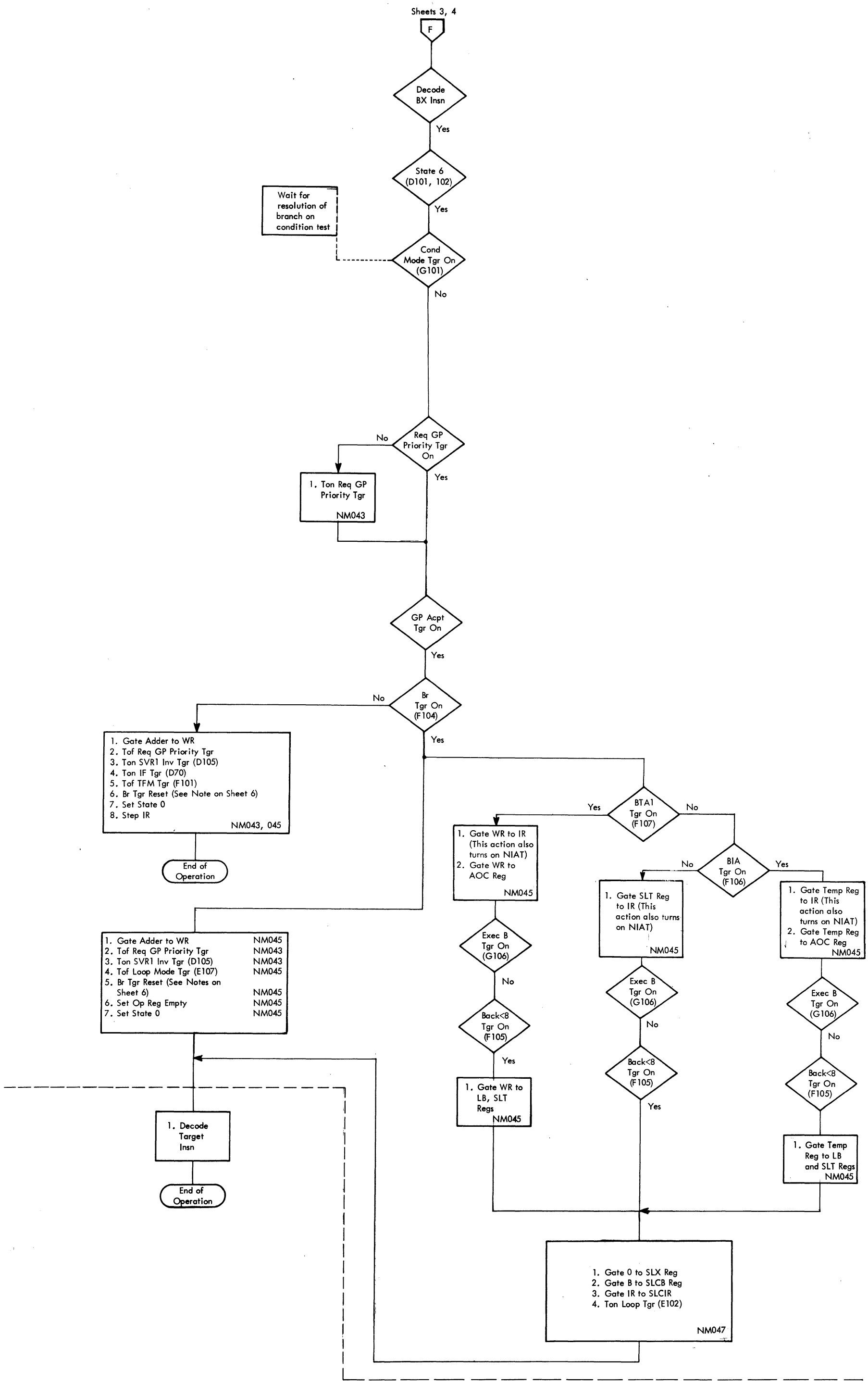
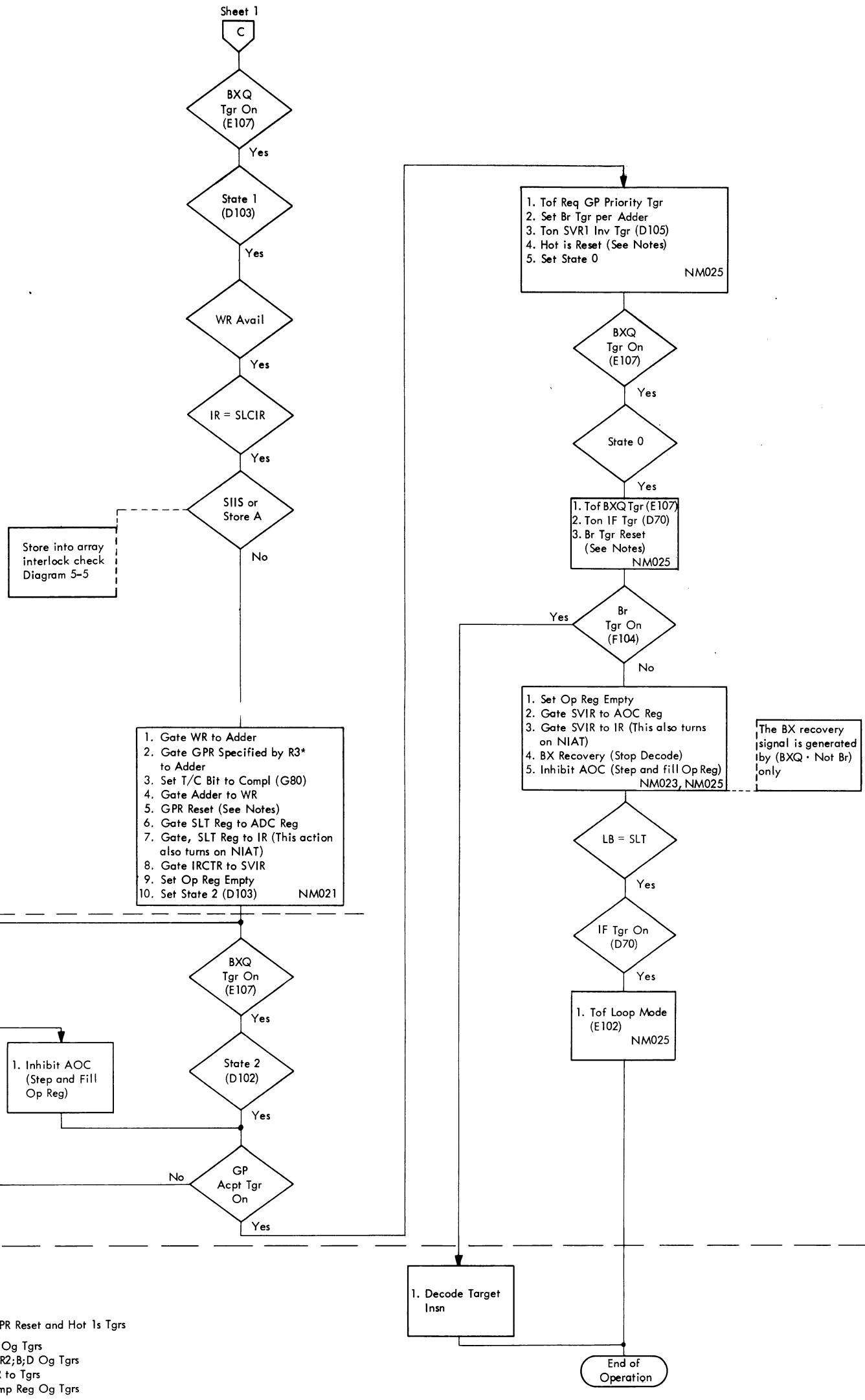


DIAGRAM 5-13. BRANCH ON INDEX SEQUENCE (SHEET 5 OF 6)



Notes:

1. Reset Og to Adder = GPR Reset and Hot 1s Tgrs
2. GPR Reset = (a) Tof R1 Og Tgrs
   (b) Tof X/R2;B;D Og Tgrs
   (c) Tof WR to Tgrs
   (d) Tof Temp Reg Og Tgrs
3. Hot 1s Reset =
   (a) Set T/C Tgr to T
   (b) Tof SVIR Og Tgr
   (c) Tof LB Og Tgr
   (d) Tof IRCTR Og Tgr
   (e) Tof +1 to Adder 31,29,28,TAT,25
4. Br Tgr Reset =
   (a) Tof Br Tgr
   (b) Tof BIA Tgr
   (c) Tof BIA1 Tgr
   (d) Tof Back <8 Tgr
   (e) Tof Exec B Tgr
   (f) Tof BCUNCONT
5. Minimum time from request to accept is 2 cycles
6. CP request off and priority granted, causes fixed point execution unit to gate WR into GPR

DIAGRAM 5-13. BRANCH ON INDEX SEQUENCE (SHEET 6 OF 6)

Objectives:

1. Decode and process a BAL instruction.
2. Check all interlocks which may cancel the execution of the instruction.
3. Set the rightmost 32 bits of the PSW, including the updated instruction address into the GPR staging area
4. Notify FXA to read link data into GPR specified by R1.

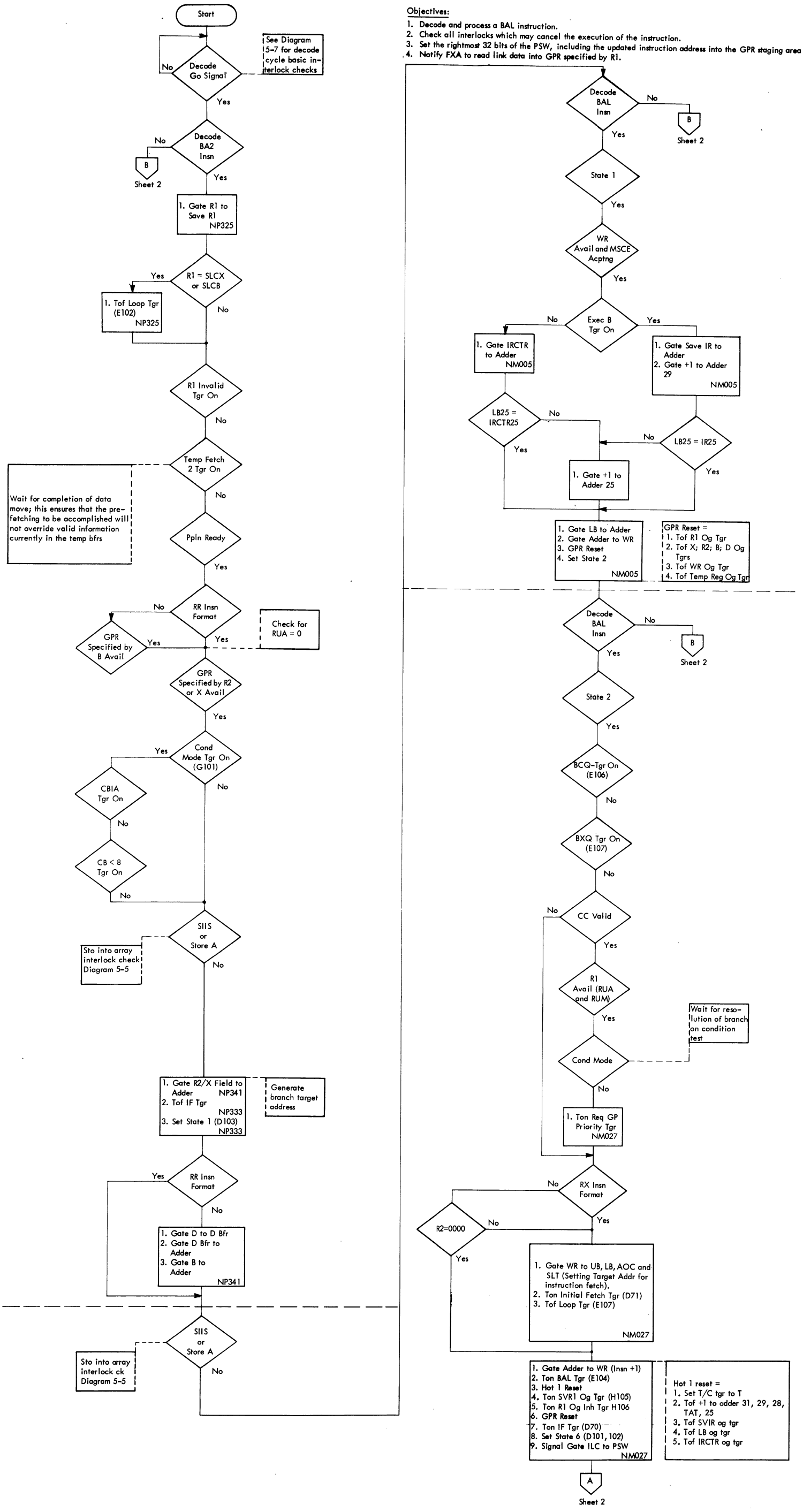


DIAGRAM 5-14. BRANCH AND LINK SEQUENCE (SHEET 1 OF 2)

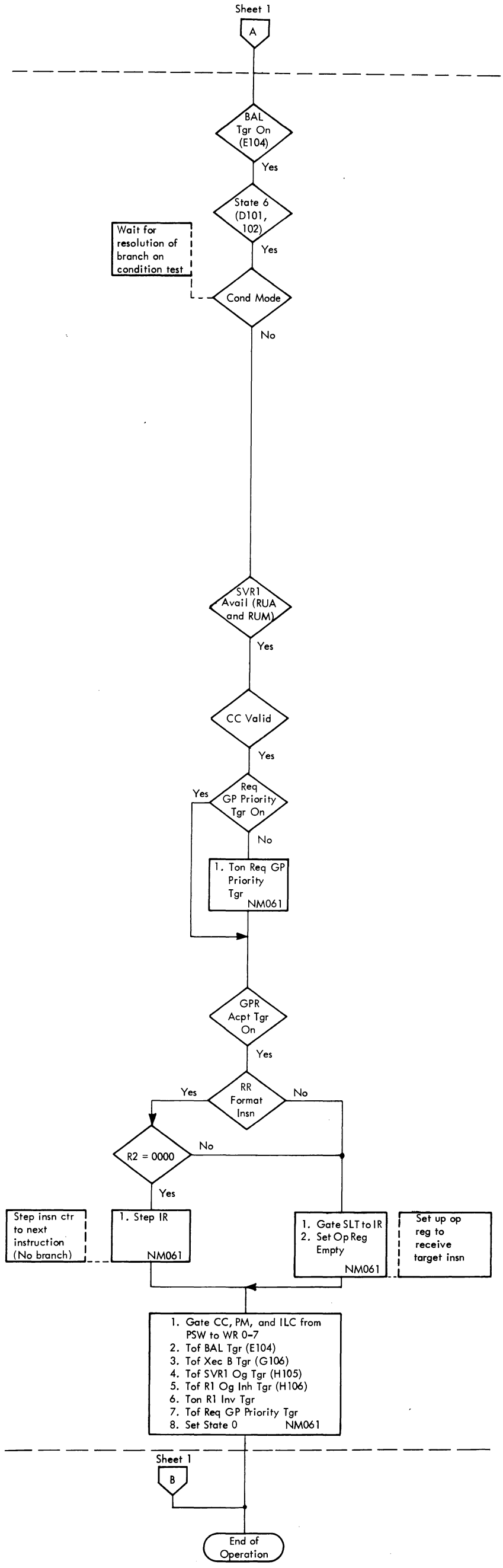


DIAGRAM 5-14. BRANCH AND LINK SEQUENCE (SHEET 2 OF 2)

Objectives:

1. To permit execution of an instruction that is not residing in the current instruction sequence.
2. To check all exceptions that can occur during the processing of an execute instruction.
3. Once the subject instruction is executed; update instruction counter, prepare op reg to recover to normal instruction processing, advance to next instruction, and resume instruction fetching.

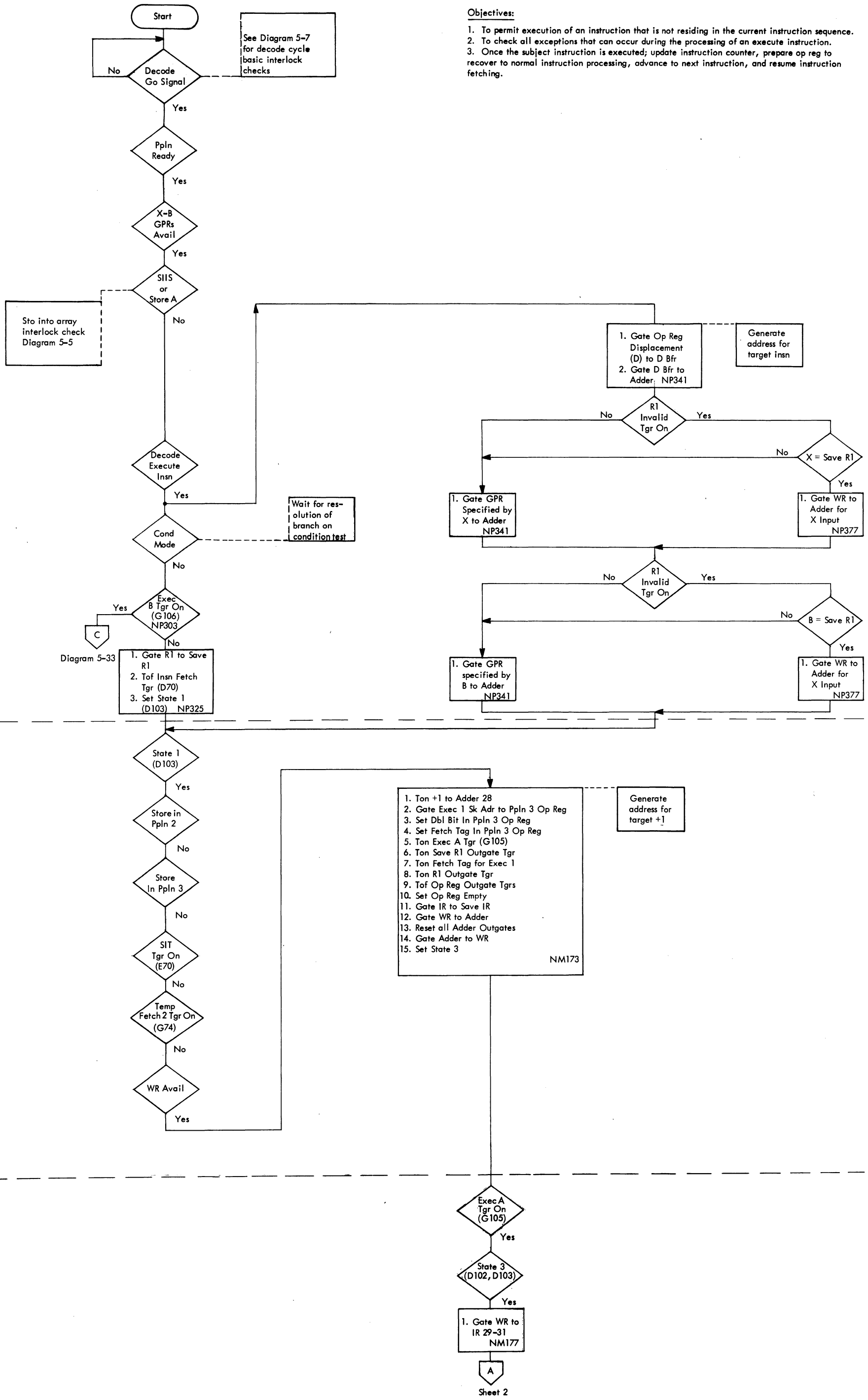


DIAGRAM 5-15. EXECUTE SEQUENCE (SHEET 1 OF 3)

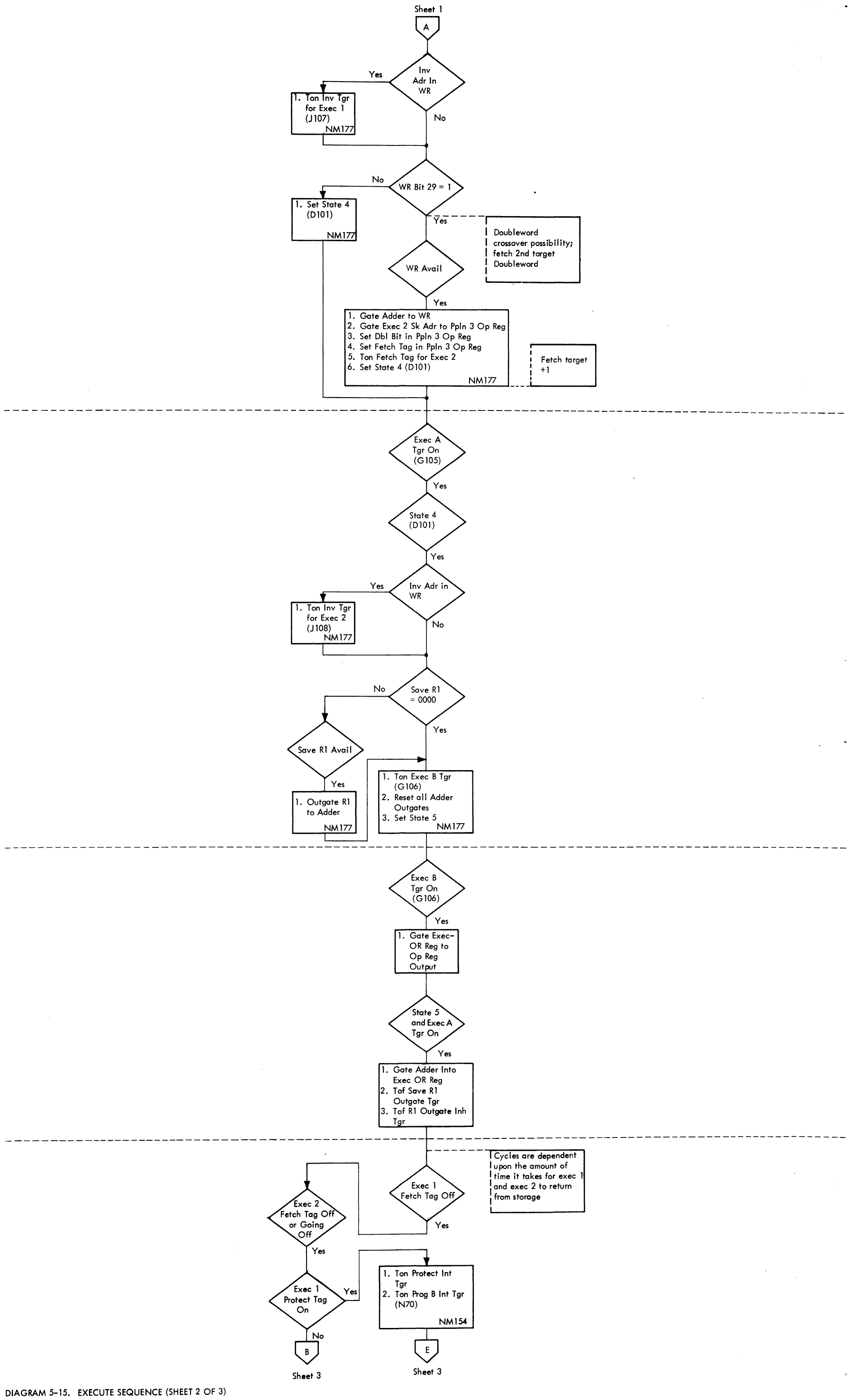


DIAGRAM 5-15. EXECUTE SEQUENCE (SHEET 2 OF 3)

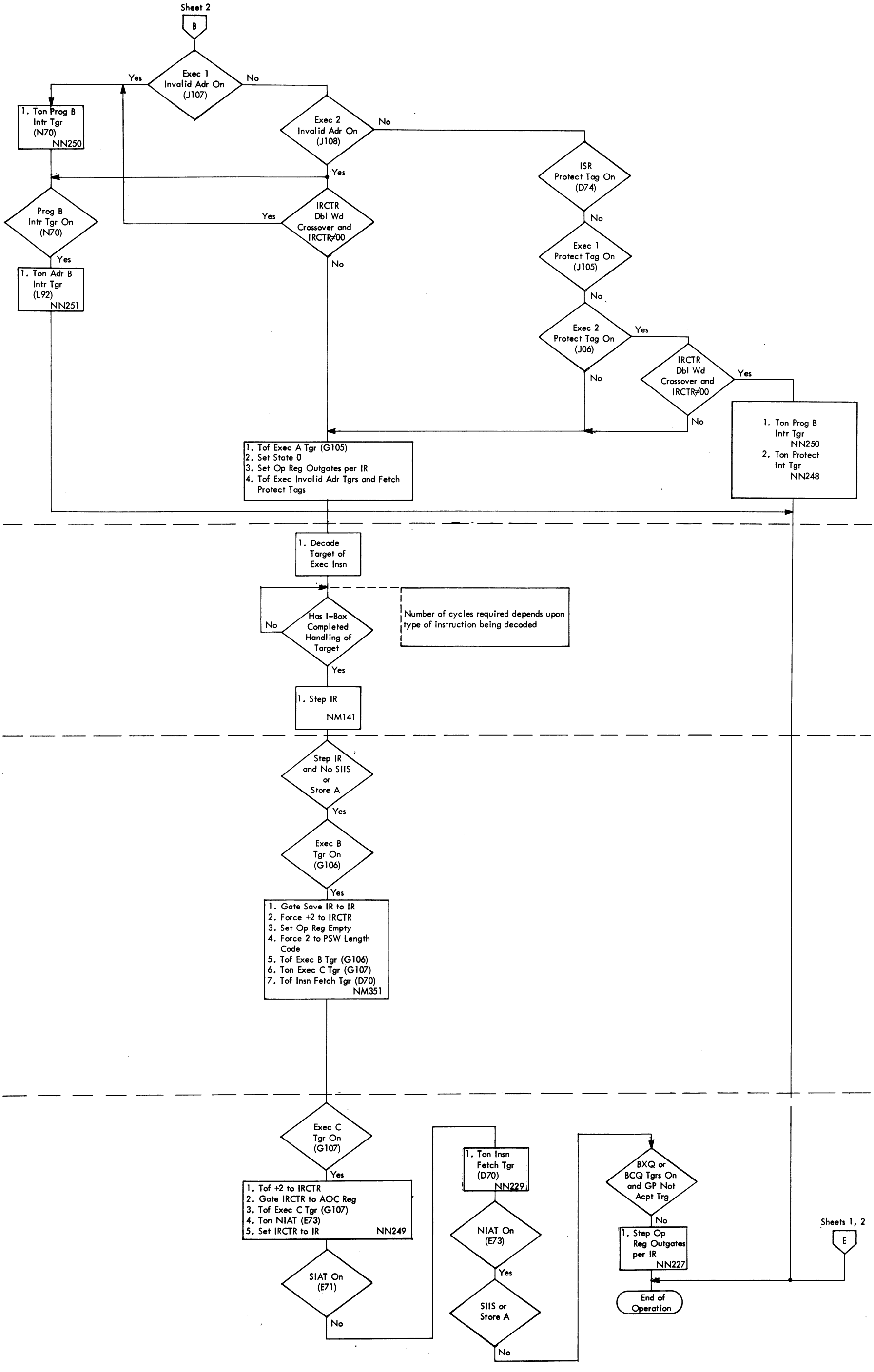


DIAGRAM 5-15. EXECUTE SEQUENCE (SHEET 3 OF 3)

Objectives:

1. PSW located at storage address specified by B1 and D1 fields.
2. Operand address must have its three low-order bits zero to designate a double word.
3. CPE must be in supervisory state.
4. Privileged operation interrupt will occur if CPE is in problem state.
5. Load PSW is only instruction available for entering problem or wait states.

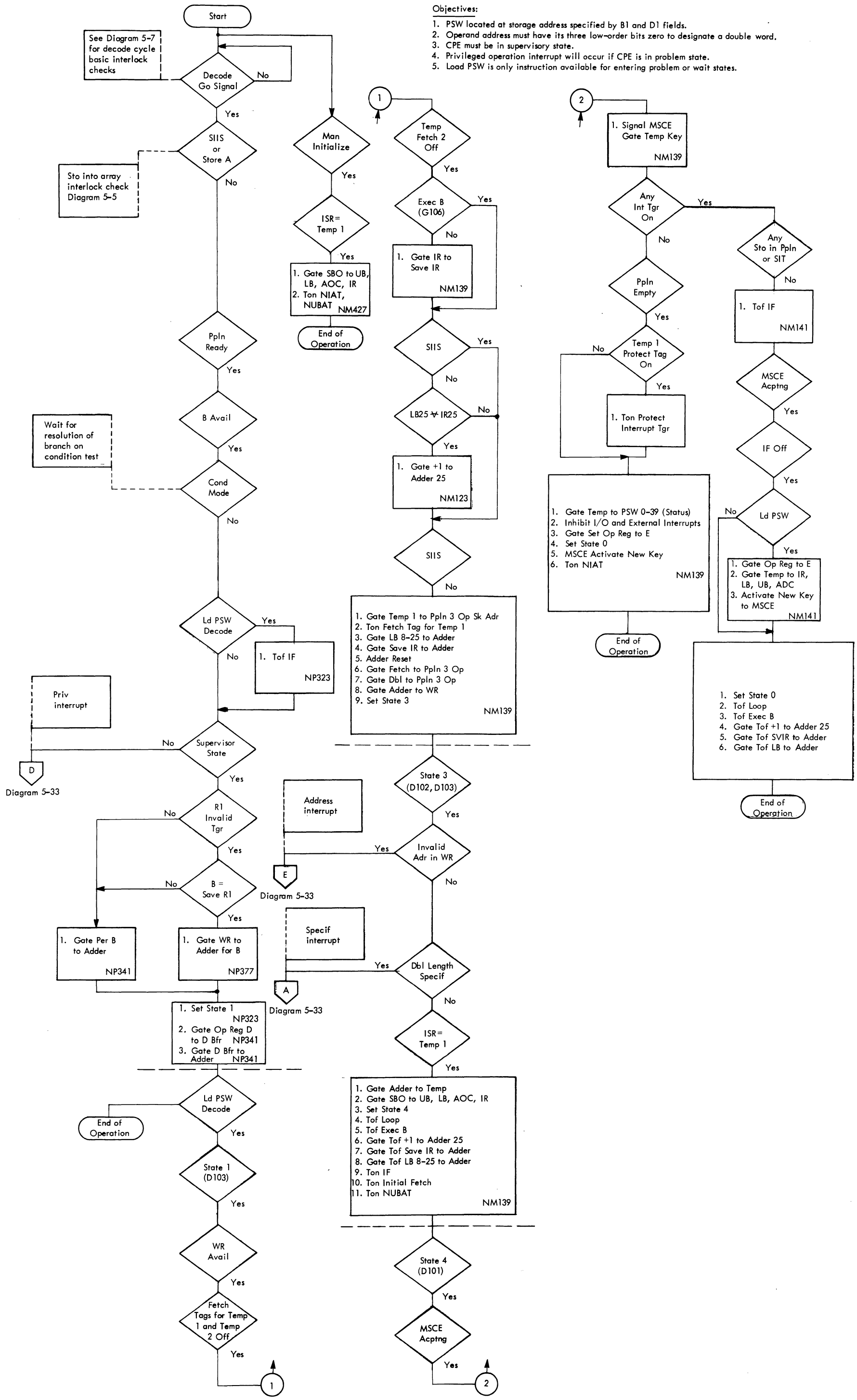


DIAGRAM 5-16. LOAD PSW SEQUENCE

Objectives:

1. Decode and issue a SPM or a SVC instruction.
2. Check all interlocks that can detain or cancel the execution of these instructions.
3. SPM - If no interrupt, set new mask and CC to PSW
4. SVC - When hardware into which the interrupt code is to be placed, become available, generate precise interrupt signal.
5. Step instruction to next instruction.

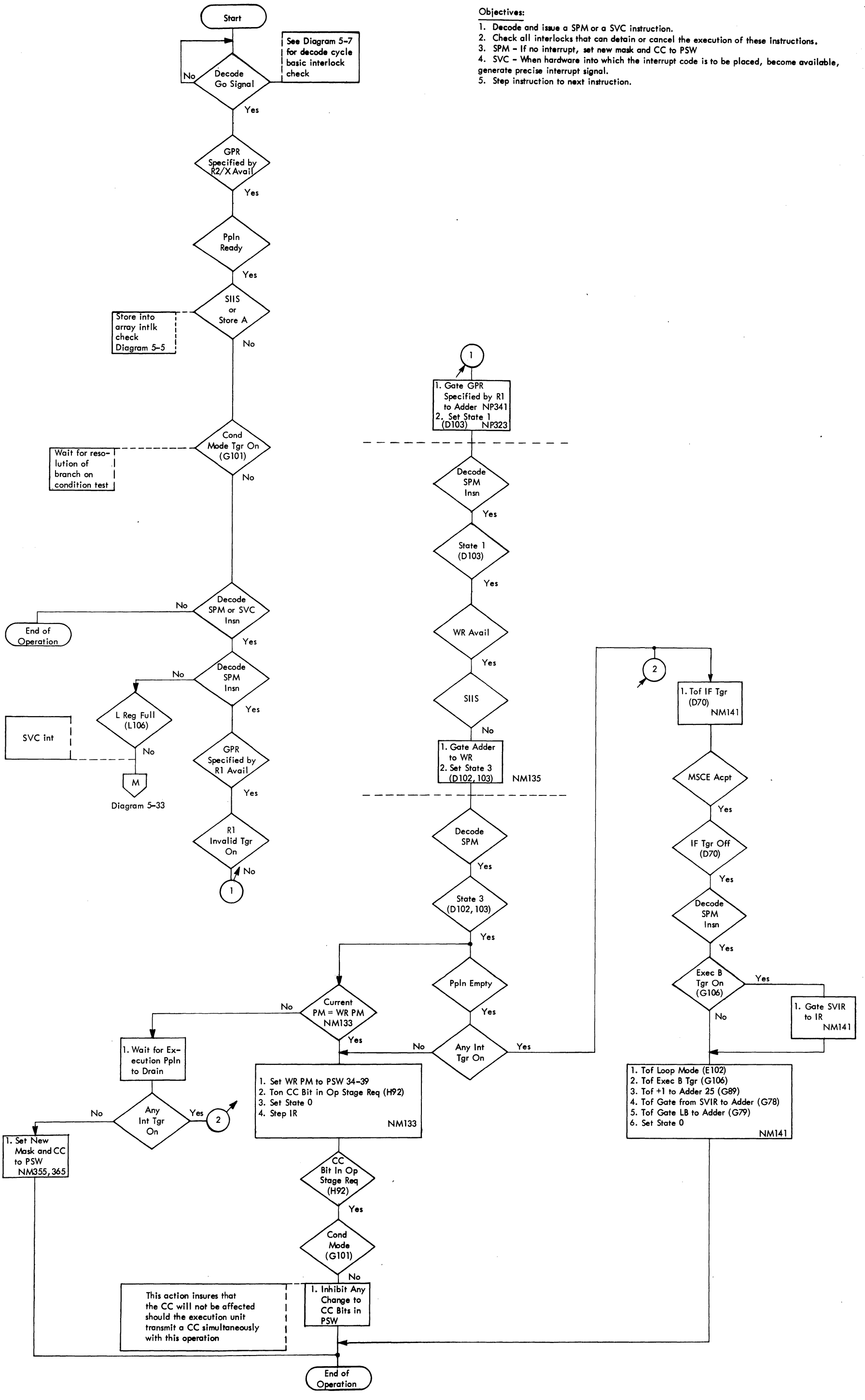


DIAGRAM 5-17. SET PROGRAM MASK AND SUPERVISOR CALL SEQUENCE

Objectives:

1. The byte located at the B1 and D1 instruction fields replaces system mask of current PSW.
2. VFLEU does byte selection for storage operand.
3. Correct system mask is stored because I/O and external interrupts are inhibited during instruction execution.

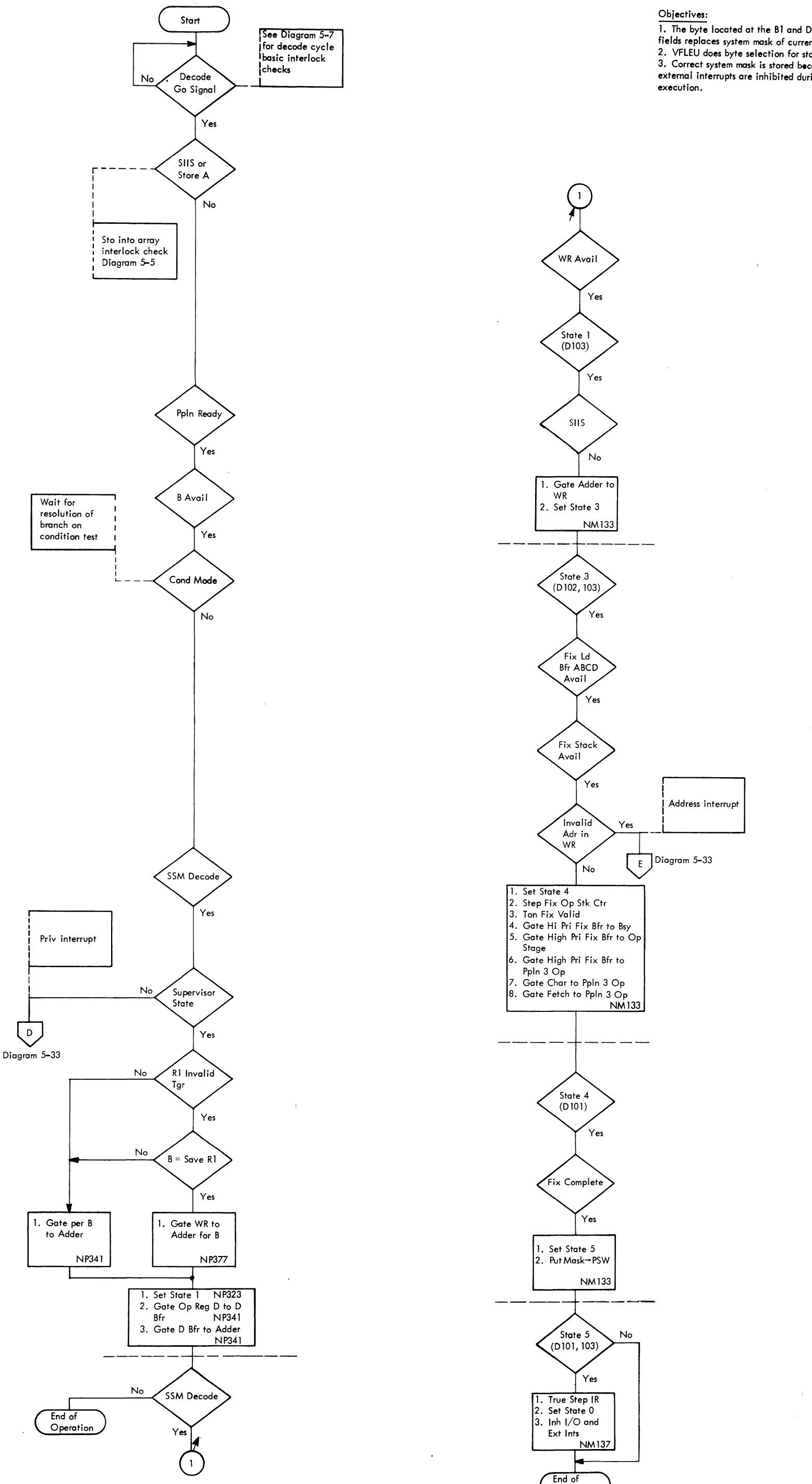


DIAGRAM 5-18. SET SYSTEM MASK SEQUENCE

Objectives:

1. To decode and issue a ISK or SSK instruction.
2. Check all interlocks that can detain or cancel the execution of these instructions.
3. ISK - Allocate buffers as appropriate; issue instruction to FXA; place address into MSCE interface and initiate storage fetch; establish sink interlock on GPR specified by R1.
4. SSK - Place data into MSCE interface; notify MSCE; delay one cycle.
5. Step instruction counter to next instruction and set length into PSW.

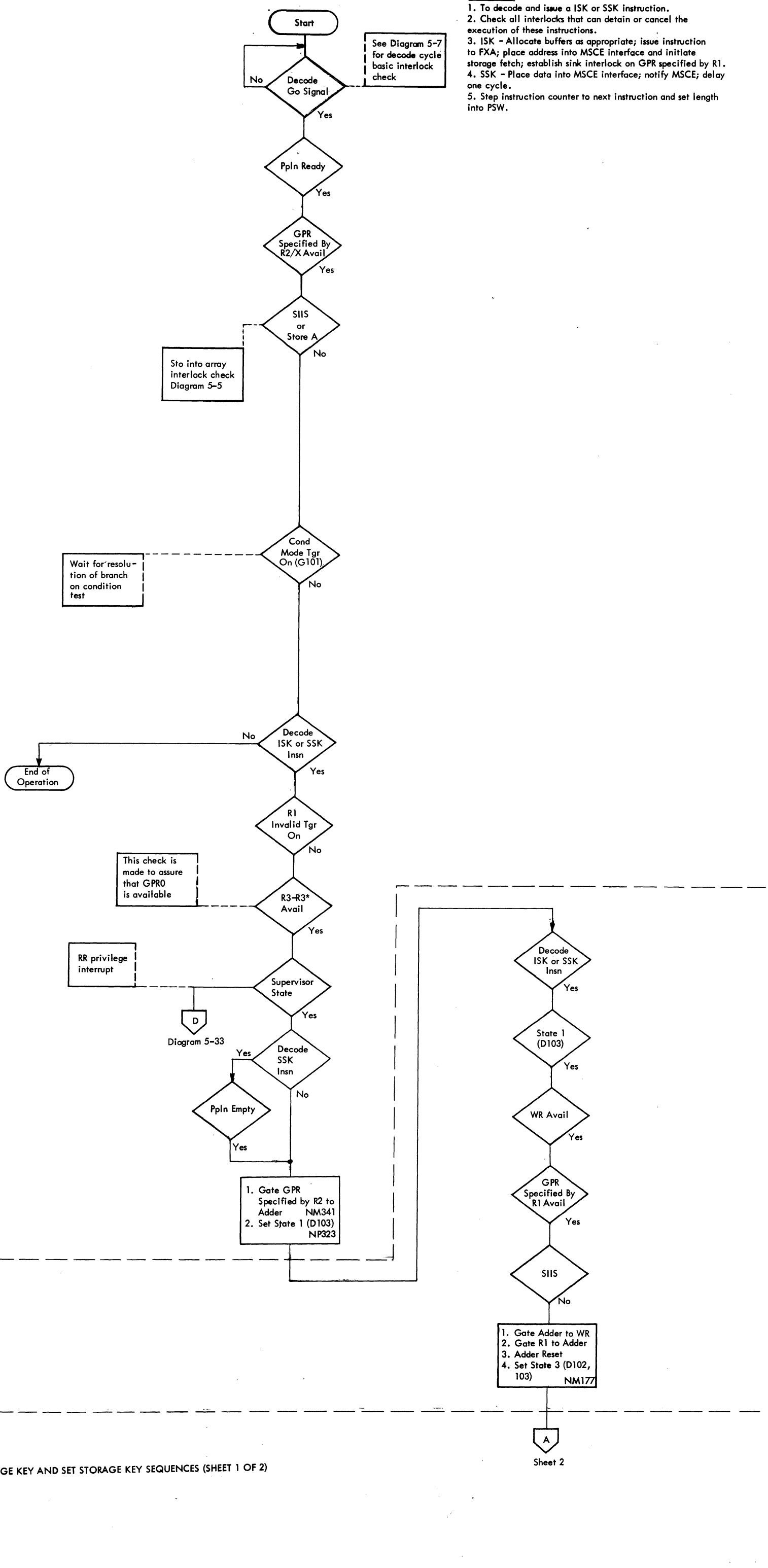


DIAGRAM 5-19. INSERT STORAGE KEY AND SET STORAGE KEY SEQUENCES (SHEET 1 OF 2)

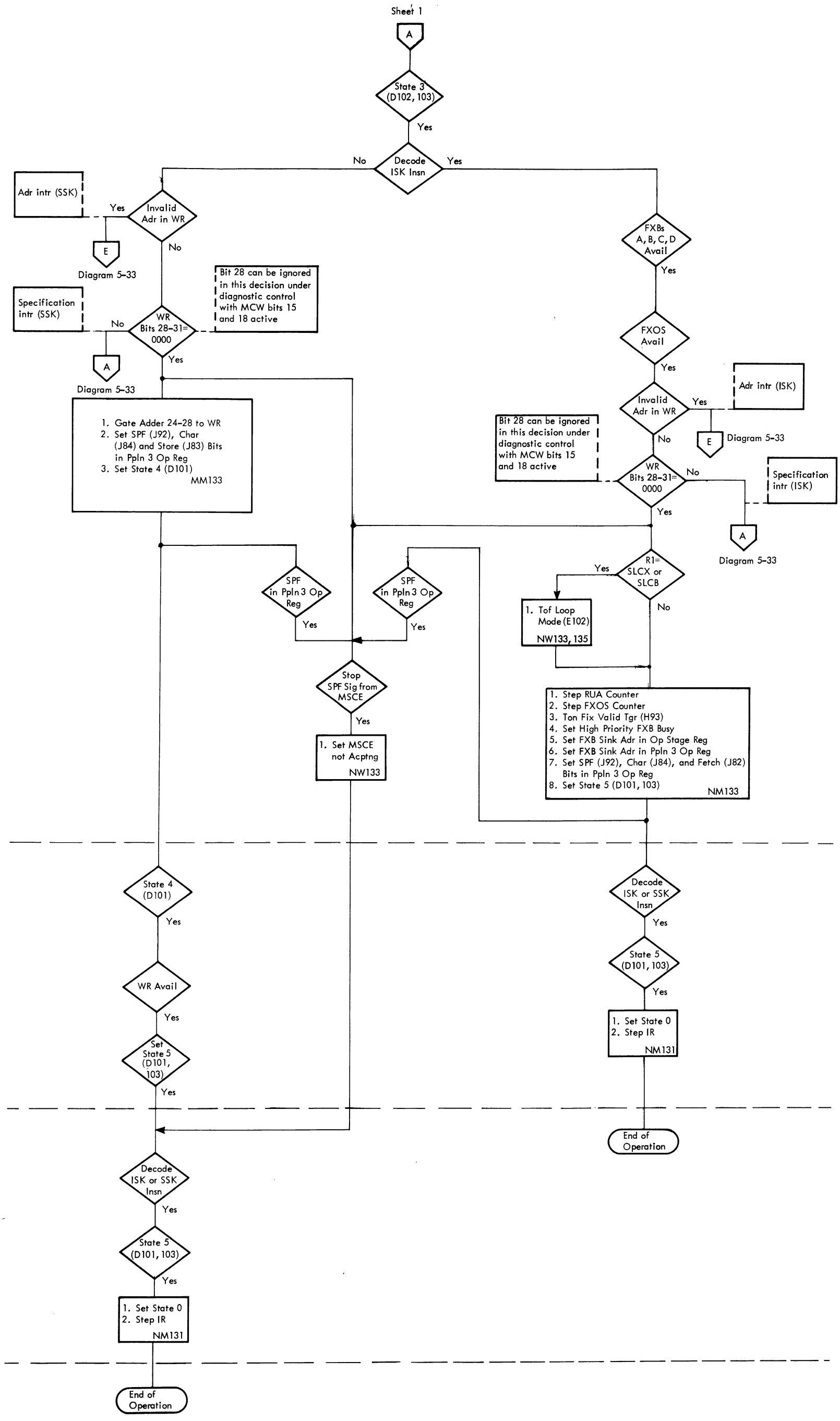


DIAGRAM 5-19. INSERT STORAGE KEY AND SET STORAGE KEY SEQUENCES (SHEET 2 OF 2)

Objectives:

1. Decode and issue a RDD or WRD instruction.
2. Check all interlocks that can detain or cancel these instructions.
3. RDD – Generate address and test for valid storage; check for FXOS and SAR availability; issue instruction to FXA; wait for FXA completion of this instruction before proceeding to decode cycle.
4. WRD – Generate address and test for valid storage; check for FXOS and FXB availability; issue instruction to FXA and proceed to decode cycle.

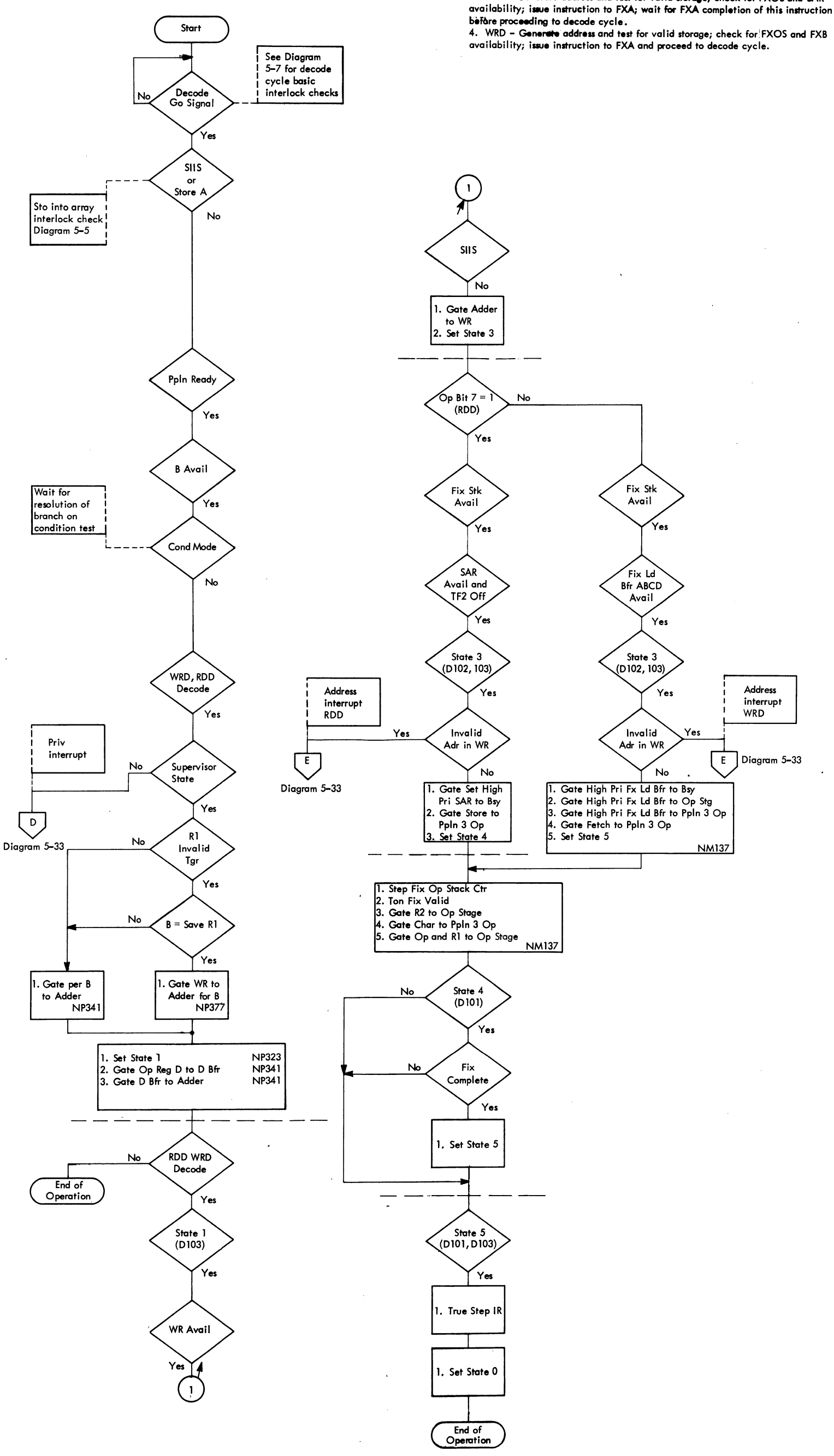


DIAGRAM 5-20. READ, WRITE DIRECT SEQUENCE

Objectives:

1. Decode and issue diagnose instruction.
2. Set MCW register which controls CPE and channel functions.
3. Verification of proper CPU equipment functions.

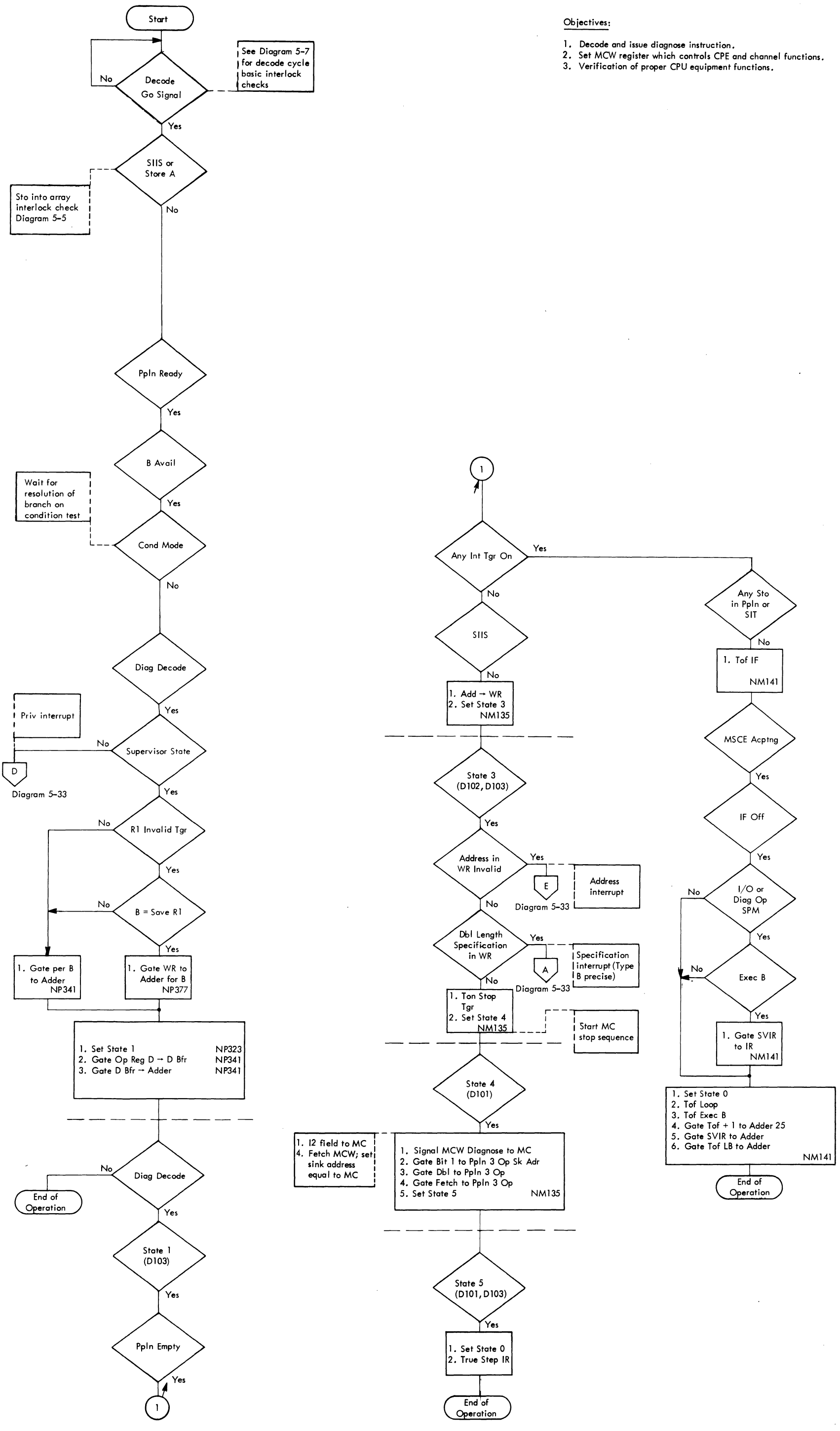


DIAGRAM 5-21. DIAGNOSE SEQUENCE

Objectives:
1. Decode and issue a TS, IC, or STC instruction.
2. Check all interlocks that can detain or cancel the execution of these instructions.
3. Issue instruction to FXA.
4. Allocate buffers as appropriate.
5. Step instruction counter to next instruction.

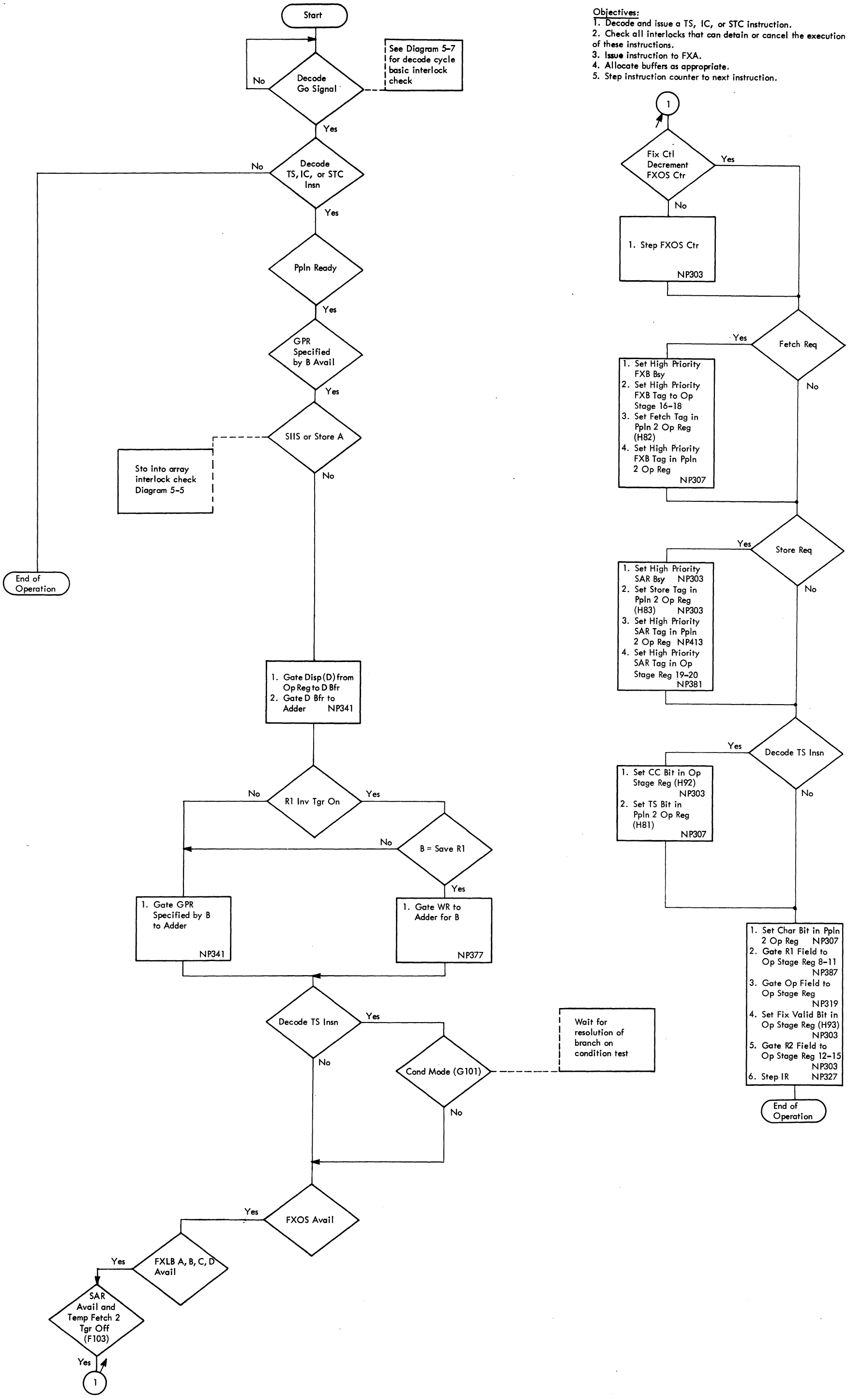


DIAGRAM 5-22. INSERT CHARACTER, STORE CHARACTER, AND TEST AND SET SEQUENCE

Objectives:

1. Decode and issue a LA instruction to the FXA.
2. Place the address into the staging area for transfer to the FXA.
3. Notify the FXA to take the address.
4. Move instruction counter to next instruction.

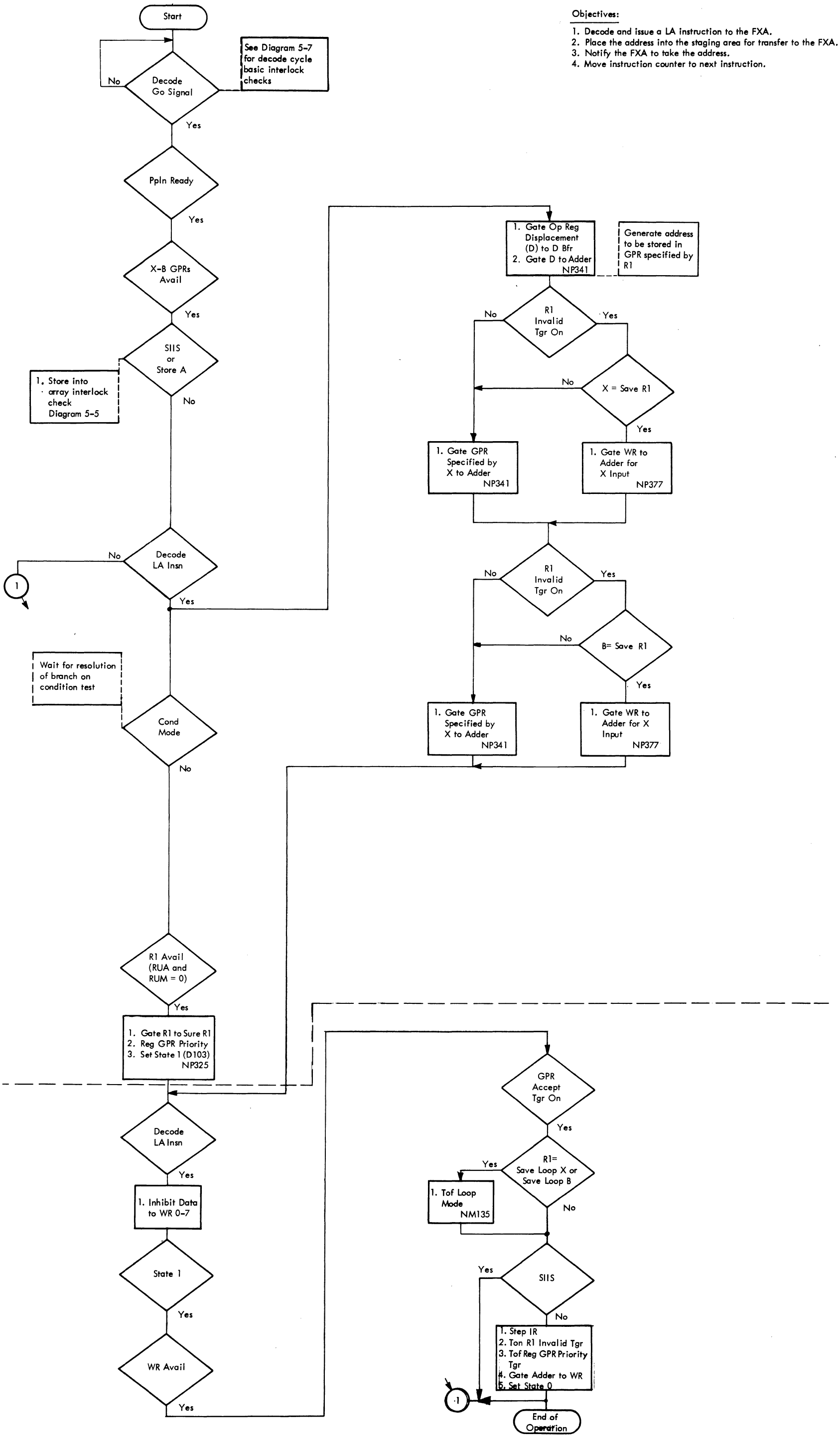


DIAGRAM 5-23. LOAD ADDRESS SEQUENCE

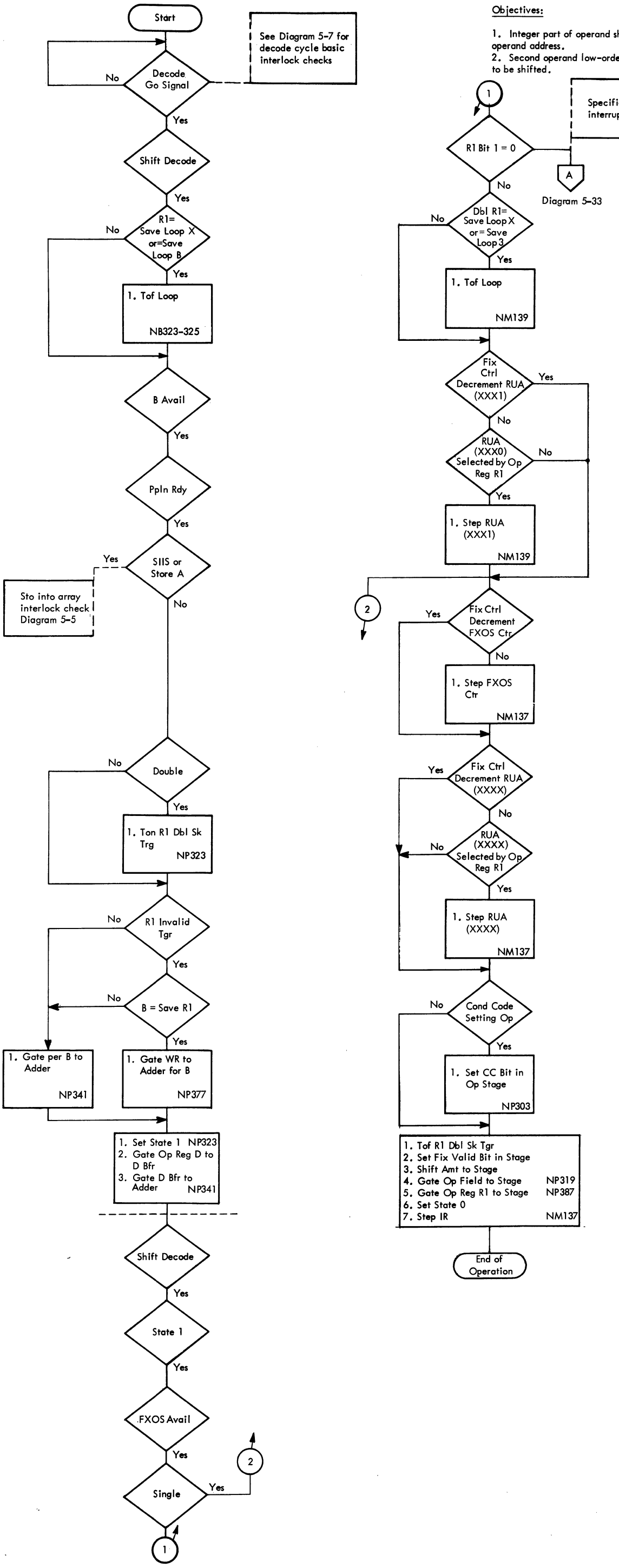


DIAGRAM 5-24. SHIFT SEQUENCE

Objectives:
1. Decode and issue the CVB or CVD instruction to the FXA
2. Place the sink address into the MSCE interface hardware and establish control signals as appropriate for storage.
3. Allocate FXP and FLP buffers as appropriate.
4. Establish the sink interlock on the R1 specified GPR.
5. Set length into the PSW.
6. Move instruction counter to next instruction.

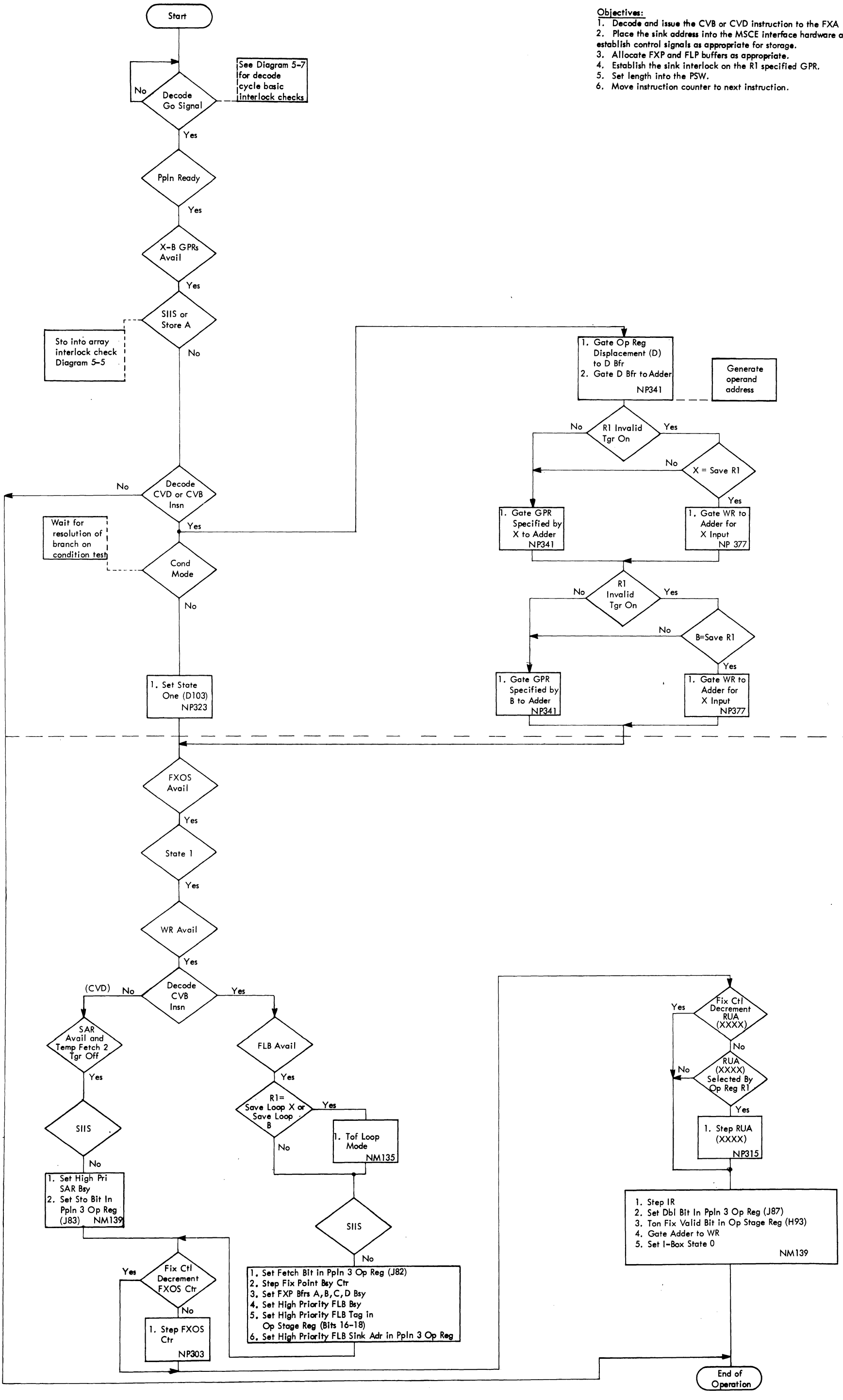
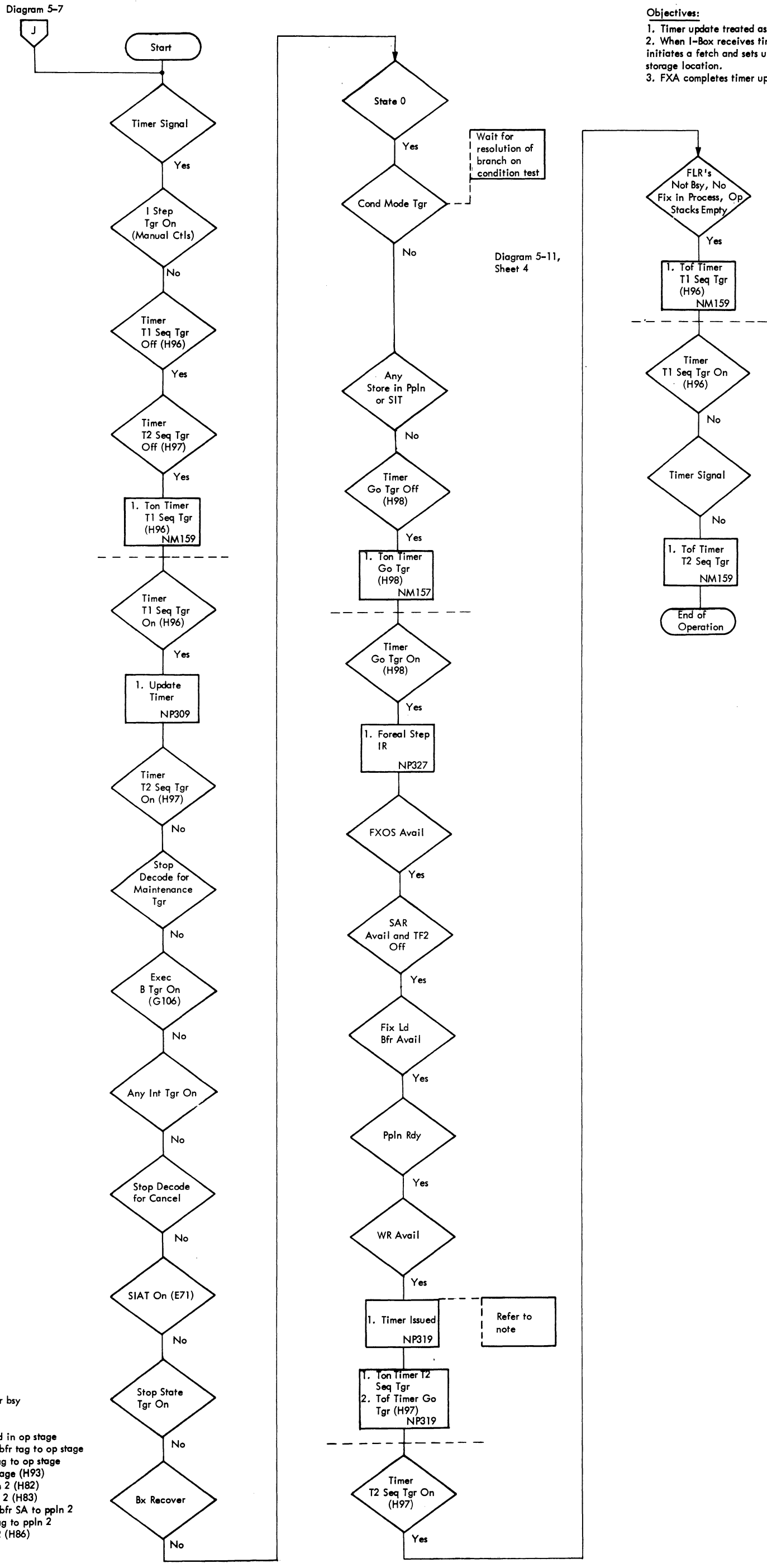


DIAGRAM 5-25. CONVERT SEQUENCE

Diagram 5-7

Objectives:
1. Timer update treated as fixed point operation.
2. When I-Box receives timer signal, it effectively initiates a fetch and sets up store for timer storage location.
3. FXA completes timer update and store operation.



Note:
Timer issued leads to:
1. Set high pri fix ld bfr bsy
2. Set high pri SAR bsy
3. Step FXOS ctr
4. Gate zero to op field in op stage
5. Gate high pri fix ld bfr tag to op stage
6. Gate high pri SAR tag to op stage
7. Set fix valid in op stage (H93)
8. Ton fetch tag in ppln 2 (H82)
9. Ton store tag in ppln 2 (H83)
10. Gate high pri fix ld bfr SA to ppln 2
11. Gate high pri SAR tag to ppln 2
12. Set full tag in ppln 2 (H86)
13. Ton timer adr tgr

DIAGRAM 5-26. TIMER UPDATE SEQUENCE

Objectives:

1. Decode and issue an I/O instruction.
2. Check all interlocks that can detain or cancel the execution of this instruction.
3. Gate bits 16-33 of B1 and D1 field (contained in WR) and a line specifying one of the I/O insn, to all channels.
4. When selected channel responses, set condition code and terminate instruction.

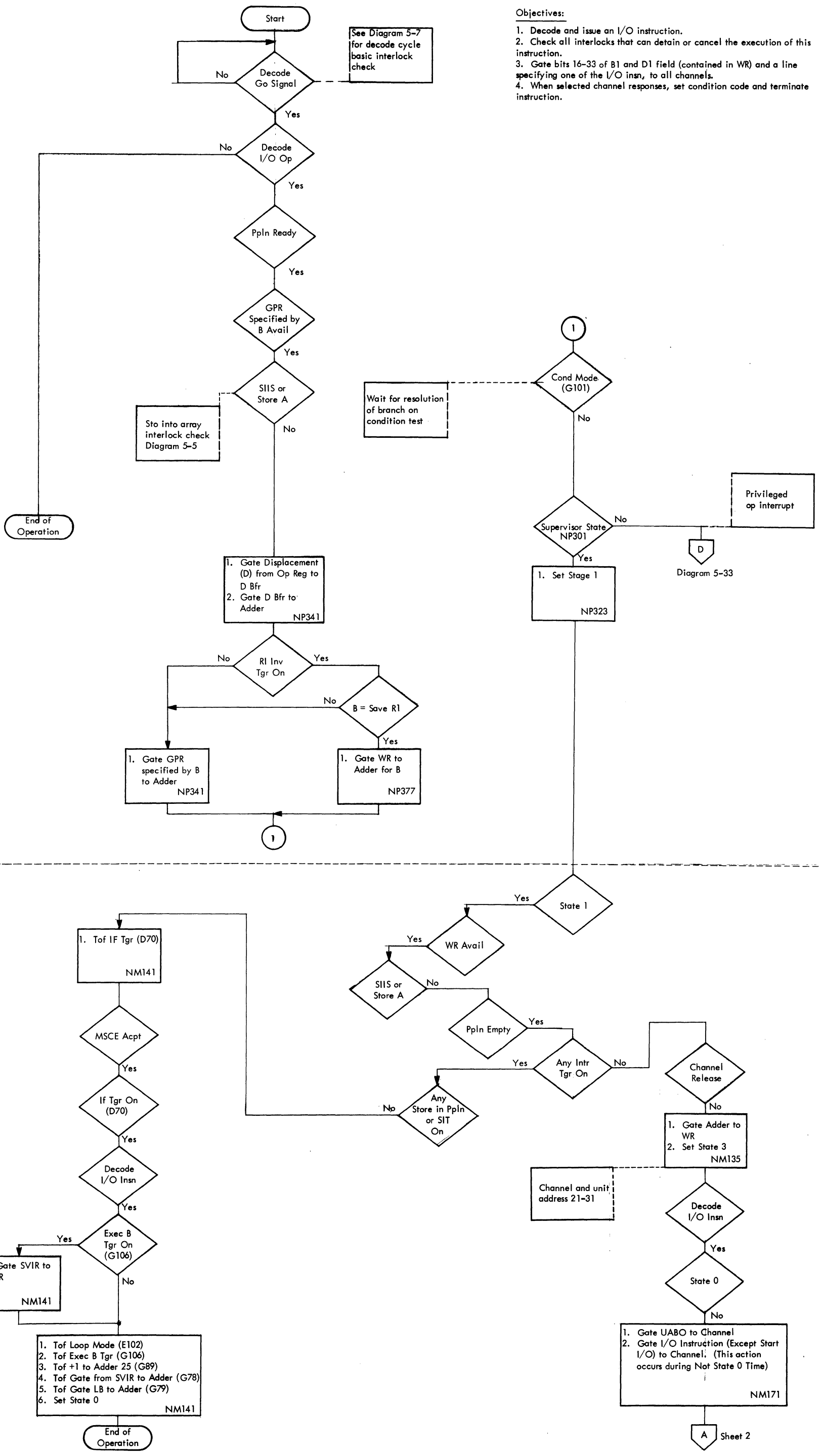


DIAGRAM 5-27. I/O SEQUENCE (SHEET 1 OF 2)

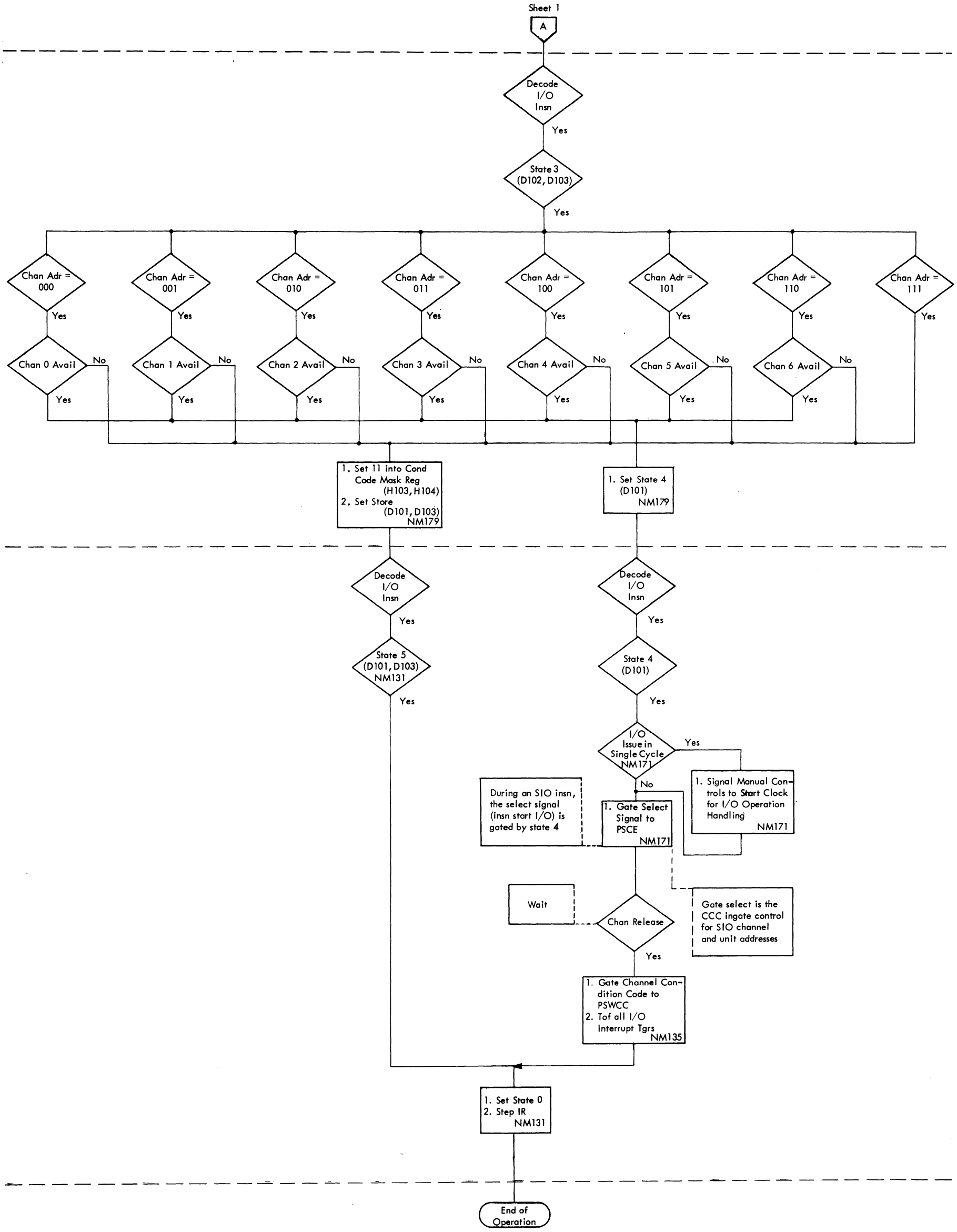


DIAGRAM 5-27. I/O SEQUENCE (SHEET 2 OF 2)

Objectives:

1. Decode and issue LM or STM instruction.
2. Check all interlocks that might detain or cancel the instruction.
3. LM - Fetch a doubleword; inform FXA of the FLB assigned for this fetch, the position of the data within the doubleword; and the GPR(s) that the data should fill.
4. STM - Set up a store for a doubleword; inform FXA of the SAR assigned for this store, the position of the data within the doubleword, and the GPR(s) to be used as source(s) for the data.

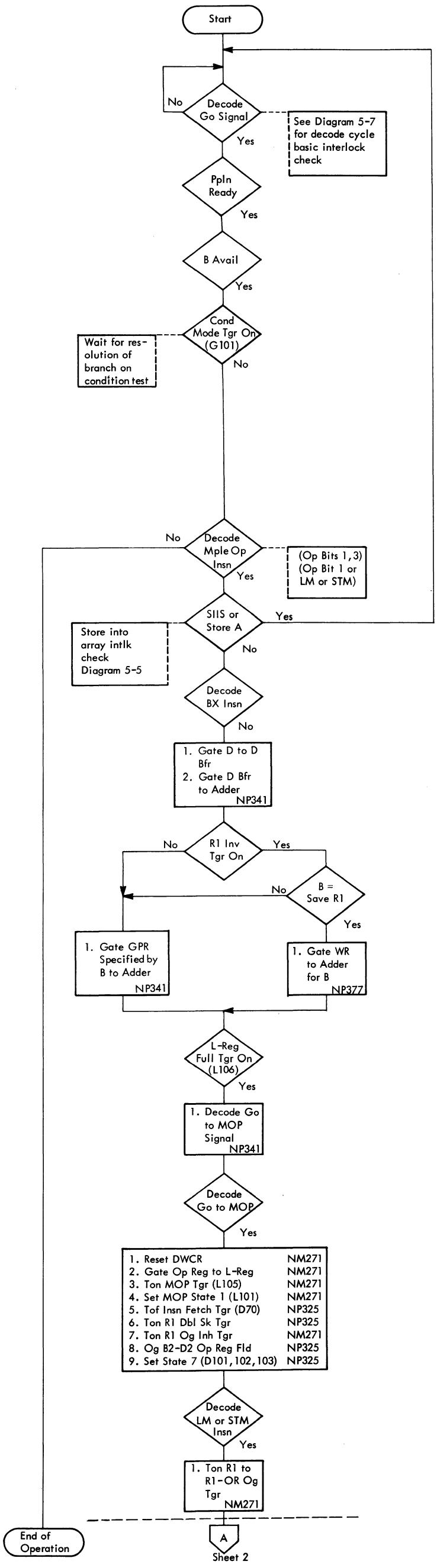


DIAGRAM 5-28. LM, STM SEQUENCE (SHEET 1 OF 3)

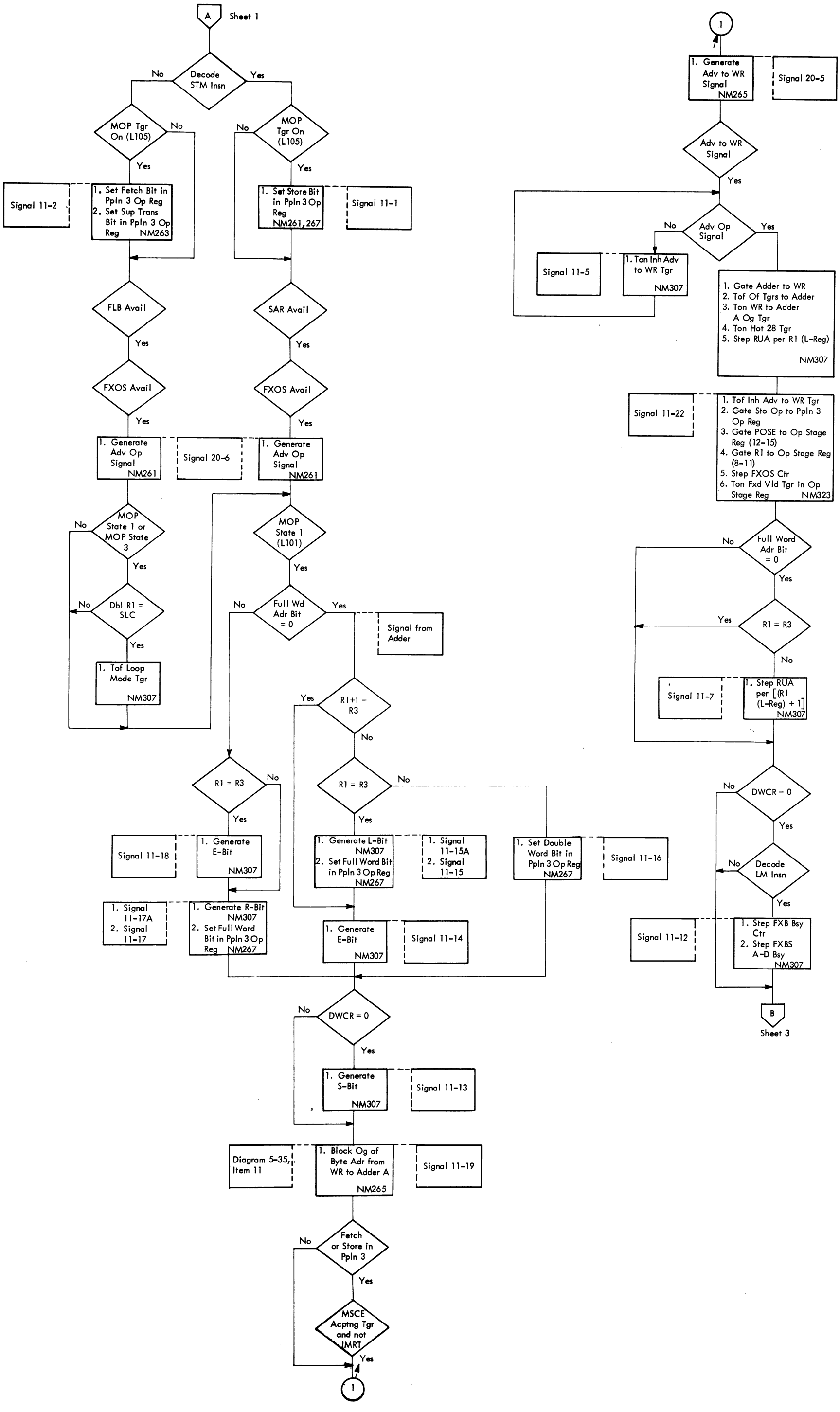


DIAGRAM 5-28. LM, STM SEQUENCE (SHEET 2 OF 3)

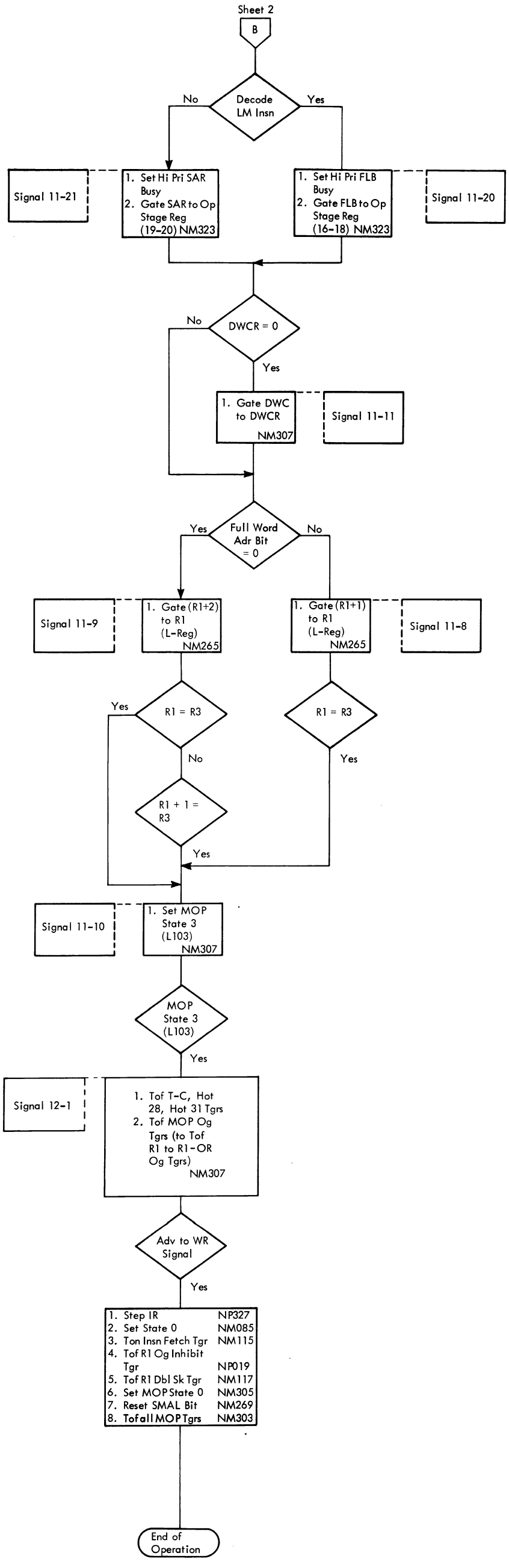


DIAGRAM 5-28. LM, STM SEQUENCE (SHEET 3 OF 3)

Objectives:

1. Decode and issue TR or TRT instruction.
2. Check all interlocks that might detain or cancel the instruction.
3. TR-Fetch and set up store for argument double word(s), and inform FXA of the assigned FLB(s) and SAR(s); suppress MSCE multi-accessing; fetch table word(s), and inform FXA of the assigned FLB(s).
4. TRT-Fetch argument double word(s), and inform FXA of the assigned FLB(s); fetch table word(s) and inform FXA of the assigned FLB(s); generate required argument byte address and save it in GPR 1.

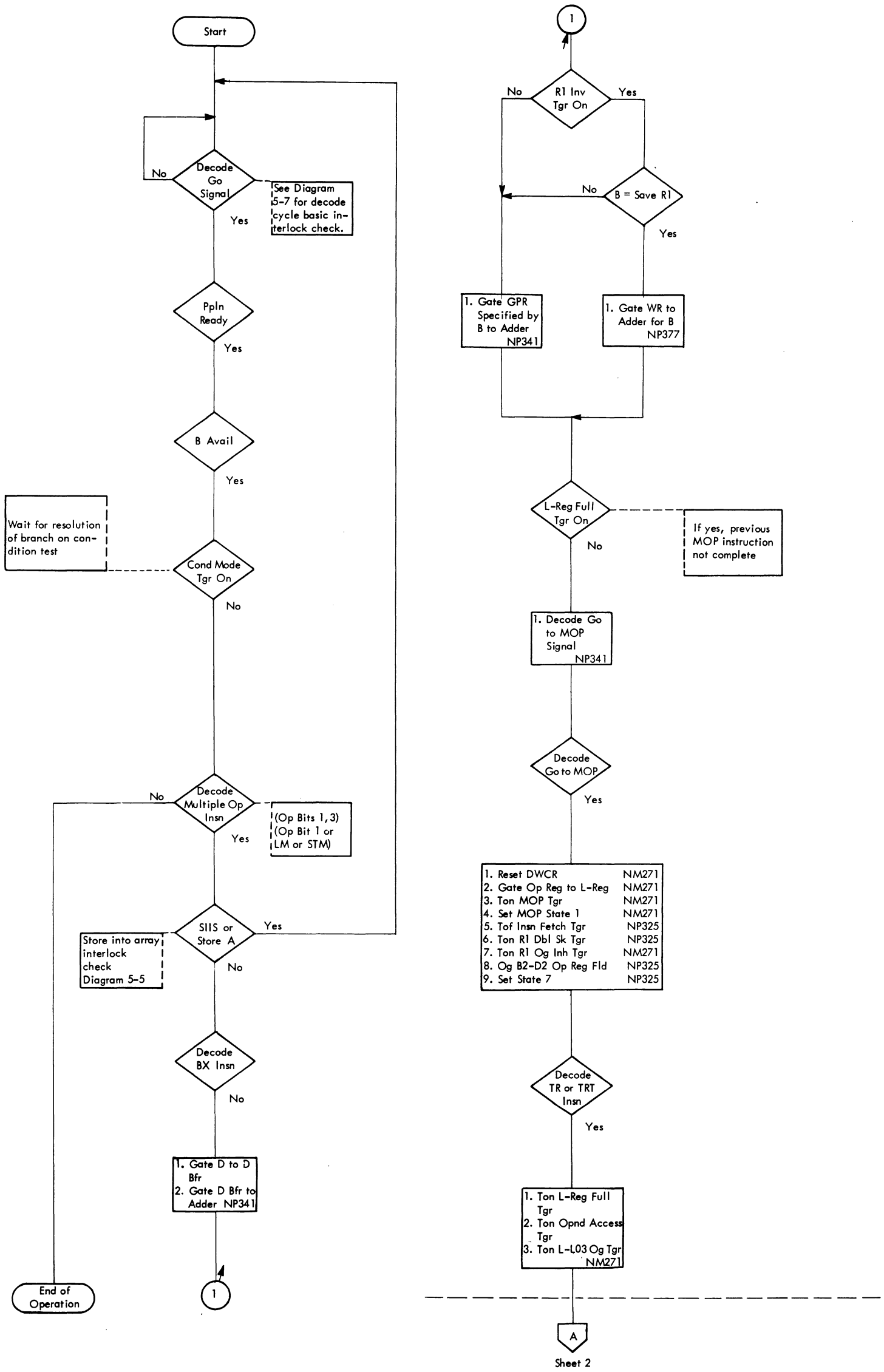


DIAGRAM 5-29. TR, TRT SEQUENCE (SHEET 1 OF 5)

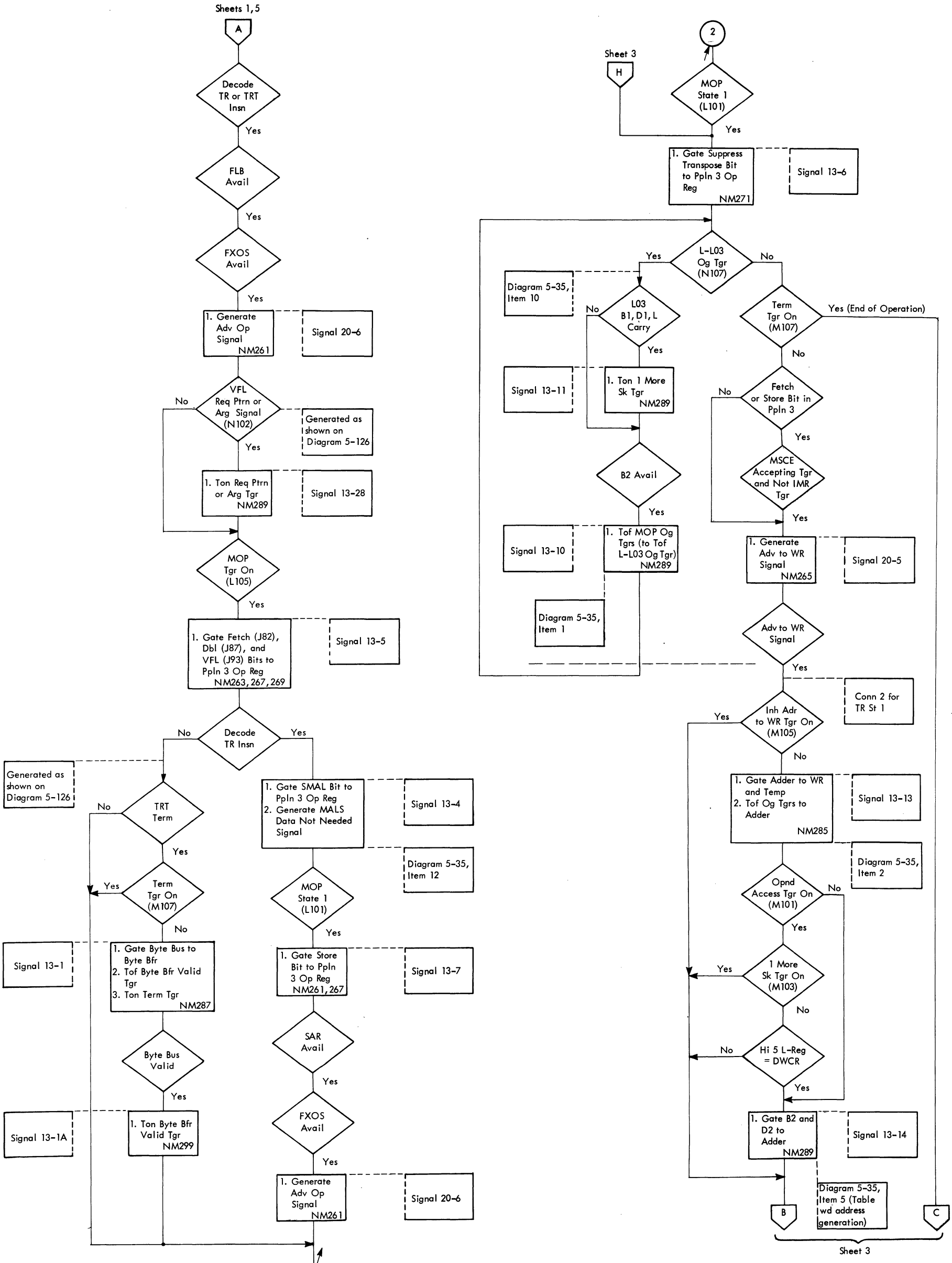


DIAGRAM 5-29. TR, TRT SEQUENCE (SHEET 2 OF 5)

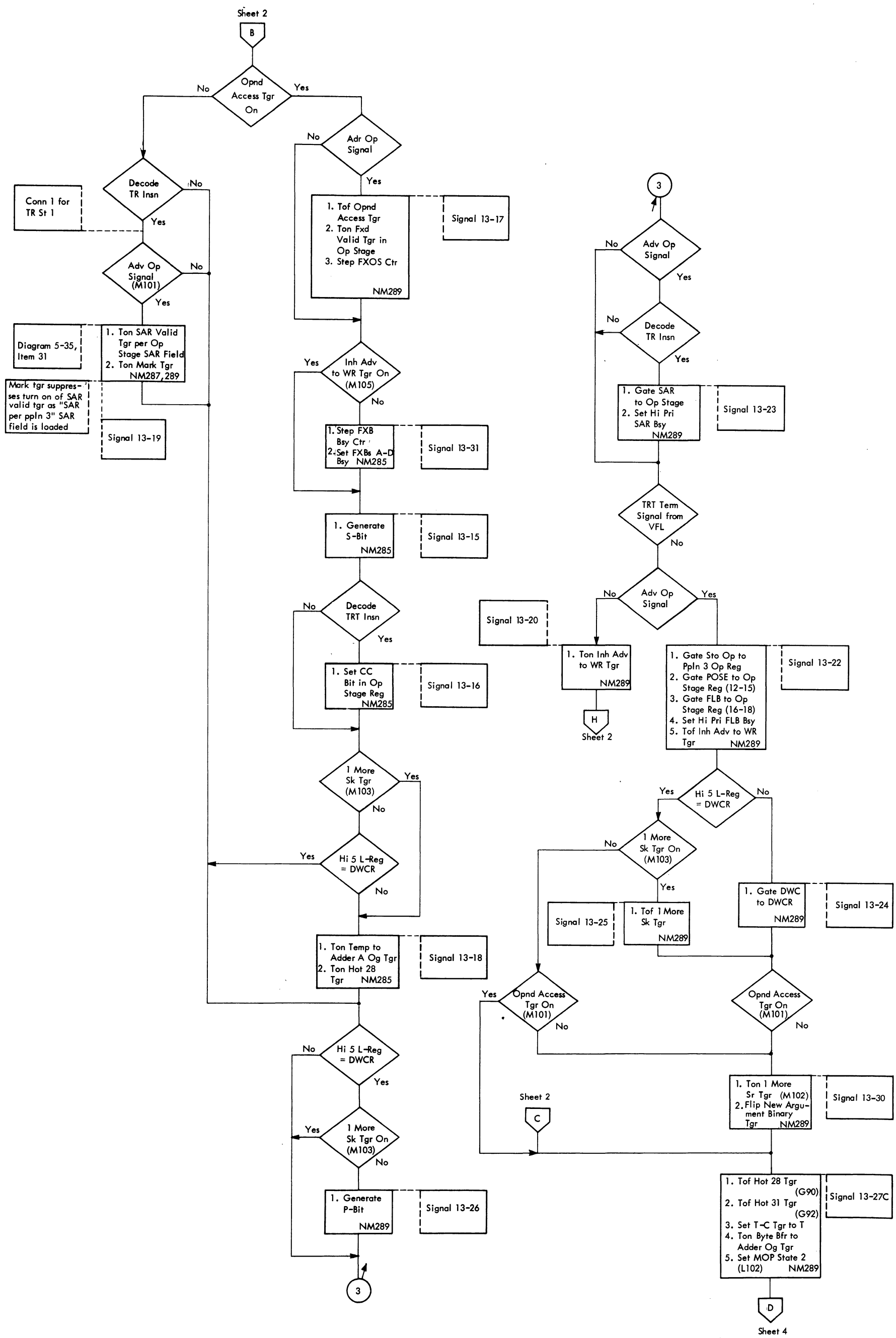


DIAGRAM 5-29. TR, TRT SEQUENCE (SHEET 3 OF 5)

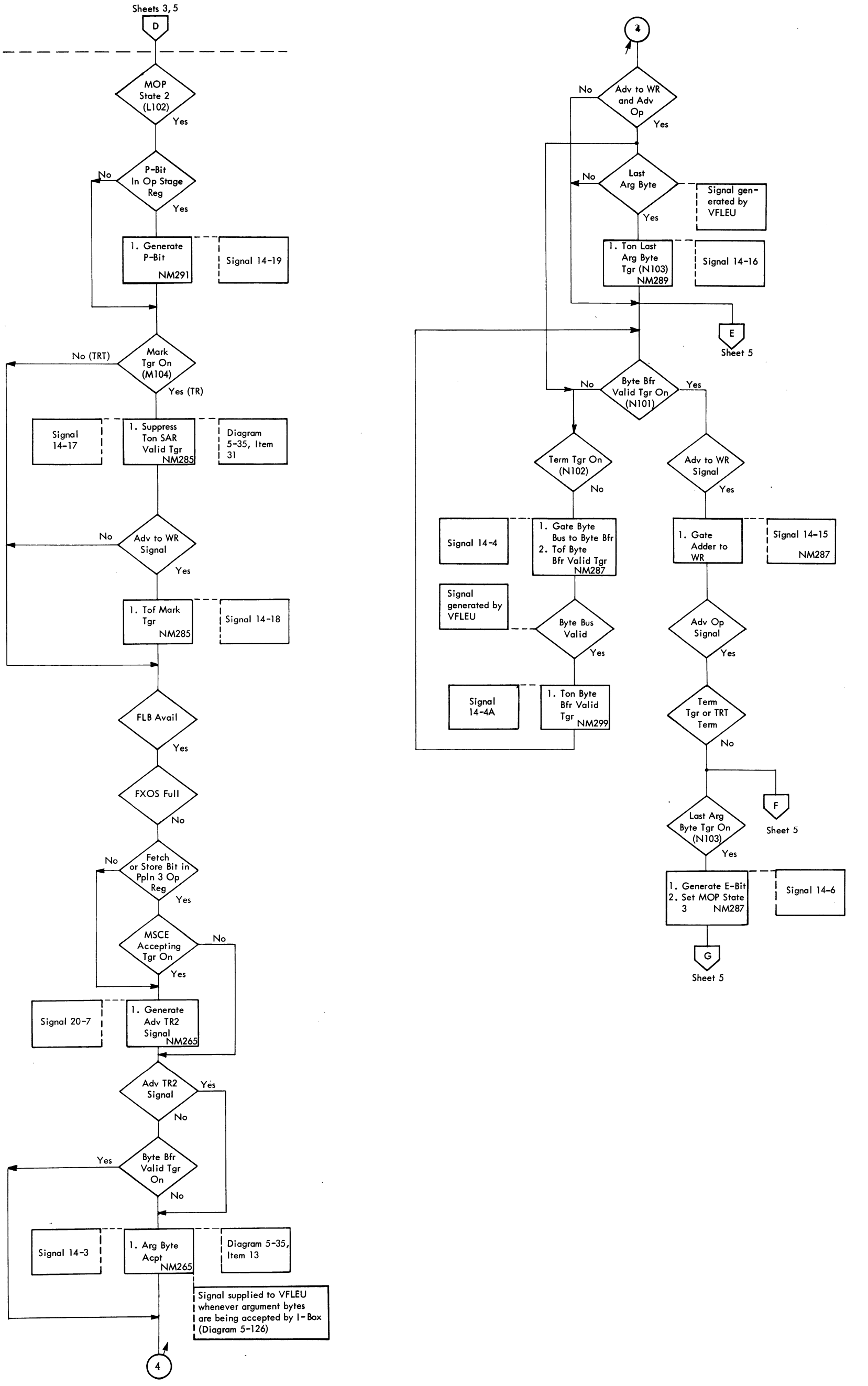


DIAGRAM 5-29. TR, TRT SEQUENCE (SHEET 4 OF 5)

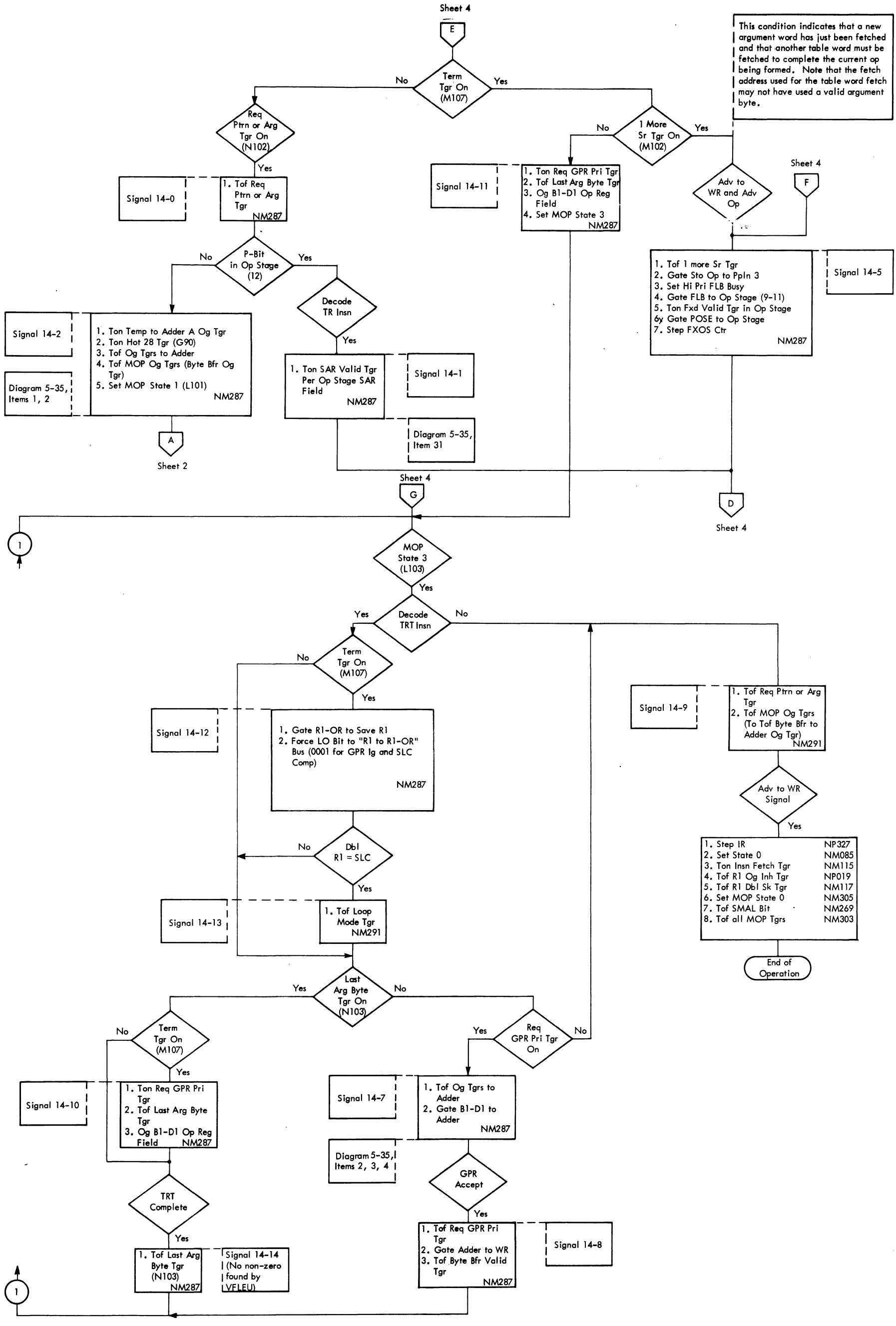
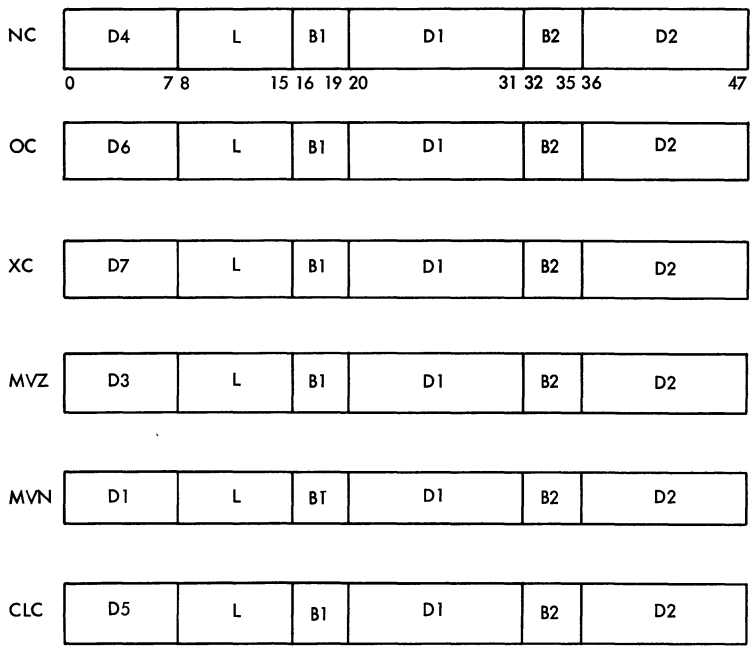


DIAGRAM 5-29. TR, TRT SEQUENCE (SHEET 5 OF 5)

Objective:

Decode and issue the following instructions:

NC, OC, XC, MVZ, MVN, and CLC Instructions (NOXCM Class)



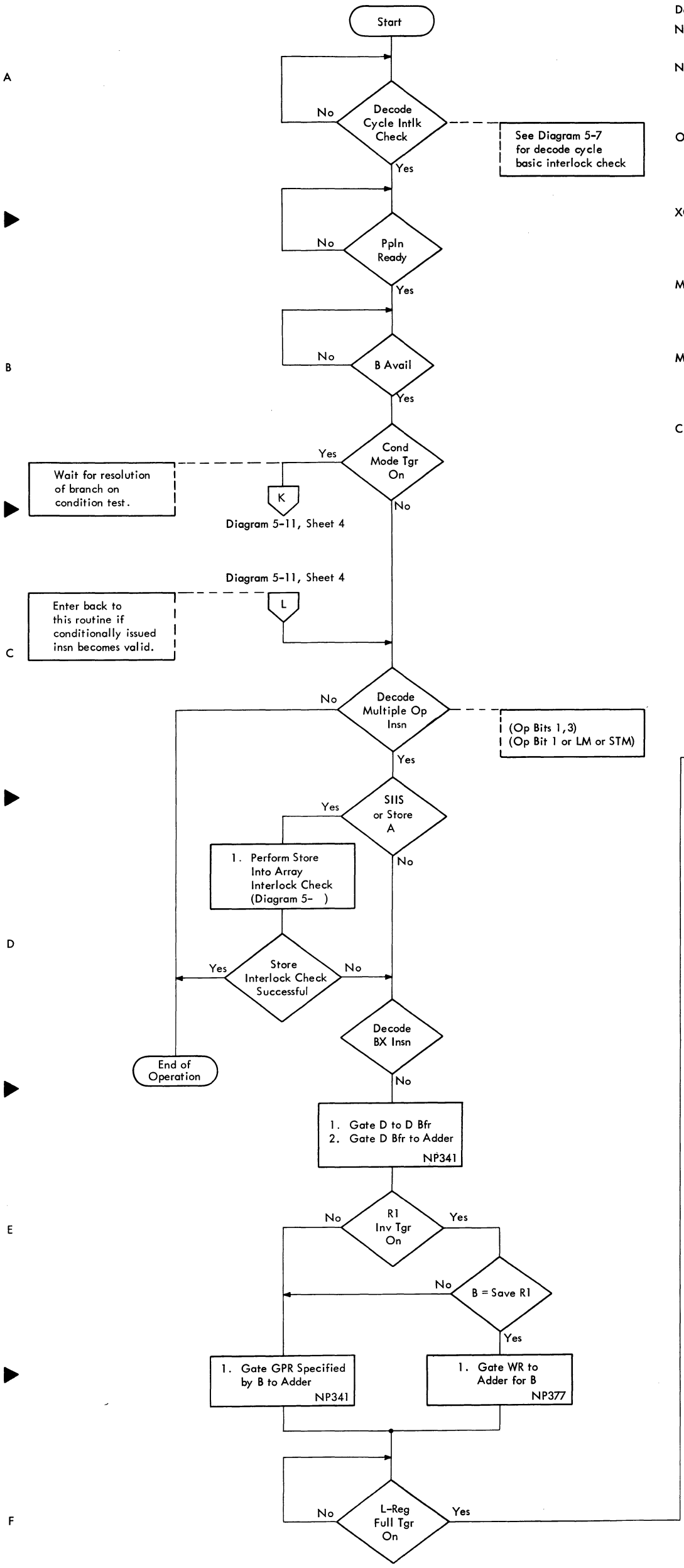


DIAGRAM 5-30. NOXCM SEQUENCE (SHEET 1 OF 3)

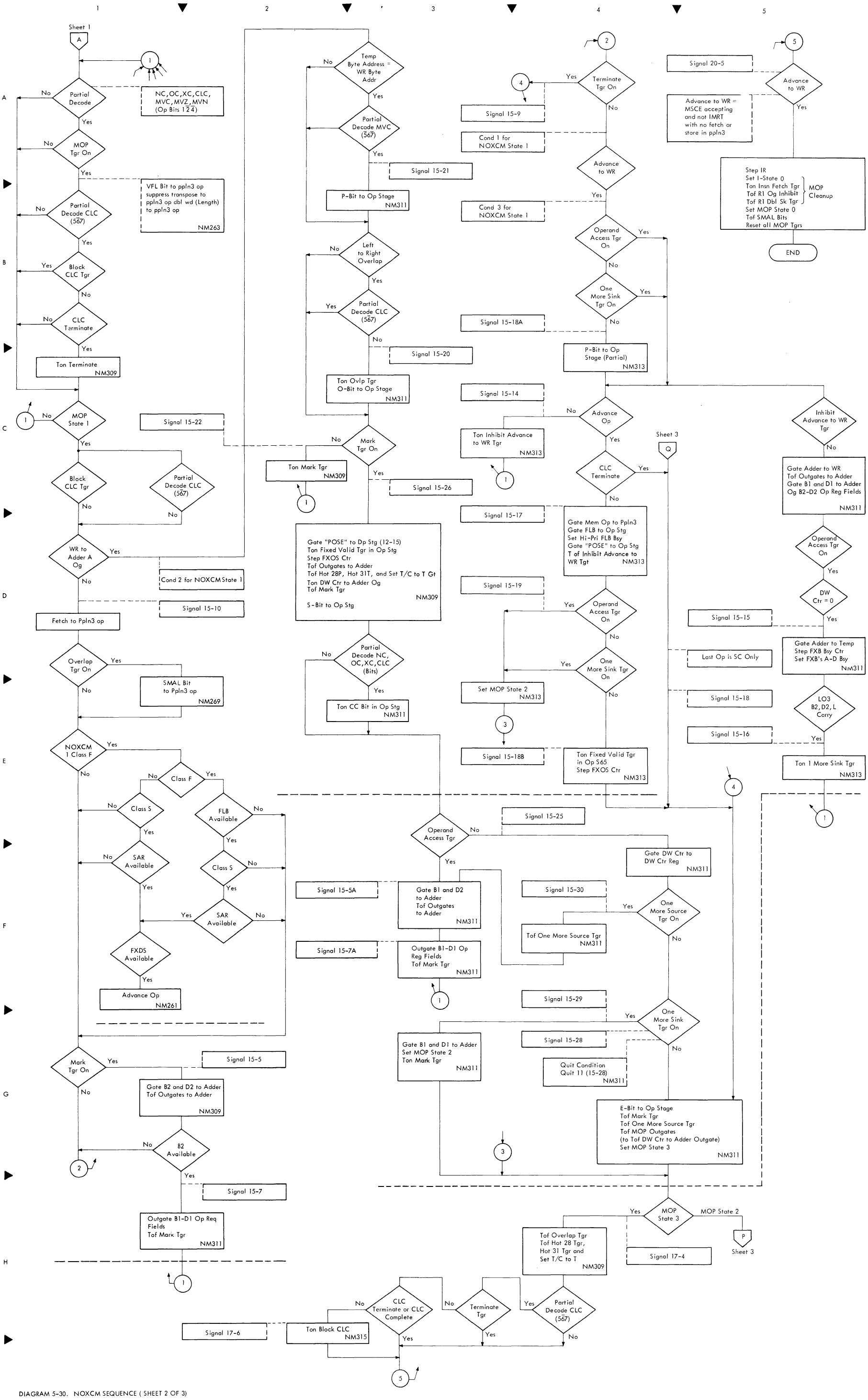


DIAGRAM 5-30. NOXCM SEQUENCE (SHEET 2 OF 3)

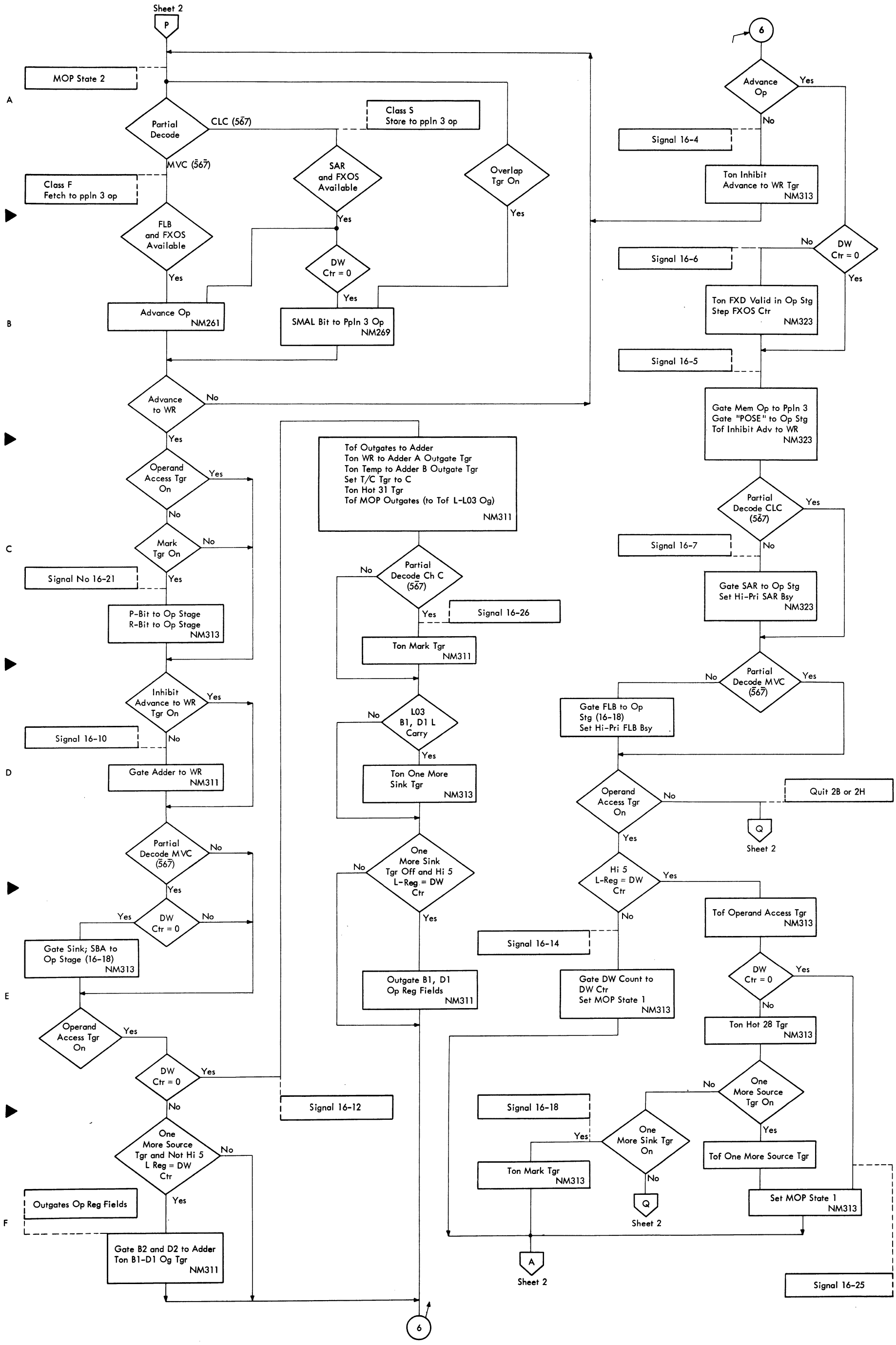
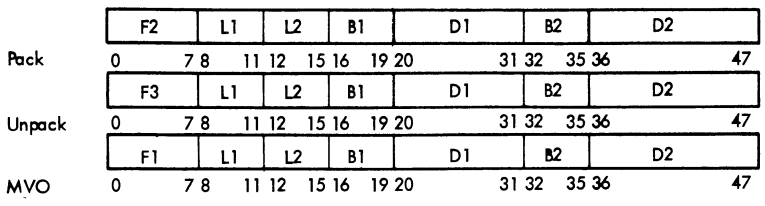


DIAGRAM 5-30. NOXCM SEQUENCE (SHEET 3 OF 3)

Objective:

Decode and issue the following instructions: Pack, Unpack, and Move With Offset



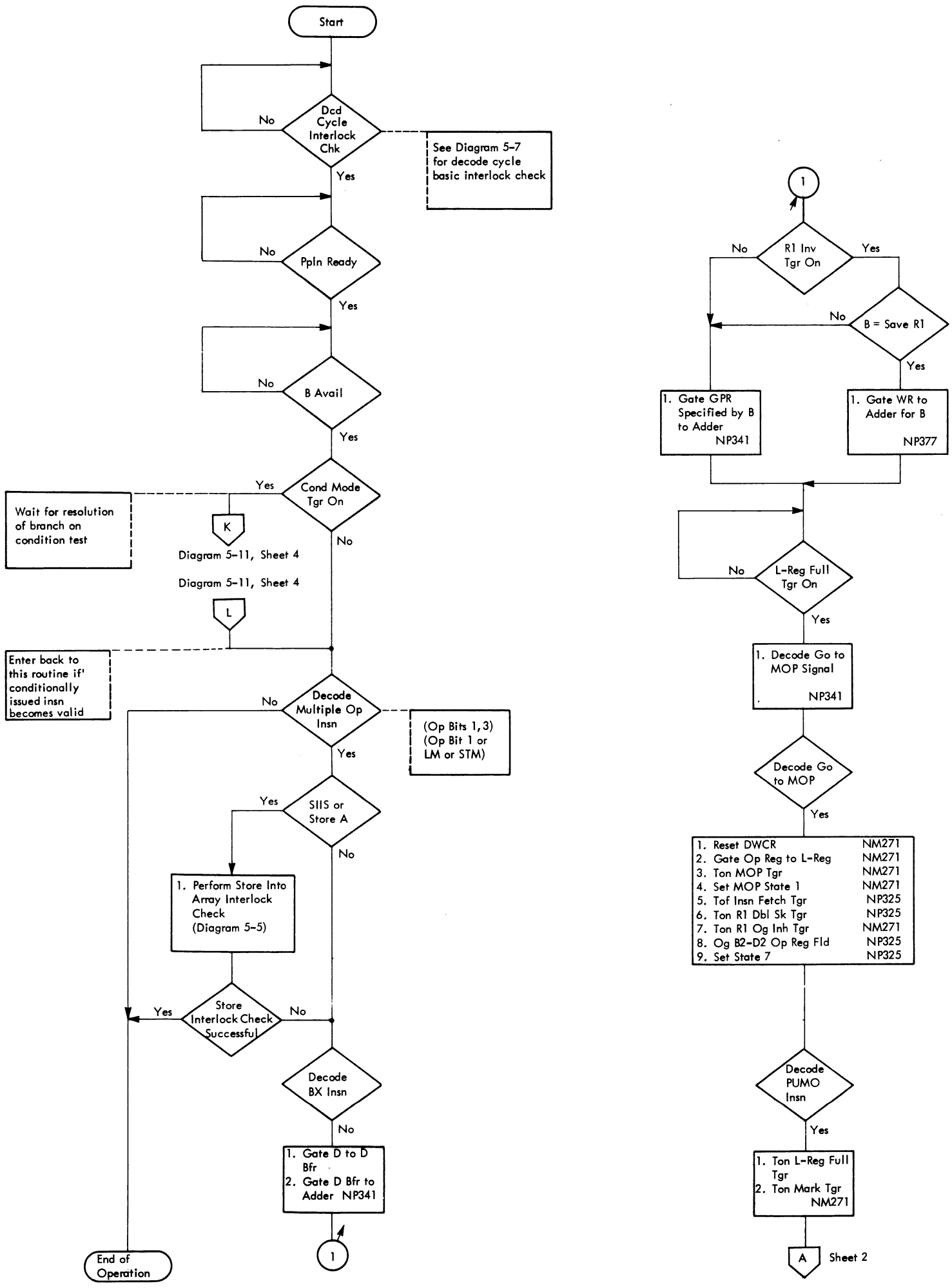


DIAGRAM 5-31. PUMO SEQUENCE (SHEET 1 OF 3)

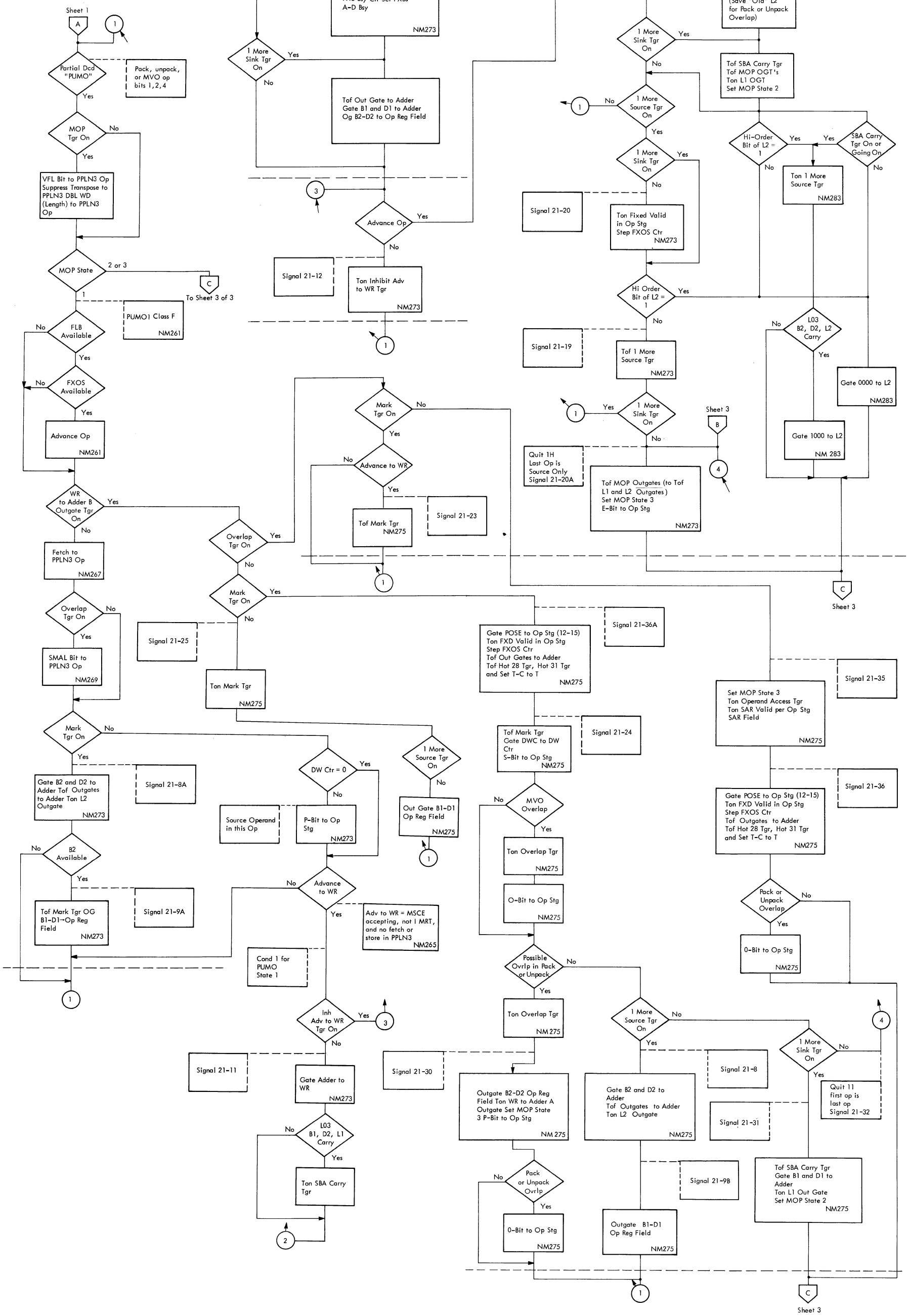


DIAGRAM 5-31. PUMO SEQUENCE (SHEET 2 OF 3)

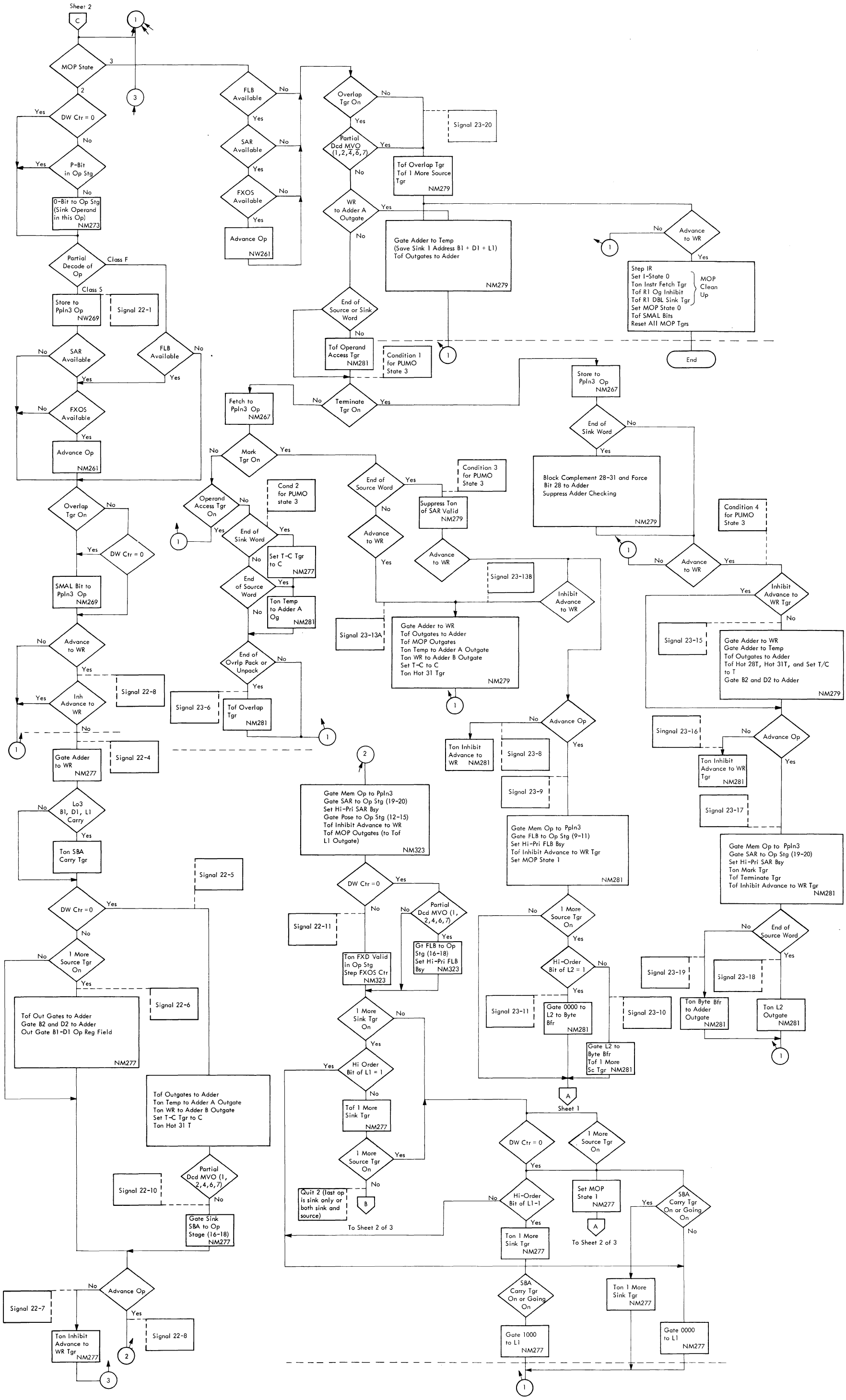
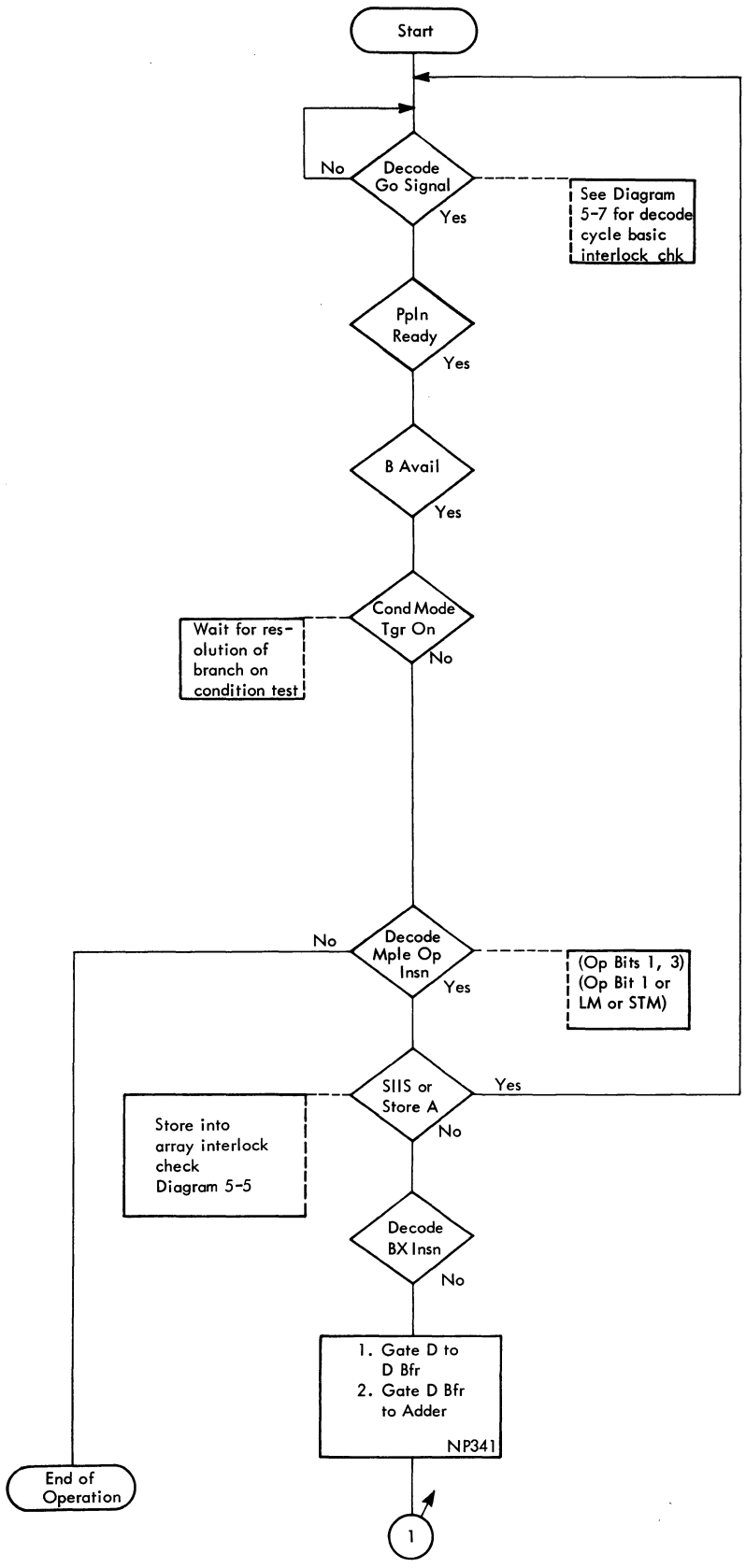


DIAGRAM 5-31. PUMO SEQUENCE (SHEET 3 OF 3)

Objectives:

1. Decode and issue an ED or EDMK instruction.
2. Check all interlocks that may detain or cancel the instruction.
3. Initially fetch two pattern and source words for VFLEU.
4. After first pattern (or source) word is requested by VFLEU, a subsequent reference for a pattern (or source) word is made each time the VFLEU finishes processing a pattern (or source) word.
5. VFLEU steps off an Op whenever it requests a pattern word, and the I-Box issues an op whenever it references a pattern word.
6. EDMK - Record byte address for first significant result digit by forming byte address and causing this byte address to be stored in GPR1.



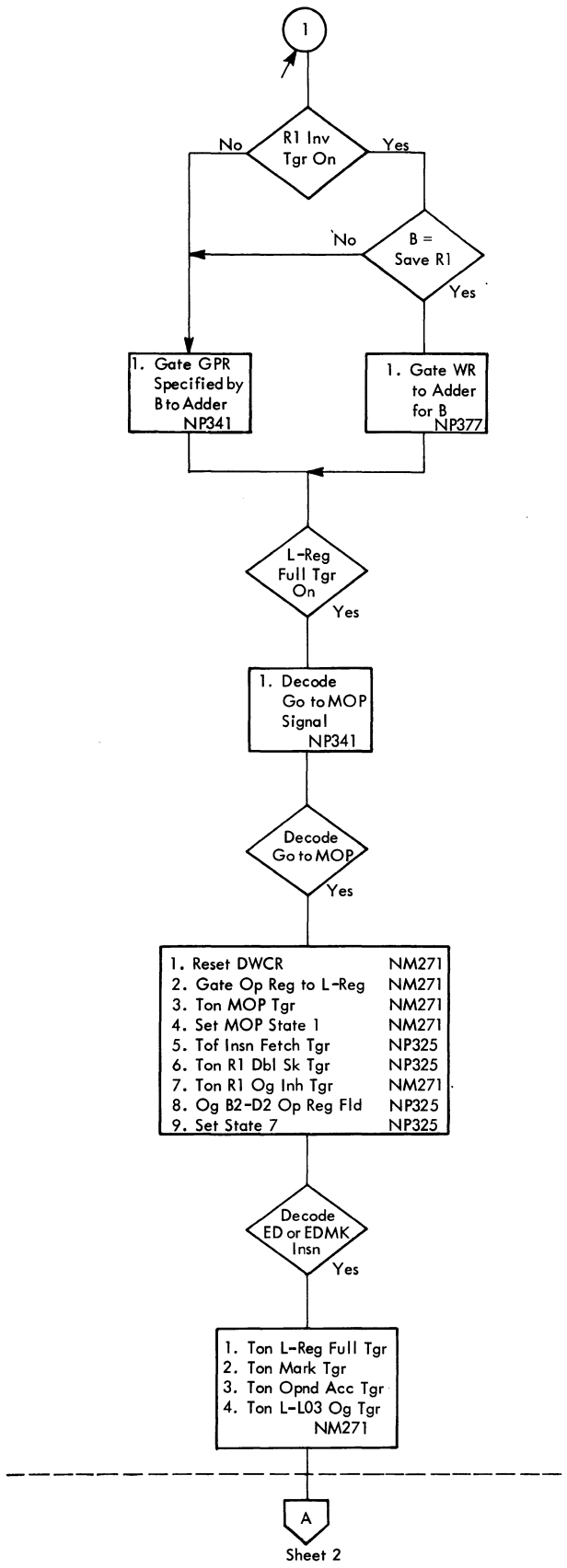


DIAGRAM 5-32. ED, EDMK SEQUENCE (SHEET 1 OF 4)

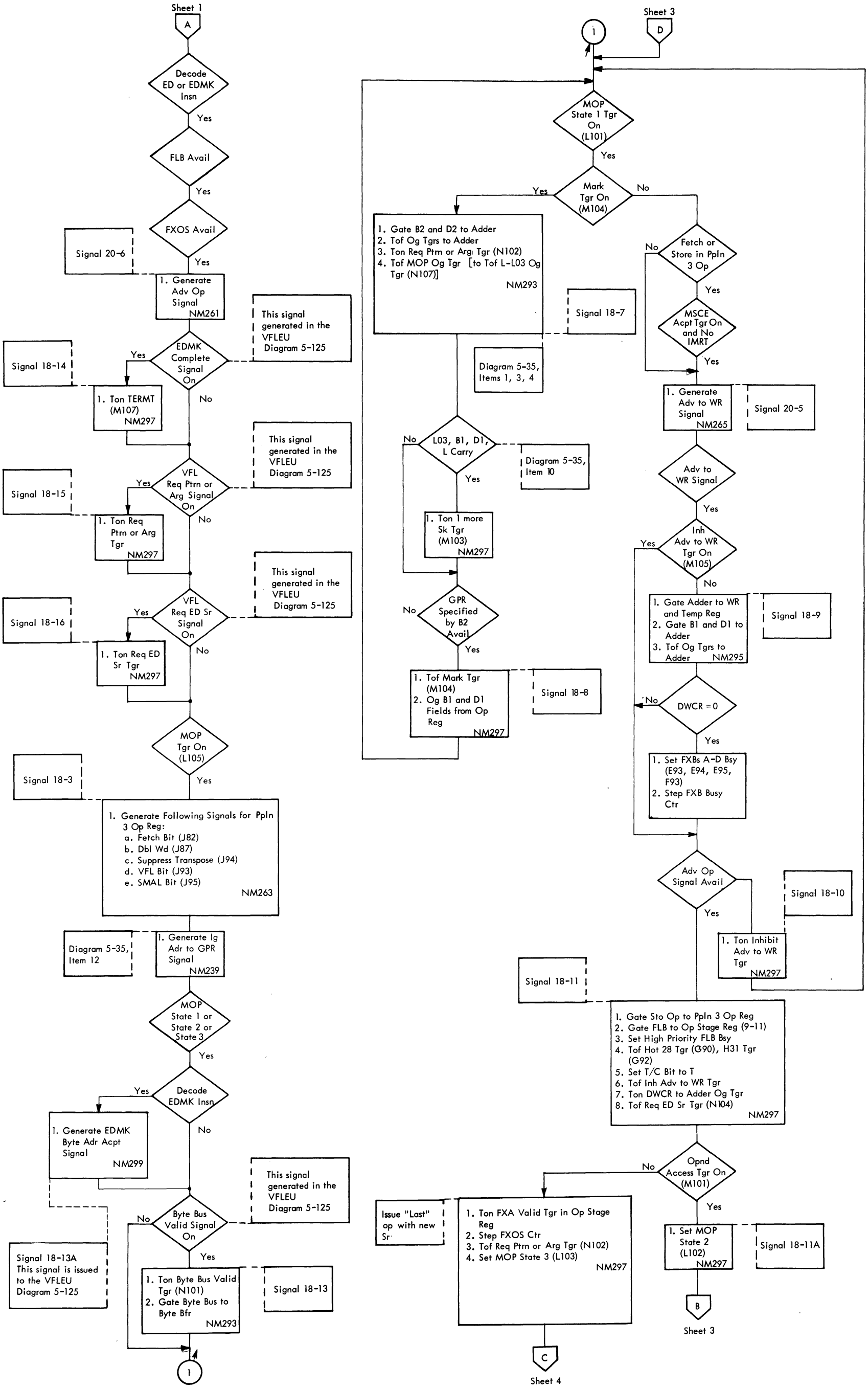


DIAGRAM 5-32. ED, EDMK SEQUENCE (SHEET 2 OF 4)

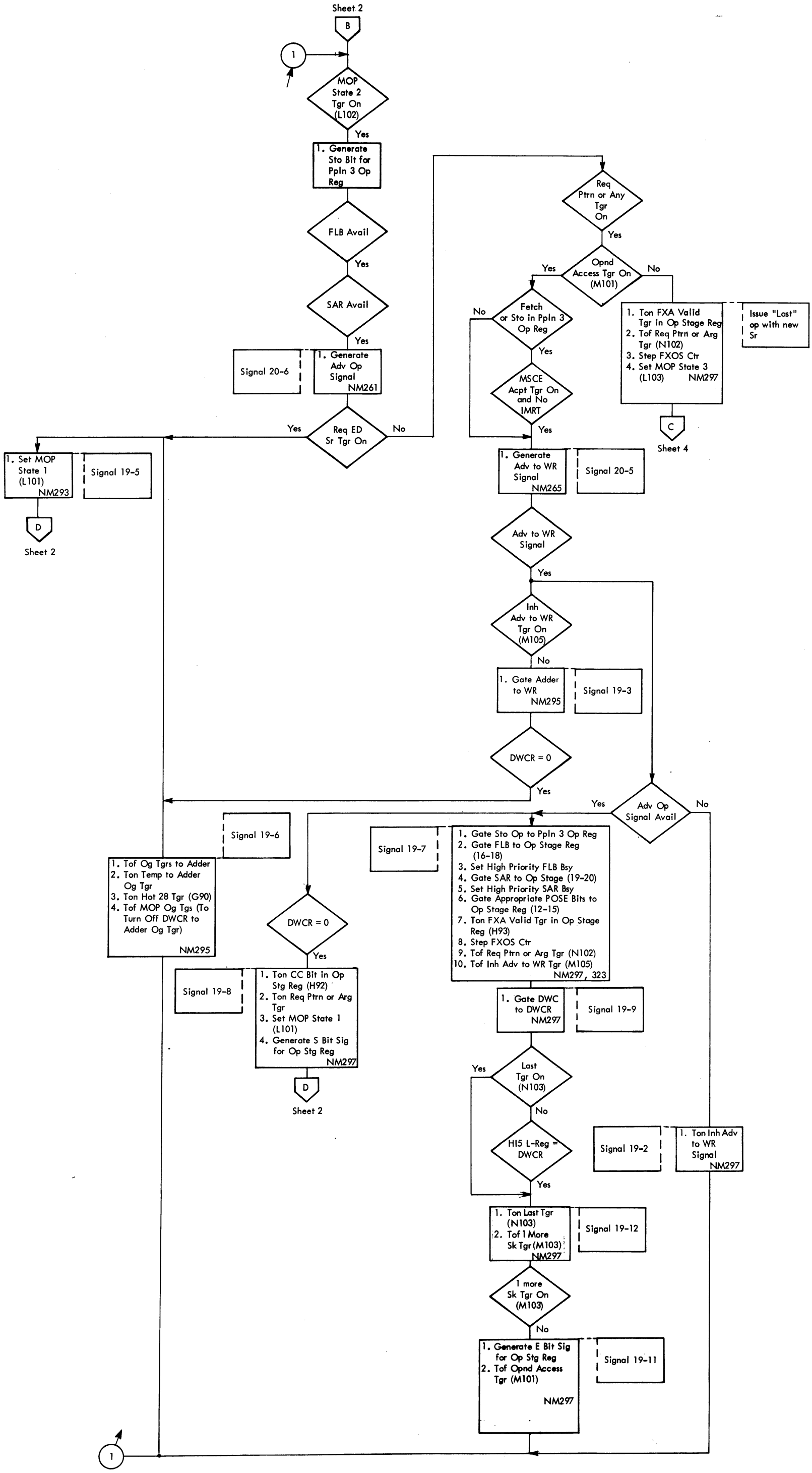


DIAGRAM 5-32. ED, EDMK SEQUENCE (SHEET 3 OF 4)

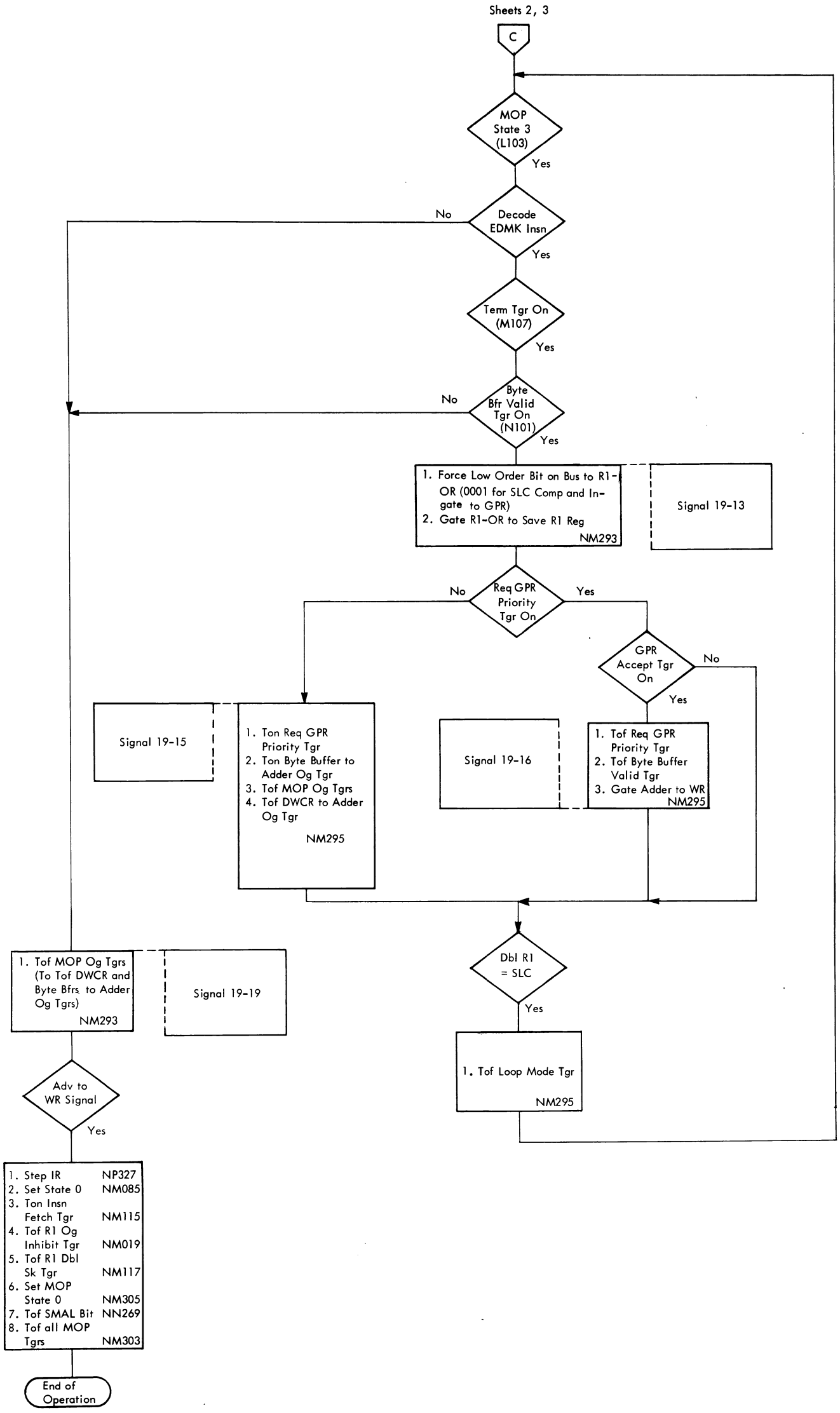


DIAGRAM 5-32. ED, EDMK SEQUENCE (SHEET 4 OF 4)

Precise Interrupt Signals

Objectives:

1. Generate the various interrupt signals

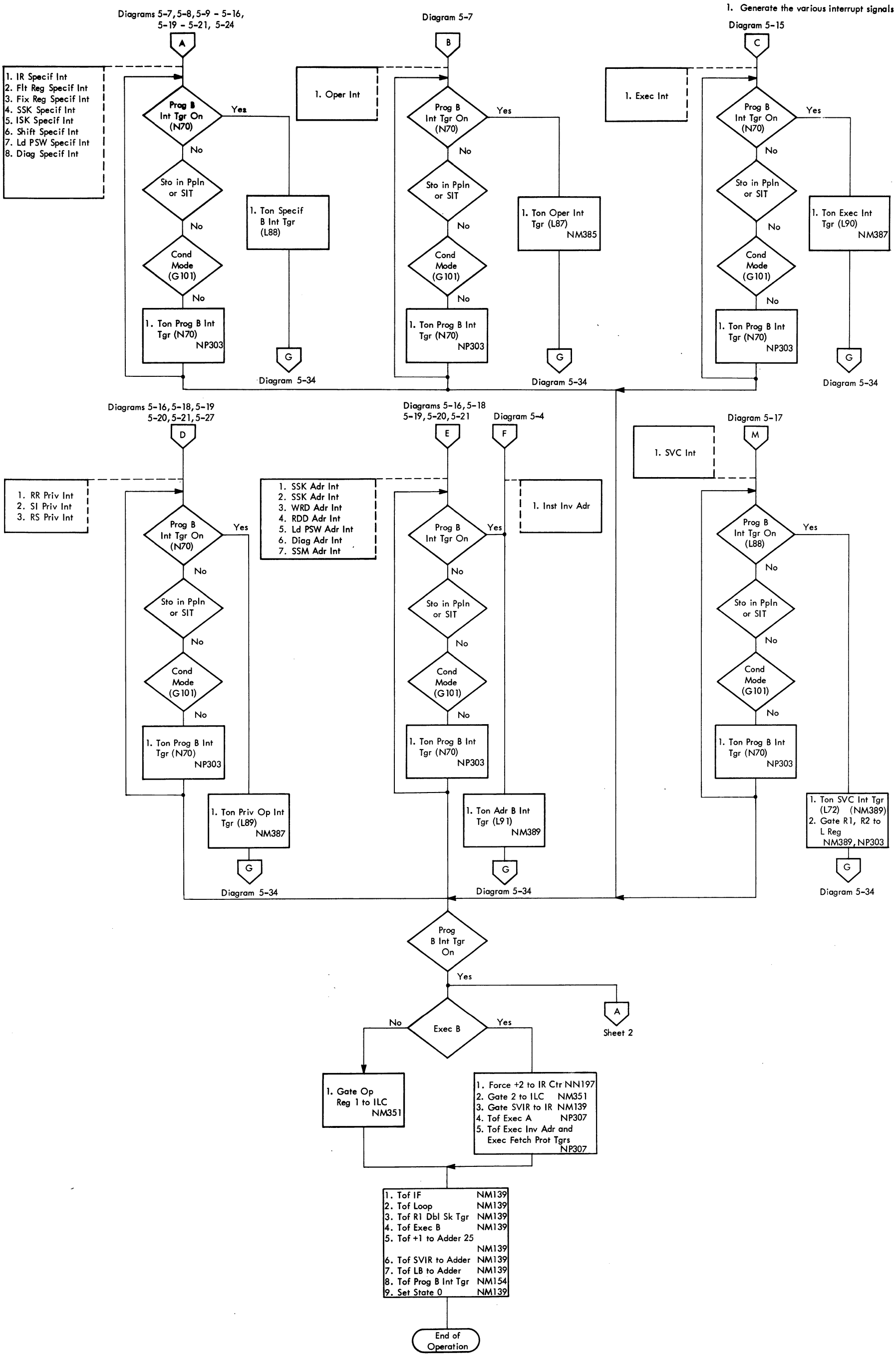
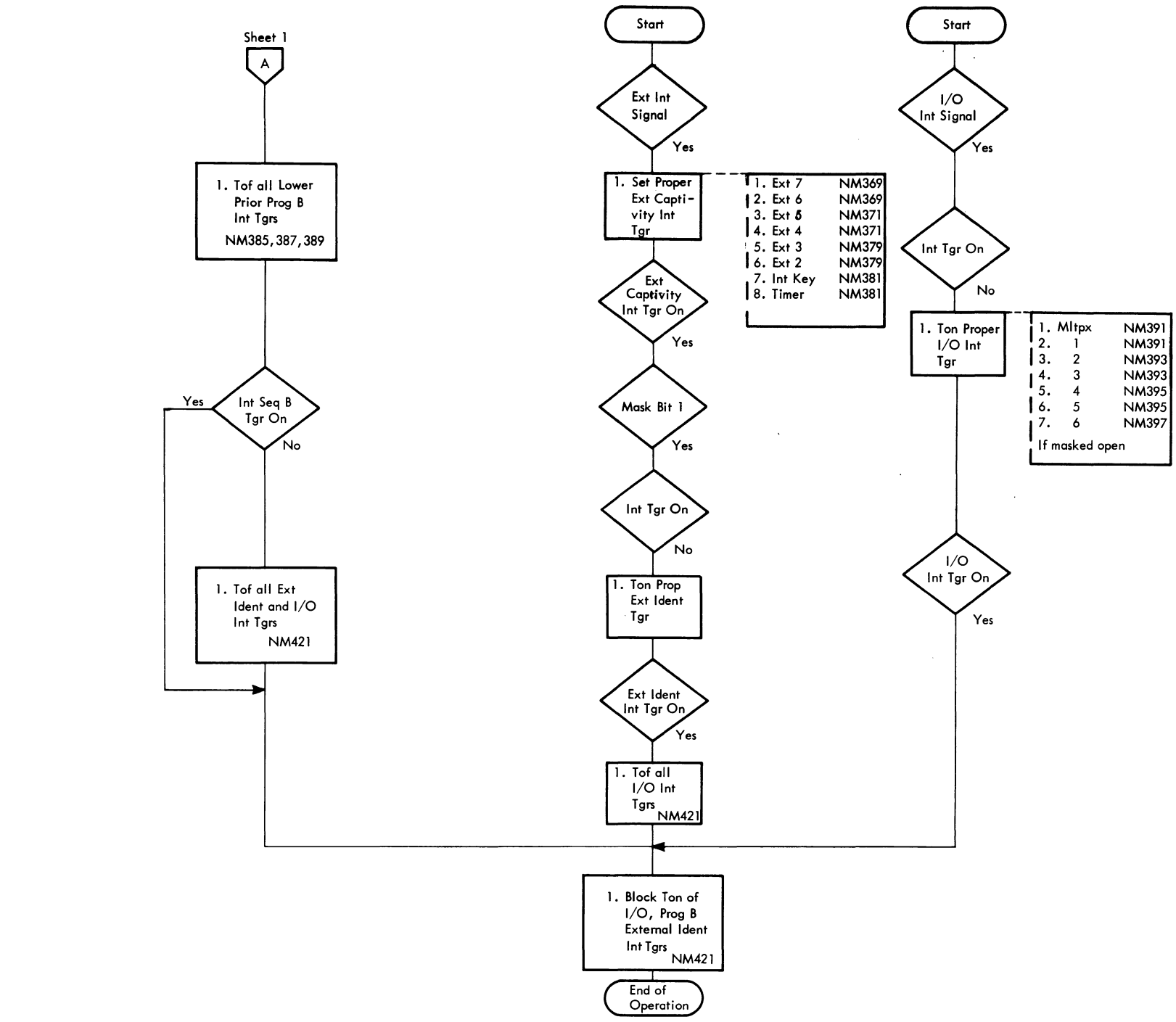


DIAGRAM 5-33. INTERRUPT SIGNALS (SHEET 1 OF 3)

Program B, External, and I/O Interrupt Signaling and Priority



Machine Check and Imprecise Interrupt Signals

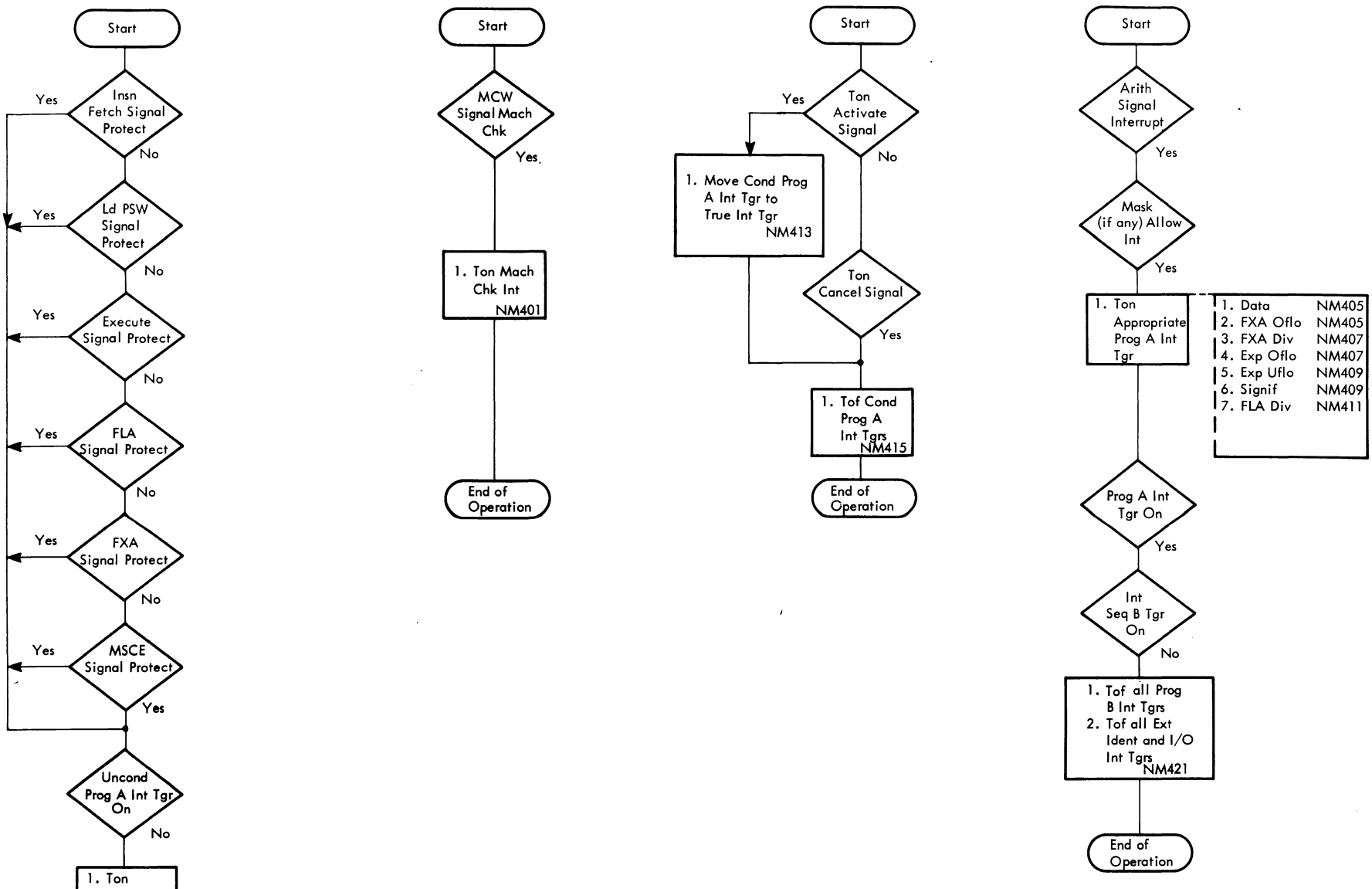
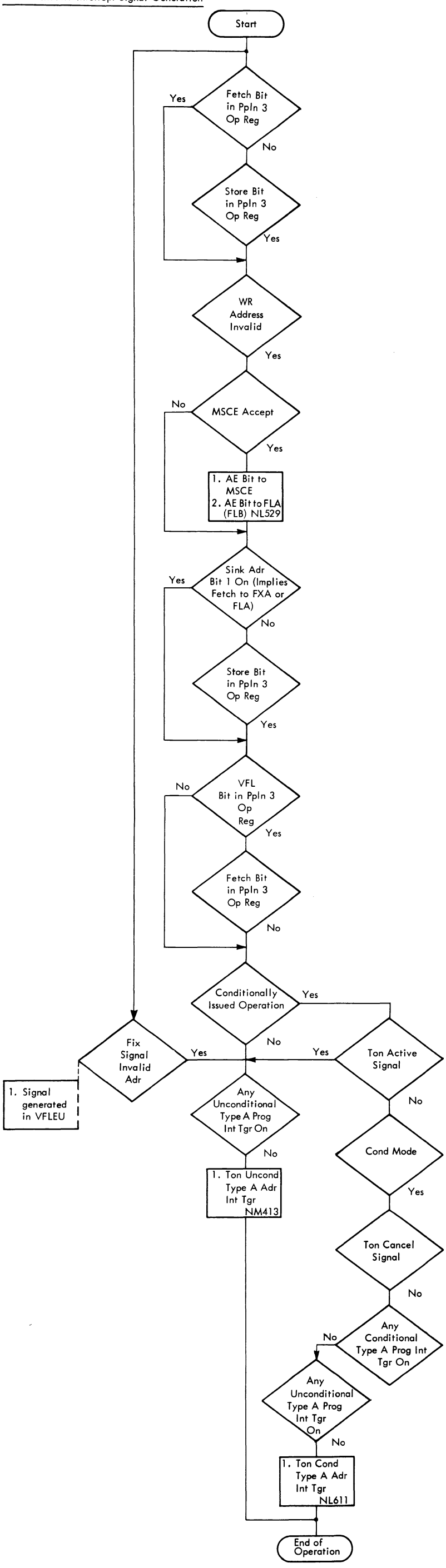


DIAGRAM 5-33. INTERRUPT SIGNALS (SHEET 2 OF 3)

Address Invalid Interrupt Signal Generation



Imprecise Address Specification Interrupt Signal Generation

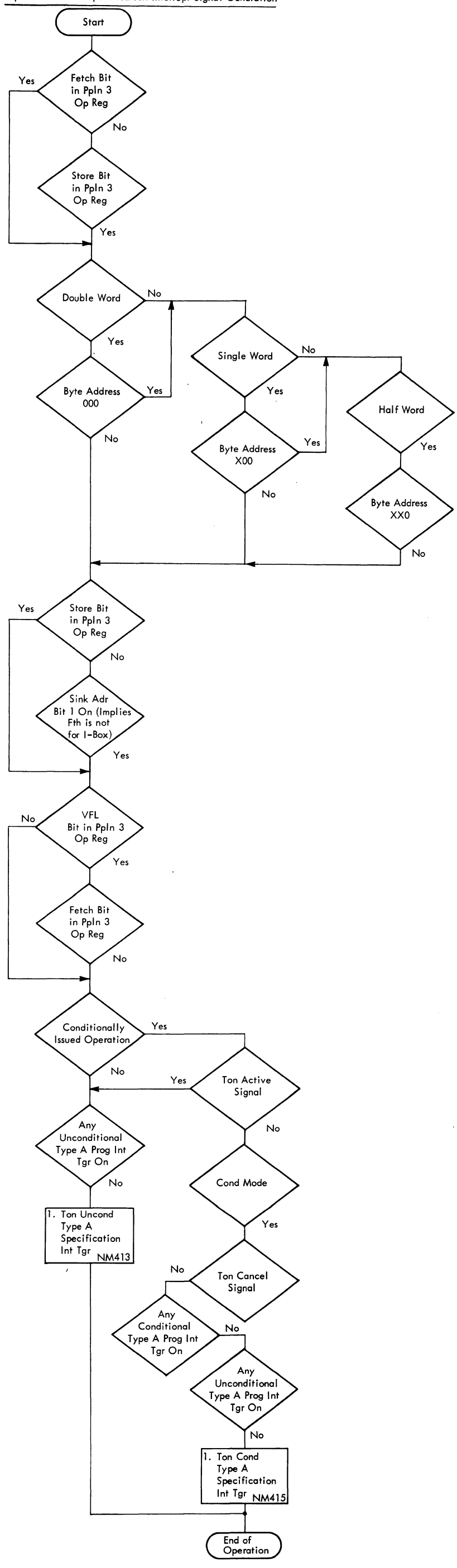


DIAGRAM 5-33. INTERRUPT SIGNALS (SHEET 3 OF 3)

Diagram 5-33

Objectives:

1. Process all interruptions defined for the IBM System/360.
2. General steps during interruption sequencing are as follows.
   a. Stop I-Box decode
   b. Wait for I-Box to complete current instruction
   c. Fetch new PSW
   d. Initialize instruction fetch mechanism with instruction address in new PSW
   e. Wait for execution pipeline to empty.
   f. Store old PSW
   g. Return I-Box to decode
   h. Store IC of instruction (N)if interrupt is imprecise (ILC = 00)
   i. Store IC of instruction (N + 1) if interrupt is precise (ILC = True ILC)

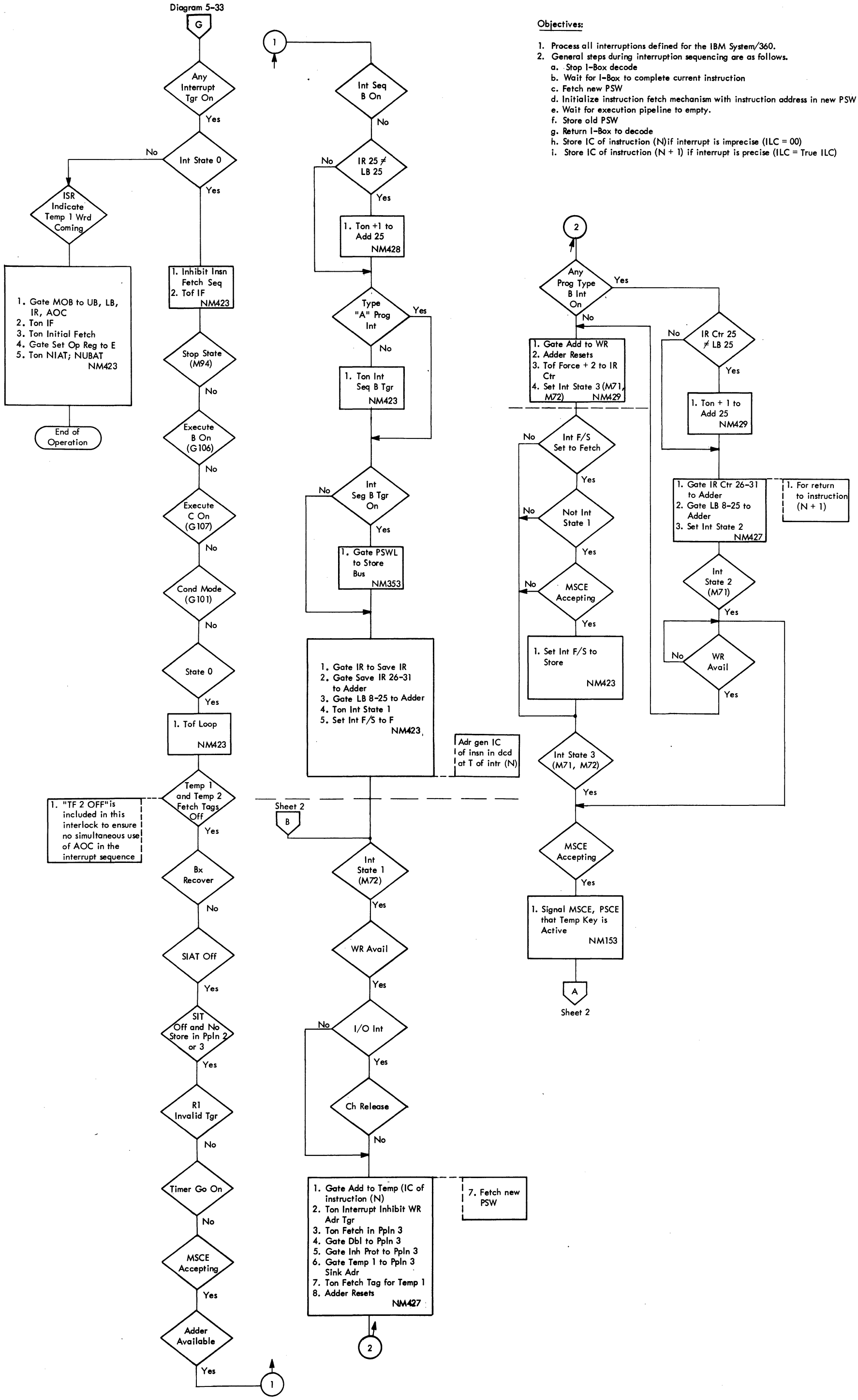


DIAGRAM 5-34. INTERRUPT SEQUENCING (SHEET 1 OF 2)

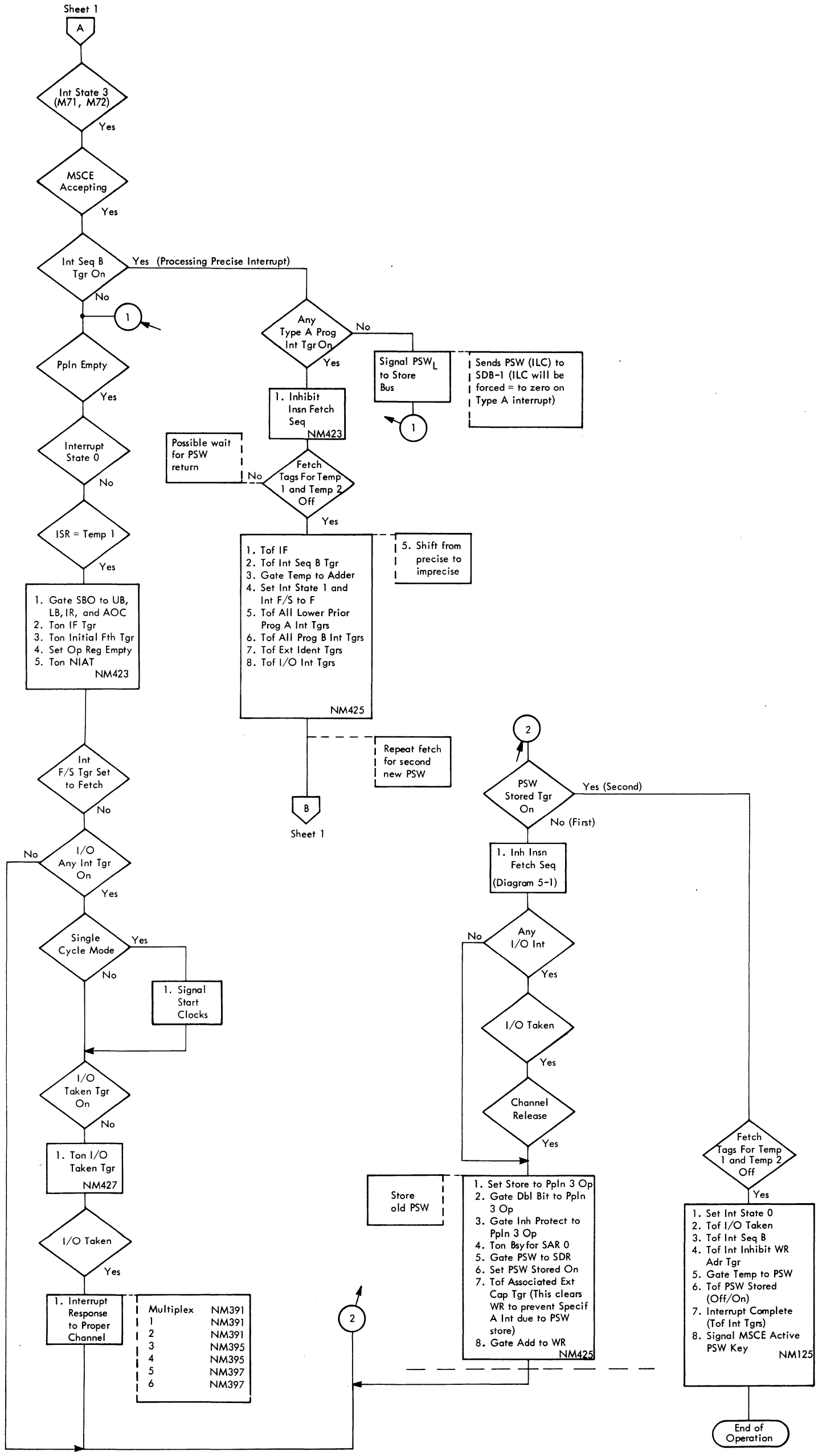


DIAGRAM 5-34. INTERRUPT SEQUENCING (SHEET 2 OF 2)

1. Tof MOP Og Tgrs - This line turns off the following outgate triggers: DWCR; Byte Buffer; R1 to R1 - OR; L1; L2; and L-L03.

2. Tof Og Tgrs to Adder - This line turns off the following outgate triggers: all "B" and "X" outgate triggers; D-Bfr; Temp to Adder A; Temp to Adder B; WR to Adder A; WR to Adder B.

3. Gate B1 (B2) to Adder - This line turns on the B outgate trigger which is specified by a decode of the B-field outgated from the Instruction Register.

4. Gate D1 (D2) to Adder - This line turns on the D-Bfr outgate trigger and ingates to the D-Bfr the D-Field which is outgated from the Instruction Register.

5. Gate B1 (B2) and D1 (D2) to Adder - This line combines the functions of the lines described in items 3 and 4.

6. Lf to Rt Olap - This signal is used in the NOXCM sequence to indicate an overlapping condition between the sink and source operand fields which requires special handling by the VFLEU. This line is conditioned by: $0 <$ [Sink Starting Address (to byte level) - Source Starting Address (to byte level)] < 8; i.e., $0 <$ [(B1 + D1) - (B2 + D2)] < 8.

7. MVO Olap - This signal is used in the PUMO sequence to indicate an overlapping condition between the sink and source operand fields which requires special handling by the VFLEU. This line is conditioned by: $0 < D < 8$, where $D = [(B2 + D2 + L2) - (B1 + D1 + L1)]$ = (Source Starting Address - Sink Starting Address).

8. Possible Olap in PACK or UNPK - This signal is used in the PUMO sequence to indicate an overlapping condition between the sink and source operand fields which may require special handling by the VFLEU. (See also item 9.) This line is conditioned by the following logical relationship: (Decode UNPK) $(-8 \leq D \leq 0)$ or $(0 < D < 16)$. (D is defined in item 7.) This condition will prompt the PUMO sequence to go into a special "hand-in-hand" relationship with the VFLEU.

9. PACK or UNPK Olap - This signal is used in the PUMO sequence to indicate that a particular source doubleword has the same address as a particular sink doubleword and that special handling of this pair of operands is required in the VFLEU. This line is conditioned when: $(-8 \leq D < 8)$ (Sink Dbl Wd Adr Bit, Bit 28) = (Source Dbl Wd Adr Bit, Bit 28); i.e., Sink Double Word Address = Source Double Wd Address. (D is defined in item 7.)

10. LO3, B, D, L Carry - This signal is used as operand fetching is begun, to aid in determining how many doublewords are involved in a particular operand stream. A three-bit sum is determined by the addition of the low-order three bits of the register specified by B to the low-order three bits of the D field. This sum is then added to the low-order three bits of the L-field; a carry out of the high-order position of this three-bit add is known as an "LO3, B, D, L Carry".

11. Block Og of Byte Adr from WR to Adder A - When this line is conditioned, it suppresses the outgating of bits 29-31 from the WR which would otherwise take place when the WR to Adder A Og Tgr is on.

12. MALS Data Not Needed - This line signals the MSCE that the data for a fetch are not needed.

13. Arg Byte Accp - This signal is generated by the MOP sequence for TR and TRT. Its presence indicates to the VFLEU that any argument byte which is presently on the Byte Bus may be taken off. Note that this signal consists of two lines to the VFLEU - the signal is present only when both lines are negative.

14. Blk End Cancel - This line is conditioned by the MOP logic whenever the Term Tgr is on. It is used in connection with the op cancelling process. When the line is conditioned, it prevents the VFLEU from completing its cancelling process until the MOP sequence can guarantee that no more ops will be issued to the FXOS.

15. Byte Bus Valid - This signal from the VFLEU indicates to the TR or TRT sequence that a valid argument byte is on the Byte Bus, or to the ED or EDMK sequence that a byte count is on the Byte Bus.

16. CLC Complete - This signal is generated by the VFLEU when it finds that the operands in a CLC instruction are equal. The signal is used in the NOXCM sequence as part of the control for the Blk CLC Tgr.

17. CLC Term - This signal is generated by the VFLEU based on the fact that a pair of CLC operand bytes are unequal. If the unequal byte pair is compared during cycle n, this signal is transmitted to the I-Box during cycle (n+1). CLC Term is used in the NOXCM sequence to prevent further operand accessing and as part of the control for the Blk CLC Tgr.

18. EDMK Byte Adr Accp - This signal is generated by the MOP sequence for ED and EDMK. Its presence indicates that the byte count has been gated into the Byte Buffer.

19. EDMK Complete - This signal is generated by the VFLEU and is transmitted to the I-Box in parallel with the transmission of the last pattern word to an SDB. It causes the ED or EDMK sequence to turn on the Term Tgr which will then lead to an exit from the sequence. This communication relative to the completion of EDMK execution is necessary since the I-Box must stand-by until there is no further chance that a byte address may have to be generated and stored in GPR1.

20. End of Olap PACK or UNPK - This signal is generated by the VFLEU in connection with the special approach used for particular overlap situations in PACK and UNPK. It is transmitted to the I-Box in parallel with the transmission of the last sink word to an SDB and is used in the PUMO sequence to motivate an exit.

21. End of Sr Wd - This signal, generated by the VFLEU, is used only in certain overlapped PACK or UNPK situations. It is transmitted to the I-Box during the cycle following the outgate of the last byte from the source word and is used in the PUMO sequence to initiate the fetch for the next source word and the setting up of another store for the present sink word.

22. End of Sk Wd - This signal, generated by the VFLEU, is used only in certain overlapped PACK or UNPK situations. It is transmitted to the I-Box during the cycle following the ingate of the last byte into the sink word, and is used in the PUMO sequence to initiate a store for the next sink word.

23. GPR Accept - This signal is the output of the GPR Accept Trigger which is a part of the GPR ingate priority scheme in the FXA. If the GPR Accept Tgr is turned on at the start of cycle n, data from the I-Box are gated into an addressed GPR at the start of cycle (n+2).

24. Ig Adr to GPR - This line is conditioned by the MOP logic whenever the MOP Tgr is on to block the ingate to the addressed GPR of the high-order eight bits from the I-Box. This covers the byte-address-saving situations which arise in TRT and EDMK, and requires the high-order eight bits of GPR1 to remain unchanged.

25. Last Arg Byte - This signal accompanies the last argument byte to be sent from the VFLEU to the I-Box in the course of a TR or TRT instruction.

26. TRT Complete - This signal is generated by the VFLEU when there is no non-zero function byte in the course of a TRT instruction. It is transmitted to the I-Box during the cycle following the examination of the last function byte and causes an exit from the TR or TRT sequence.

27. TRT Term - This signal is generated by the VFLEU based on the fact that a non-zero function byte has been found. It is transmitted to the I-Box during the cycle following the examination of the non-zero function byte, and in parallel with the transmission to the I-Box of the byte count necessary for the generation of the culprit argument byte address. TRT Term is recorded by the Term Tgr, and motivates argument byte address generation and storing into GPR1. It then leads to an exit from the TR or TRT sequence.

28. VFLEU - This signal is sent to the I-Box from the FXA during the FXOS decode cycle for an SS instruction. Its presence will cause the L-Reg Full Tgr to be turned off.

29. VFL Req ED Sr - This signal is sent to the I-Box by the VFLEU whenever, in the course of an ED or EDMK, it makes a request for a source word from an FLB. VFL Req ED Sr is recorded by MOP in the Req ED Sr Tgr and motivates the fetch of the next source word.

30. VFL Req Ptrn or Arg Tgr - This signal is generated by the VFLEU in connection with pattern word requests in ED or EDMK and argument word requests in TR or TRT. The signal is recorded by Req Ptrn or Arg Tgr. The TR or TRT sequence is notified for all argument word requests except the first; and the Req Ptrn or Arg Tgr initiates the fetch of the next argument word. The ED or EDMK sequence receives this signal for all pattern word requests except the first (and in one case the end of the pattern word from the "END" Op); and the Req Ptrn or Arg Tgr initiates the fetch of the next pattern word.

31. Ton SAR Valid Tgr - Associated with each of the three SARs is a valid trigger. A SAR is set "Valid" when a new address is set into the SAR. Once the SAR is valid, its address is compared with all incoming addresses. Situations arise in the multiple op sequences for handling TR and PACK or UNPK in which an address may be set into a SAR before it is desirable to subject it to comparisons. In such cases it becomes necessary to modify the general procedure for setting a SAR valid. The logic used to accomplish this modification makes use of the signals: Suppress Ton SAR Valid Tgr, and Ton SAR Valid Tgr per Op Stage SAR Field. The flow chart below shows how these special signals interact with the general procedure for setting a SAR valid.

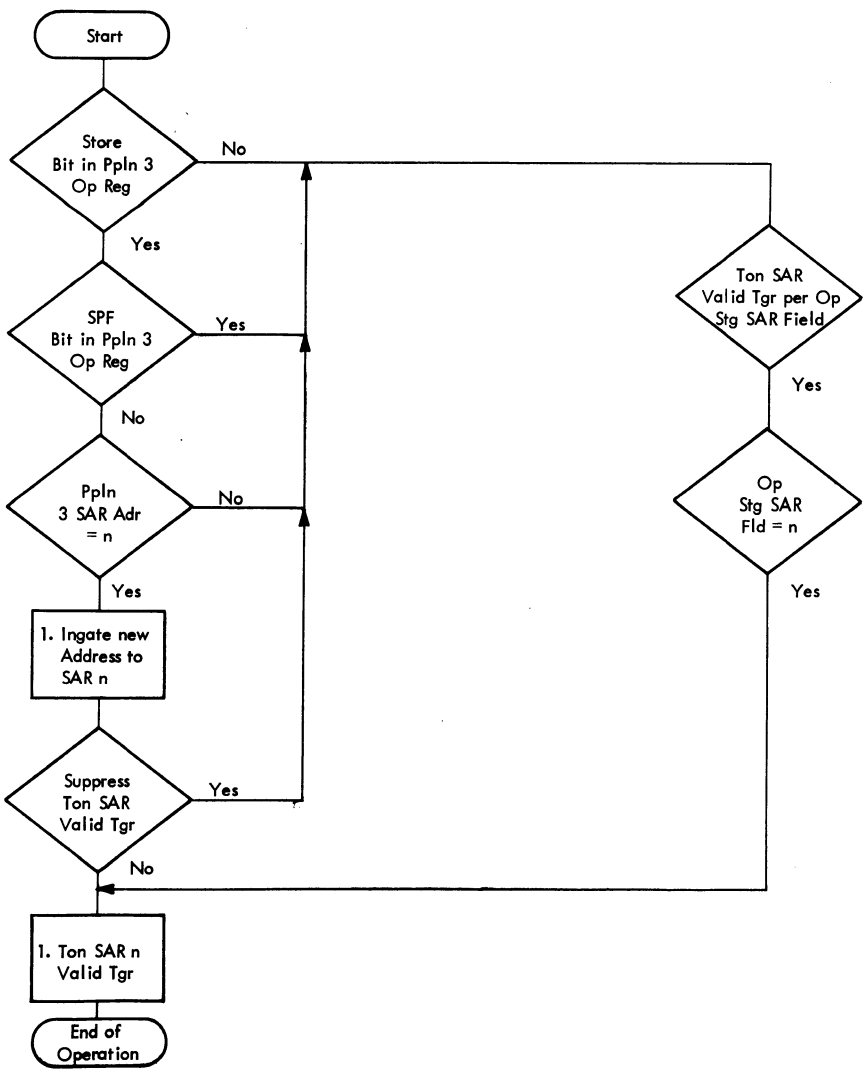
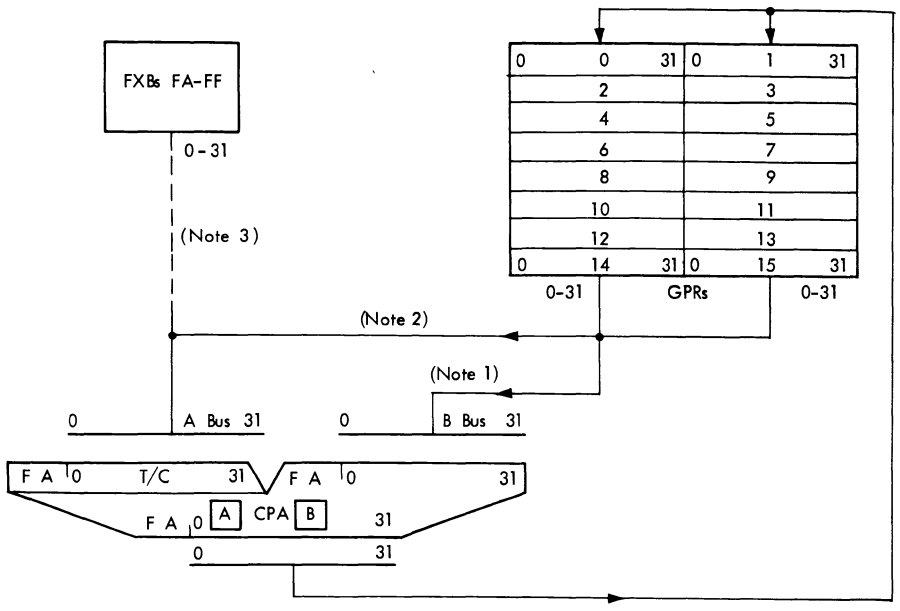


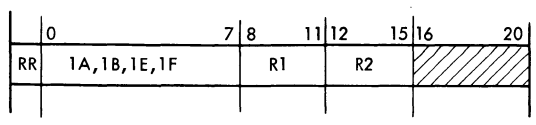
DIAGRAM 5-35. MOP DEFINITIONS



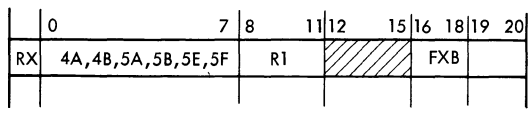
Notes:

1. R1 Field Data (RR and RX Formats).
2. R2 Field Data (RR Format Only).
3. FXB Field Data (RX Format Only).

1A	AR	Add
1B	SR	Subtract
1E	ALR	Add Logical
1F	SLR	Subtract Logical



4A	AH	Add Halfword
4B	SH	Subtract Halfword
5A	A	Add
5B	S	Subtract
5E	AL	Add Logical
5F	SL	Subtract Logical



CONDITION CODE

Arithmetic Ops:

0 = Results are zero
1 = Results less than zero
2 = Results greater than zero
3 = Overflow

Logical Ops:

0 = Results are zero (no carry)
1 = Results are not zero (no carry)
2 = Results are zero (carry)
3 = Results are not zero (carry)

1. Add and subtract operations gate data from the GPRs or storage (FXB) to the carry propagate adder.
2. Adder circuits perform all the add and subtract functions.
3. Results are placed back in the R1 GPR.
4. Logical operations treat the data as 32 bit values without signs.
5. In halfword operations, 16 bits of data from the FXB are gated to the adder. The high-order bits of the word are changed to agree with the halfword sign.
6. Normal execution time for ops 1A, 1B, 1E, and 1F is one machine cycle.
7. Ops 4A,4B,5A,5B,5E, and 5F require an undetermined number of machine cycles because input data must come from storage via an FXB.

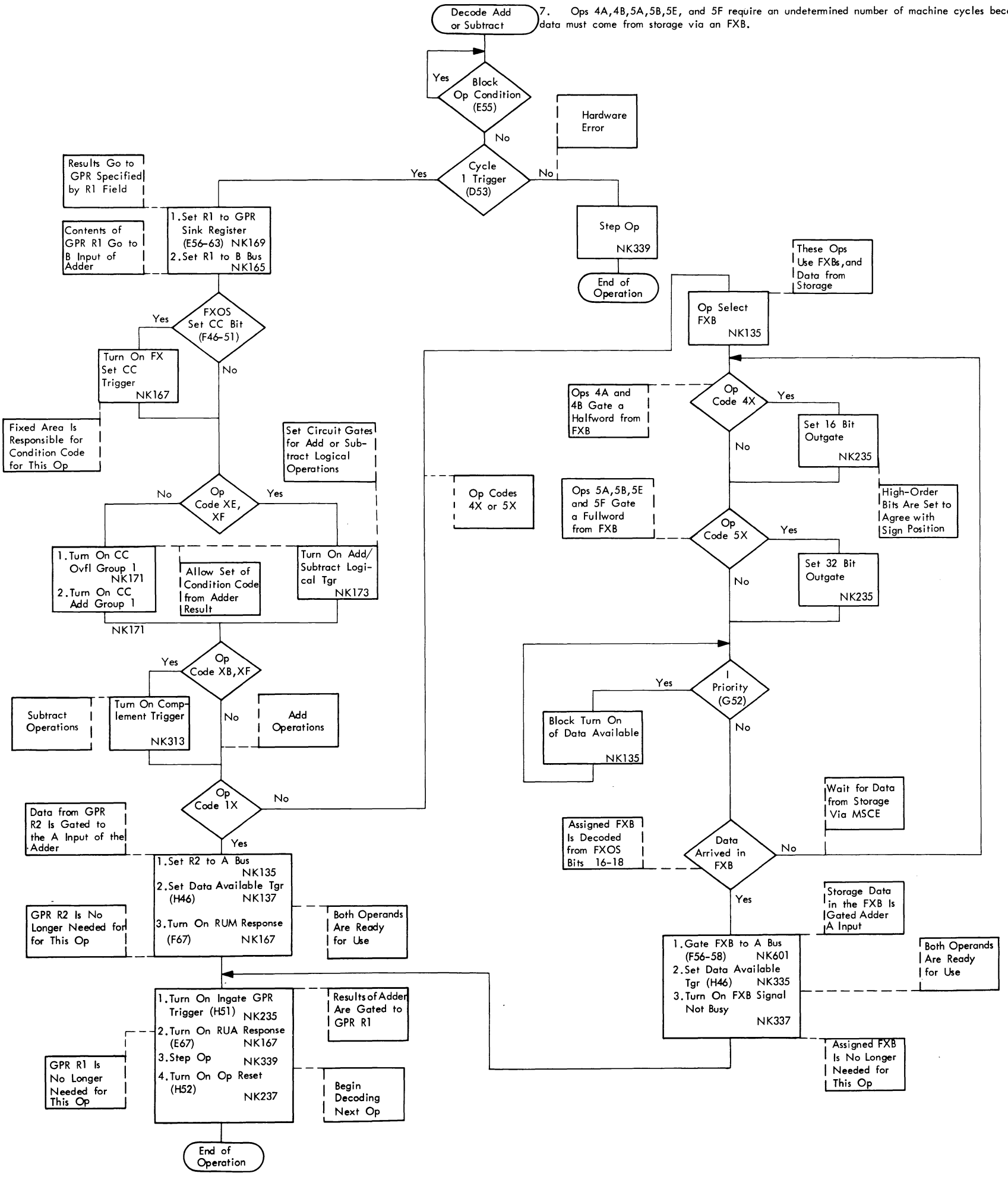
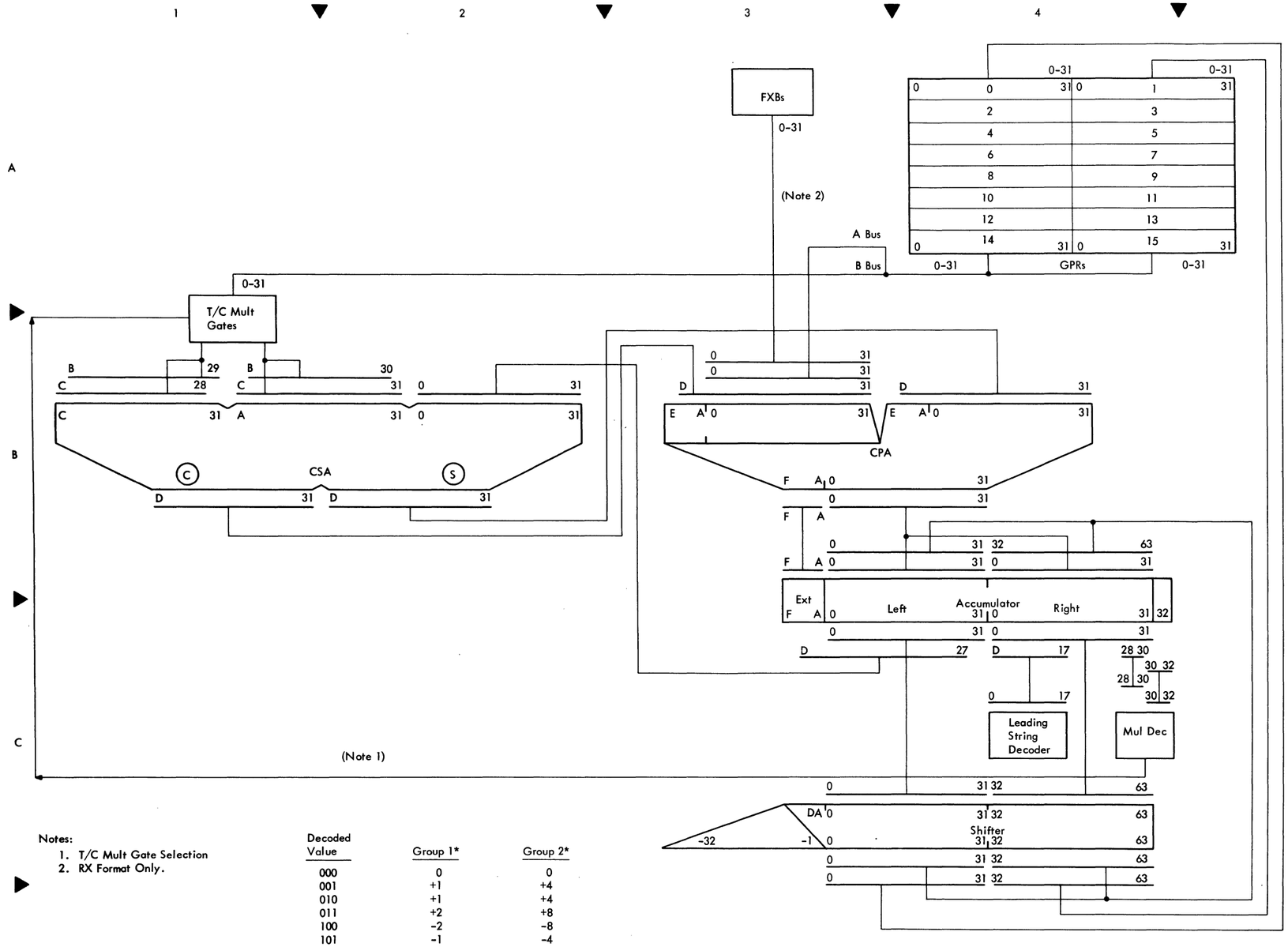


DIAGRAM 5-100. ADD AND SUBTRACT



Notes:
1. T/C Mult Gate Selection
2. RX Format Only.

Decoded Value	Group 1*	Group 2*
000	0	0
001	+1	+4
010	+1	+4
011	+2	+8
100	-2	-8
101	-1	-4
110	-1	-4
111	0	0

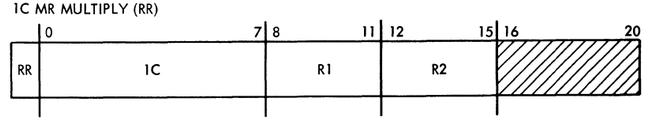
*True and complement form indicated by (+) and (-), respectively. Numbers indicate number of shift positions made for a particular decoded value.

T/C Multiply Gate Decoding

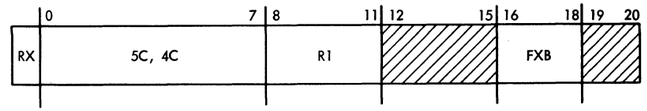
DIAGRAM 5-101. MULTIPLY (SHEET 1 OF 2)

The condition code remains unchanged during multiply ops

1C MR MULTIPLY (RR)



5C M MULTIPLY (RX)
4C MH MULTIPLY HALFWORD (RX)



Objectives:

1. During MULTIPLY operations, information from GPR's or Storage (FXB's) is multiplied and the product placed in a GPR.
2. The multiplicand is specified by the R1 field.
3. The multiplier is specified by the R2 field for RR multiply and by the FXB field for RX multiply or multiply halfword.
4. Actual multiplication is performed by the True/Complement multiply gates at the input of the carry save adder (CSA), Each hex digit of the multiplier is decoded in turn. The decoded gates determine the multiplied value of the multiplicand entered into the partial product during each iteration.
5. T/C Mult. gates multiply by shifting the multiplicand 0, 1, 2 or 3 positions as it is gated into the CSA. By gating into the CSA in true or complement form and shifting the correct number of positions, any value of multiplication can be produced from 0 to 16 (the binary value of the multiplier hex digit).
6. The partial product is held in the left side of the accumulator and shifted right 4 positions for each iteration. The final product occupies both the left and right sides of the accumulator.
7. At the end of the multiply operation the final product is transferred to the GPR's.
8. Multiply operations can be performed in less than the normal 8 iterations (11 cycles) if the multiplier high order positions are an unbroken string of 9 or 17 zero bits or one bits.
9. During the last cycle the product is shifted the correct number of bits to position it in the sink GPR's.

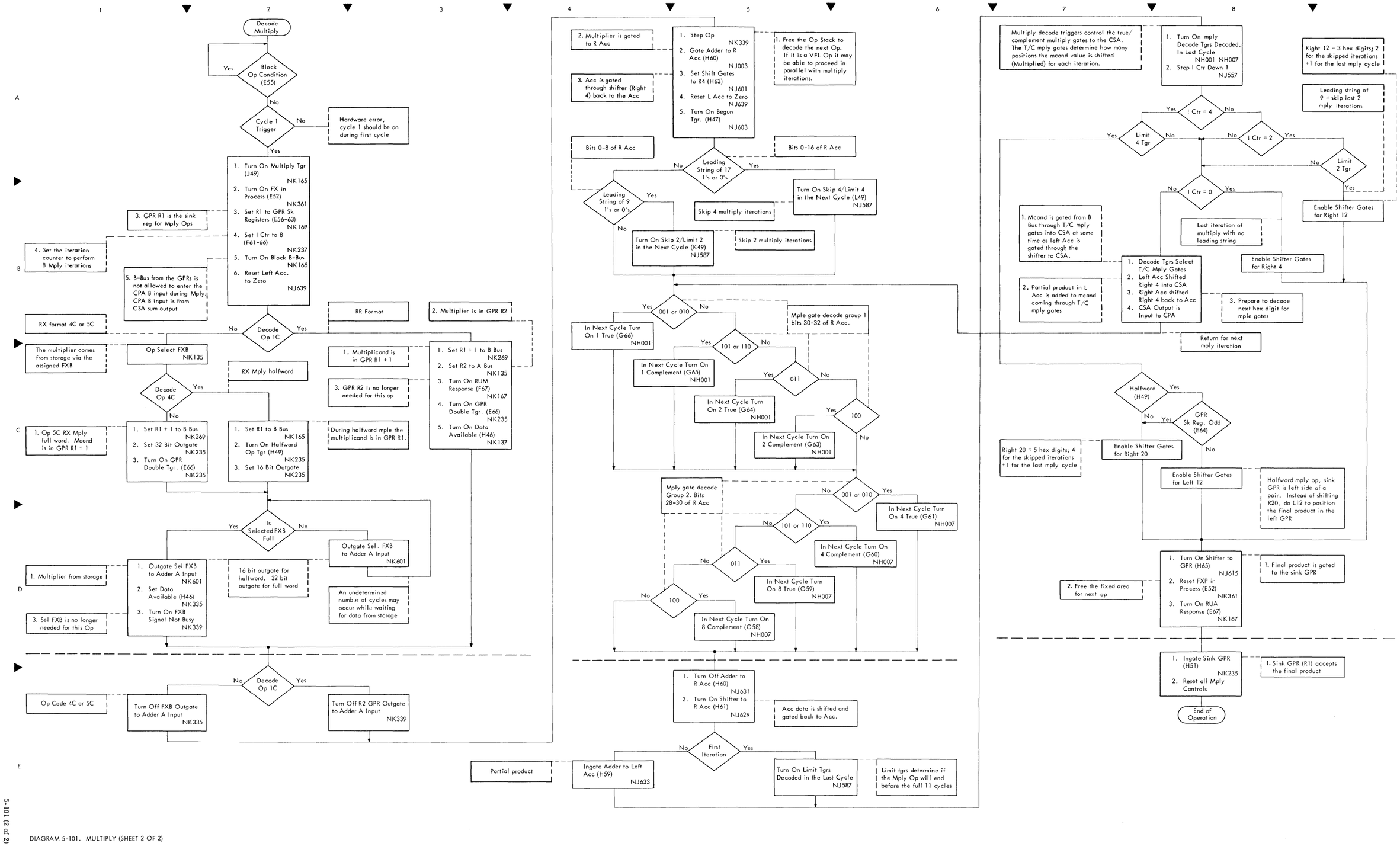


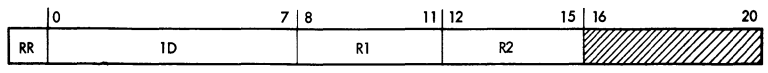
DIAGRAM 5-101. MULTIPLY (SHEET 2 OF 2)

Objectives:

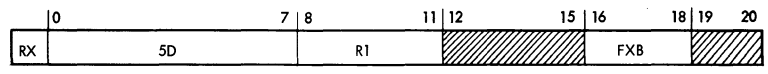
1. During divide operations the value in GPR R1 and R1+1 (Dividend) is divided by the value (Divisor) in either GPR R2 (RR) or an assigned FXB (RX). The results are placed back in GPR R1 (Remainder) and GPR R1+1 (Quotient).
2. Division is performed by repetitively subtracting the divisor from the dividend and at the same time shifting 1 bit position for each of 32 iterations.
3. The quotient is constructed bit by bit from the results of each subtraction.
4. In the last iteration, subtraction may cause the remainder to go negative (less than zero). In this case the divisor must be added back in once to correct the remainder.
5. Remainder, Divisor and Dividend signs determine the ending sequence which adjusts the results to conform to standard mathematical sign rules.
6. During the ending sequence the quotient is transferred to GPR R1+1 and the remainder is transferred to GPR R1.
7. Divide operations may be aborted if the factors exceed the capacity of machine circuits.
8. A complete divide operation requires 36 or 37 machine cycles depending on how many cycles are required in the ending sequence.

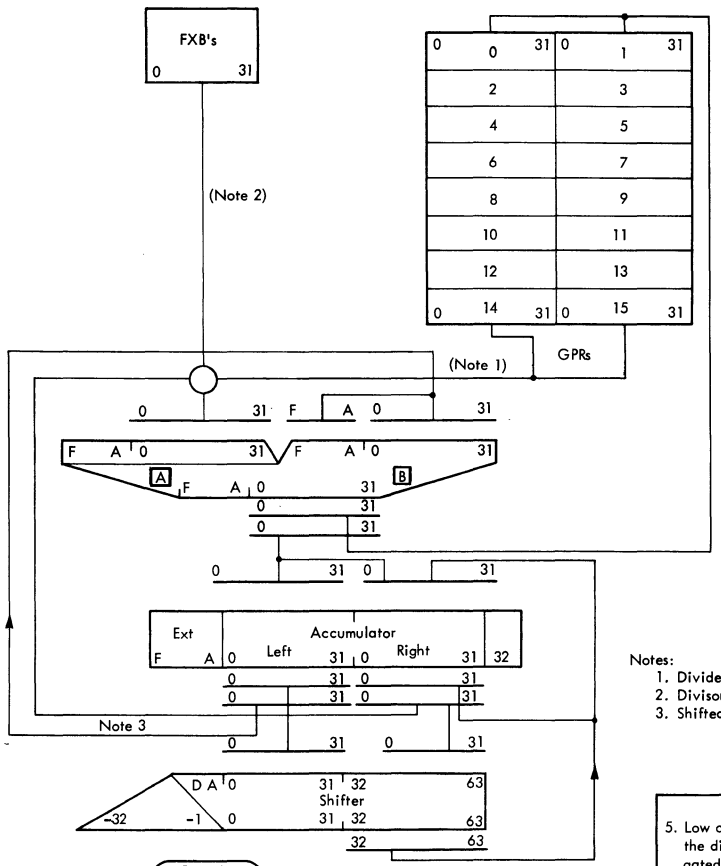
Condition Code Remains Unchanged For Divide Operations.

1D DR Divide (RR)



5D D Divide (RX)





Notes:
1. Dividend and divisor for RR format.
2. Divisor for RX format.
3. Shifted left 1.

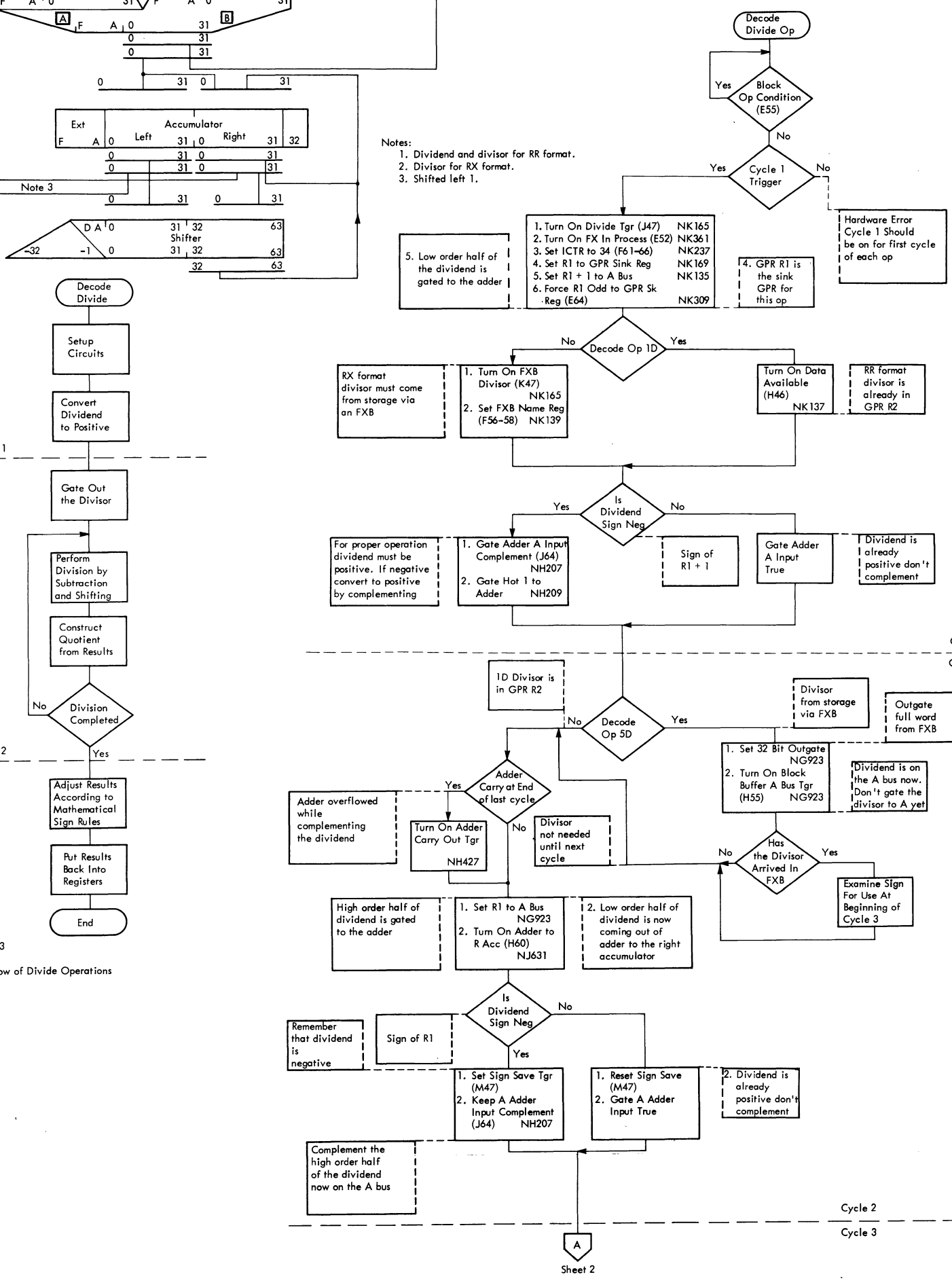


DIAGRAM 5-102. DIVIDE (SHEET 1 OF 3)

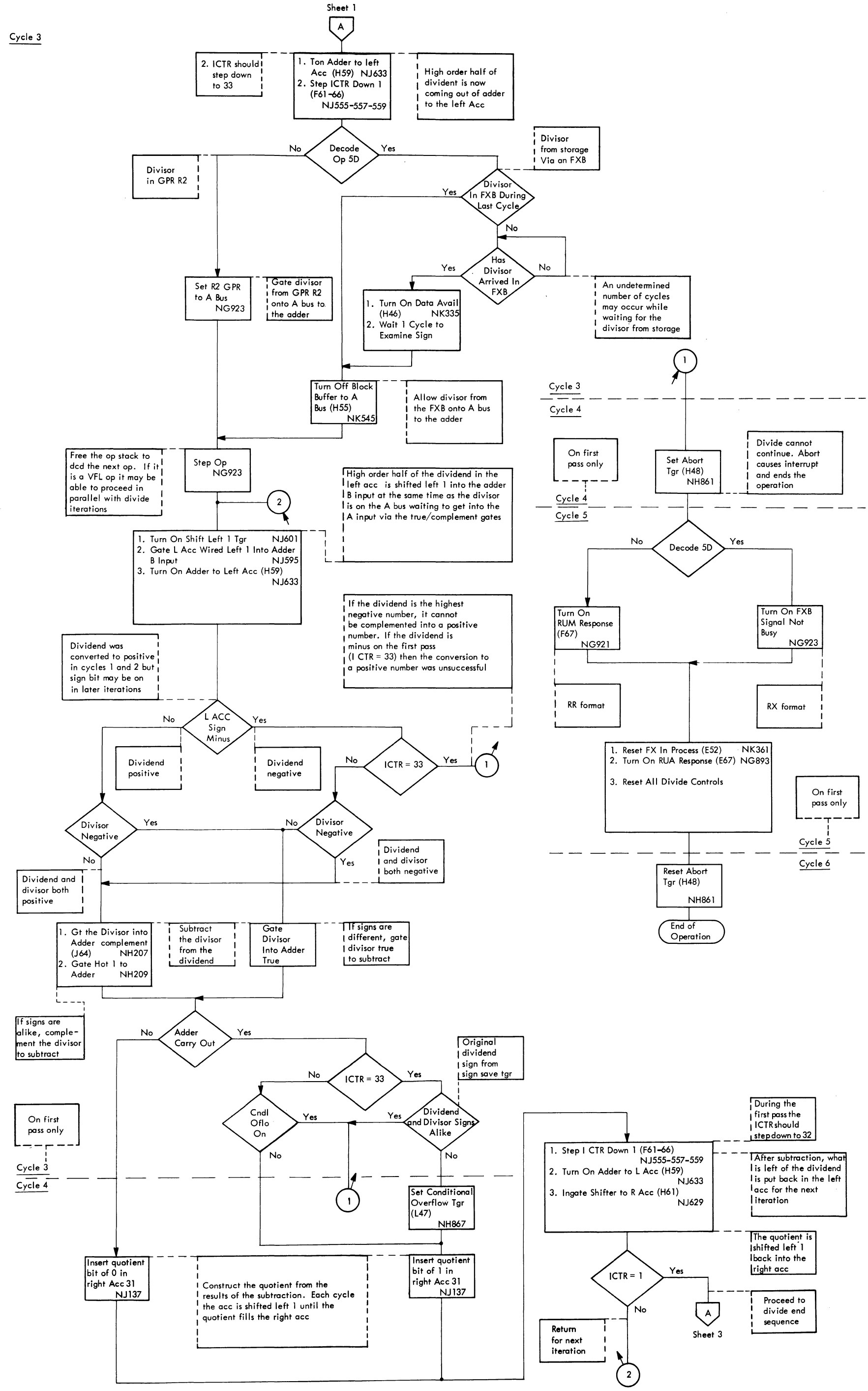
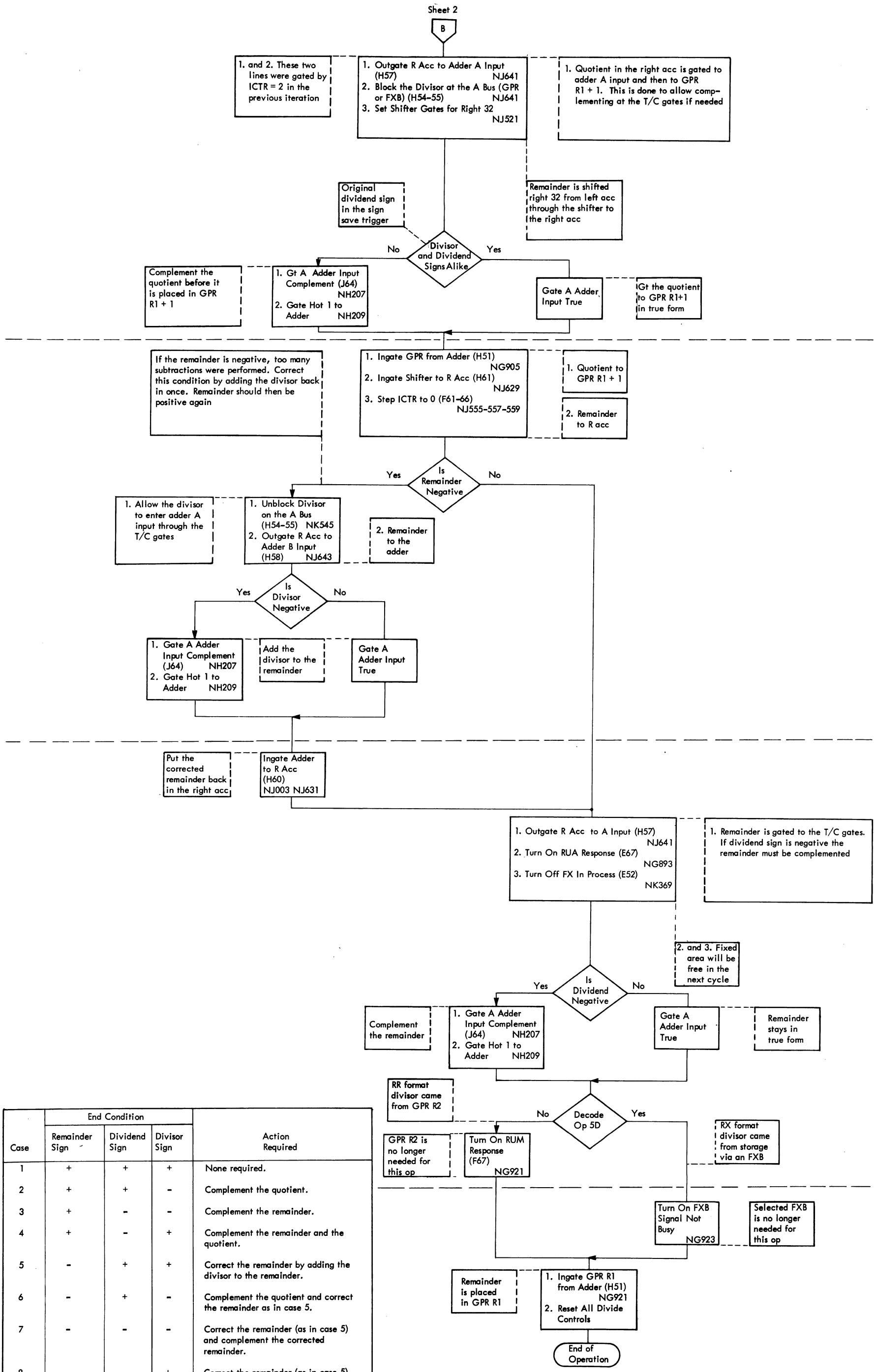
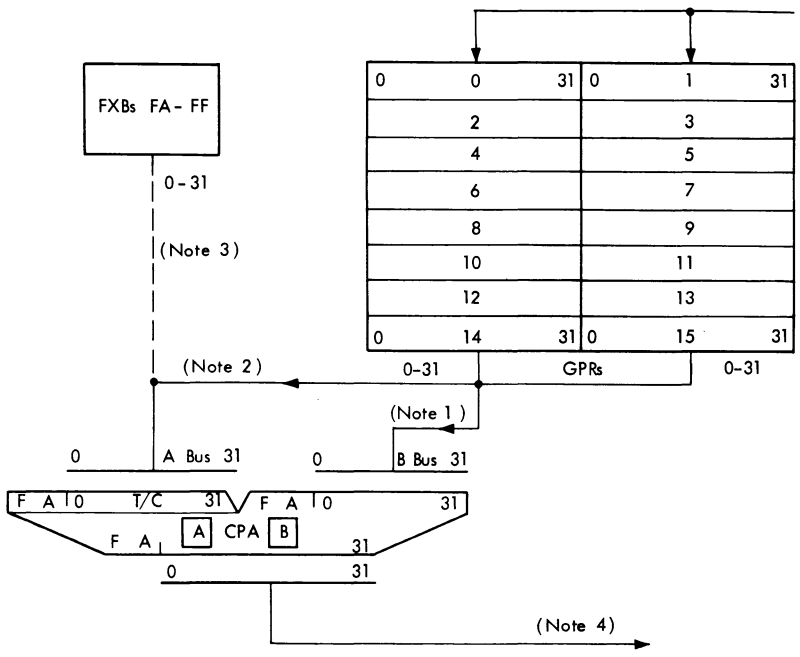


DIAGRAM 5-102. DIVIDE (SHEET 2 OF 3)



Case	End Condition: Remainder Sign	End Condition: Dividend Sign	End Condition: Divisor Sign	Action Required
1	+	+	+	None required.
2	+	+	-	Complement the quotient.
3	+	-	-	Complement the remainder.
4	+	-	+	Complement the remainder and the quotient.
5	-	+	+	Correct the remainder by adding the divisor to the remainder.
6	-	+	-	Complement the quotient and correct the remainder as in case 5.
7	-	-	-	Correct the remainder (as in case 5) and complement the corrected remainder.
8	-	-	+	Correct the remainder (as in case 5) and complement the quotient and the corrected remainder.

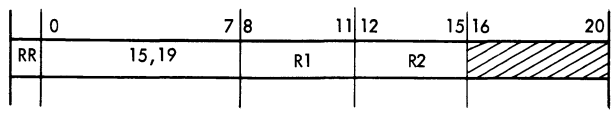
DIAGRAM 5-102. DIVIDE (SHEET 3 OF 3)



Notes:

1. R1 Field Data (RR and RX Formats).
2. R2 Field Data (RR Format Only).
3. FXB Field Data (RX Format Only).
4. Results set the CC but are not gated back to a GPR.

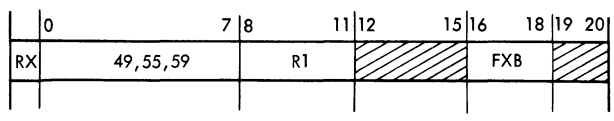
15	CLR	Compare Logical
19	CR	Compare



Condition Code

0 = Operands are equal
1 = First Operand is low
2 = First operand is high

49	CH	Compare Halfword
55	CL	Compare Logical
59	C	Compare



1. Compare operations gate data from the GPR's or Storage (FXB) to the carry propagate adder.
2. The value of R2 is subtracted from the value of R1 and the results set the condition code.

   Operands are not changed by the compare operations.
4. Logical compares treat the data as 32 bit values without signs.
5. During halfword compare, 16 bits of data from the FXB are gated to the adder. The high order bits of the word are changed to agree with the halfword sign.
6. Normal execution time for ops 15 and 19 is 1 machine cycle.
7. Ops 49, 55 and 59 require an undetermined number of machine cycles because input data must come from storage via an FXB.

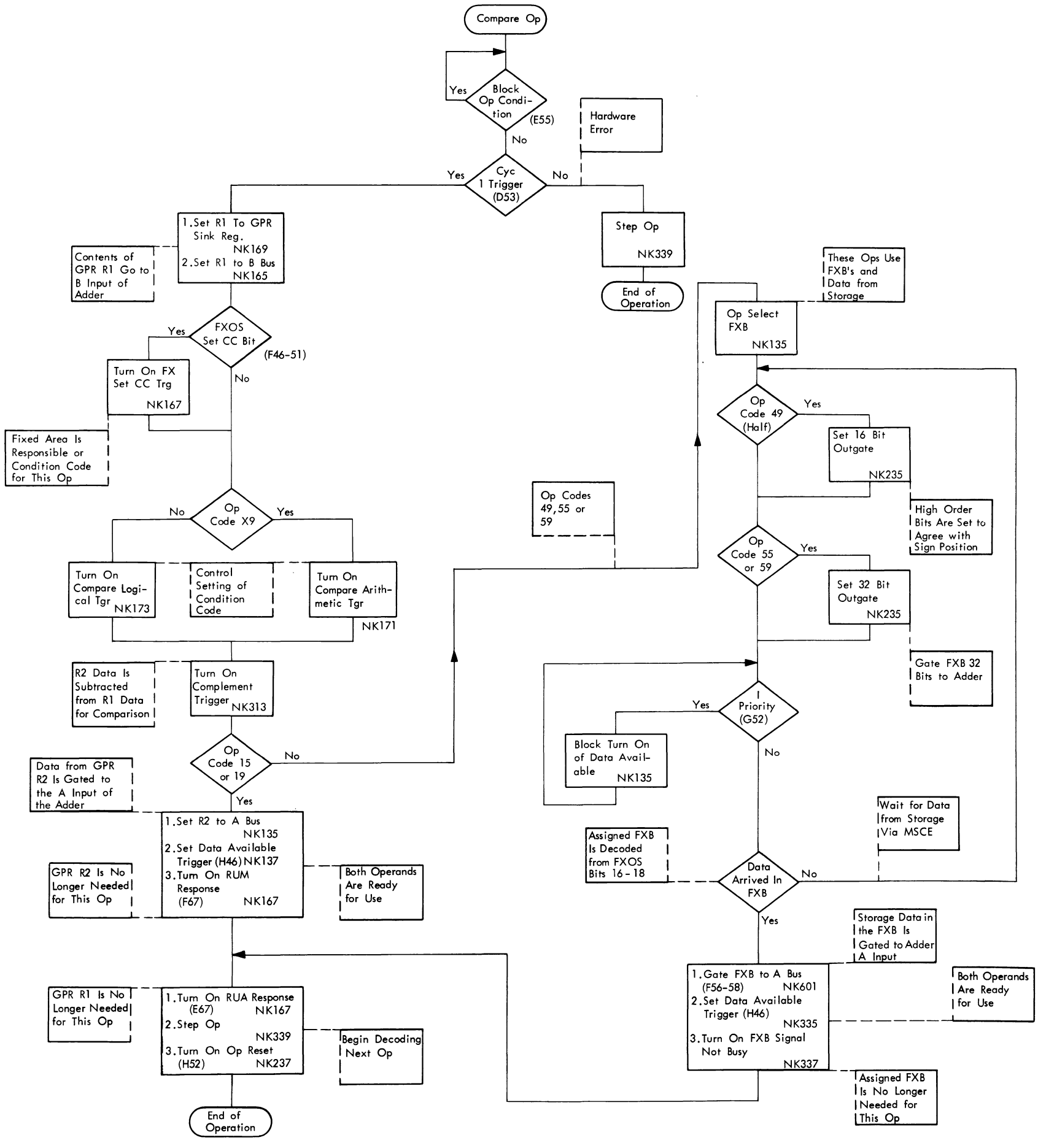
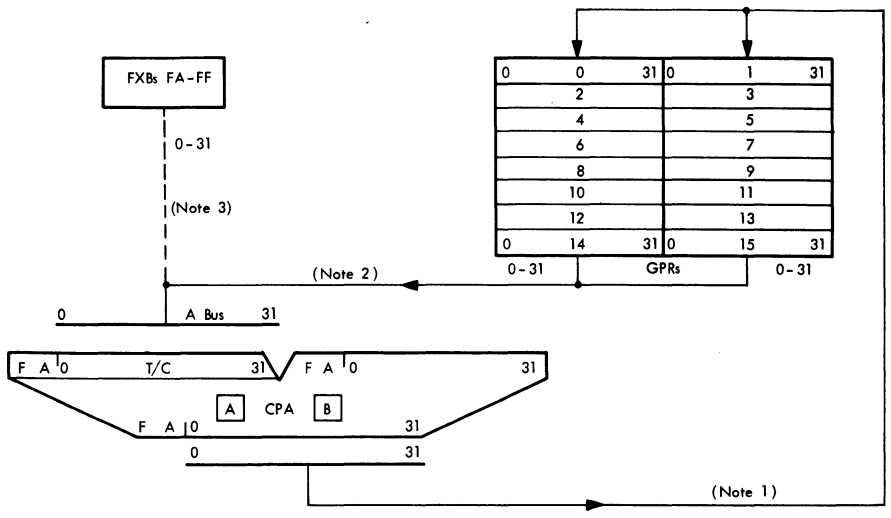
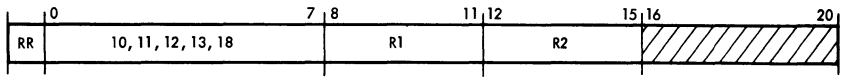


DIAGRAM 5 - 103. COMPARE

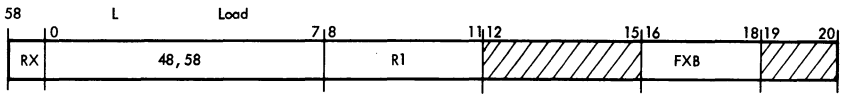


Notes:
1. R1 Field Data (RR and RX Formats).
2. R2 Field Data (RR Format Only).
3. FXB Field Data (RX Format Only).

10	LPR	Load Positive
11	LNR	Load Negative
12	LTR	Load and Test
13	LCR	Load Complement
18	LR	Load



48	LH	Load Halfword
58	L	Load



For ops 18, 48 and 58 the CC remains unchanged.

Ops 10, 11, 12 and 13 set the CC as follows:

CC Adder Condition

00 = Result is zero
01 = Result less than zero
10 = Result greater than zero
11 = Overflow

1. Load operations place information in a GPR (R1) from either another GPR (R2) or from storage via an FXB.
2. I-unit initiates the storage fetch (ops 48 and 58) and assigns an FXB to accept the data from MSCE.
3. All data is routed through the A input of the carry propagate adder to R1 to allow complementing when required (for ops 10, 11 and 13).
4. Load operations (except LH) move 32 bits of information in parallel into GPR R1.
5. LH moves 16 bits of information into the low order half of R1 and propagates the sign bit through the high order half of R1.
6. Operations 10, 11, 12, 13 and 18 require 1 machine cycle.
7. Operations 48 and 58 must wait for storage data and therefore require an undetermined number of machine cycles.

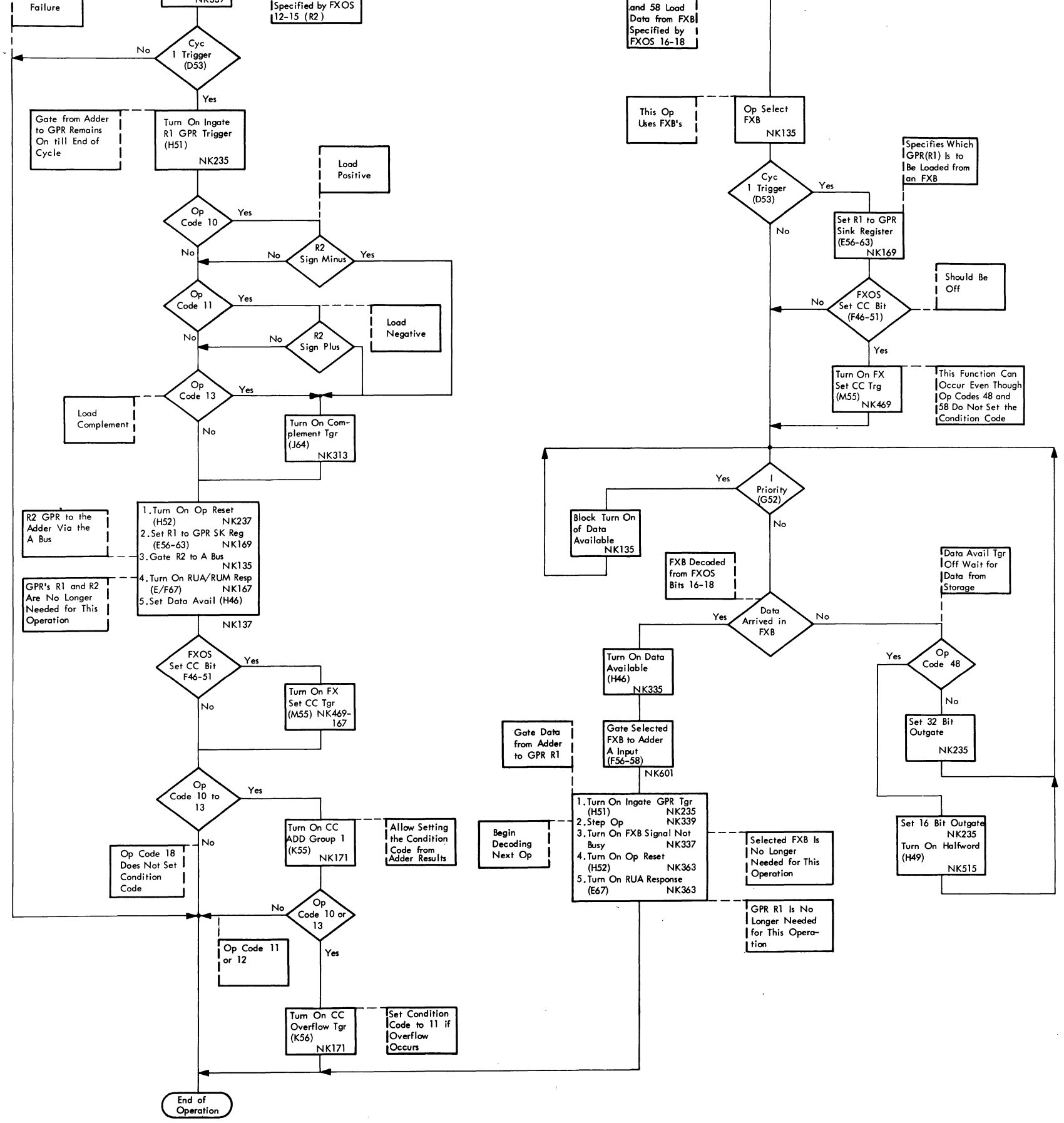
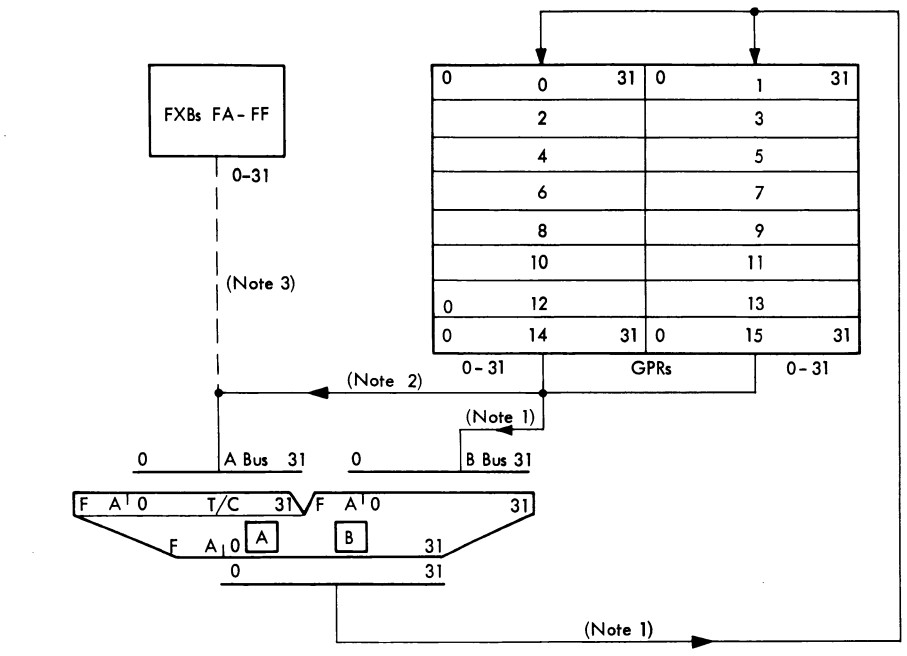


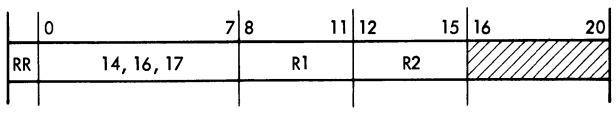
DIAGRAM 5-104. LOADS



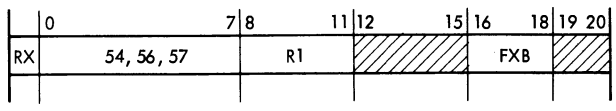
Notes:

1. R1 Field Data (RR and RX Formats)
2. R2 Field Data (RR Format Only)
3. FXB Field Data (RX Format Only)

14	NR	And
16	OR	Or
17	XR	Exclusive Or



54	N	And
56	O	Or
57	X	Exclusive Or



Condition Code

CC = 0: Results Zero
CC = 1: Results Not Zero

1. And, Or, and Exclusive Or operations gate data from the GPR's or storage (FXB) to connective circuits in the carry propagate adder.
2. The connective circuits perform all the And, Or and Exclusive Or functions.
3. Results are placed back in the R1 GPR.
4. Normal execution time for ops 14, 16 and 17 (RR format) is 1 machine cycle.
5. Ops 54, 56 and 57 (RX format) require an undetermined number of cycles because input data must come from storage via an FXB.

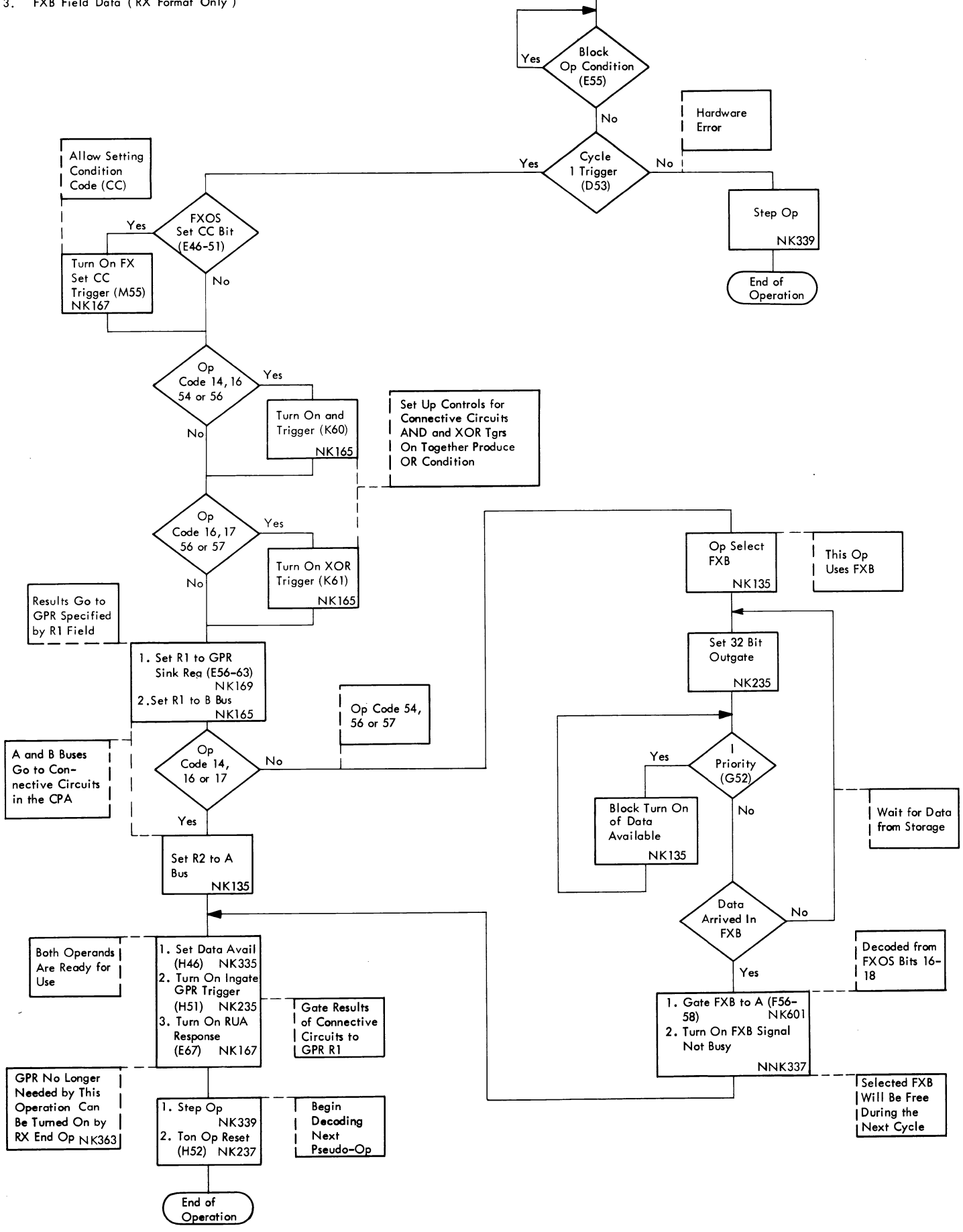
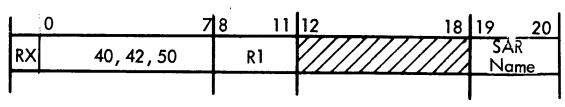


DIAGRAM 5-105. AND, OR, AND EXCLUSIVE OR

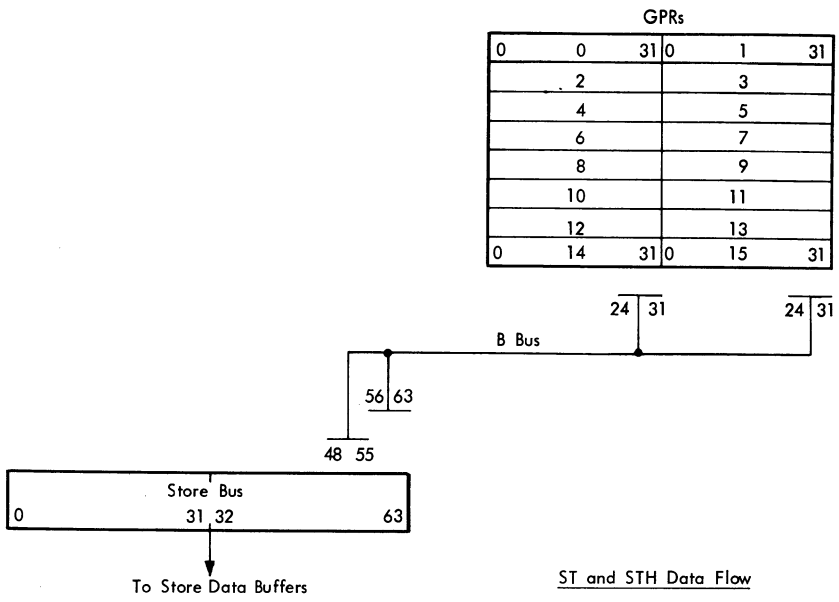
CC Remains Unchanged

40	STH	Store Halfword
42	STC	Store Character
50	ST	Store (Word)

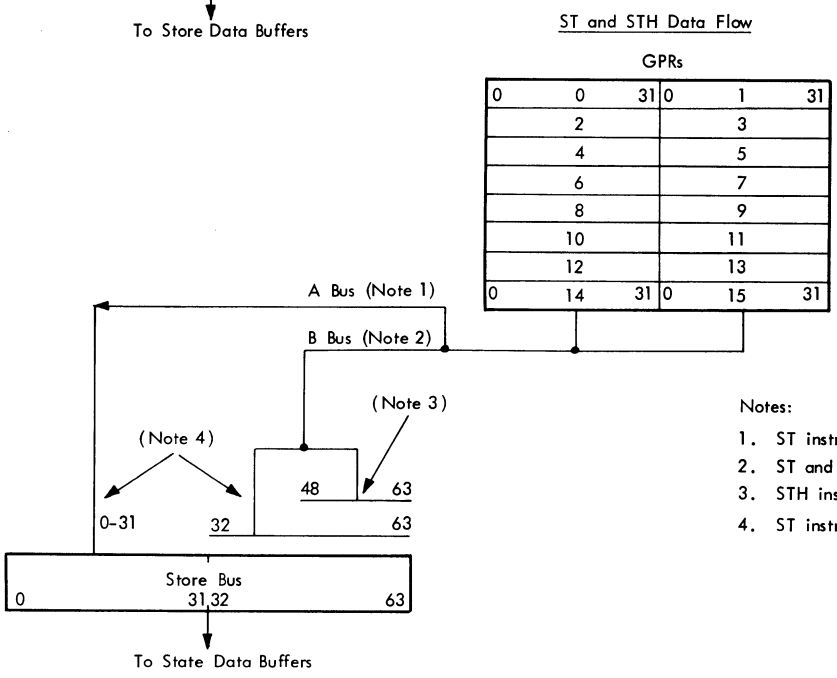


1. Data is transferred from the GPR, specified by R1, to storage via the fixed area store bus and MSCE storage data buffers (SDB).
2. During a store operation, all 32 bits of R1 are placed in storage via bus positions 0-31 and 32-63.
3. During a store halfword operation, bits 16-31 of R1 are placed in storage via bus positions 48-63.
4. During a store character operation, bits 24-31 of R1 are placed in storage via bus positions 48-55 and 56-63.
5. I-unit notifies MSCE what portion of the SDB data is to be stored.
6. Normal execution time is 1 machine cycle.

STC Data Flow



ST and STH Data Flow



Notes:
1. ST instruction only.
2. ST and STH instructions.
3. STH instruction only.
4. ST instruction only.

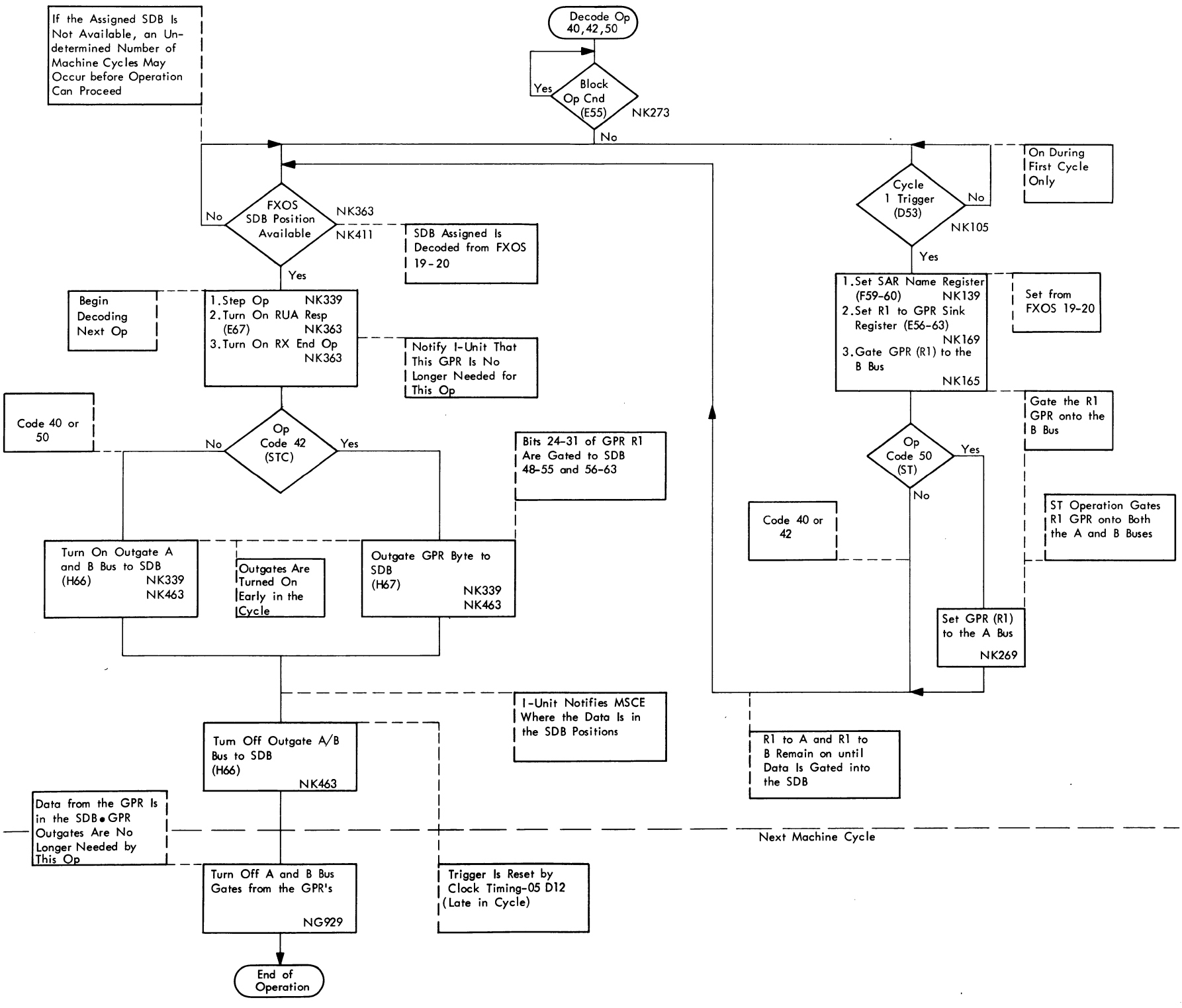
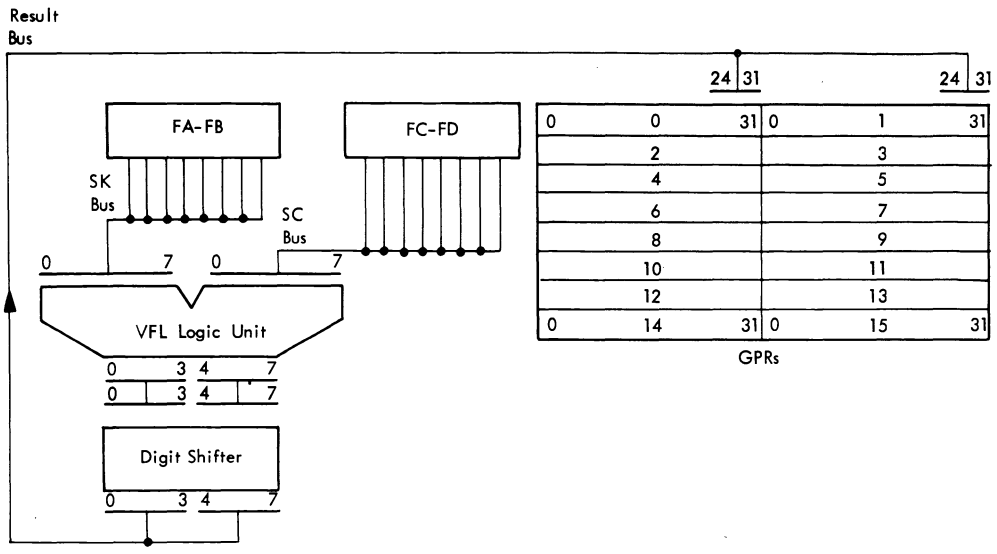
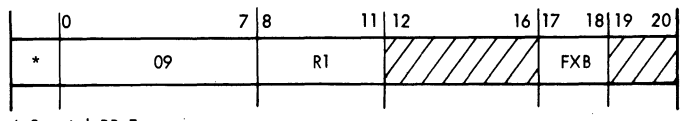


DIAGRAM 5-106. STORE, STORE HALFWORD AND STORE CHARACTER



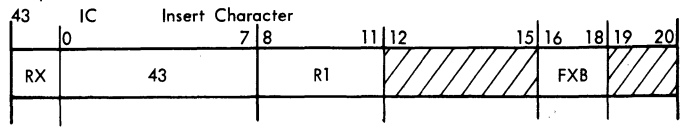
Condition Code
Remains Unchanged

09 ISK Insert Storage Key



* Special RR Format

43 IC Insert Character



1. Insert operations move a byte of information from storage into bits 24-31 of GPR R1.

2. ISK obtains the information from storage protect memory (a 5 bit storage key).

3. IC fetches a byte from a storage address.

4. Data path for the information is from storage through the FXB, VFL logic unit, digit shifter (straight through), and the result bus into GPR R1.

5. Insert ops require an undetermined number of machine cycles because data must come from storage.

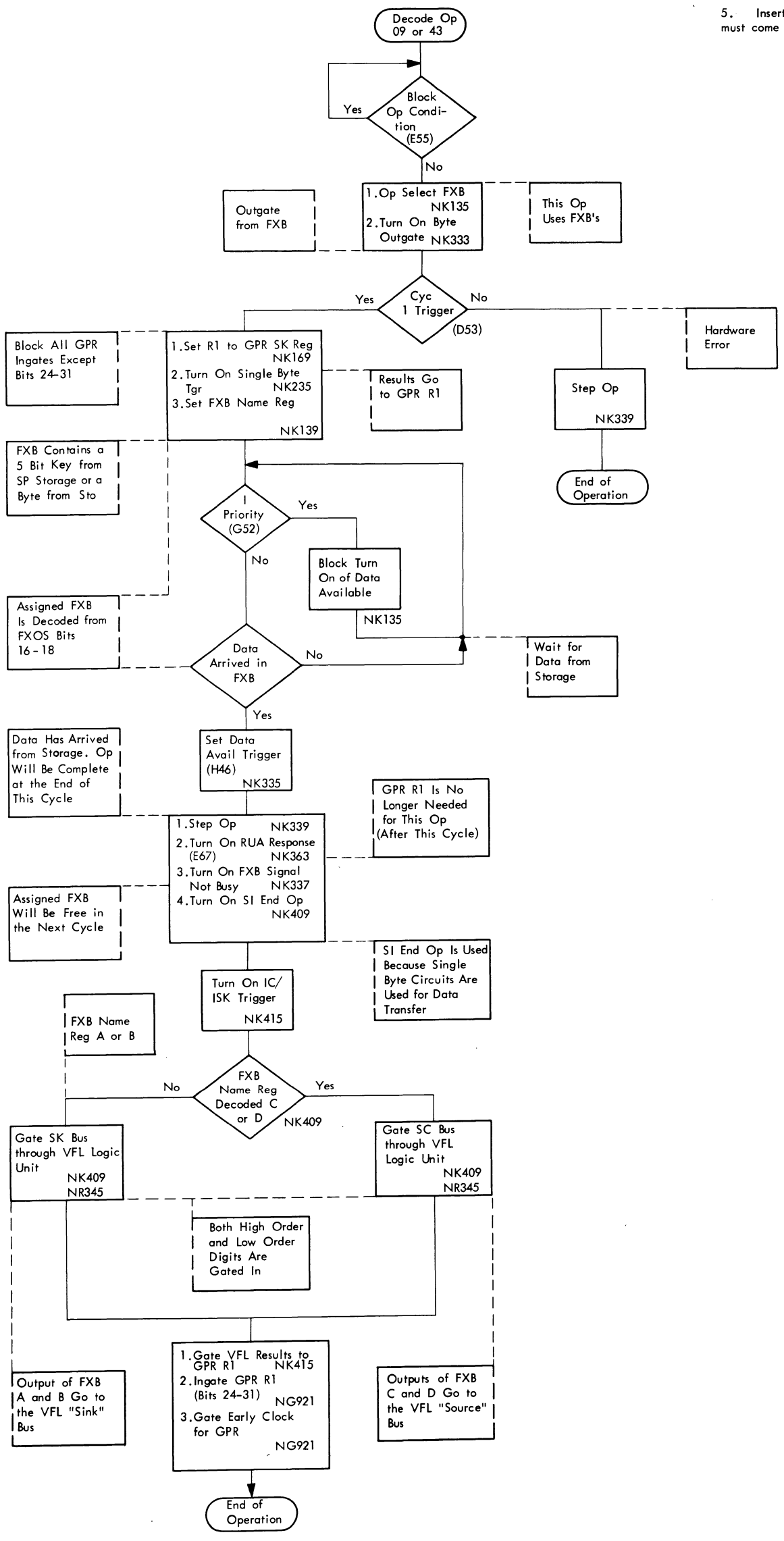
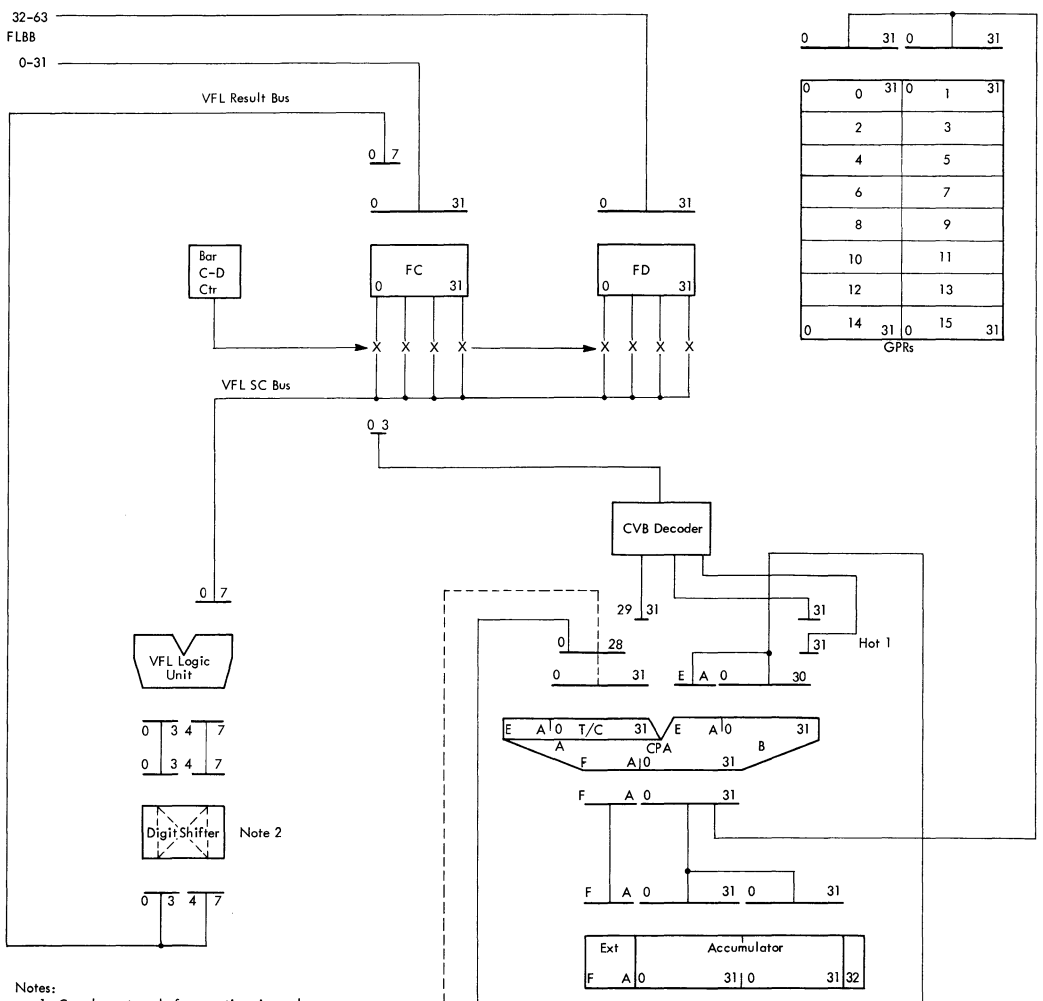


DIAGRAM 5-107. INSERT

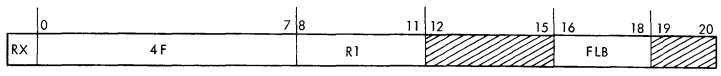
CVB Data Flow



Notes:
1. Complement cycle for negative sign only.
2. Shifter capable of straight through transfer or transposition of high-order and low-order digits.

The Condition Remains Unchanged for CVB

4 F CVB Convert to Binary



Objectives:

1. Convert to Binary Processes a doubleword of packed decimal information from storage and converts the information to Binary.
2. The doubleword from storage is held temporarily in an FLB in the floating point unit before being transferred to FXB C and D in the fixed point unit.
3. Each digit (4 bits) of the doubleword is moved in turn into the high order 4 bits of FXB C only, decoded and added into the binary value in the accumulator.
4. As each digit is decoded and added in, the binary value is multiplied by 10_{10} to simulate shifting 1 decimal position. The high order digit is decoded first so it will be multiplied by 10 (or shifted 1) for each following iteration (maximum of 15).
5. The largest number that can be held in packed decimal form in a doubleword would produce a binary number larger than a 32 bit register can hold. An overflow detection circuit monitors left accumulator positions 0, 1 and 2 to detect this condition. When results are too big to be stored in a GPR, FXP Divide Interrupt occurs.
6. If any digit in the source doubleword (except sign) contains a number higher than 9 (hexidecimal A to F) the operation will abort and cause an interrupt.
7. At the end of the operation the sign digit of the double word is examined. If the sign digit is a legitimate negative, the binary value is complemented to a negative form.
8. The resulting binary number is placed in the GPR specified by the R1 field of the pseudo-op.

Digit Value	CPA A Bit Positions 29 30 31	CPA B Bit Position 31	CPA B Hot 1 Bit 31
0	0 0 0	0	0
1	0 0 1	0	0
2	0 1 0	0	0
3	0 1 1	0	0
4	1 0 0	0	0
5	1 0 1	0	0
6	1 1 0	0	0
7	1 1 1	0	0
8	1 1 1	1	0
9	1 1 1	1	1

CVB Decoder Chart

Digit values are decoded and added into the binary result in the accumulator during each iteration of a CVB operation (objective 3).

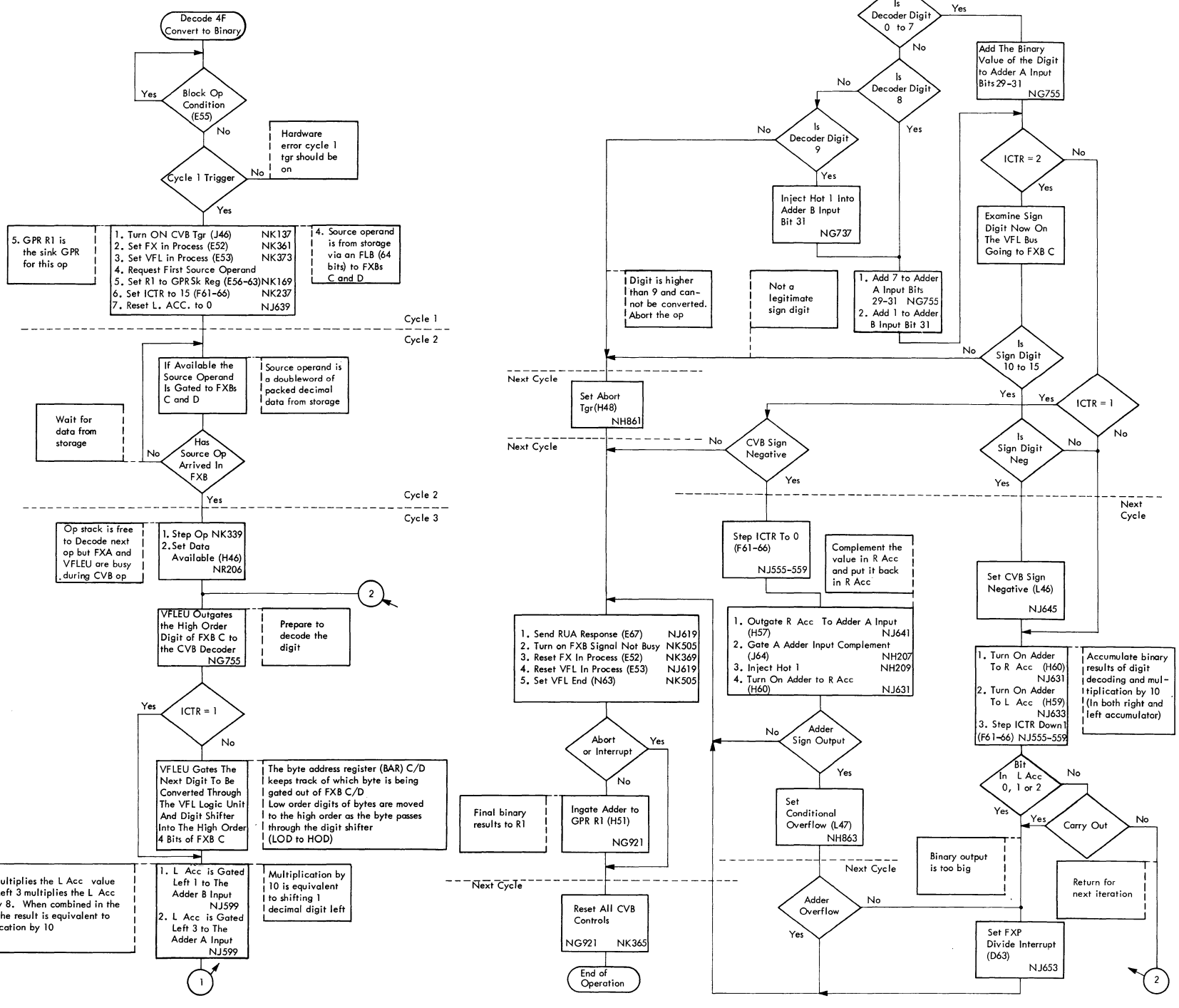
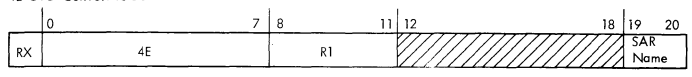


DIAGRAM 5-108. CONVERT TO BINARY

The Condition Code Remains Unchanged During CVD Operations

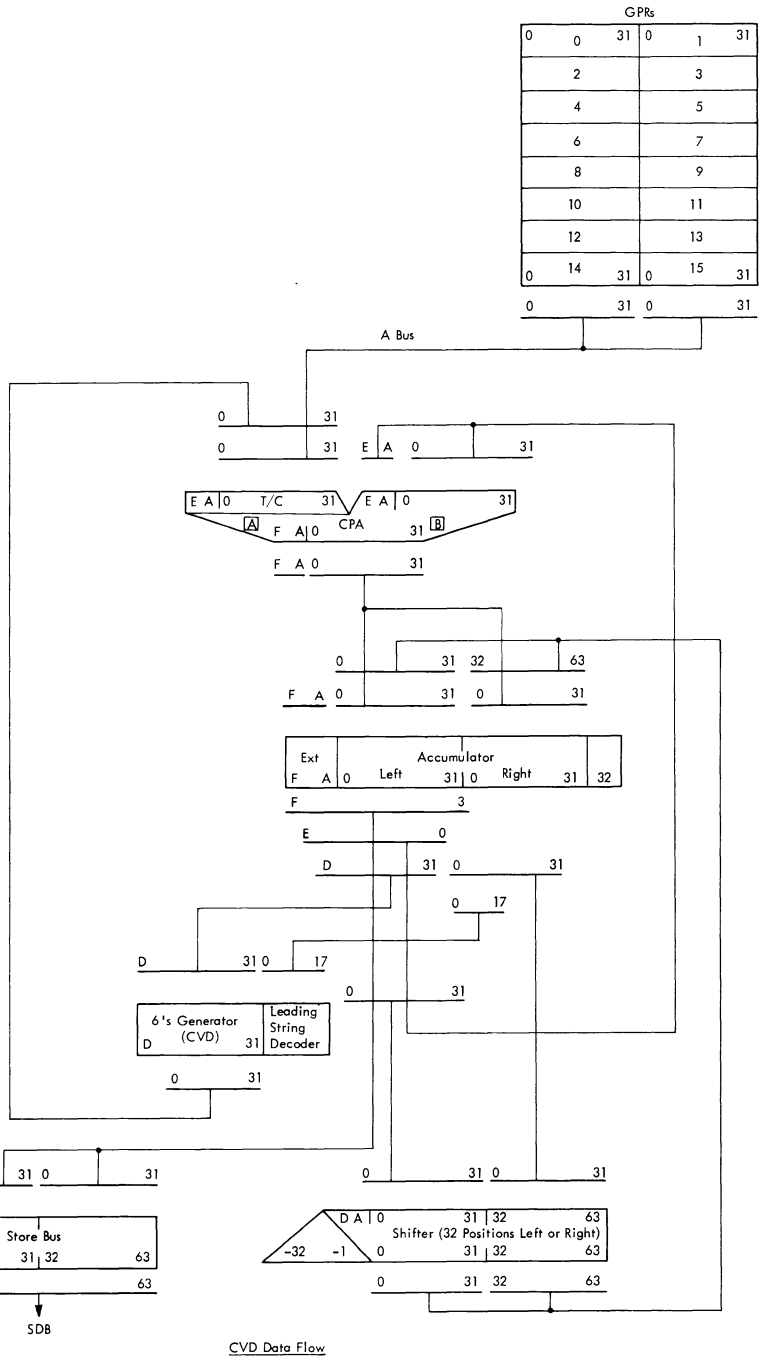
4E CVD Convert to Decimal



Objectives:

1. A CVD operation converts the data in GPR R1 from binary to packed decimal and stores the results via the SDB.
2. The address in the R1 field is normally a sink GPR but for CVD it is used as a source. Storage is the sink for CVD.
3. The sign of R1 is stored in the sign save trigger. The sign digit of the converted data is set to agree with the sign save trigger and format specified by PSW 12 (PSW 12 ON for ASCII; OFF for EBCDIC)
4. Conversion is done by decoding the output of the accumulator and inserting a 6 in the adder for each digit position that contains 5 or greater. The result will be a 38 bit decimal output from the adder back to the accumulator. (with sign bits the result is 42 bits)
5. The 42 bit result is transferred to storage in a doubleword (64 bits) via the SDB.
6. Timing of CVD depends on the number of leading zeros in the high order of R1 as follows:

Leading Zeros	Iterations	Total Cycles
0-4	No Skip = 28	32
5-8	Skip 4 = 24	29
9-12	Skip 8 = 20	25
13-16	Skip 12 = 16	21
17 or More	Skip 16 = 12	17



CVD Data Flow

SHIFT AMOUNTS IN CVD					
SHIFT AMOUNT	WIRED	STAGE 1	STAGE 2	STAGE 3	STAGE 4
L1	R16	L1	0	L16	0
L4	R16	0	L4	L16	0
L8	R16	0	0	0	L24
L12	R16	0	L4	0	L24
L16	R16	0	0	L8	L24

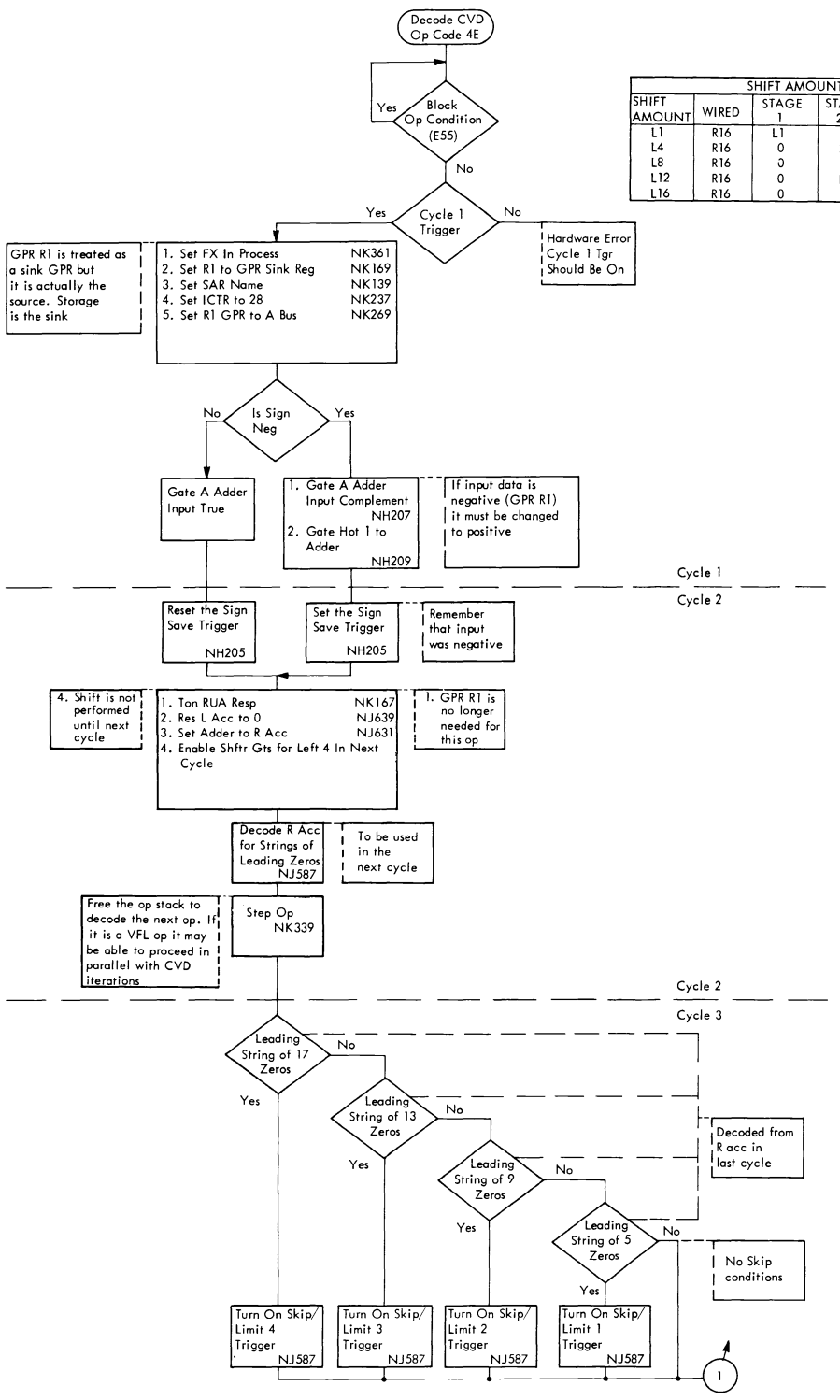
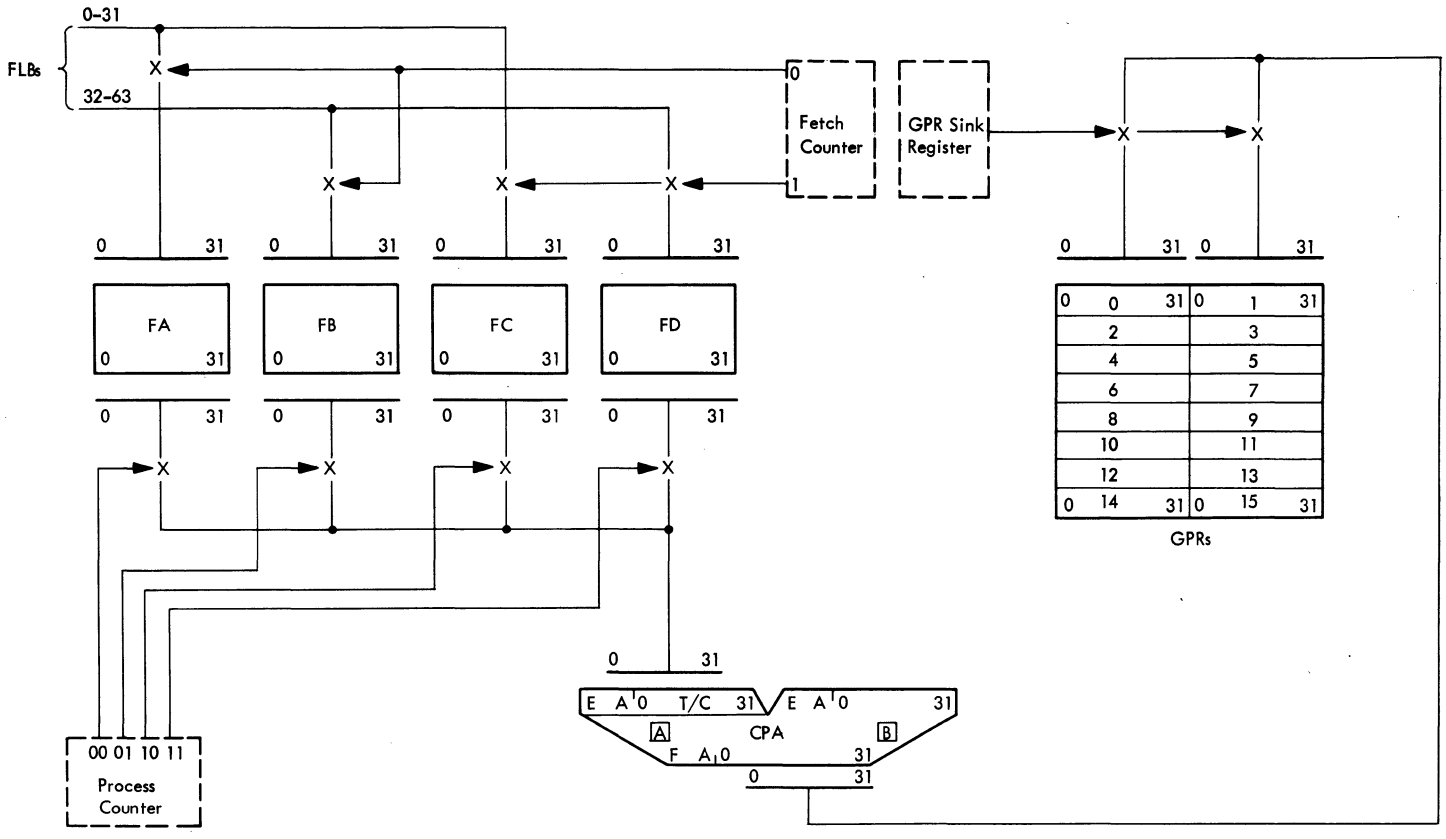


DIAGRAM 5-109. CONVERT TO DECIMAL



98 LM Load Multiple

0 7	8 11	12	13	14	15	16 18	19 20
RS 98	R1	L	R	S	E	FLB	

CC Remains Unchanged

1. Load multiple transmits data from storage through the floating point buffers, the fixed buffers, and the adder, to the GPRs.
2. Any number of GPRs, from 1 to 16 can be loaded with one LM operation.
3. I-unit issues pseudo op to the fixed area for each doubleword of data fetched from storage.
4. Start (S) or end (E) bits indicate the first and last pseudo ops of the operation.
5. The first and last pseudo ops may transmit only one word of data. In these cases the left (L) and the right (R) bits indicate which half of the doubleword fetch is to be used.
6. FXB A-B and C-D are used as doubleword pairs to accept data from the specified FLB. FXB E and F are not used for LM.
7. FXB are outgated, one at a time, through the adder to each GPR in sequence.

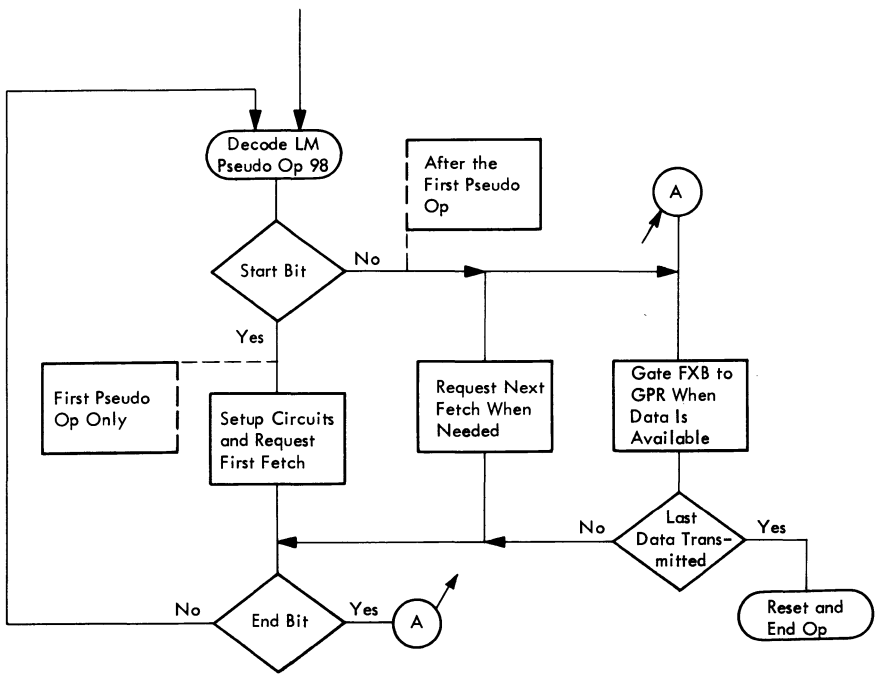
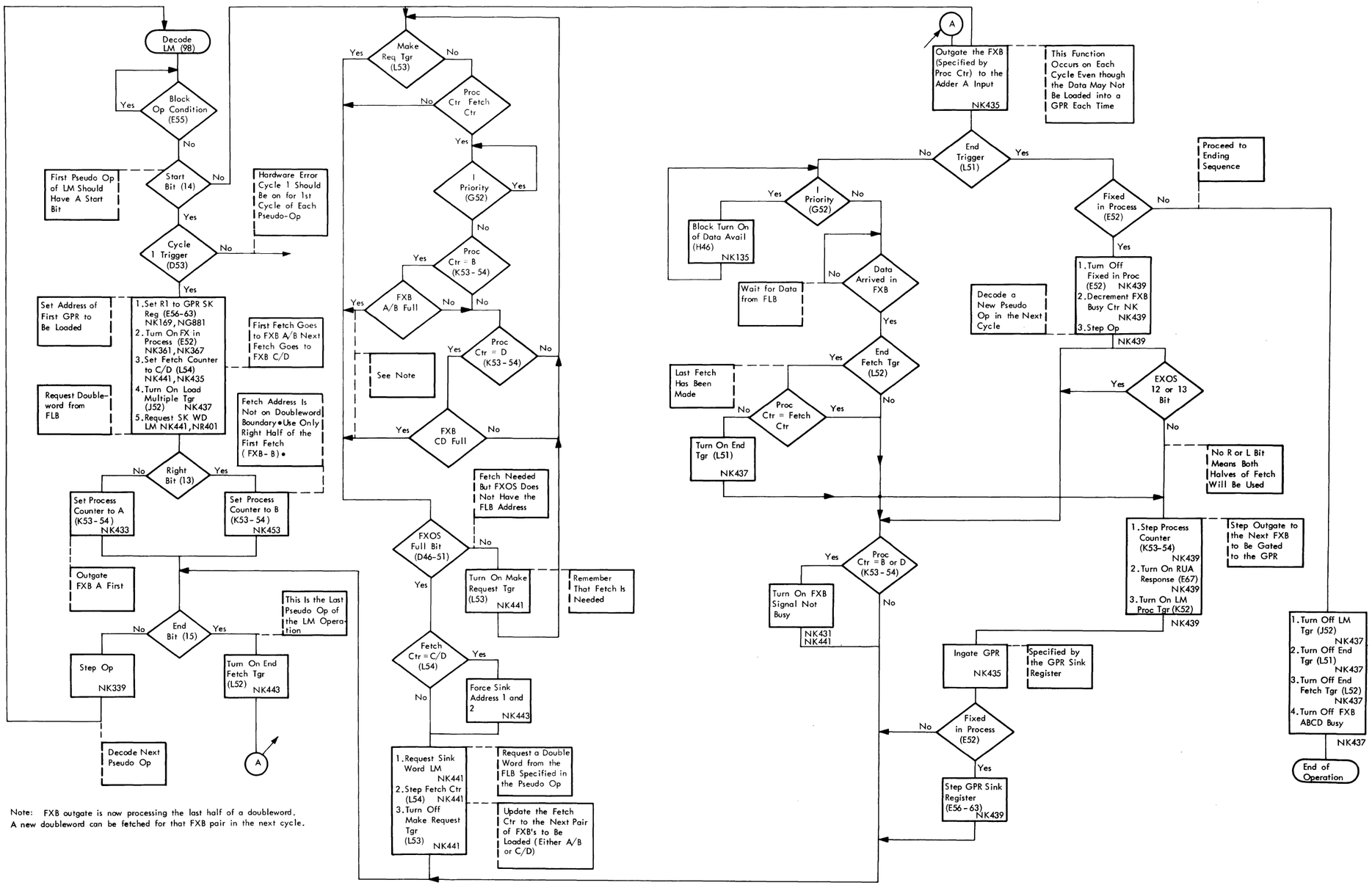


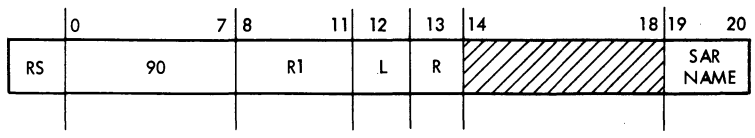
DIAGRAM 5-110. LOAD MULTIPLE (SHEET 1 OF 2)



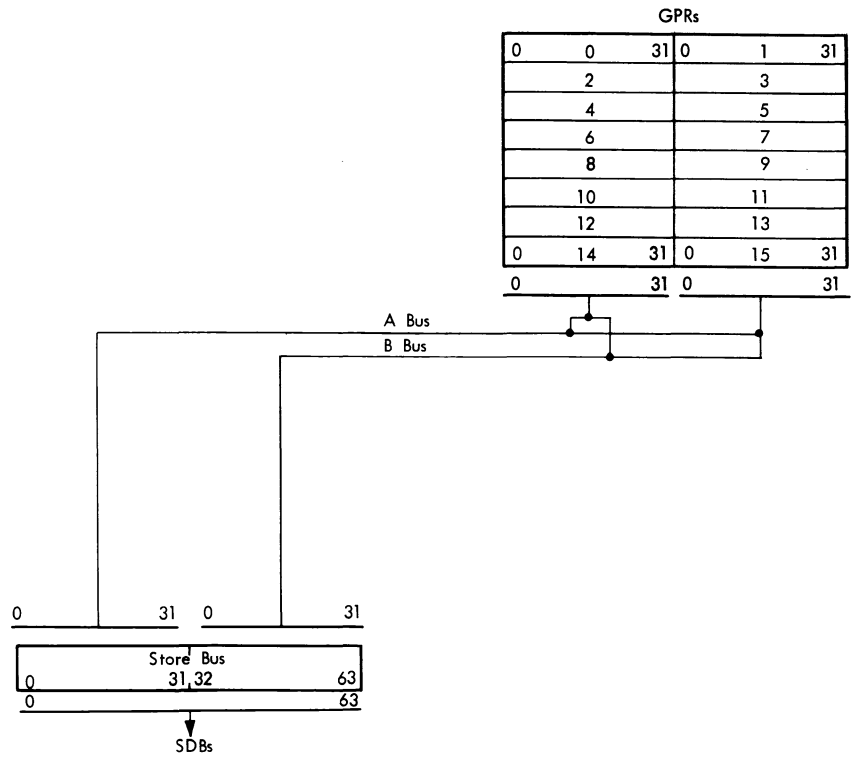
Note: FXB outgate is now processing the last half of a doubleword. A new doubleword can be fetched for that FXB pair in the next cycle.

DIAGRAM 5-110. LOAD MULTIPLE (SHEET 2 OF 2)

STM Store Multiple CC Remains Unchanged



1. A block of data is transferred from the GPR's to storage via the fixed area storage bus.
2. The R1 field specifies the first GPR of the block to be stored.
3. A pseudo-op will be issued, by the I-unit, for each doubleword of the data block.
4. Singleword stores can occur during the first and last pseudo-ops. The L (left) and R (right) bits indicate which half of the doubleword is to be stored.
5. During a singleword store two GPR's are gated out to MSCE but only one is actually stored. I-unit notifies MSCE which half of the doubleword to store.
6. Normal execution time is 1 machine cycle for each pseudo-op.



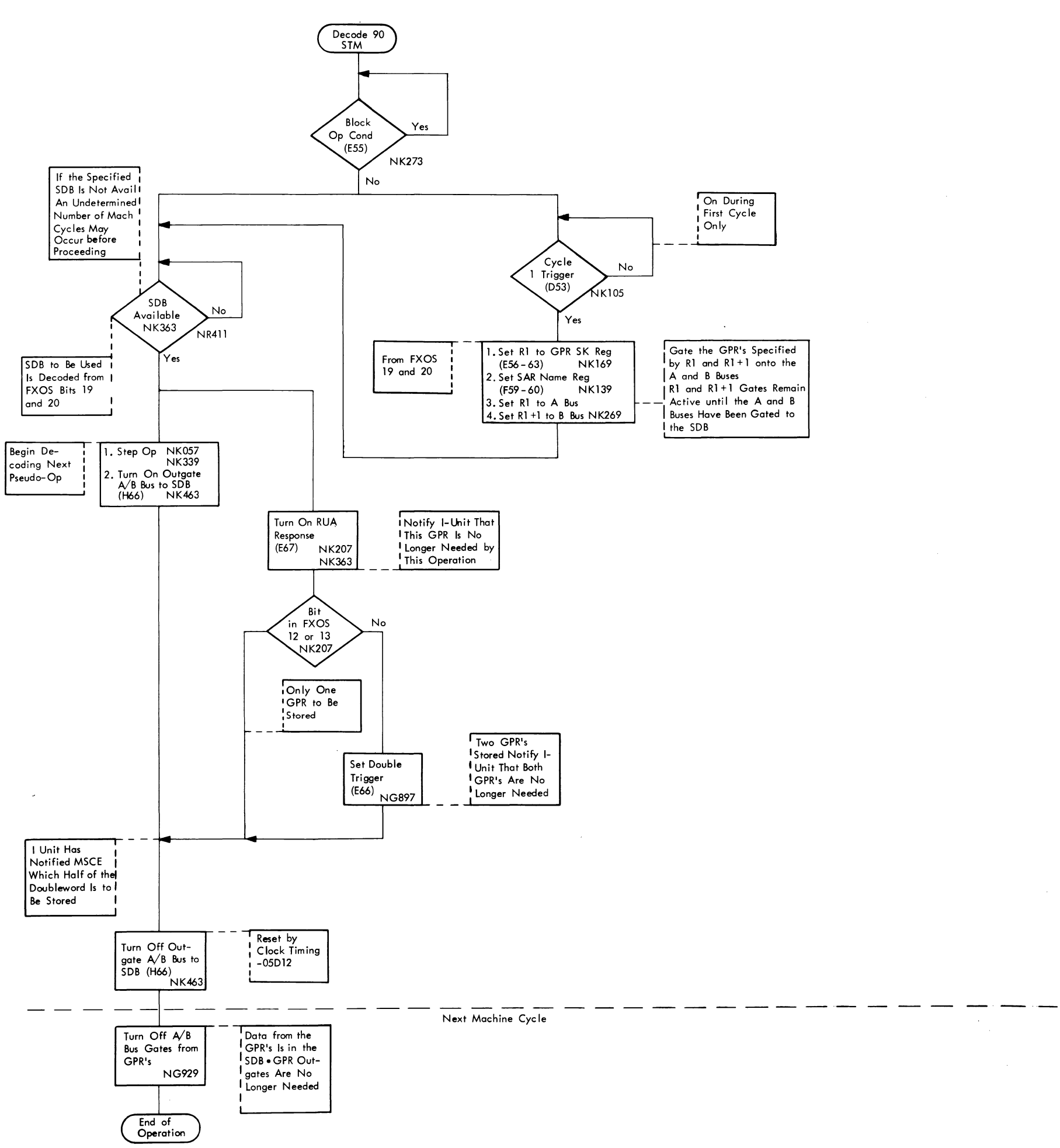
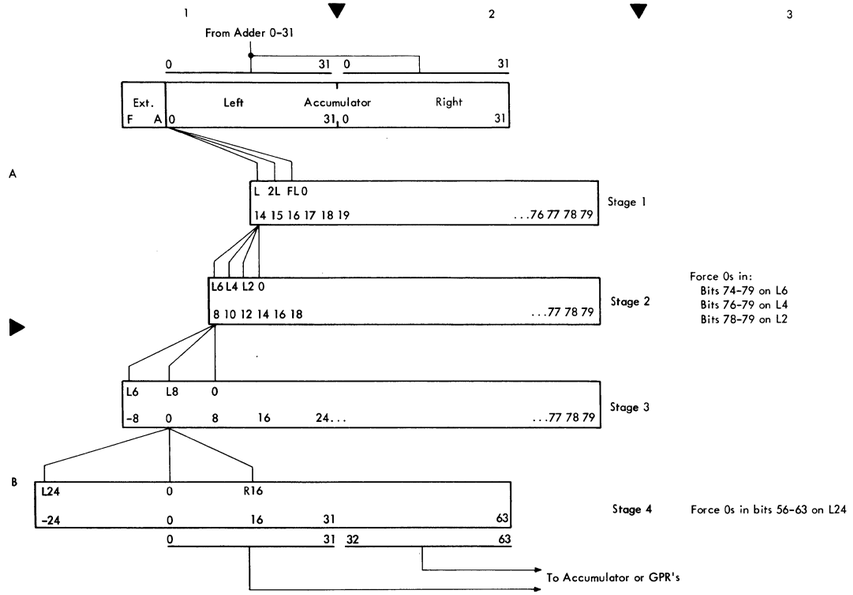


DIAGRAM 5-111. STORE MULTIPLE



Force 0s in:
Bits 74-79 on L6
Bits 76-79 on L4
Bits 78-79 on L2

Force 0s in bits 56-63 on L24

Left Shift

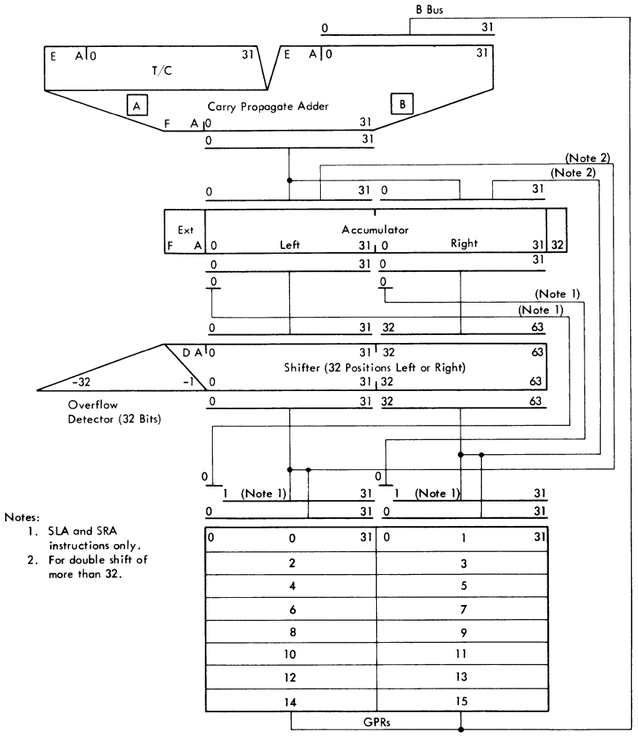
Shift Amount	Wired	Stage 1	Stage 2	Stage 3	Stage 4
0	R16	L0	L0	L16	L0
1	R16	L1	L0	L16	L0
2	R16	L0	L2	L16	L0
3	R16	L1	L2	L16	L0
4	R16	L0	L4	L16	L0
5	R16	L1	L4	L16	L0
6	R16	L0	L6	L16	L0
7	R16	L1	L6	L16	L0
8	R16	L0	L0	L0	L24
9	R16	L1	L0	L0	L24
10	R16	L0	L2	L0	L24
11	R16	L1	L2	L0	L24
12	R16	L0	L4	L0	L24
13	R16	L1	L4	L0	L24
14	R16	L0	L6	L0	L24
15	R16	L1	L6	L0	L24

Shift Amount	Wired	Stage 1	Stage 2	Stage 3	Stage 4
16	R16	L0	L0	L8	L24
17	R16	L1	L0	L8	L24
18	R16	L0	L2	L8	L24
19	R16	L1	L2	L8	L24
20	R16	L0	L4	L8	L24
21	R16	L1	L4	L8	L24
22	R16	L0	L6	L8	L24
23	R16	L1	L6	L8	L24
24	R16	L0	L0	L16	L24
25	R16	L1	L0	L16	L24
26	R16	L0	L2	L16	L24
27	R16	L1	L2	L16	L24
28	R16	L0	L4	L16	L24
29	R16	L1	L4	L16	L24
30	R16	L0	L6	L16	L24
31	R16	L1	L6	L16	L24
32	R16	L2	L6	L16	L24

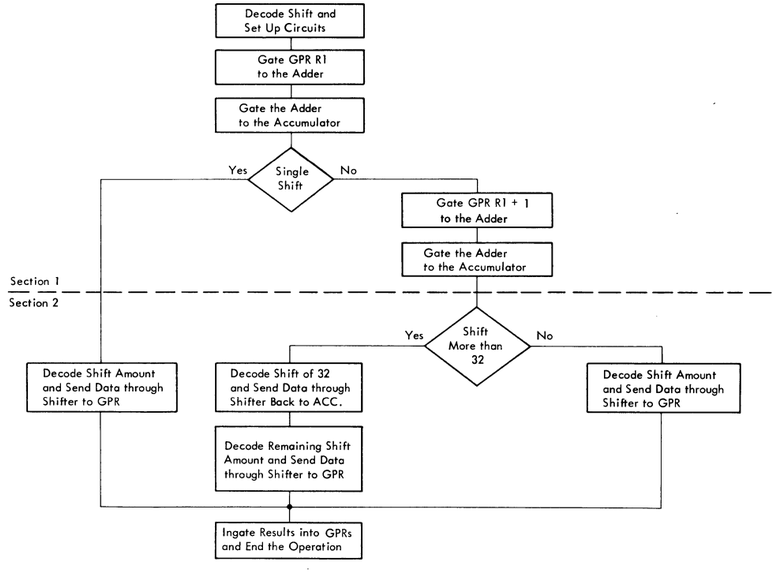
Right Shift

Shift Amount	Wired	Stage 1	Stage 2	Stage 3	Stage 4
0	R16	L2	L6	L8	L0
1	R16	L1	L6	L8	L0
2	R16	L0	L6	L8	L0
3	R16	L1	L4	L8	L0
4	R16	L0	L4	L8	L0
5	R16	L1	L2	L8	L0
6	R16	L0	L2	L8	L0
7	R16	L1	L0	L8	L0
8	R16	L2	L6	L0	L0
9	R16	L1	L6	L0	L0
10	R16	L0	L6	L0	L0
11	R16	L1	L4	L0	L0
12	R16	L0	L4	L0	L0
13	R16	L1	L2	L0	L0
14	R16	L0	L2	L0	L0
15	R16	L1	L0	L0	L0

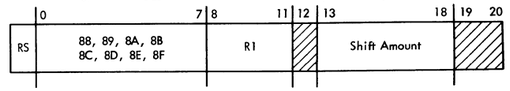
Shift Amount	Wired	Stage 1	Stage 2	Stage 3	Stage 4
16	R16	L2	L6	L8	R16
17	R16	L1	L6	L8	R16
18	R16	L0	L6	L8	R16
19	R16	L1	L4	L8	R16
20	R16	L0	L4	L8	R16
21	R16	L1	L2	L8	R16
22	R16	L0	L2	L8	R16
23	R16	L1	L0	L8	R16
24	R16	L2	L6	L0	R16
25	R16	L1	L6	L0	R16
26	R16	L0	L6	L0	R16
27	R16	L1	L4	L0	R16
28	R16	L0	L4	L0	R16
29	R16	L1	L2	L0	R16
30	R16	L0	L2	L0	R16
31	R16	L1	L0	L0	R16
32	R16	L0	L0	L0	R16



Notes:
1. SLA and SRA instructions only.
2. For double shift of more than 32.



88	SRL	Shift Right Single Logical
89	SLL	Shift Left Single Logical
8A	SRA	Shift Right Single (Arithmetic)
8B	SLA	Shift Left Single (Arithmetic)
8C	SRDL	Shift Right Double Logical
8D	SLDL	Shift Left Double Logical
8E	SRDA	Shift Right Double (Arithmetic)
8F	SLDA	Shift Left Double (Arithmetic)



Objectives:

1. Data in the GPR specified by R1 is shifted right or left the specified amount and replaced in the GPR.
2. Logical shifts move bits 0 to 31 as an unsigned value.
3. Arithmetic shifts move bits 1 to 31 and treat position 0 as a fixed sign.
4. Single shifts move the data from a single GPR.
5. Double shifts move the data from an even odd pair of GPRs as one value.
6. Data path for shift operations is from the GPRs through the carry propagate adder, the accumulator, and shifter back to the GPRs.
7. The shift amount generates gates to control the shifter.
8. Single shift operations are completed in two machine cycles with some ending functions performed in the first cycle of the following operation (Cycle 3).
9. Double shift operations are completed in three machine cycles if the shift amount is more than 32. Some ending functions are performed in the first cycle of the following operation (Cycles 4 or 5).

Condition code remains unchanged for logical shift operations

Condition code is set as follows for arithmetic shift operations:

CC	Condition
0	Result is zero
1	Result is less than zero
2	Result is greater than zero
3	Overflow * (SLA and SLDA only)

* Overflow detection circuits are of the left end of the shifter. (See data flow)

DIAGRAM 5-112. SHIFT INSTRUCTIONS (SHEET 1 OF 2)

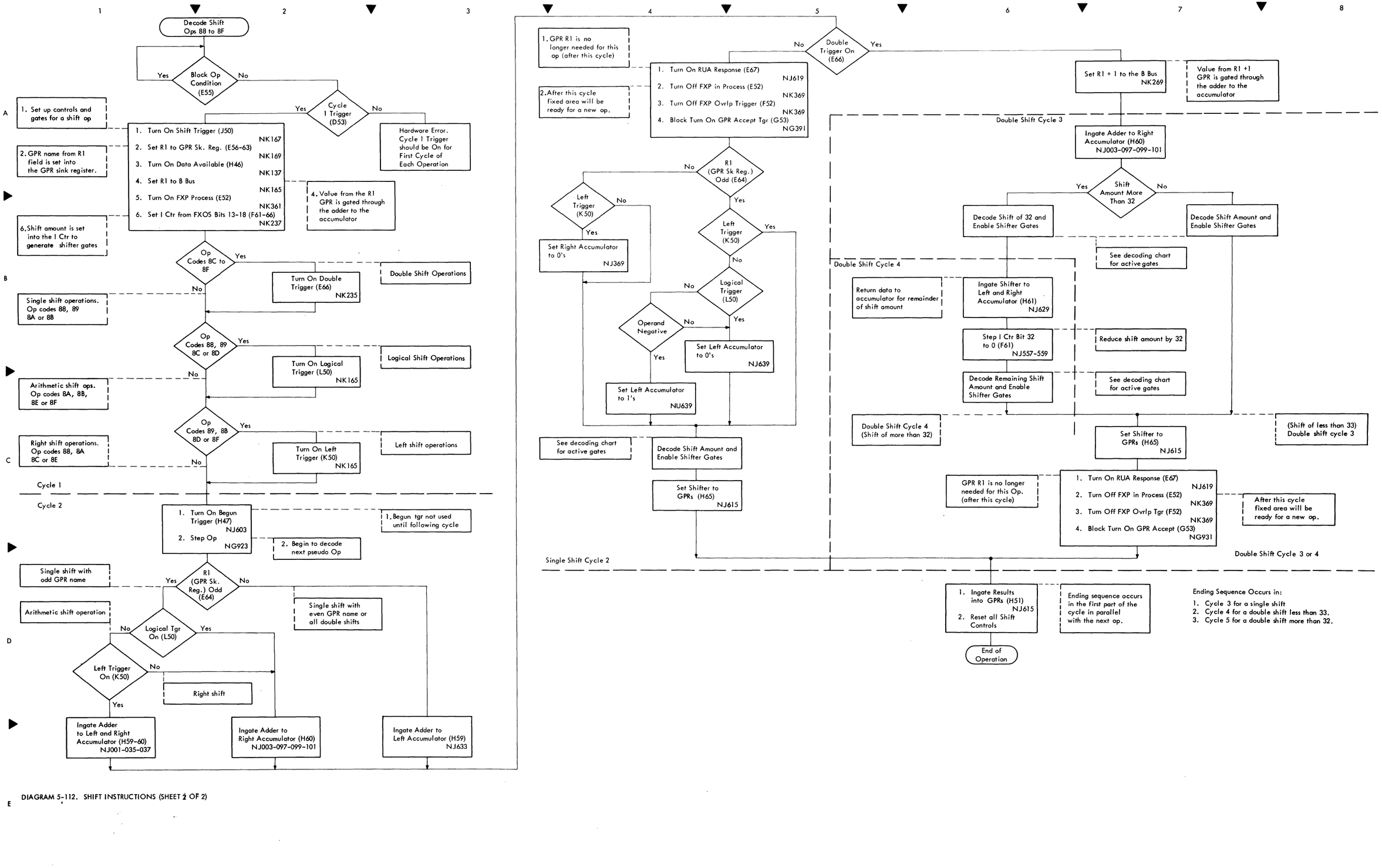


DIAGRAM 5-112. SHIFT INSTRUCTIONS (SHEET 2 OF 2)

Objectives:

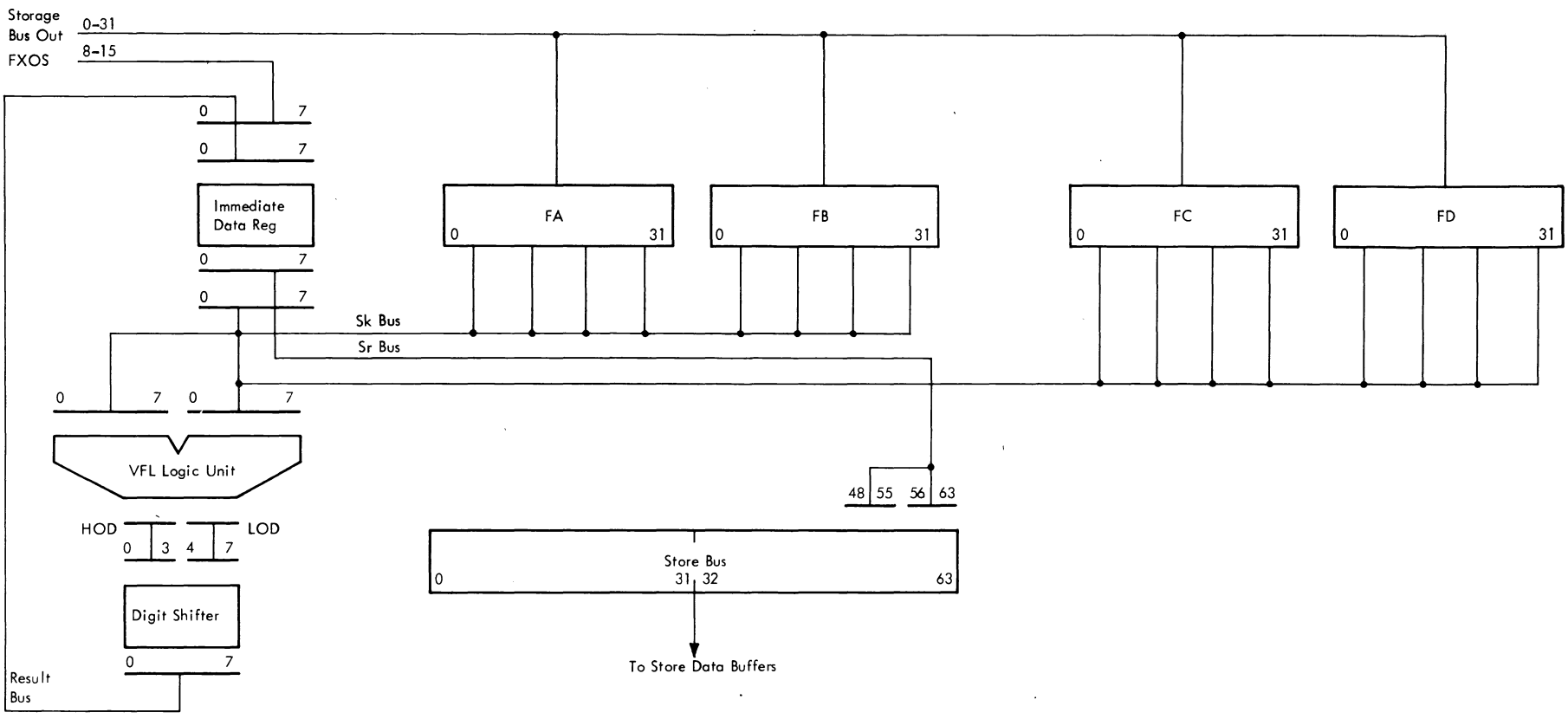
AND, OR, Exclusive OR
NI, OI, XI

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
FXOS Format	SI	94,96,97							I Field								FXB Name			SAR Name	

NI,OI, or XI Condition Codes:
0-Result is 0
1-Result is not 0
2-(Not applicable)
3-(Not applicable)

1. Specified FXB byte is ANDed (94), ORed (96) or Exclusive ORed (97) with immediate data field in VFL Logic Unit.
2. FXB data and immediate data are gated to VFL Logic Unit over opposite busses (Sk or Sr).
3. Results are outgated to specified SDB.

Data Flow



Flow Chart

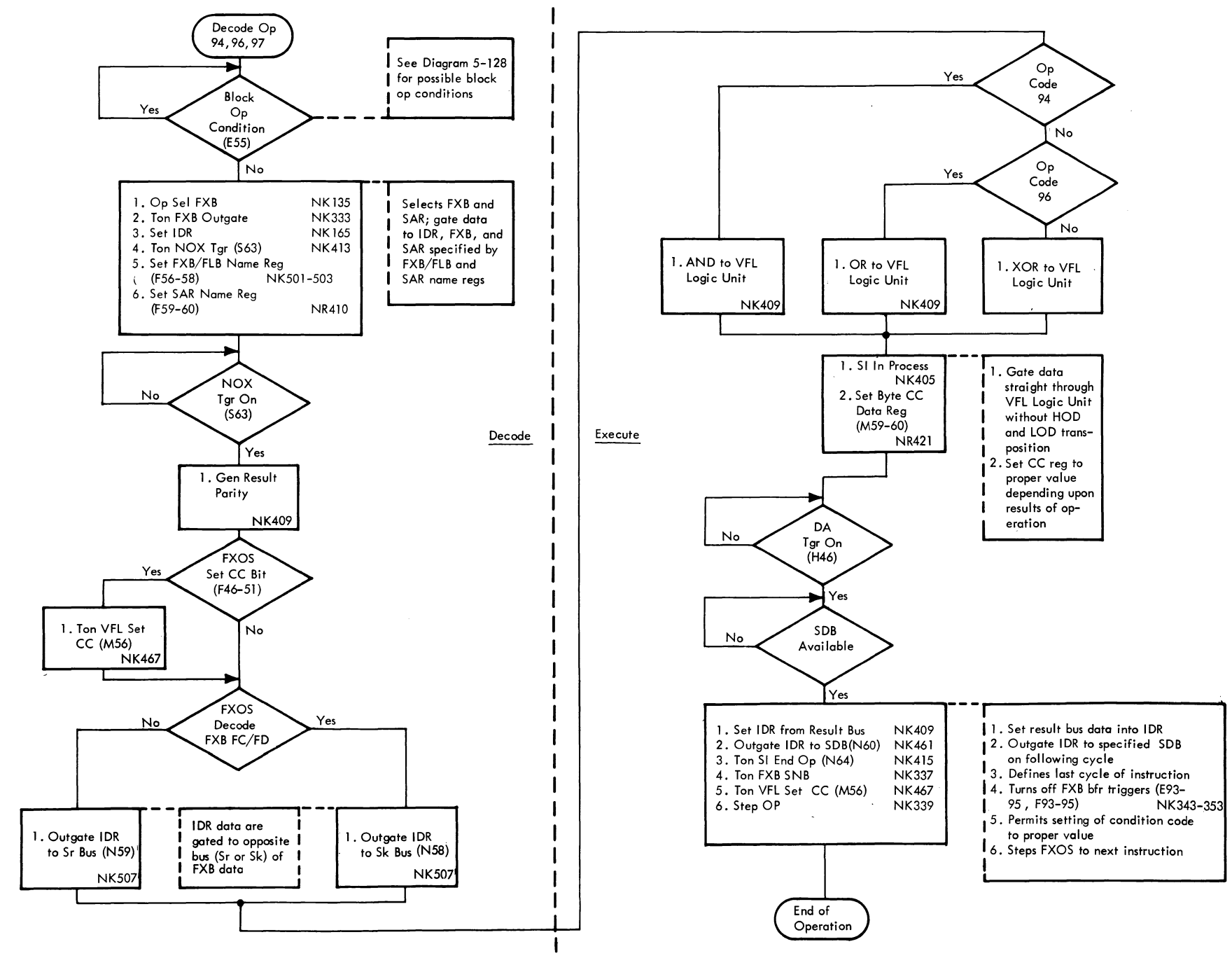
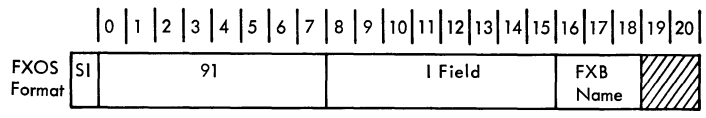


DIAGRAM 5-113. NI, OI, AND XI INSTRUCTIONS

Objectives:

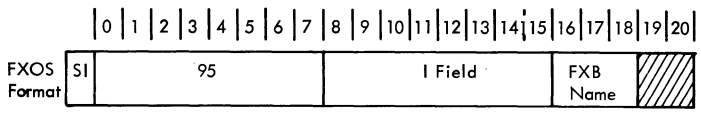
Test Under Mask
TM



1. State of bits in FXB byte selected by mask in immediate data field is used to set condition code.
2. Mask bit of 1 indicates corresponding FXB bit is to be tested; mask bit of 0 indicates corresponding bit is to be ignored.
3. FXB data and immediate data are gated to VFL Logic Unit over opposite busses (Sk or Sr).

TM Condition Code:
- 0 - Selected bits and mask are all 0
- 1 - Selected bits are mixed 0 and 1
- 2 - (Not applicable)
- 3 - Selected bits are all 1

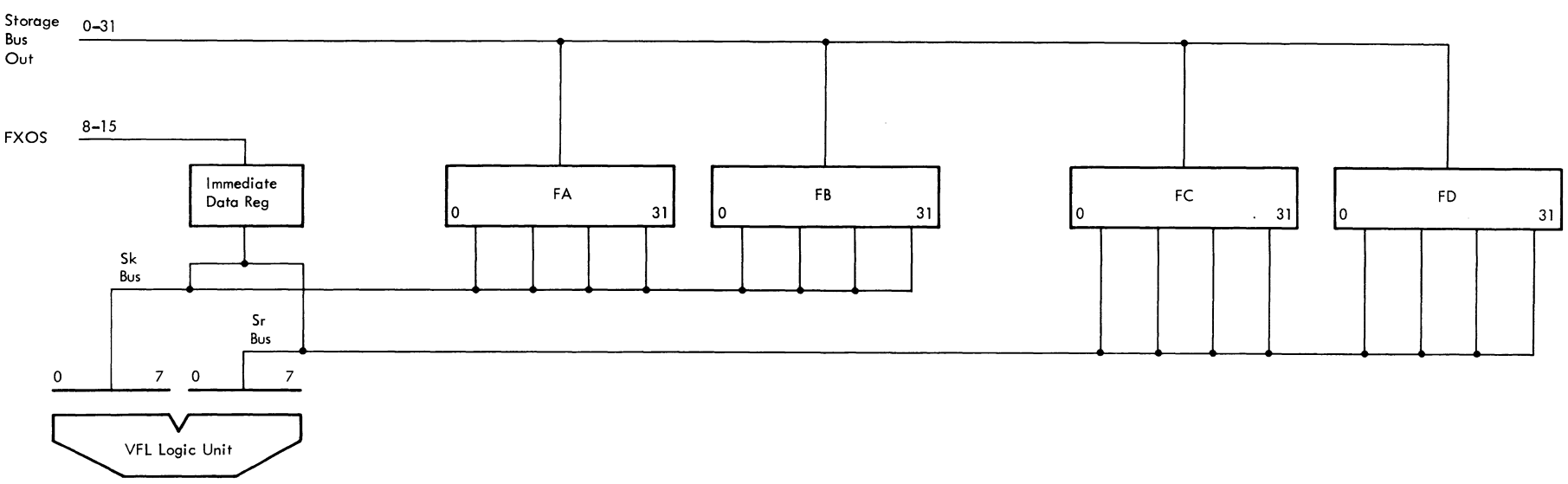
Compare Logical
CLI



1. Specified FXB byte is compared with immediate data field in VFL Logic Unit.
2. FXB data and immediate data are gated to VFL Logic Unit over opposite busses (Sk or Sr).
3. Results of operation are indicated in condition code.

CLI Condition Code:
- 0 - Operands are equal
- 1 - FXB data is low
- 2 - FXB data is high
- 3 - (Not applicable)

TM Data Flow



CLI Data Flow

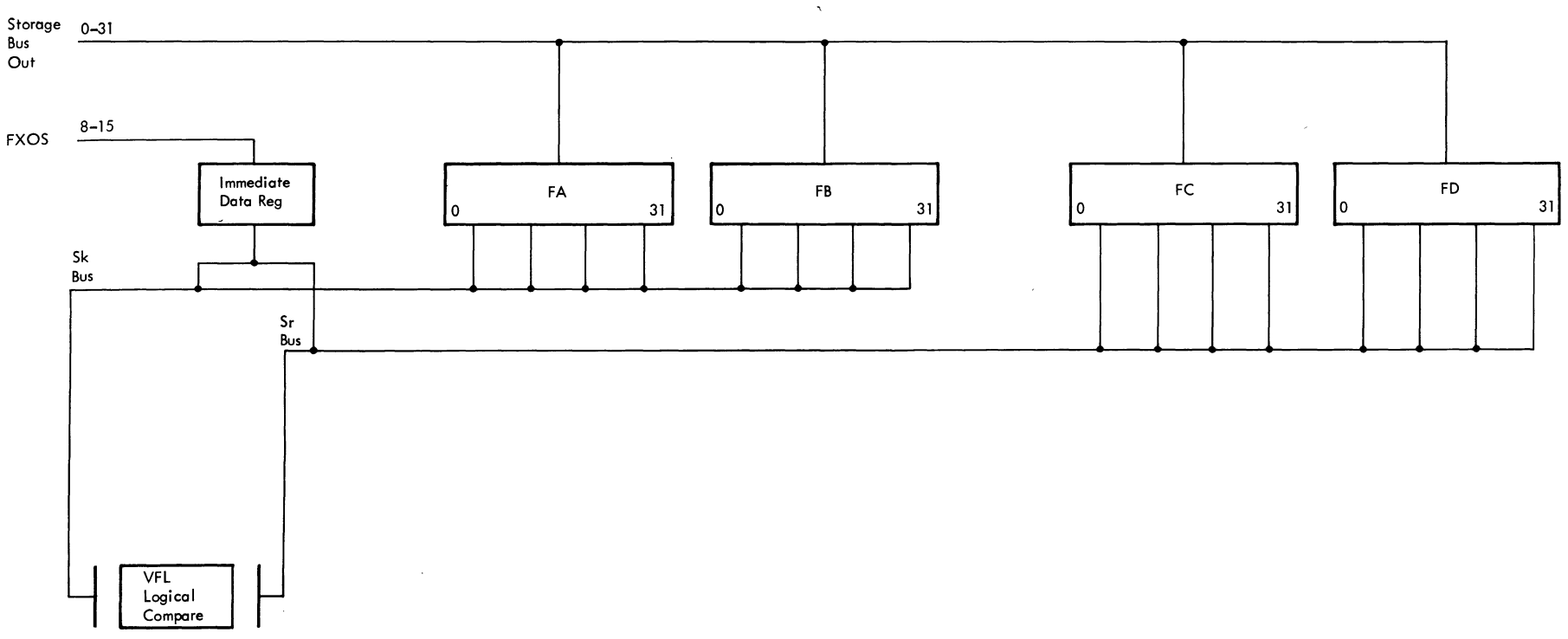


DIAGRAM 5-114. TM AND CLI INSTRUCTIONS (SHEET 1 OF 2)

Flow Chart

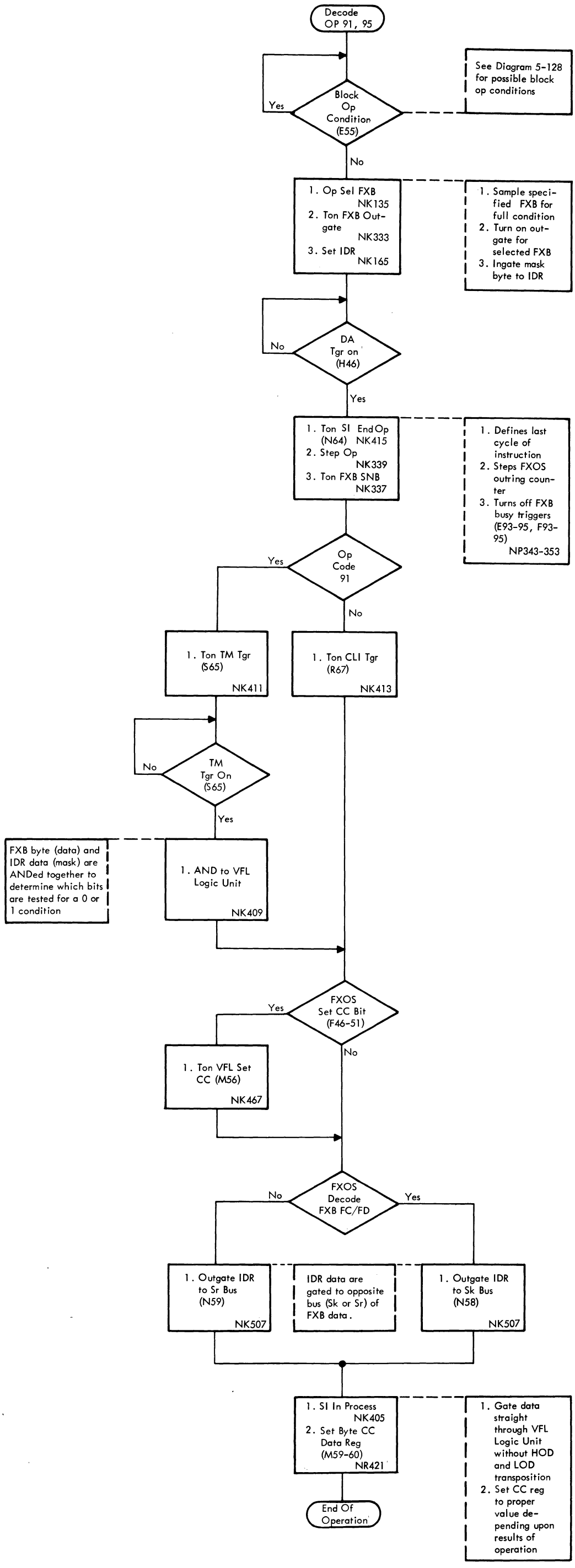
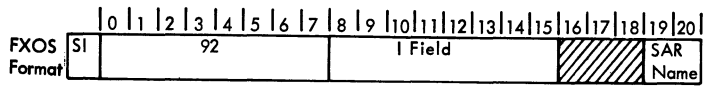


DIAGRAM 5-114. TM AND CLI INSTRUCTIONS (SHEET 2 OF 2)

Objectives:

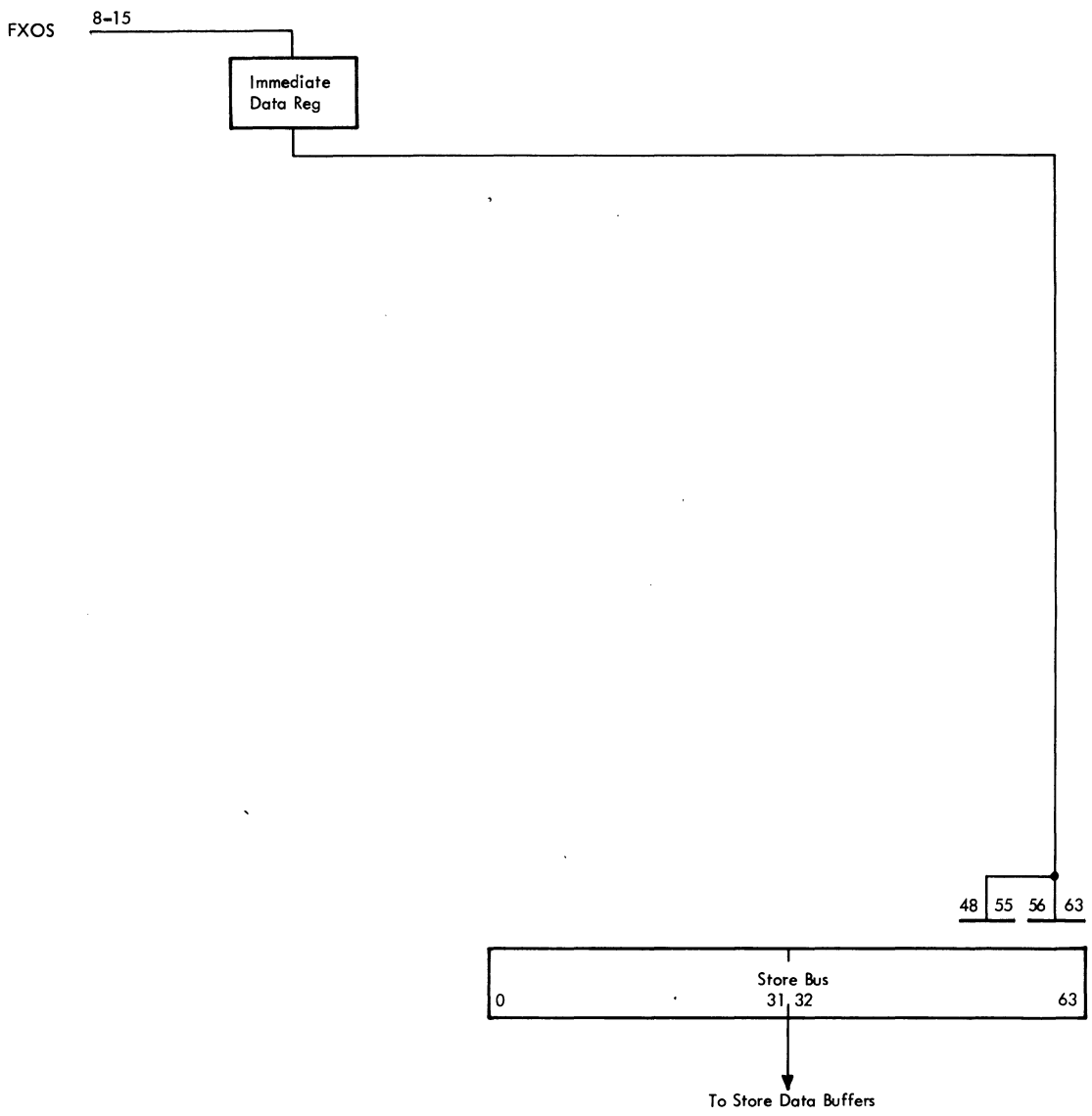
Move
MVI



1. IDR data are moved to a specified SDB
2. IDR data are gated onto store bus positions 48 through 55 and 56 through 63.

MVI Condition Code:
Code remains unchanged

Data Flow



Flow Chart

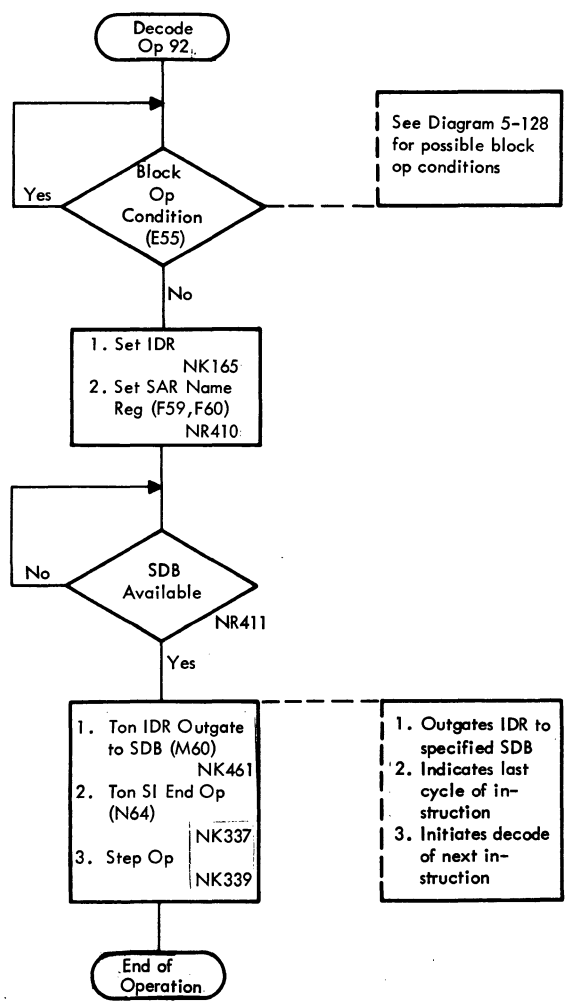
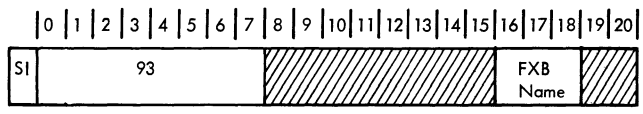


DIAGRAM 5-115. MVI INSTRUCTION

Objectives:

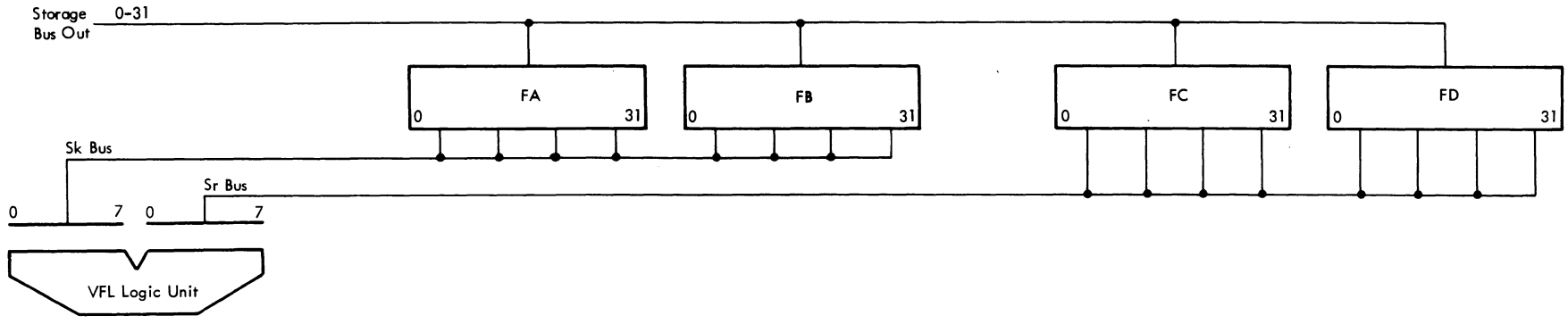
Test and Set
TS



1. VFLEU only performs part of instruction. MSCE performs other part. MSCE outgates a word from storage and (1) sends it to VFLEU, and (2) sets specified byte of word to all 1s and places it back in storage.
2. Specified byte is gated from FXB to VFL Logic Unit where bit 0 of byte is tested for a 0 or 1 condition. Condition code is set accordingly.

TS Condition Code:
- 0 - Bit 0 of specified byte is zero
- 1 - Bit 1 of specified byte is one
- 2 - (Not applicable)
- 3 - (Not applicable)

Data Flow



Flow Chart

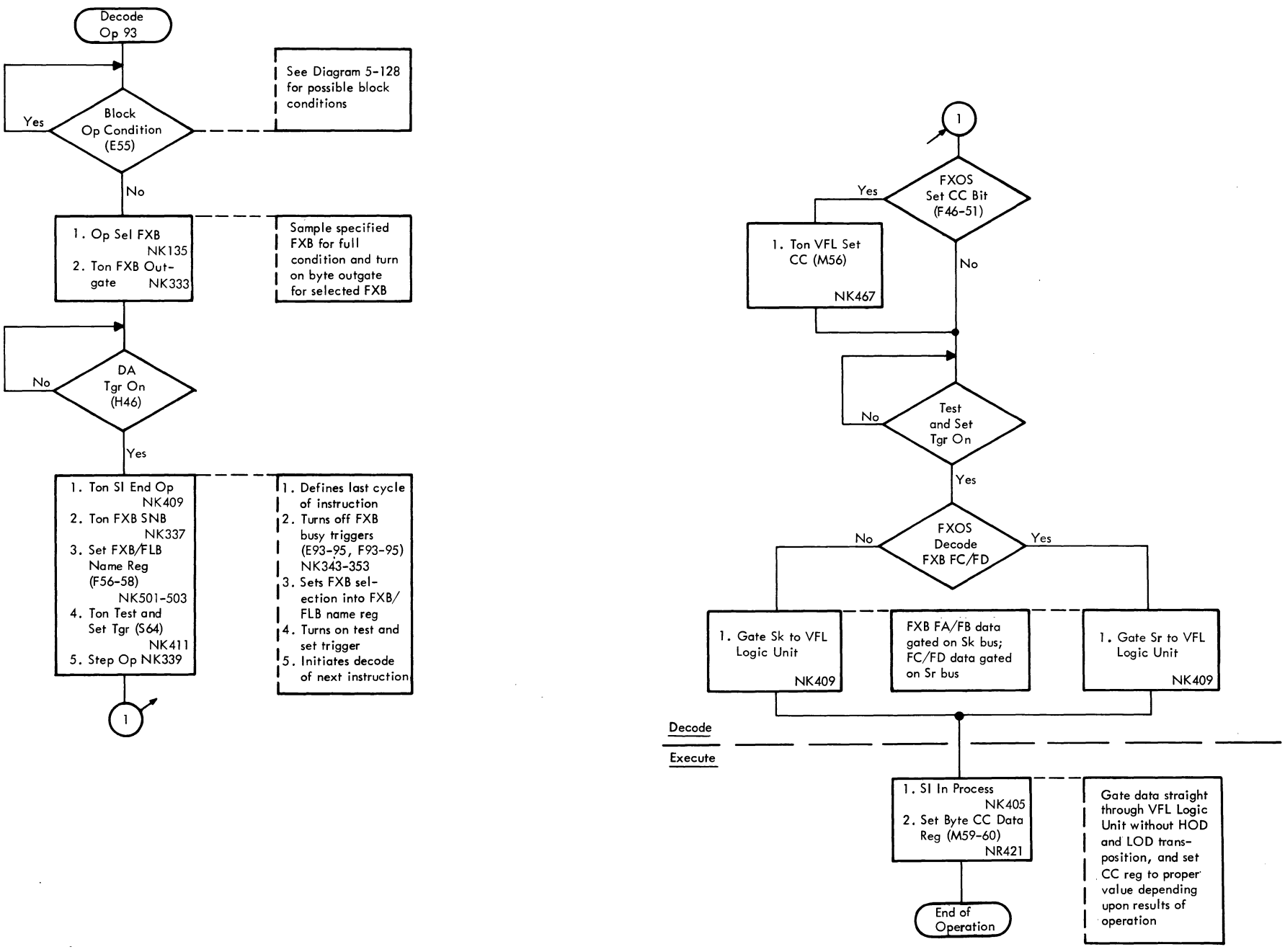
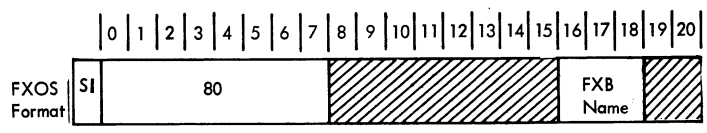


DIAGRAM 5-116. TS INSTRUCTION

Objective:

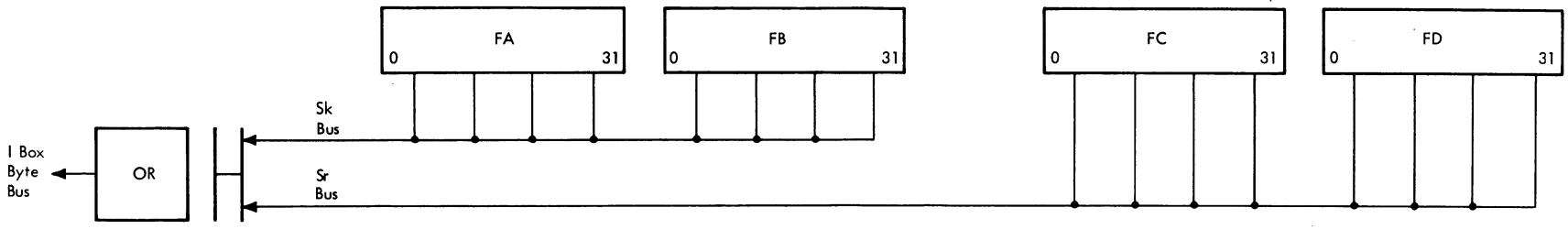
Set System Mask
SSM



1. Byte in specified FXB replaces system mask bits in current PSW.
2. FXB byte is sent to I-Box via byte bus.
3. FX complete signal is sent to I-Box causing current PSW to ingate new mask byte.

SSM Condition Code:
Code remains unchanged

Data Flow



Flow Chart

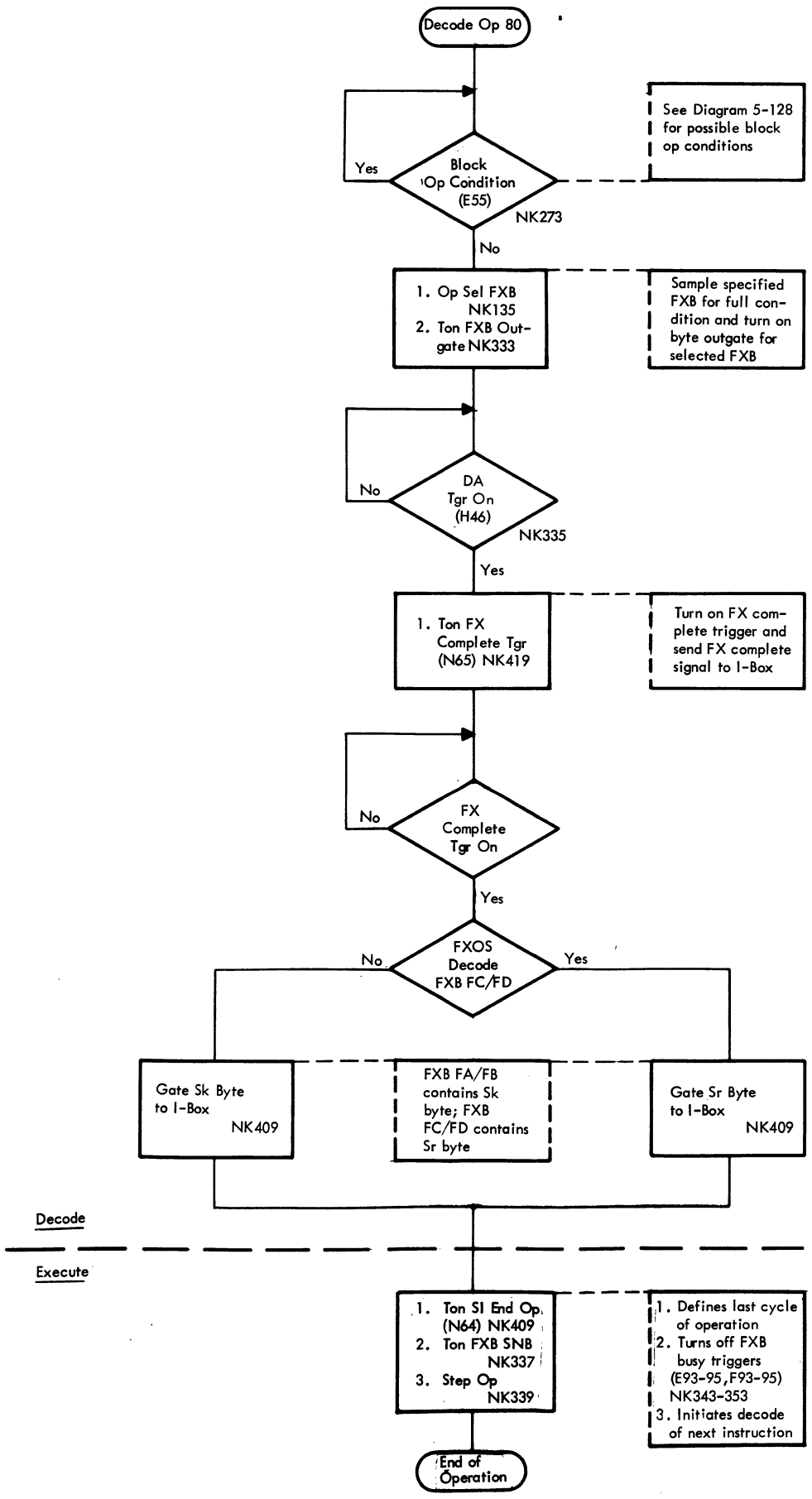
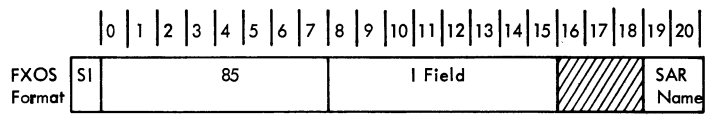


DIAGRAM 5-117. SSM INSTRUCTION

Objectives:

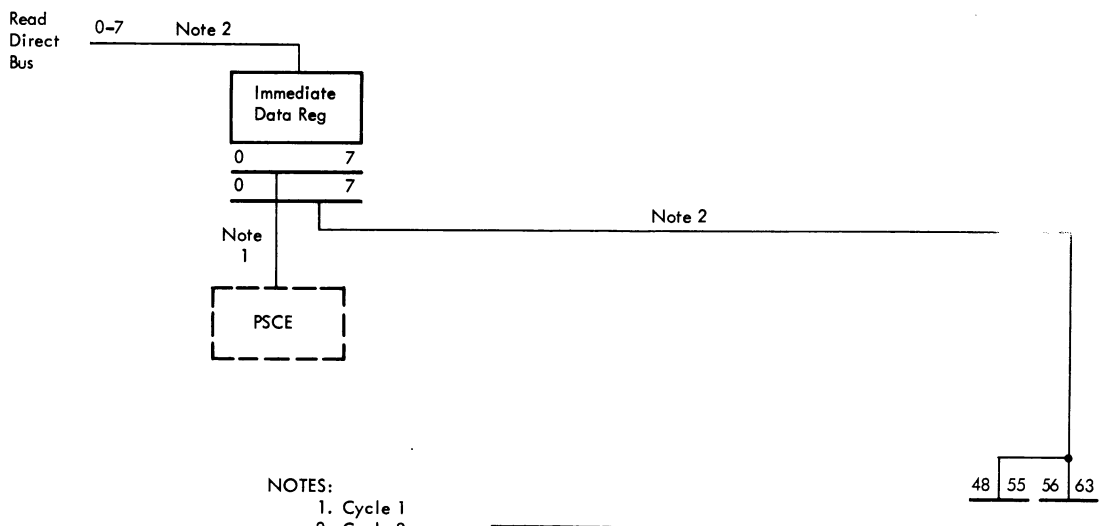
Read Direct
RDD



1. IDR data are made available as signal-out timing signals; direct-in data are placed in IDR for subsequent transfer to specified SDB.
2. IDR data are gated to single shots and direct-in data are set into IDR via result bus after single shots are activated.
3. Direct-in data are transferred from IDR to specified SDB.

RDD Condition Code:
Code remains unchanged

Data Flow



NOTES:
1. Cycle 1
2. Cycle 2

Flow Chart

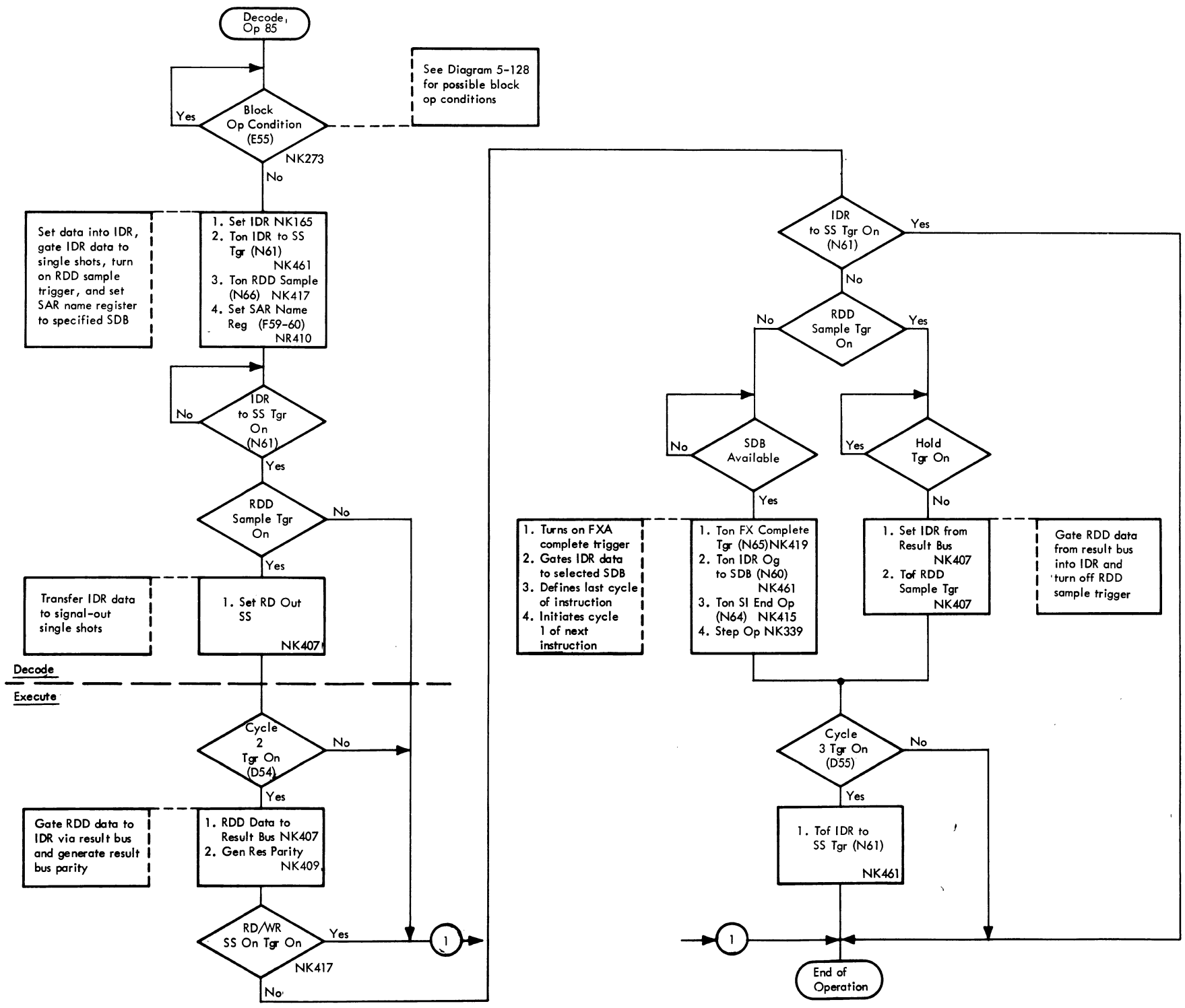
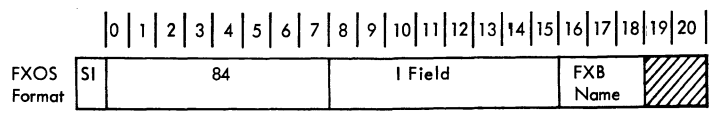


DIAGRAM 5-118. RDD INSTRUCTION

Objectives:

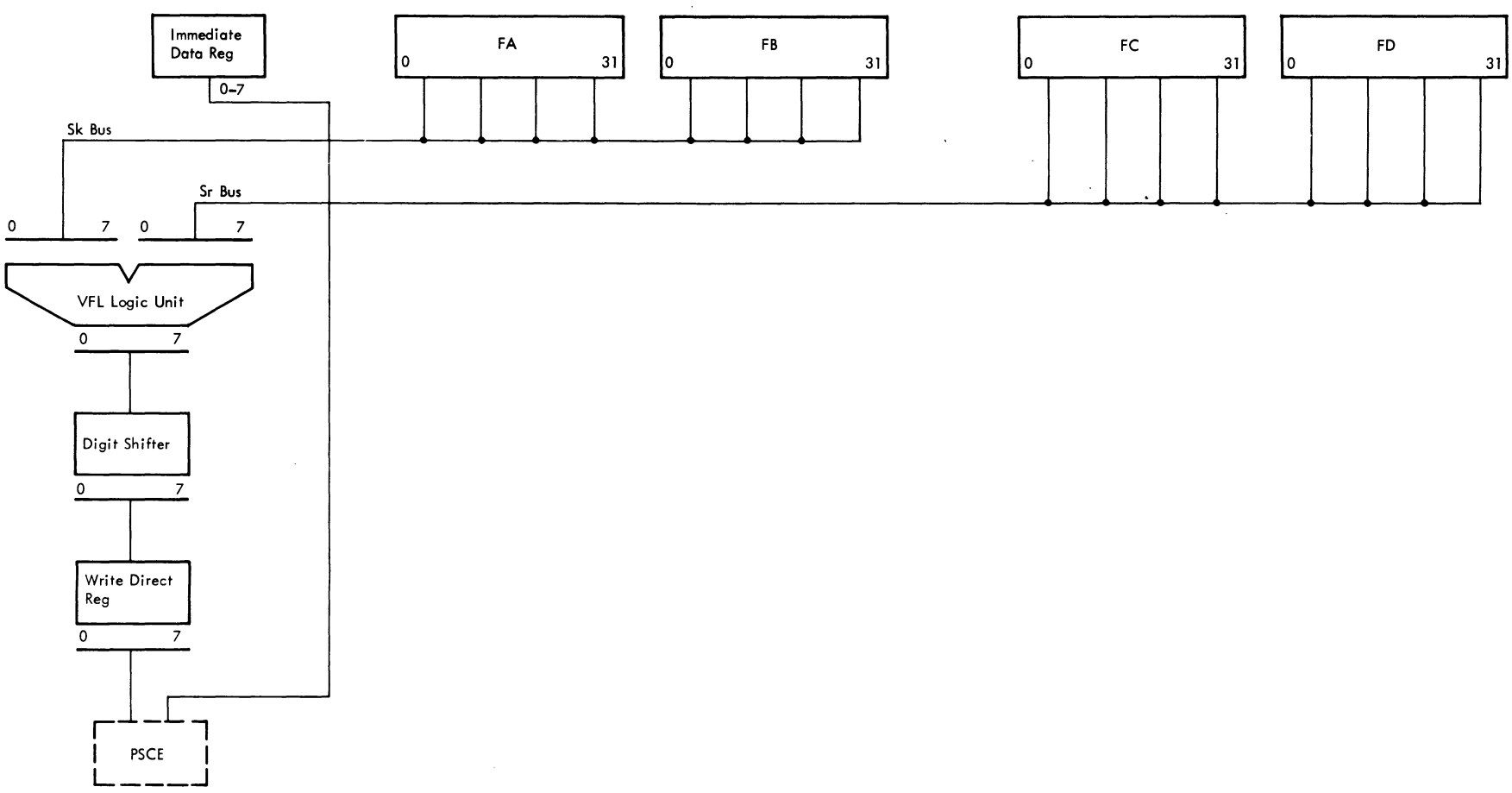
Write Direct
WRD



1. A byte in the specified FXB is made available as a set of direct-out static signals and IDR data are made available as signal-out timing signals.
2. When instruction is decoded, IDR data are gated to single shots.
3. After activation of single shots, FXB data are gated to write direct register.

WRD Condition Code:
Code remains unchanged

Data Flow



Flow Chart

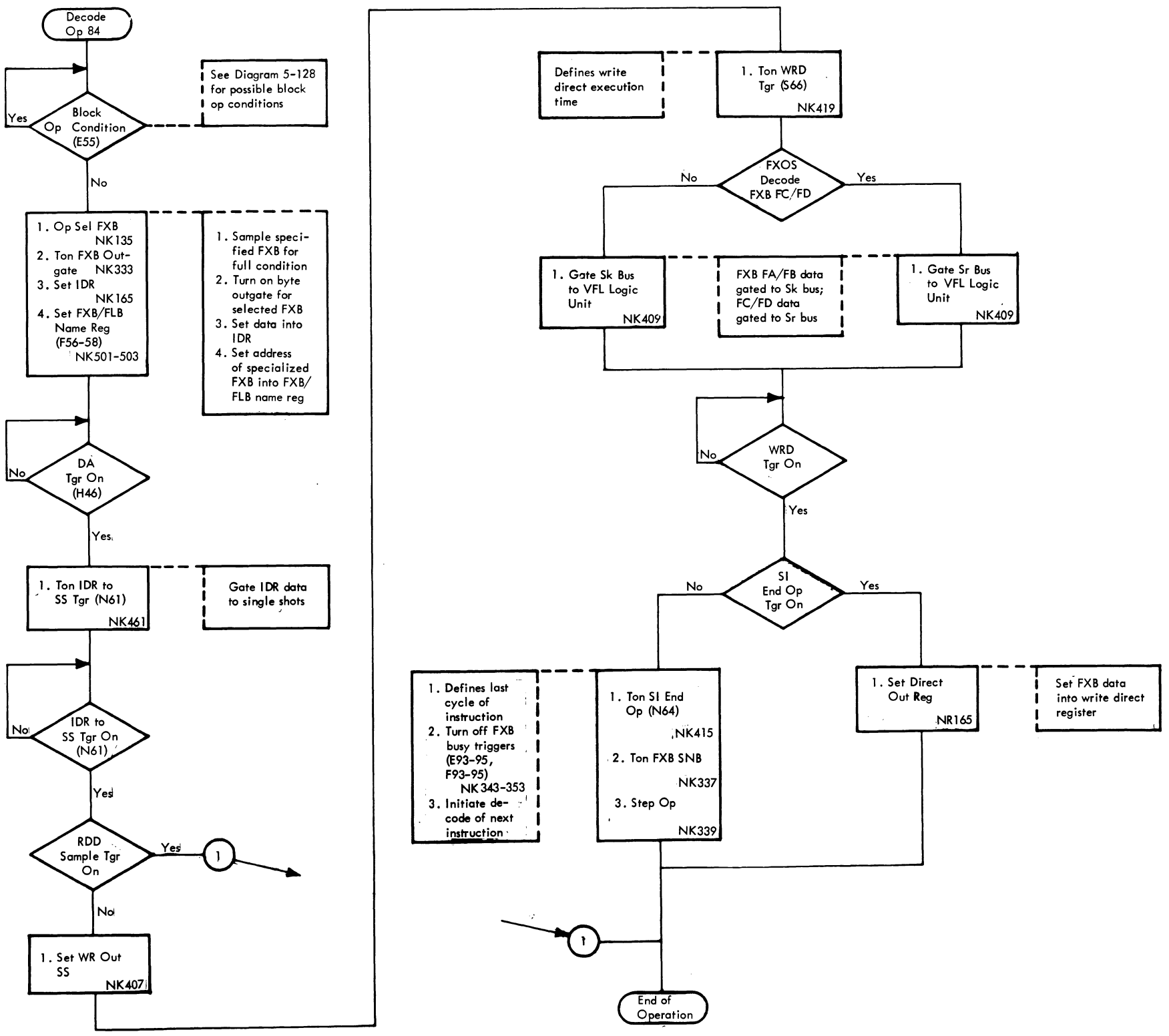


DIAGRAM 5-119. WRD INSTRUCTION

Objectives:

AND, OR, Exclusive OR, Move Numerics, Move Zones
NC, OC, XC, MVN, MVZ

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
SS	D4, D6, D7, D1, D3								FLB Source			P	O	S	E	FLB Sink			SAR Name	

1. Sink and source doublewords are gated from FLBs to FXBs FA/FB and FC/FD, respectively.
2. Sink and source bytes are gated to the VFL logic unit and are processed as follows:
AND (NC) - Logical product is obtained between the sink and source bytes.
OR (OC) - Logical sum is obtained between the sink and source bytes.
Exclusive OR (XC) - Module-two sum is obtained between the sink and source bytes.
Move Numerics (MVN) - LOD of source byte is placed in LOD position of sink byte.
Move Zones (MVZ) - HOD of source byte is placed in HOD of sink byte.
3. Resultant bytes are placed in specified positions of sink FXB (FA/FB). When FA/FB are filled with processed bytes, FA/FB data are stored in a specified SDB.

Simplified Flow

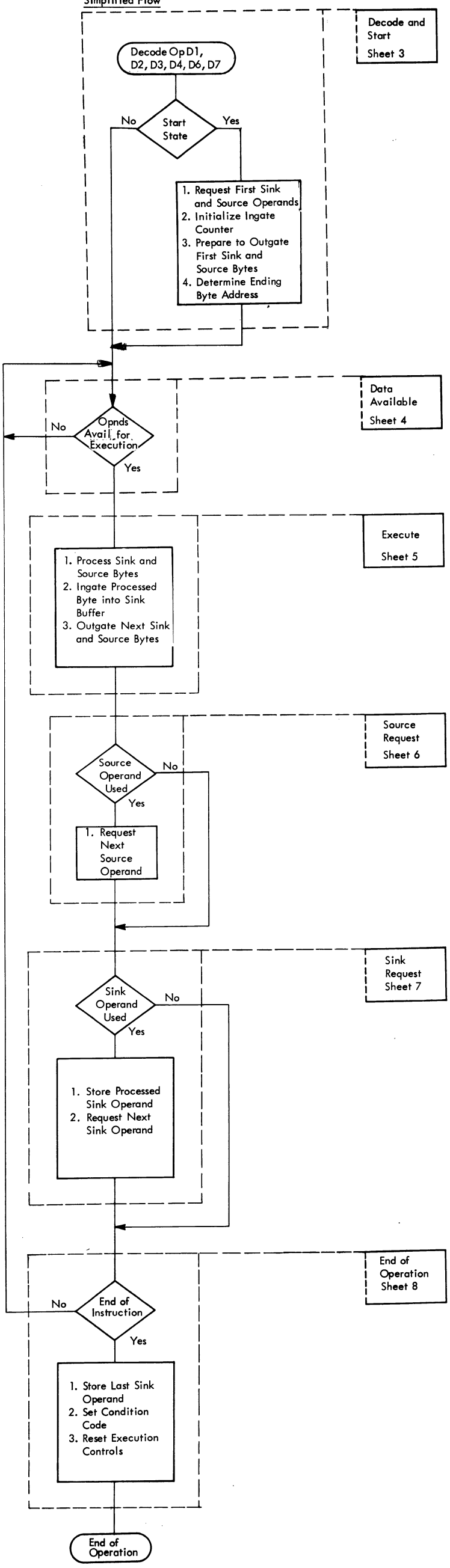
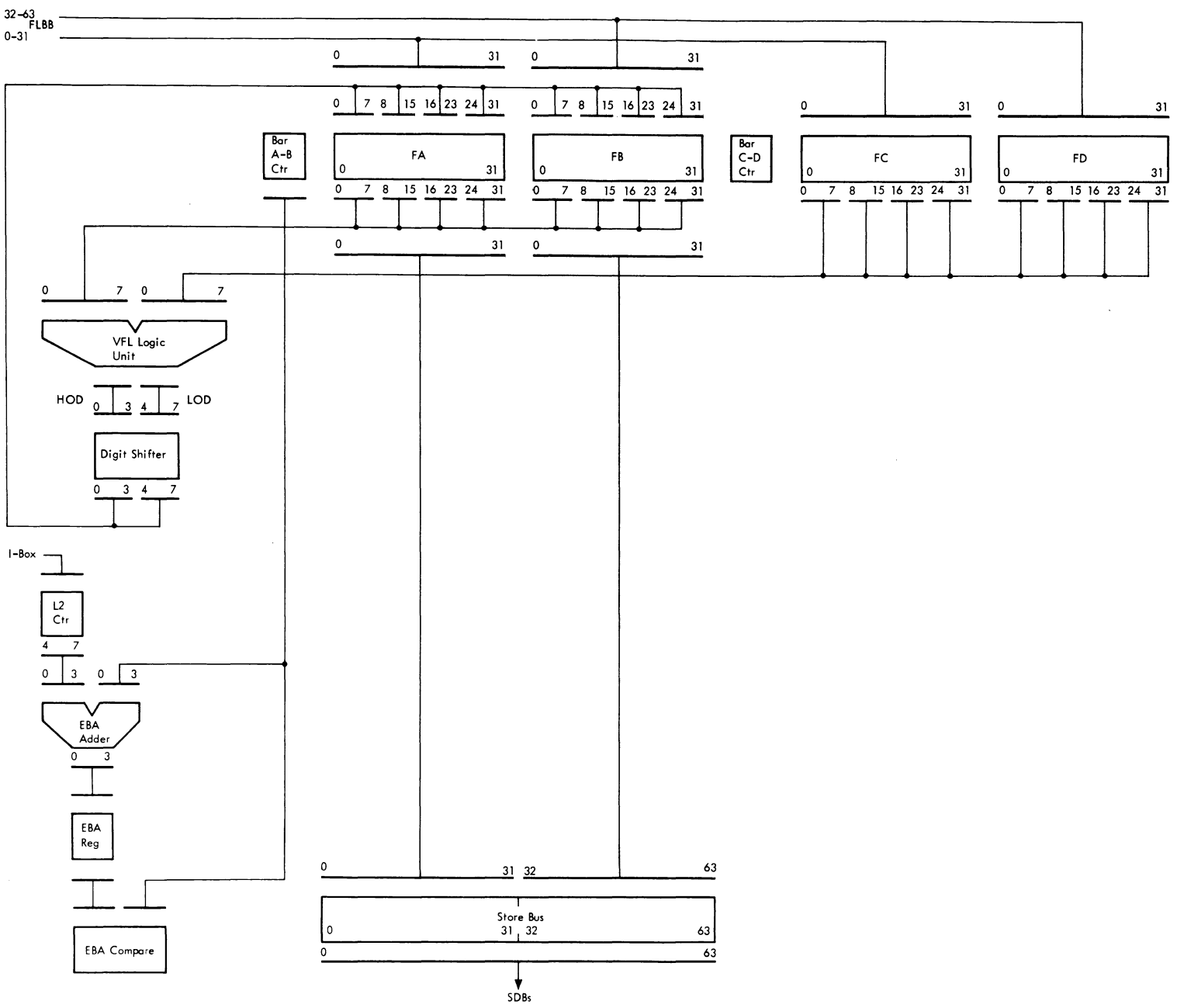


DIAGRAM 5-120. NC, OC, XC, MVN, AND MVZ INSTRUCTIONS (SHEET 1 OF 8)

MVN and MVZ Data Flow



NC, OC and XC Data Flow

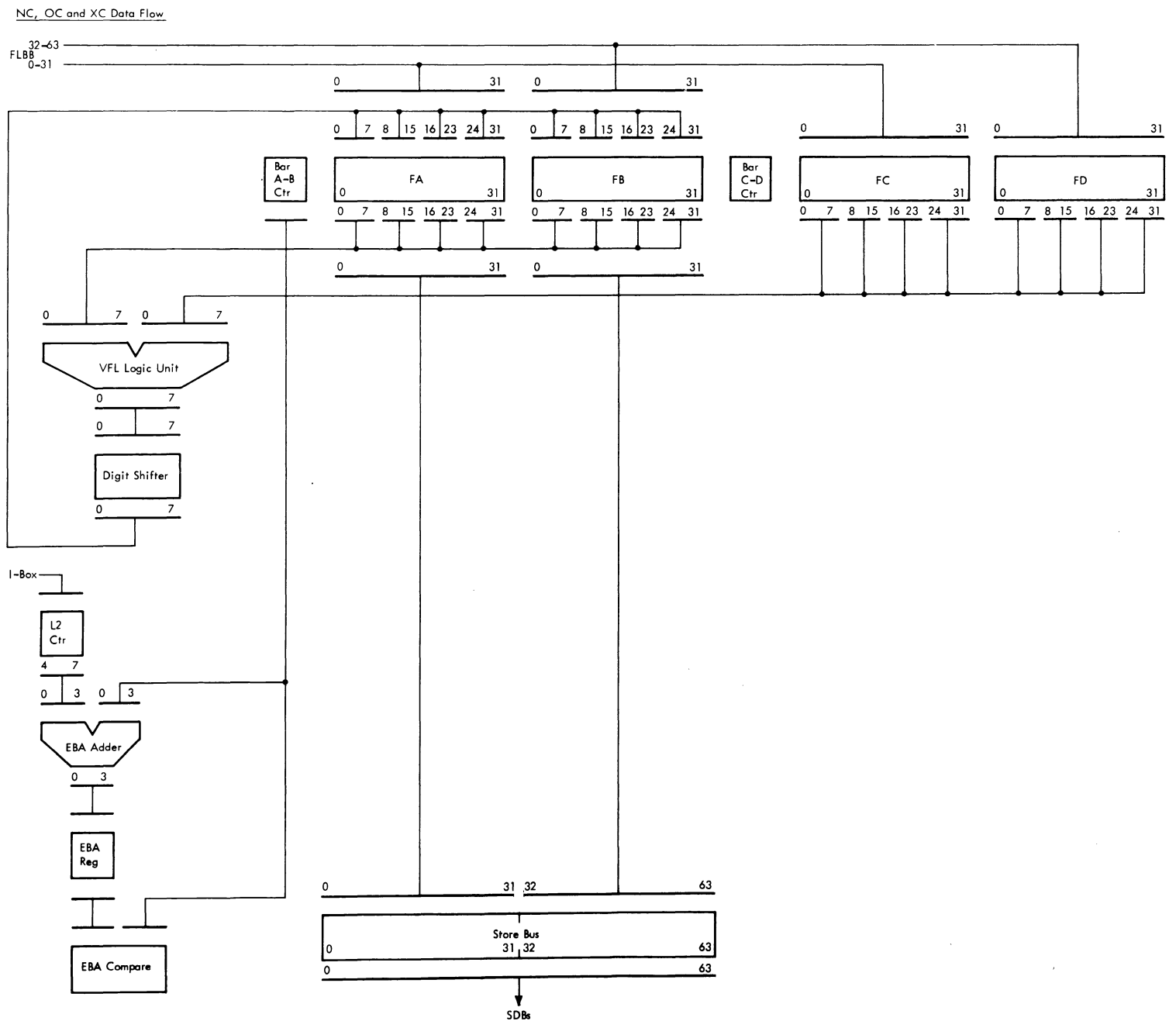


DIAGRAM 5-120. NC, OC, XC, MVN, AND MVZ INSTRUCTIONS (SHEET 2 OF 8)

Decode and Start

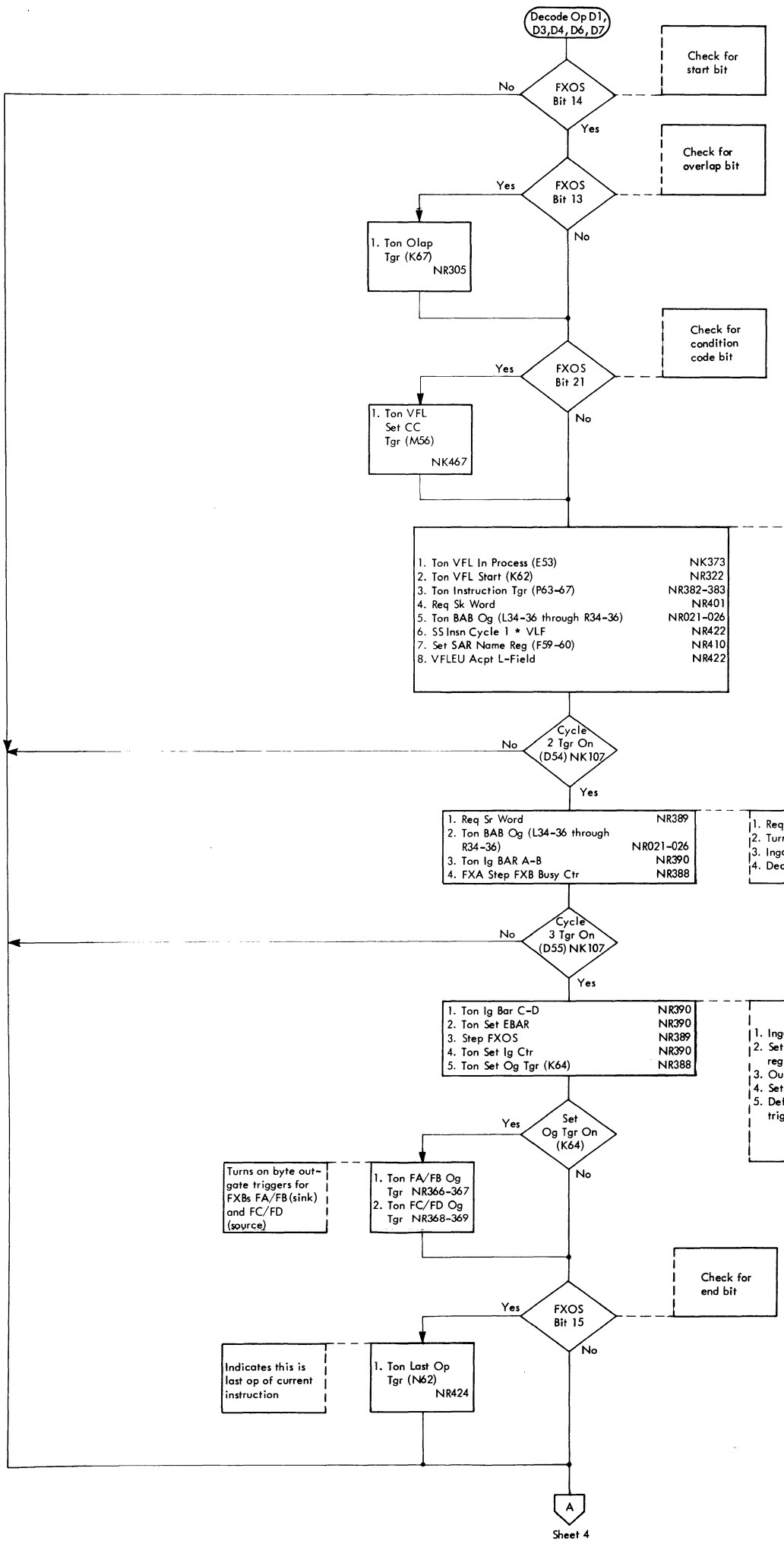


DIAGRAM 5-120. NC, OC, XC, MVN, AND MVZ INSTRUCTIONS (SHEET 3 OF 8)

Data Available

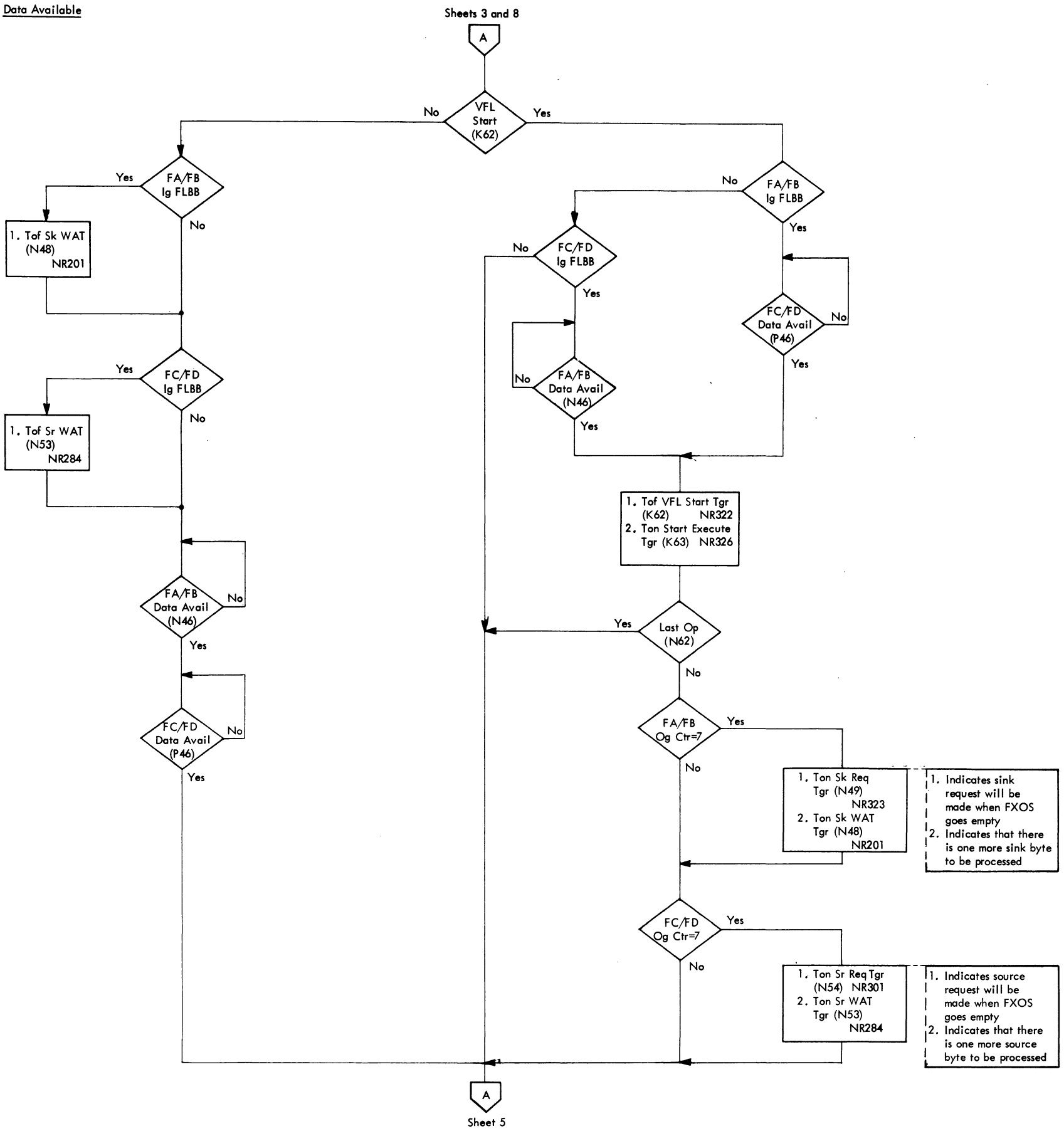


DIAGRAM 5-120. NC, OC, XC, MVN, AND MVZ INSTRUCTIONS (SHEET 4 OF 8)

Execute

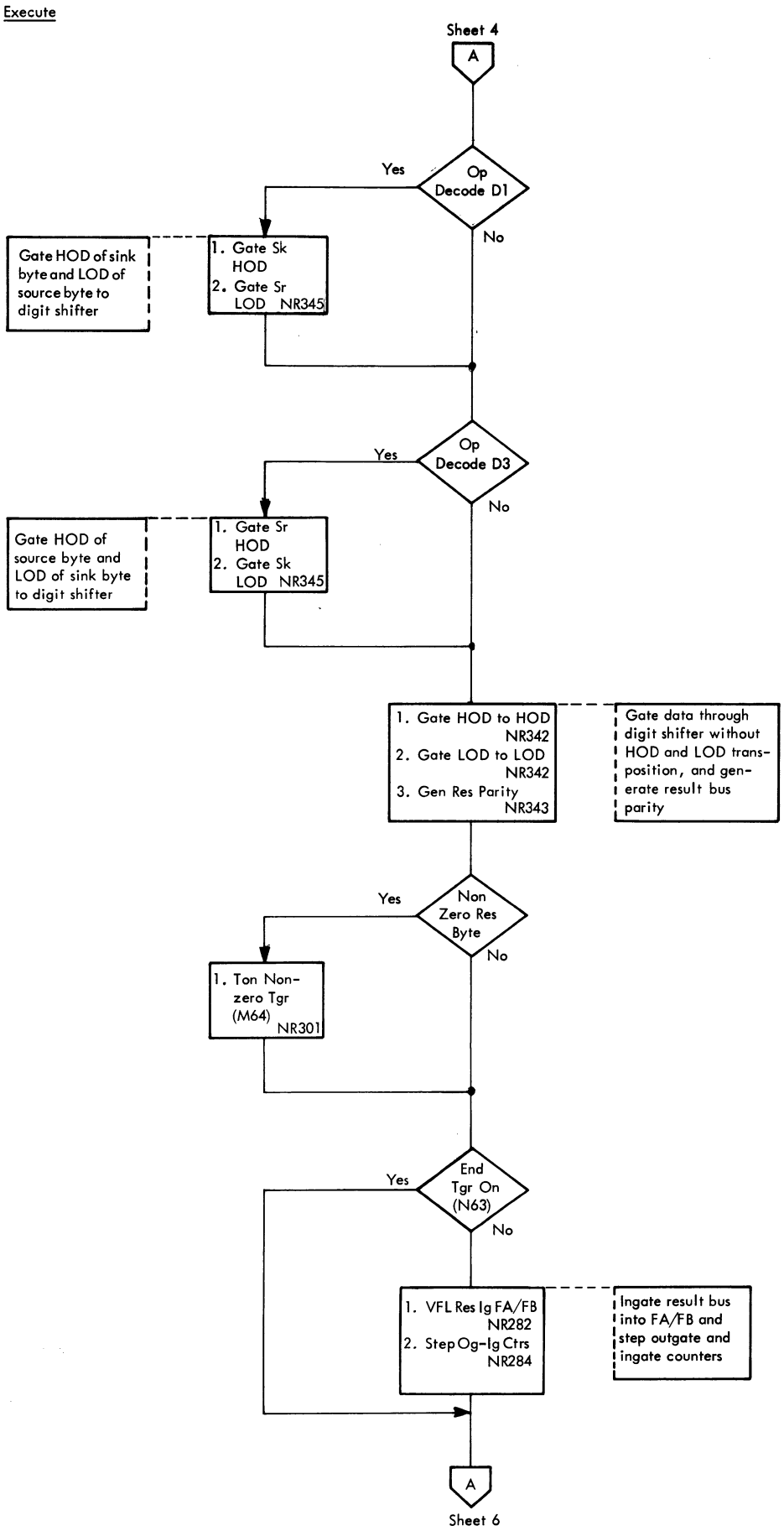


DIAGRAM 5-120. NC, OC, XC, MVN, AND MVZ INSTRUCTIONS (SHEET 5 OF 8)

Sink Request

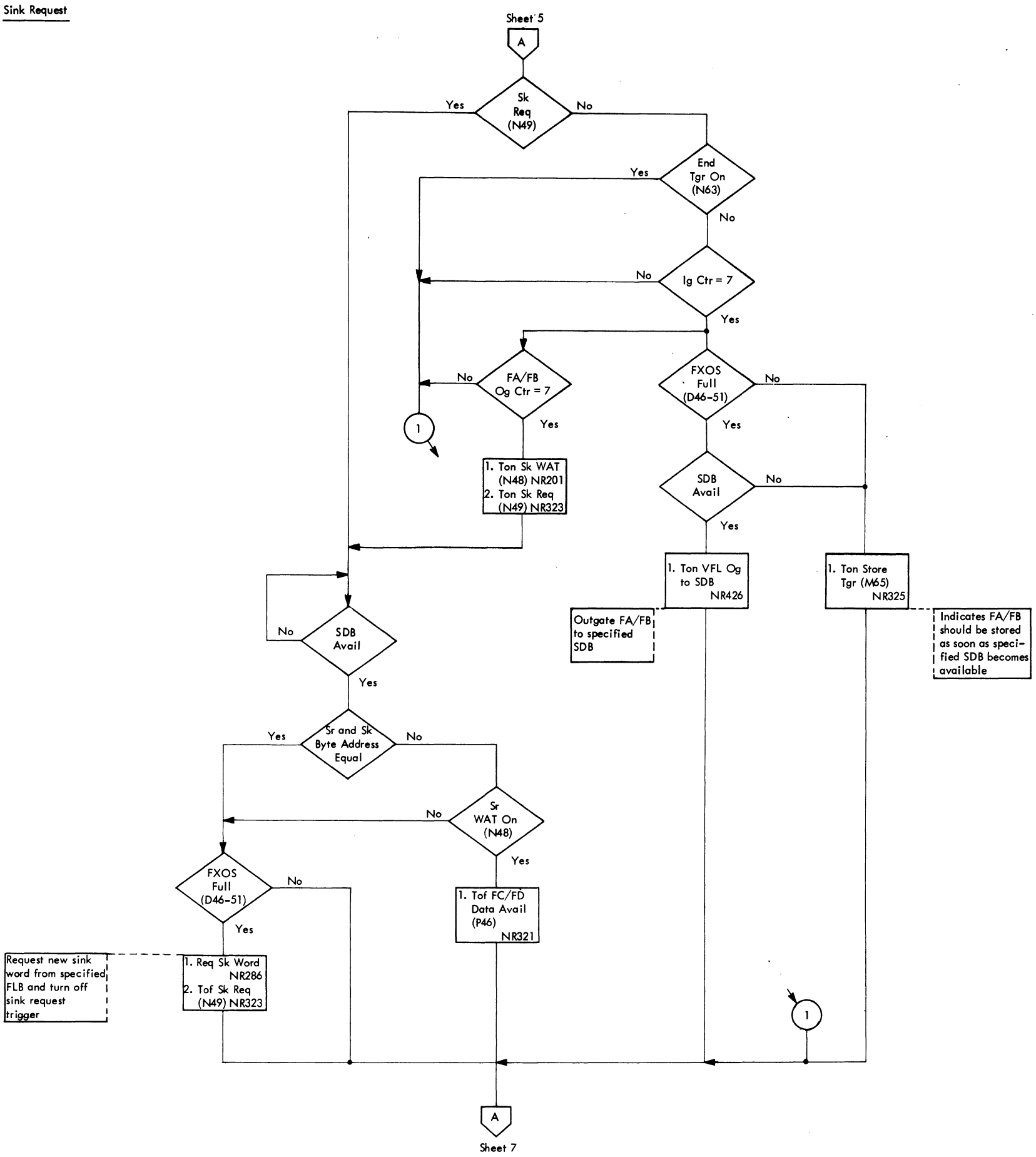


DIAGRAM 5-120. NC, OC, XC, MVN, AND MVZ INSTRUCTIONS (SHEET 6 OF 8)

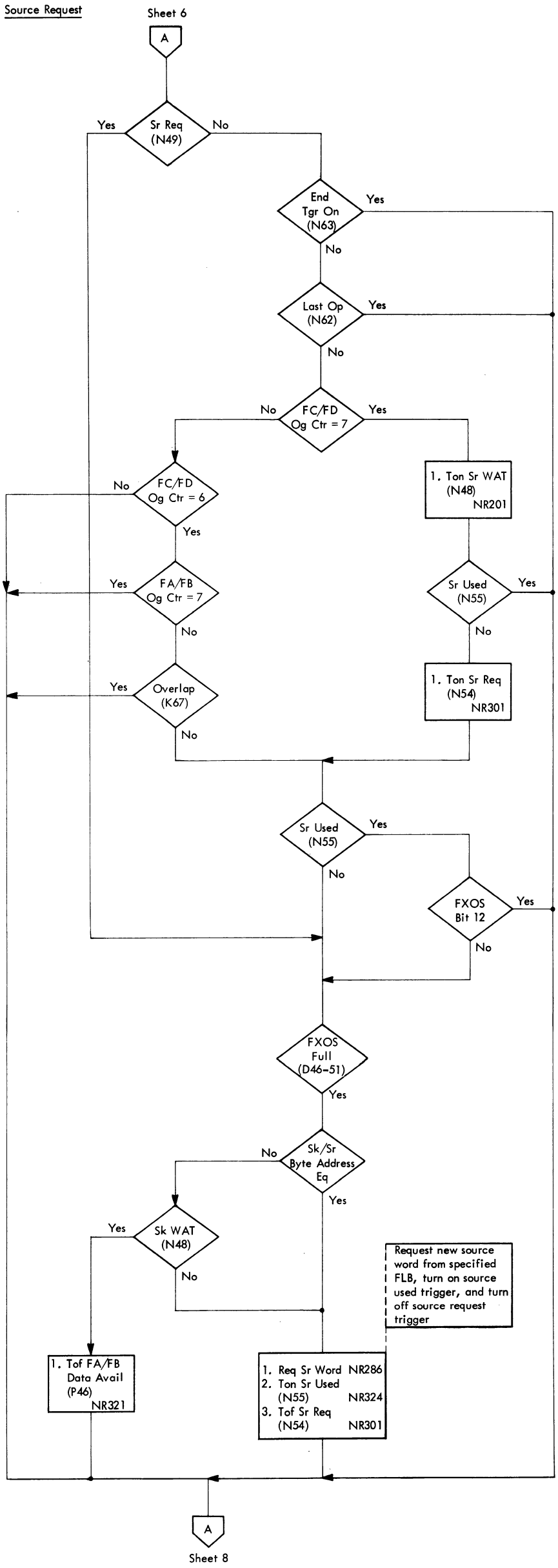


DIAGRAM 5-120. NC, OC, XC, MVN, AND MVZ INSTRUCTIONS (SHEET 7 OF 8)

End of Operation

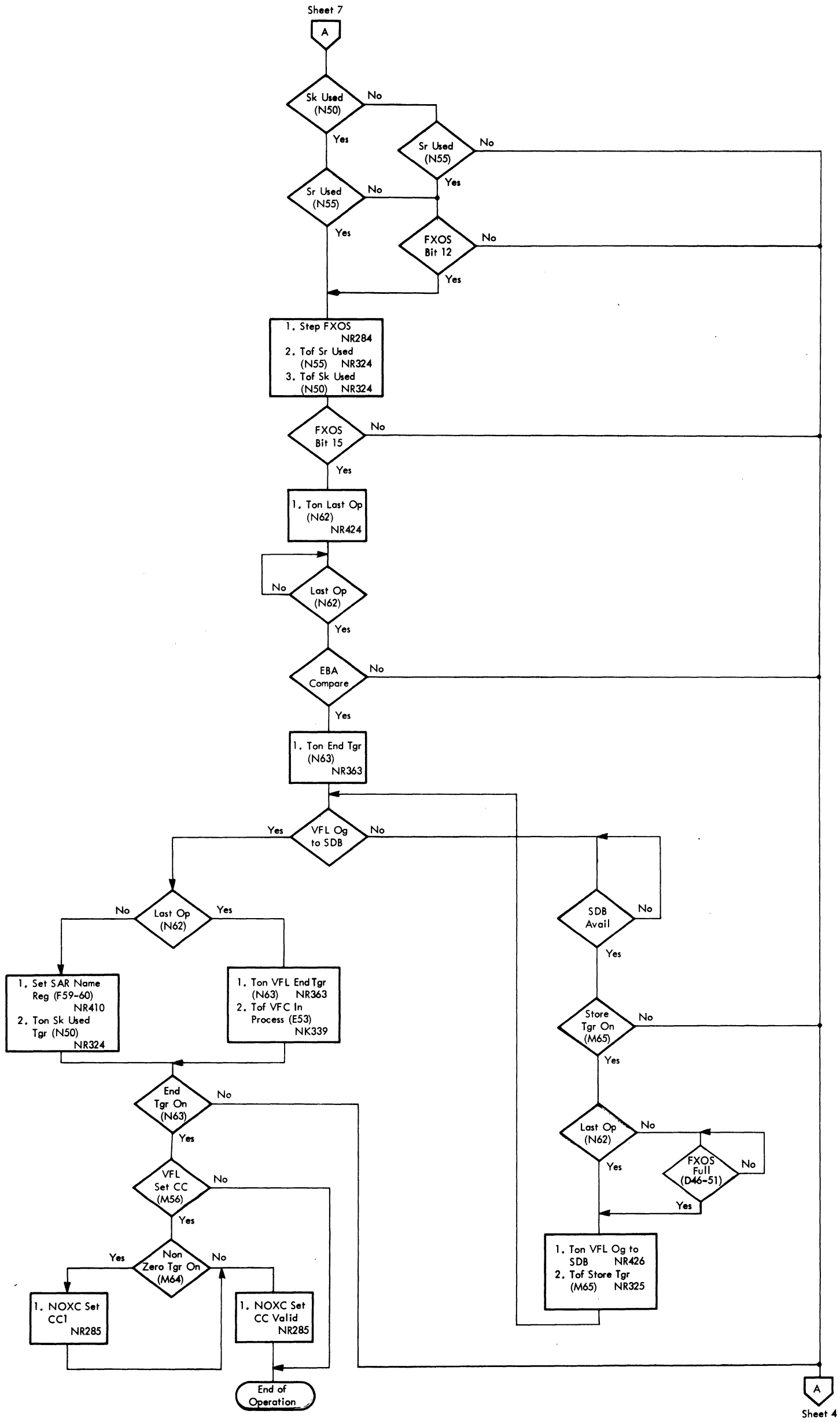
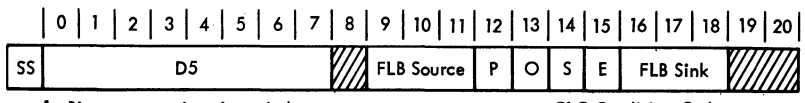


DIAGRAM 5-120. NC, OC, XC, MVN, AND MVZ INSTRUCTIONS (SHEET 8 OF 8)

Objectives:

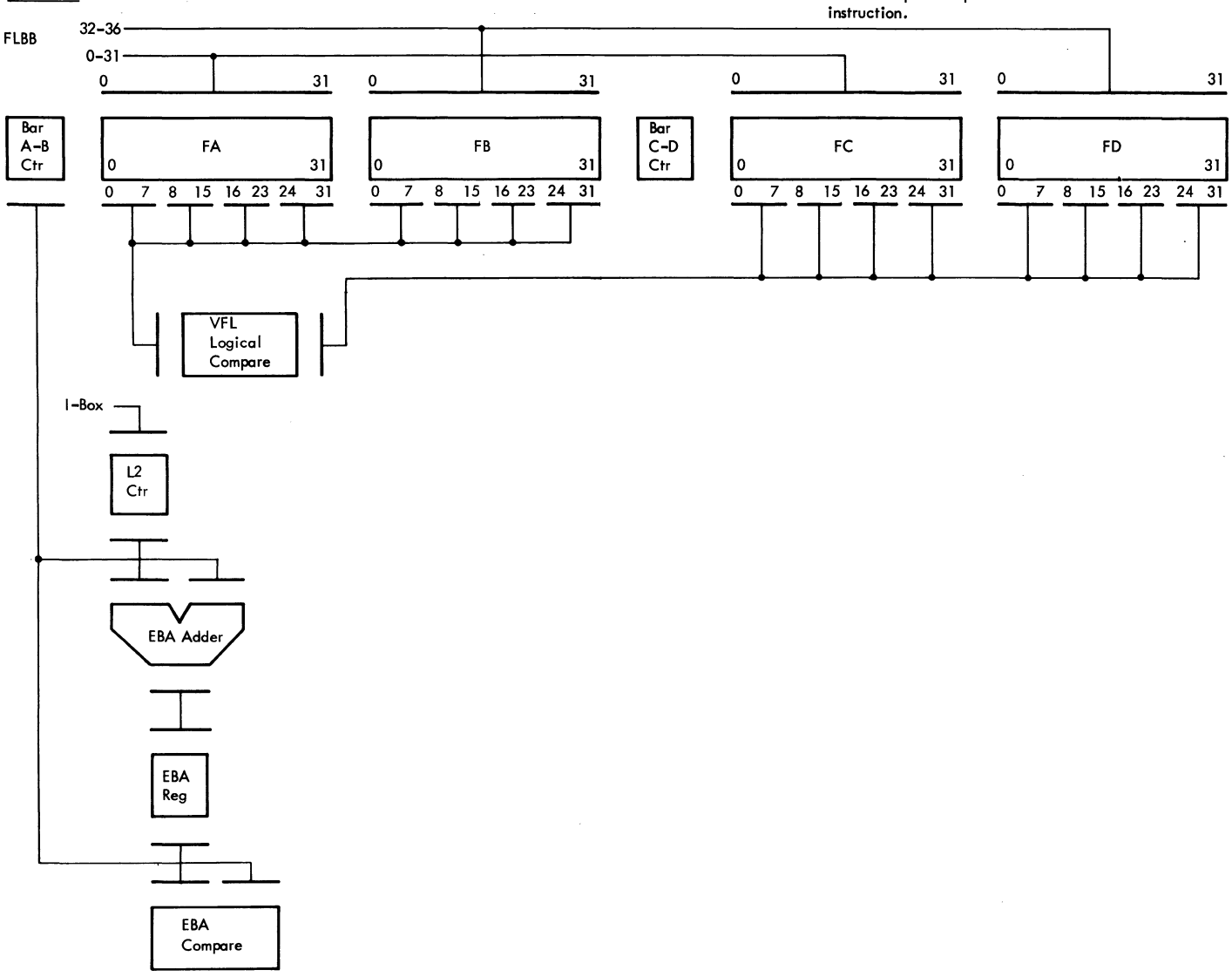
Compare Logical
CLC



1. Binary comparison is made between sink and source bytes.
2. Sink and source words are gated from FLBs to FXBs FA/FB and FC/FD, respectively.
3. Bytes are gated to VFL comparison unit where bits are binarily compared.
4. Results of comparison are used to set condition code.
5. An unequal compare terminates instruction.

CLC Condition Code:
0 - Operands are equal
1 - Sink operand is low
2 - Sink operand is high
3 - (Not applicable)

Data Flow



Simplified Flow Chart

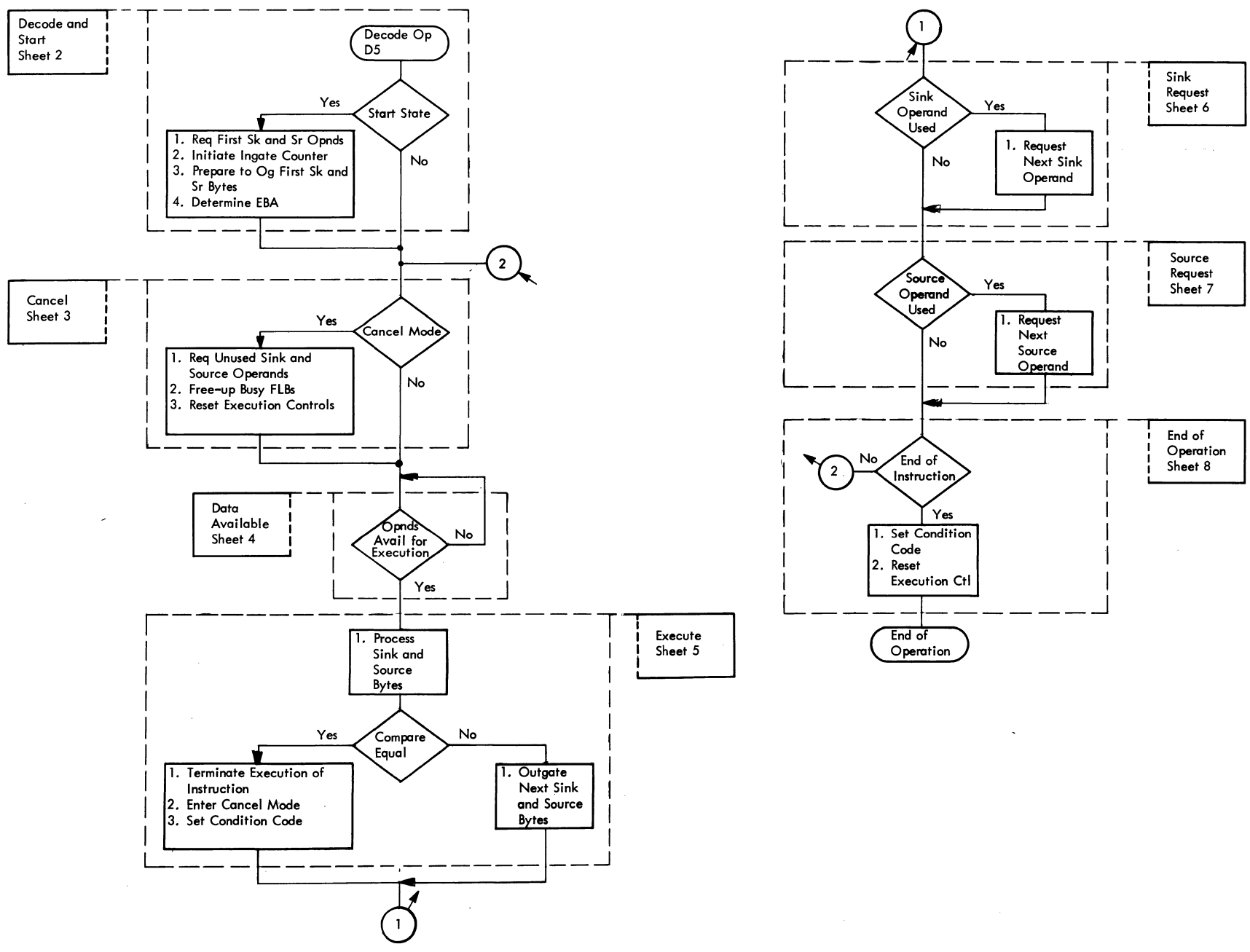


DIAGRAM 5-121. CLC INSTRUCTION (SHEET 1 OF 8)

Decode and Start

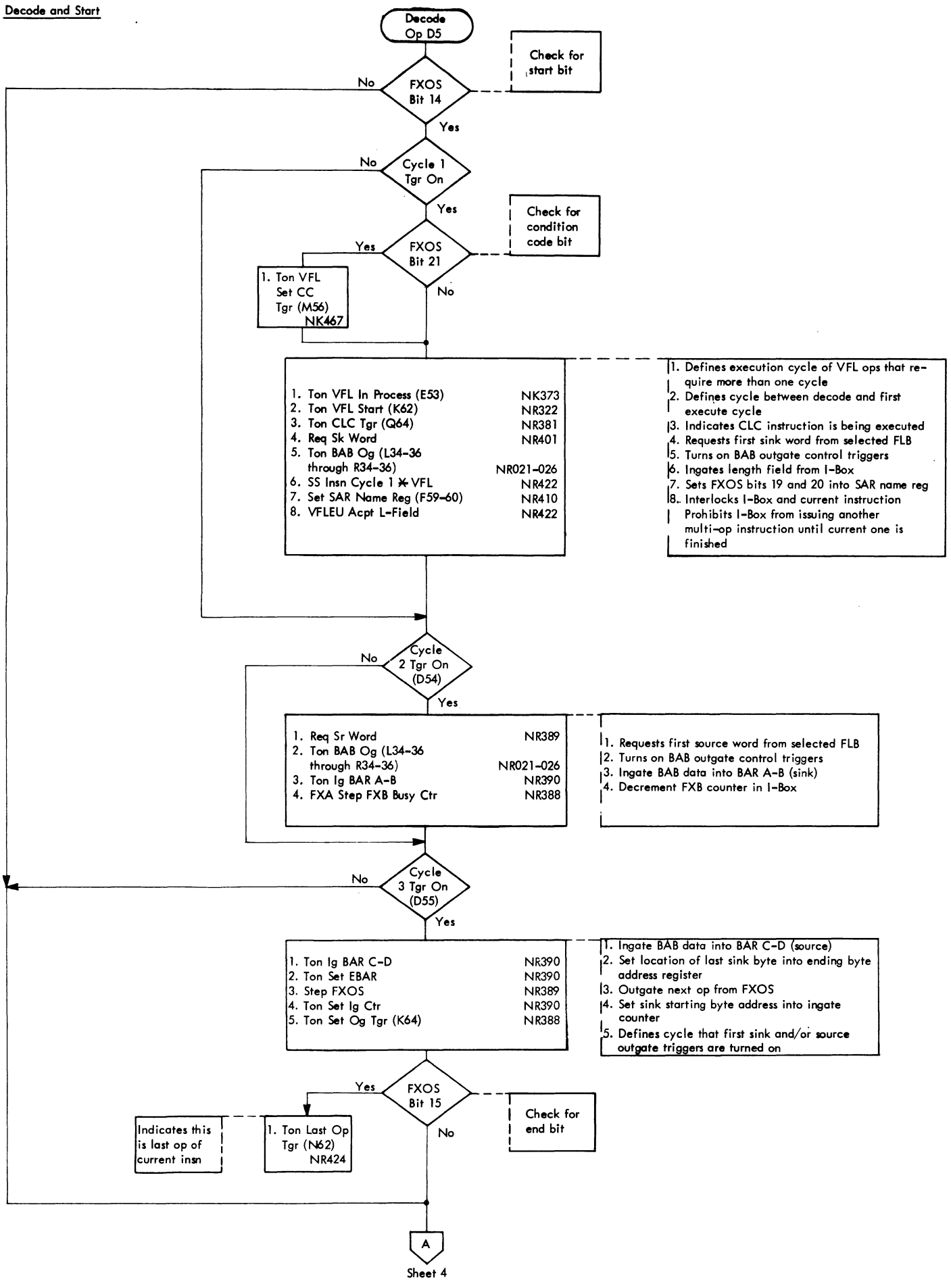


DIAGRAM 5-121. CLC INSTRUCTION (SHEET 2 OF 8)

Cancel

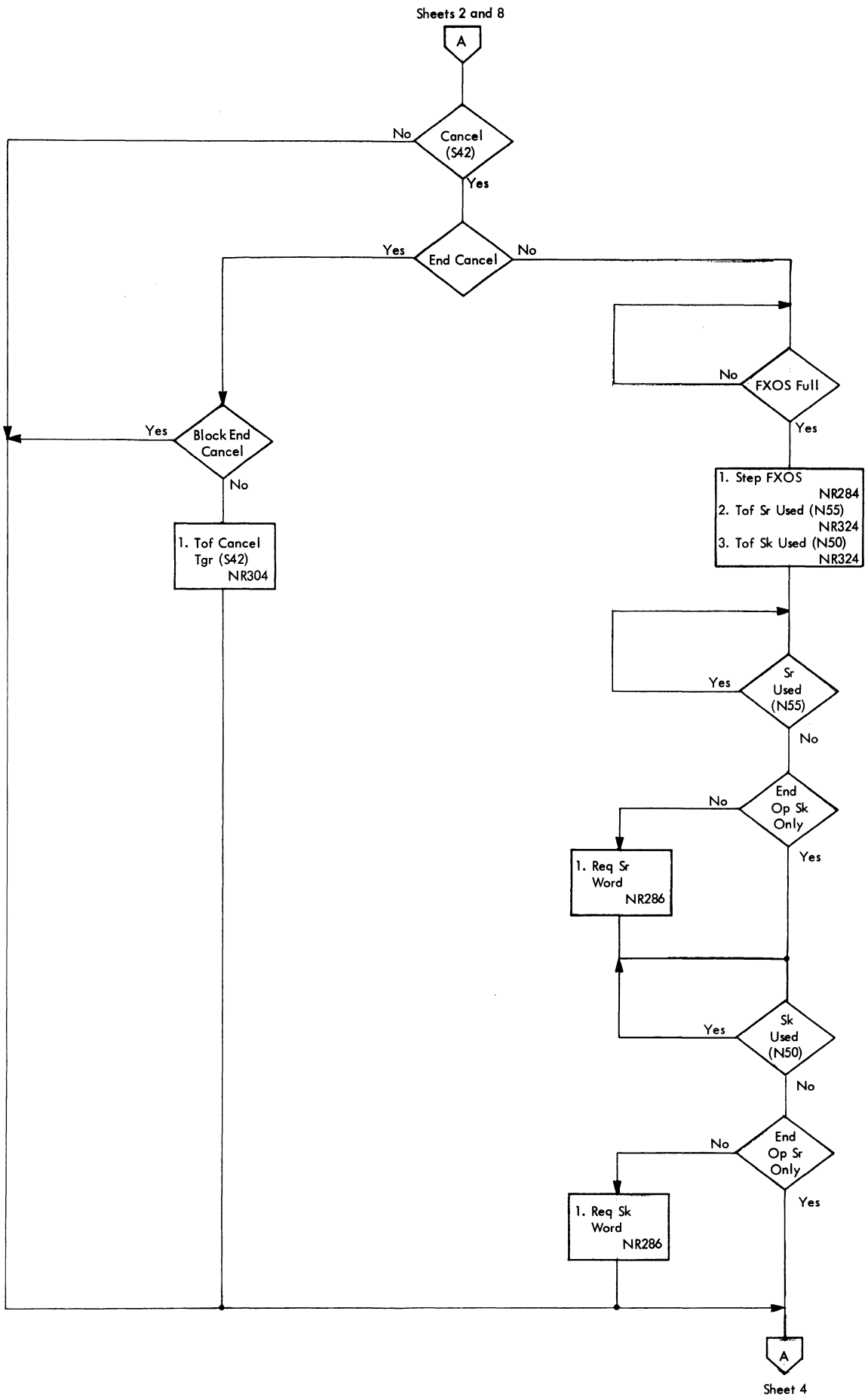


DIAGRAM 5-121. CLC INSTRUCTION (SHEET 3 OF 8)

Data Available

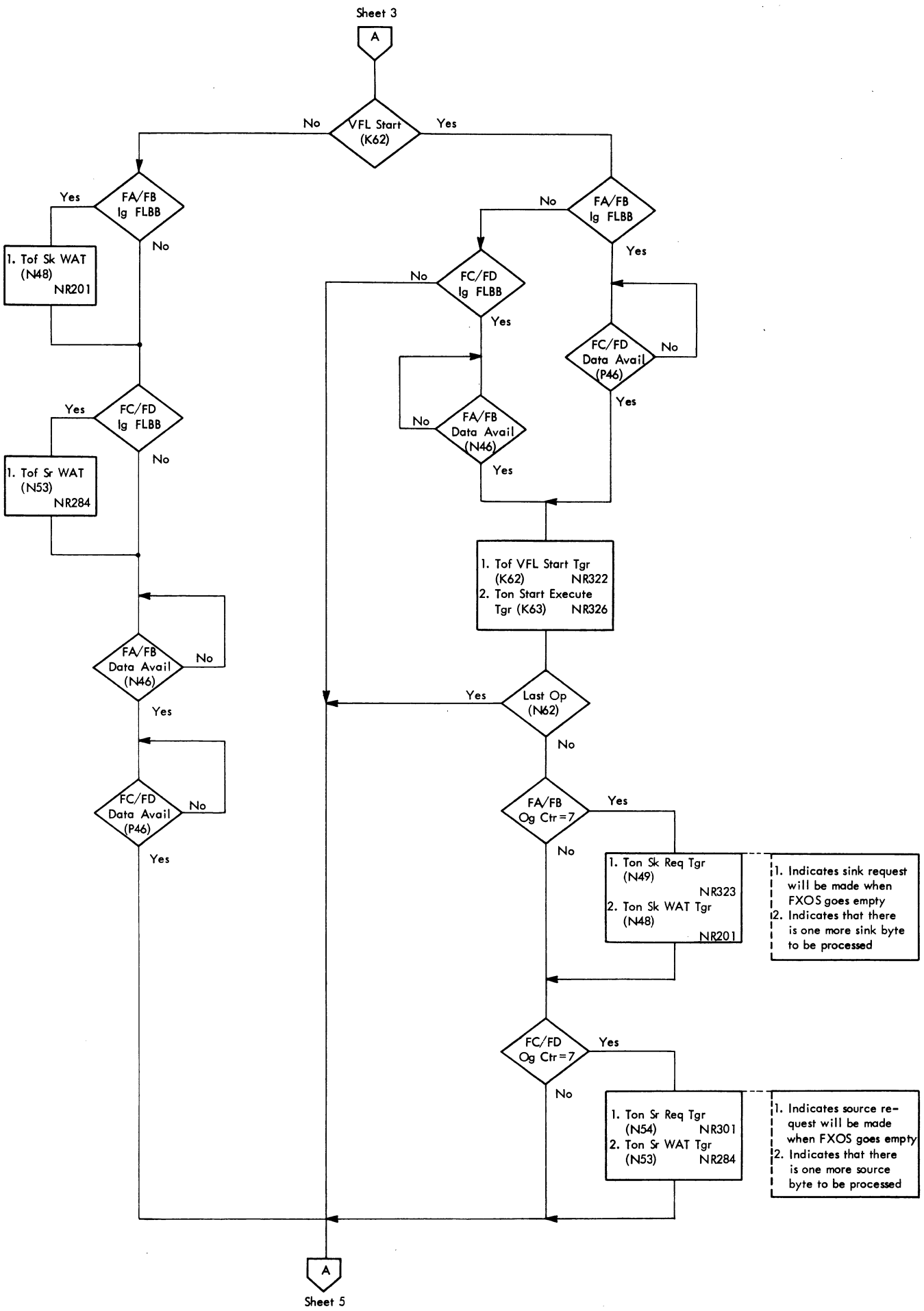


DIAGRAM 5-121. CLC INSTRUCTION (SHEET 4 OF 8)

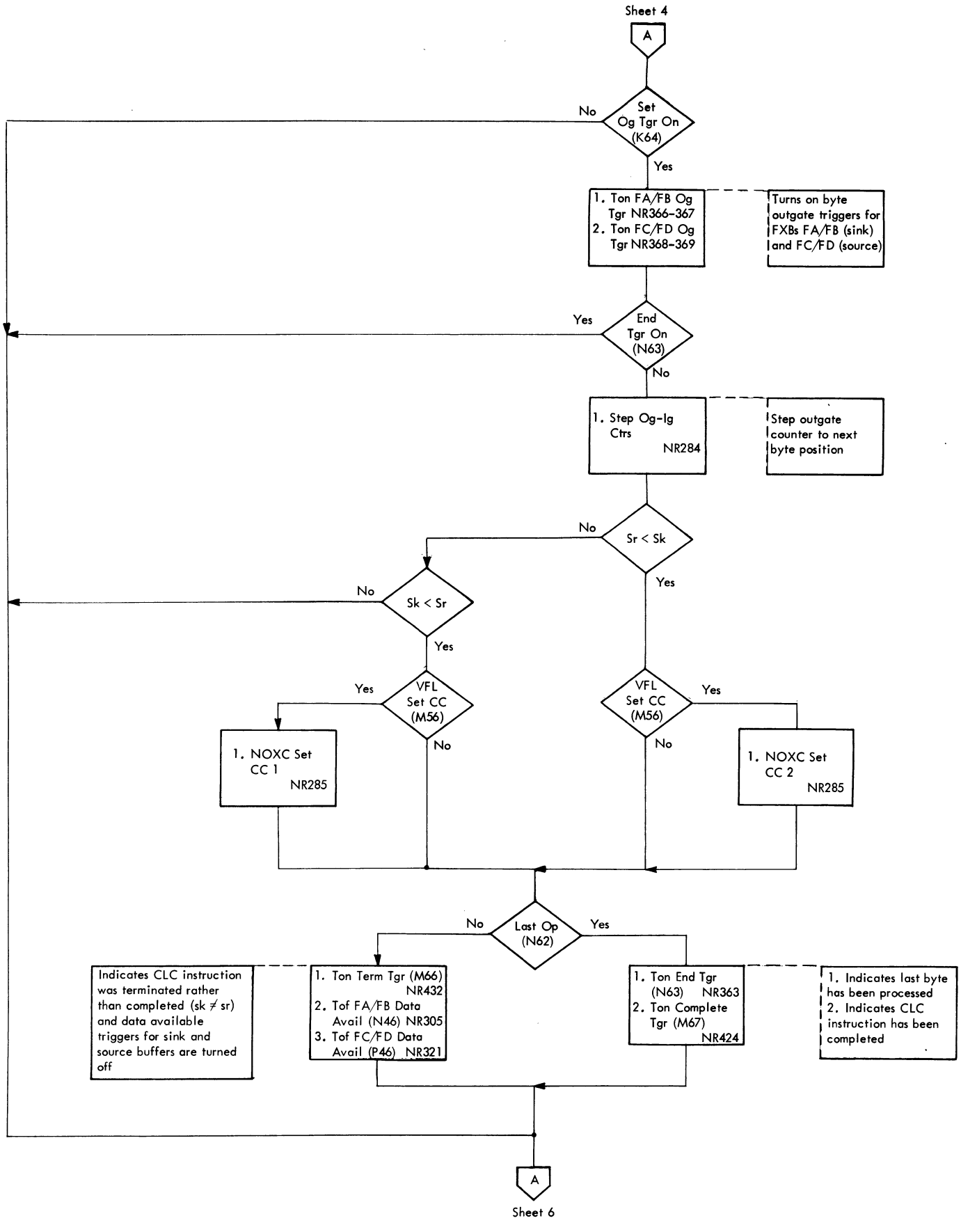


DIAGRAM 5-121. CLC INSTRUCTION (SHEET 5 OF 8)

Sink Request

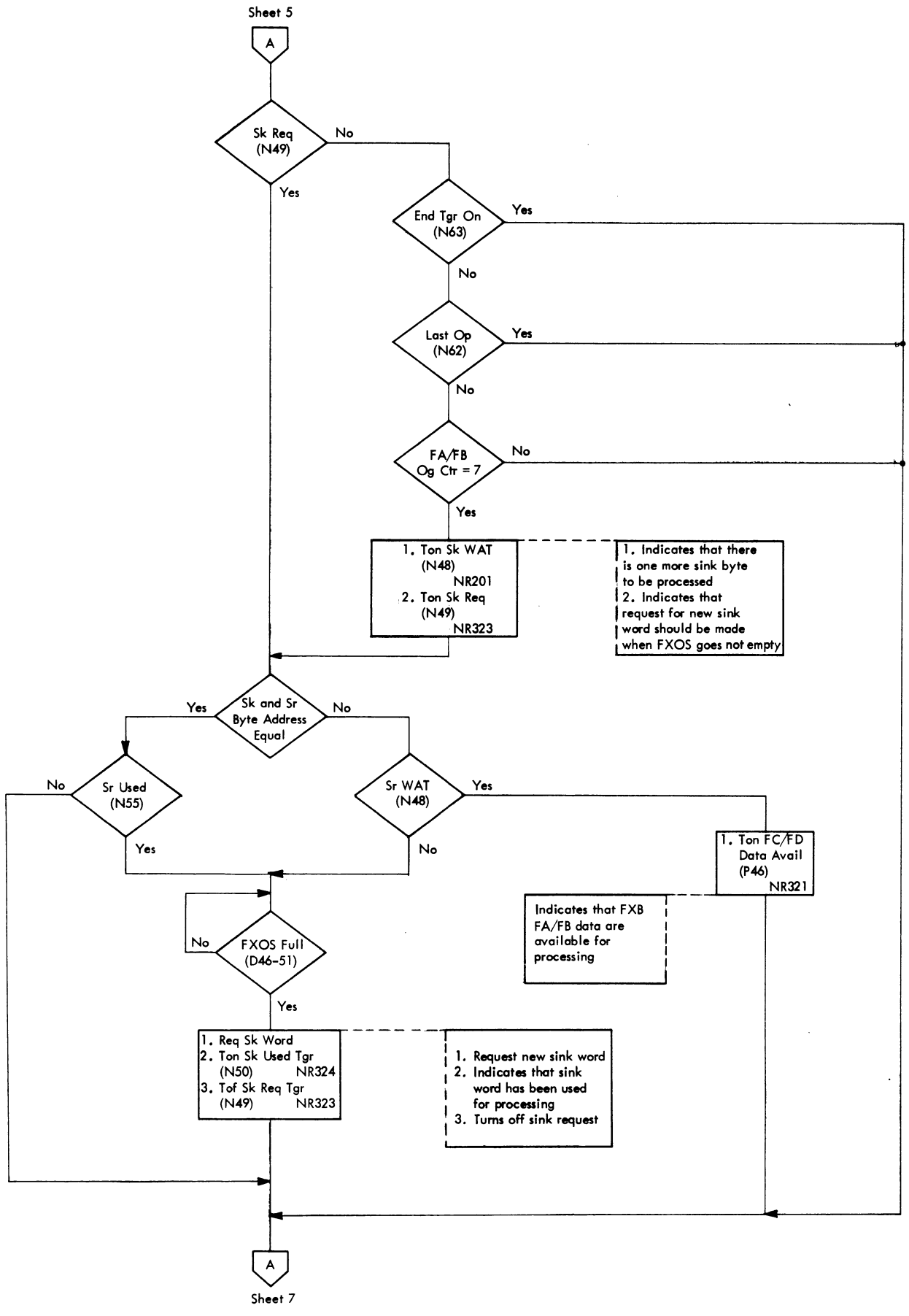
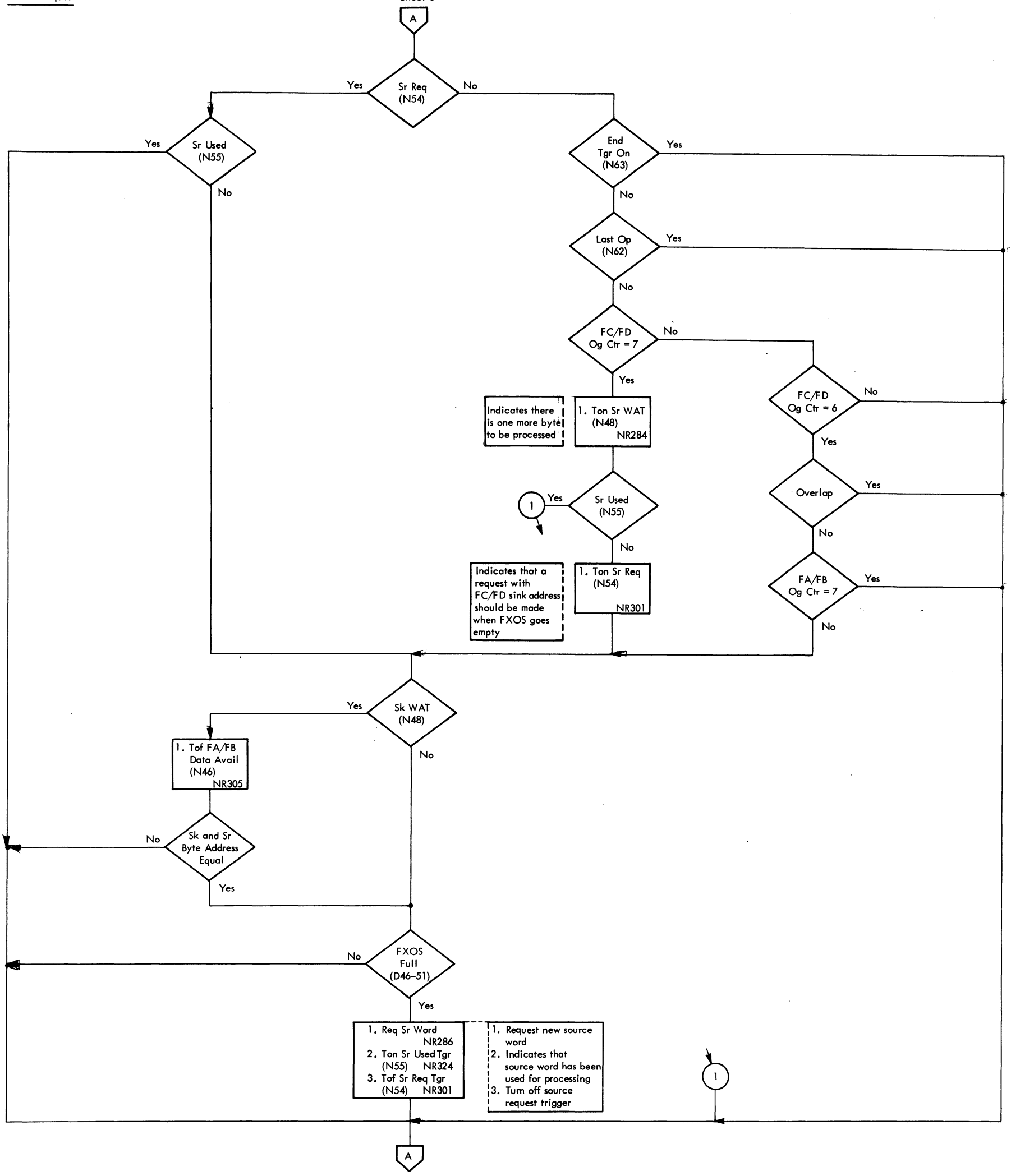


DIAGRAM 5-121. CLC INSTRUCTION (SHEET 6 OF 8)

Sheet 6



Sheet 8

DIAGRAM 5-121. CLC INSTRUCTION (SHEET 7 OF 8)

End of Operation

Sheet 7

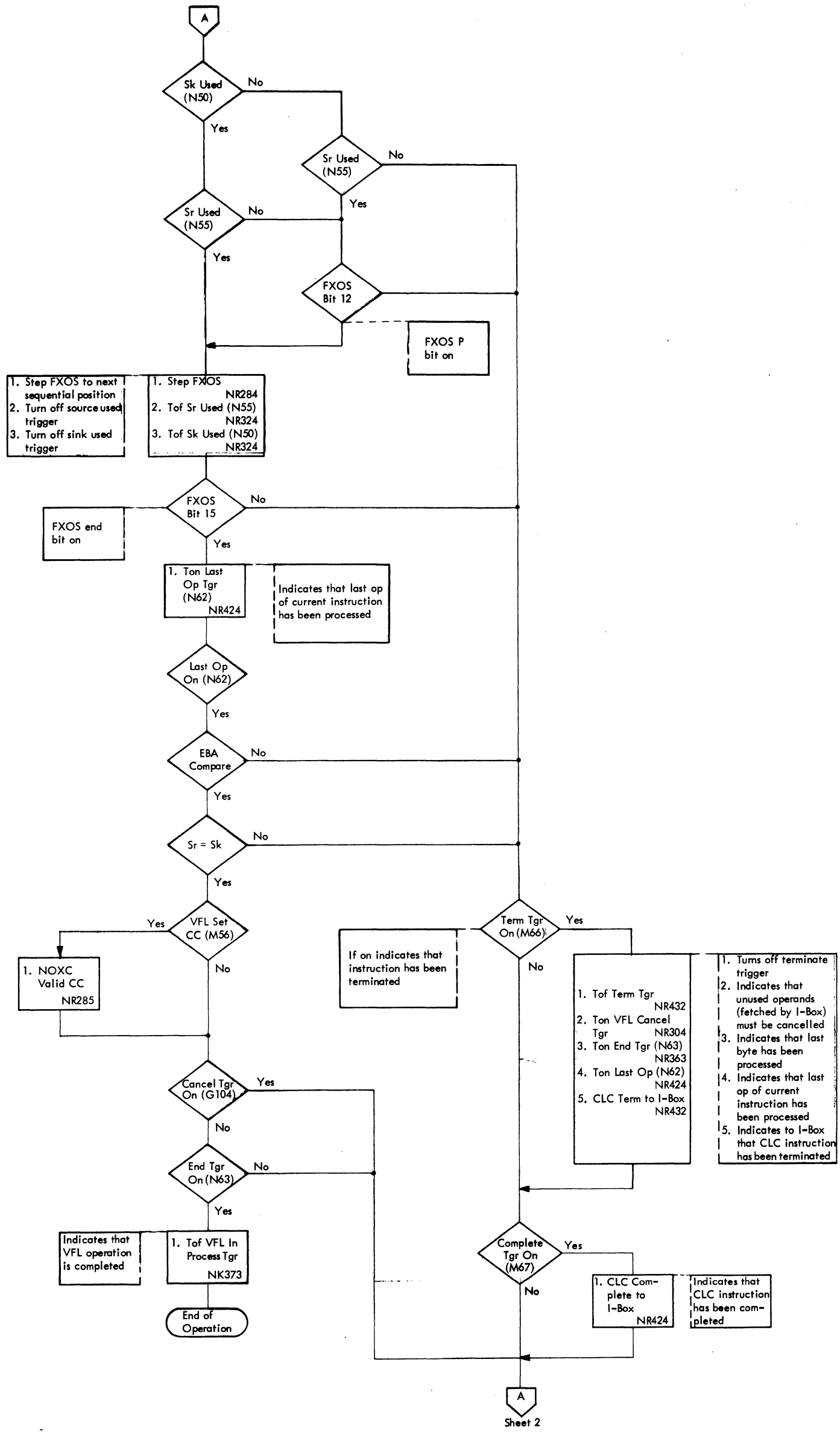


DIAGRAM 5-121. CLC INSTRUCTION (SHEET 8 OF 8)

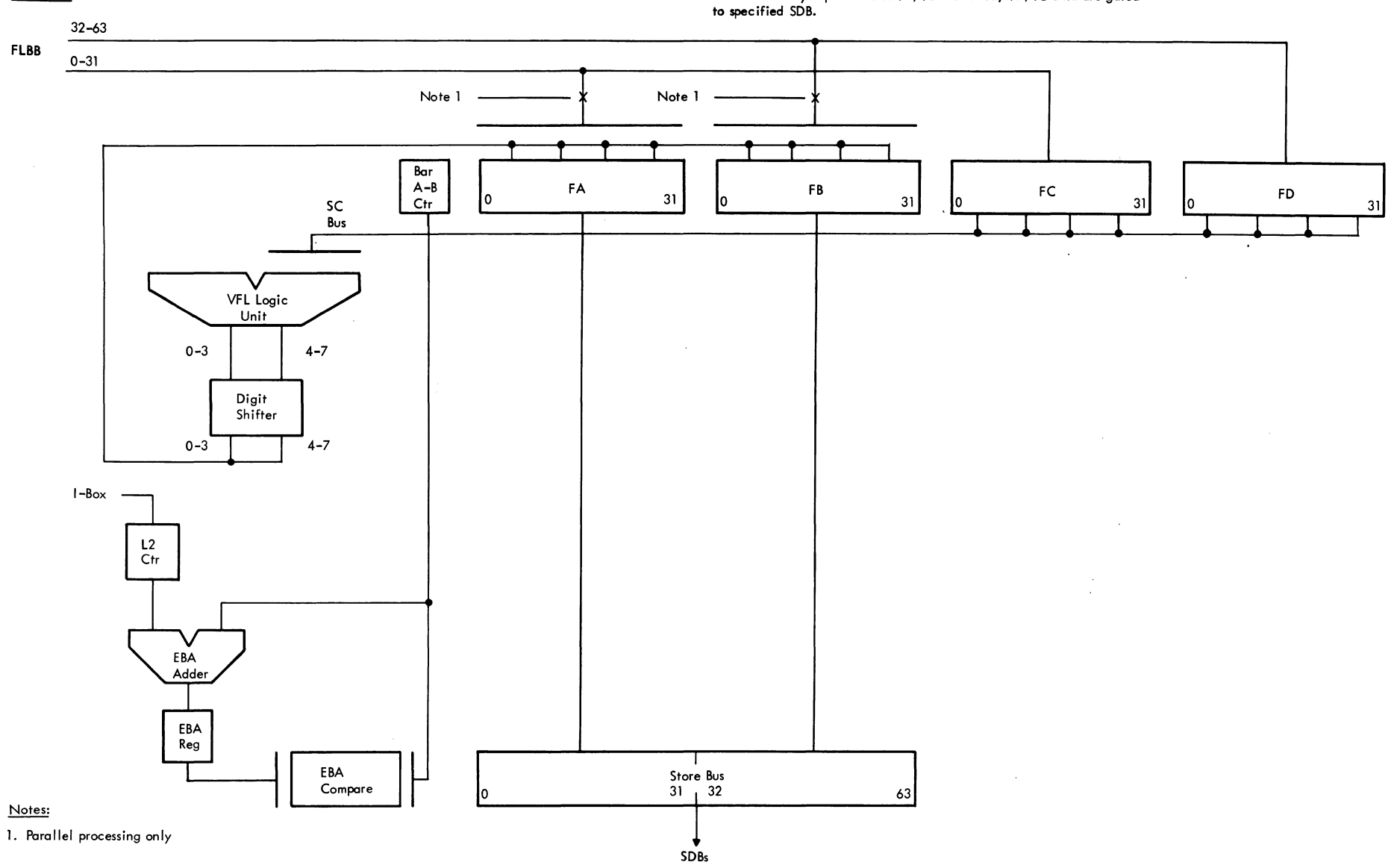
Objectives:

Move Character
MVC

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
SS	D2								FLB Name			P	O	S	E	Starting Byte Adr			SAR Name	

1. Specified FLB is gated to FA/FB and FC/FD during parallel operation (P bit = 1) or FC/FD only during serial operation (P bit = 0).
2. Data are moved one byte at a time from FC/FD to FA/FB during serial operation.
3. When all byte positions of FA/FB are filled, FA/FB data are gated to specified SDB.

Data Flow



Notes:

1. Parallel processing only

Simplified Flow Chart

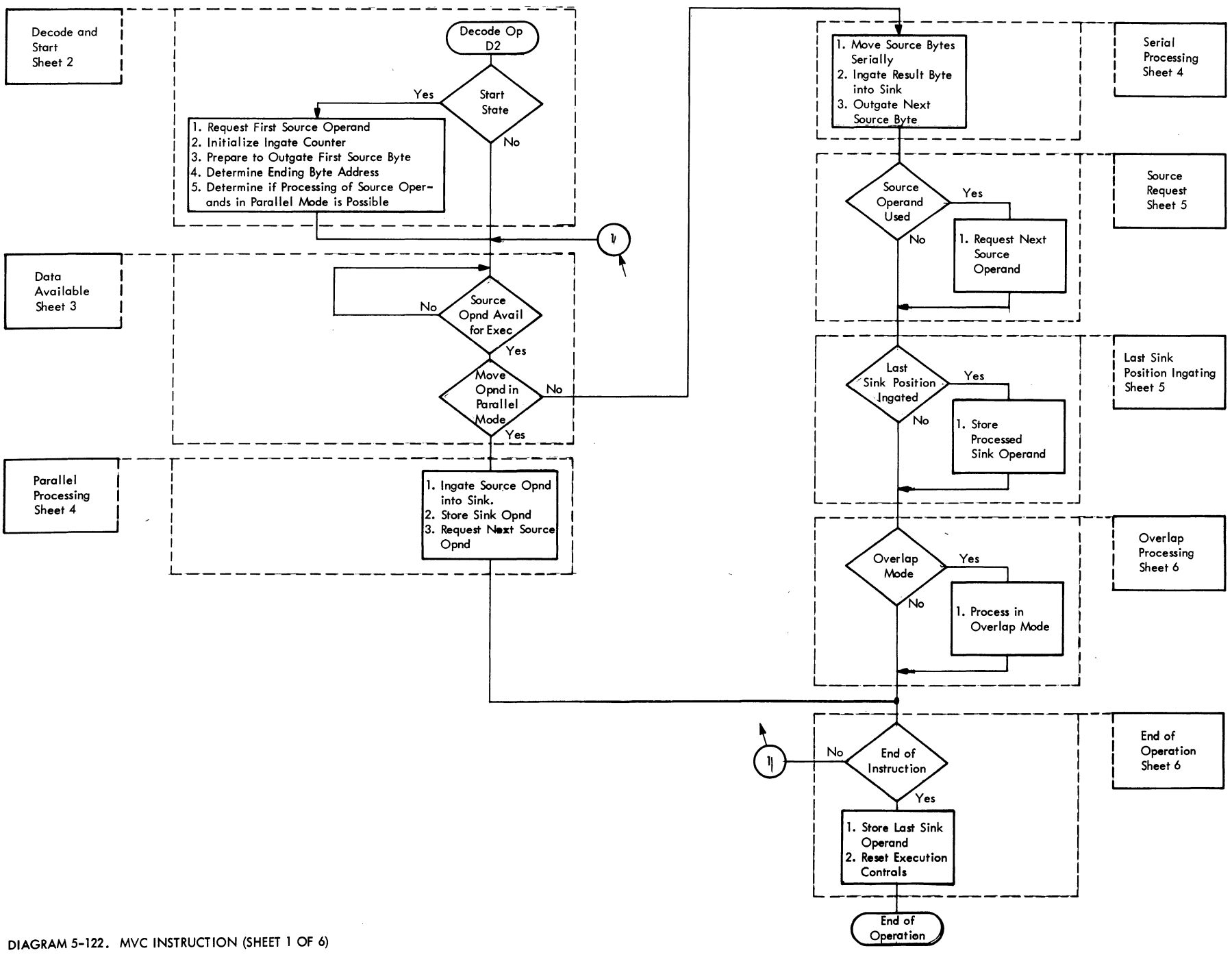


DIAGRAM 5-122. MVC INSTRUCTION (SHEET 1 OF 6)

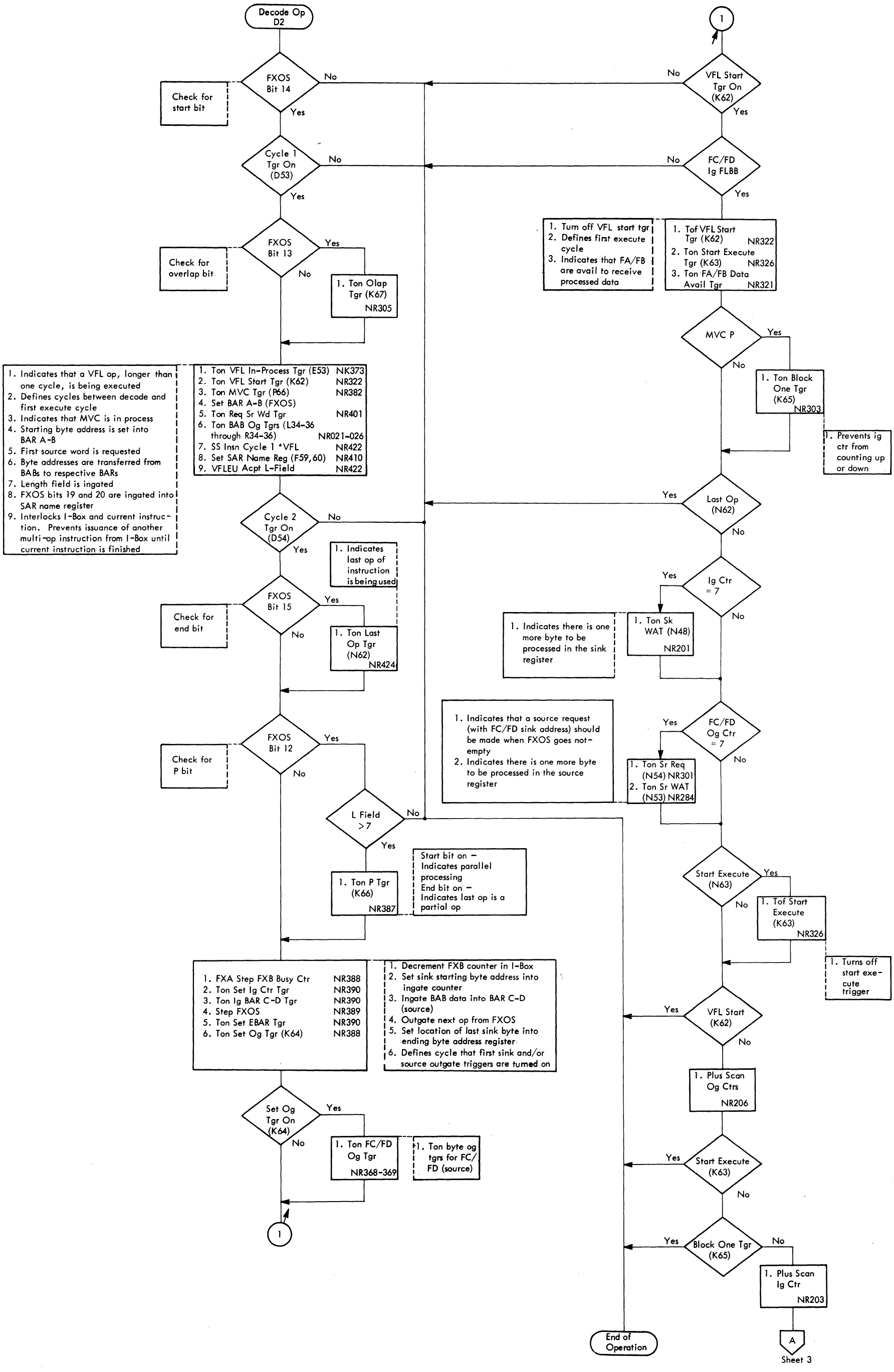


DIAGRAM 5-122. MVC INSTRUCTION (SHEET 2 OF 6)

Data Available

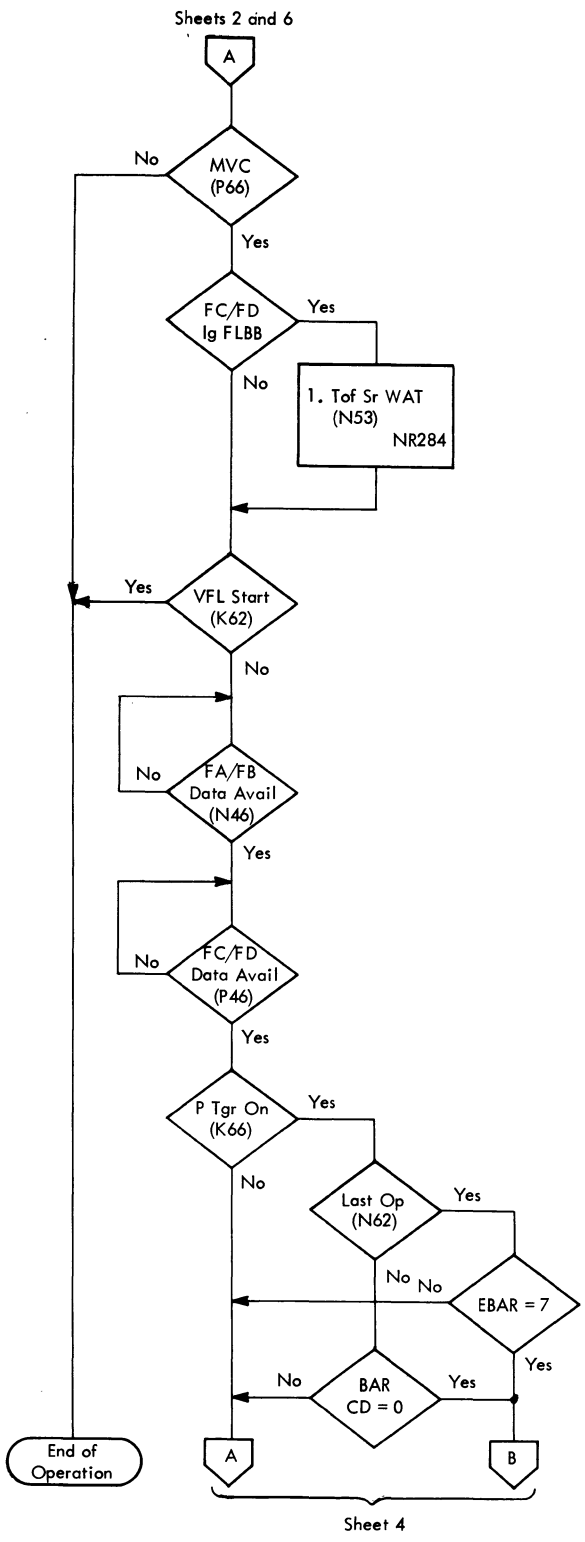
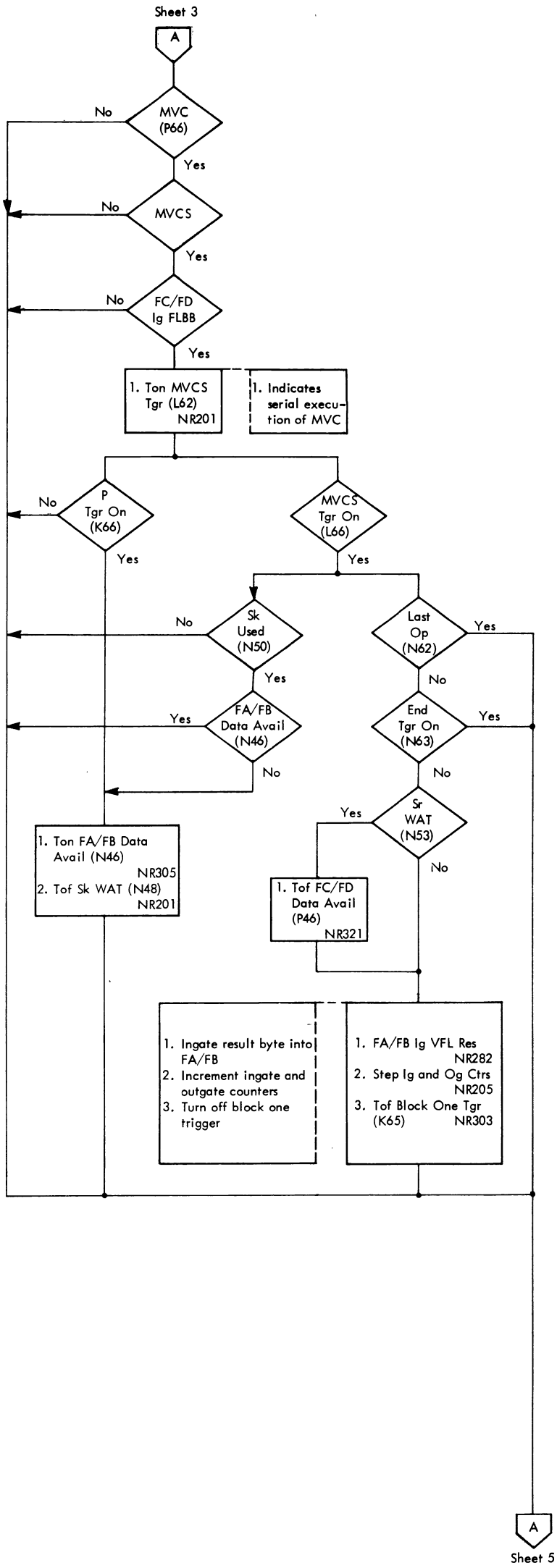


DIAGRAM 5-122. MVC INSTRUCTION (SHEET 3 OF 6)

Serial Processing



Parallel Processing

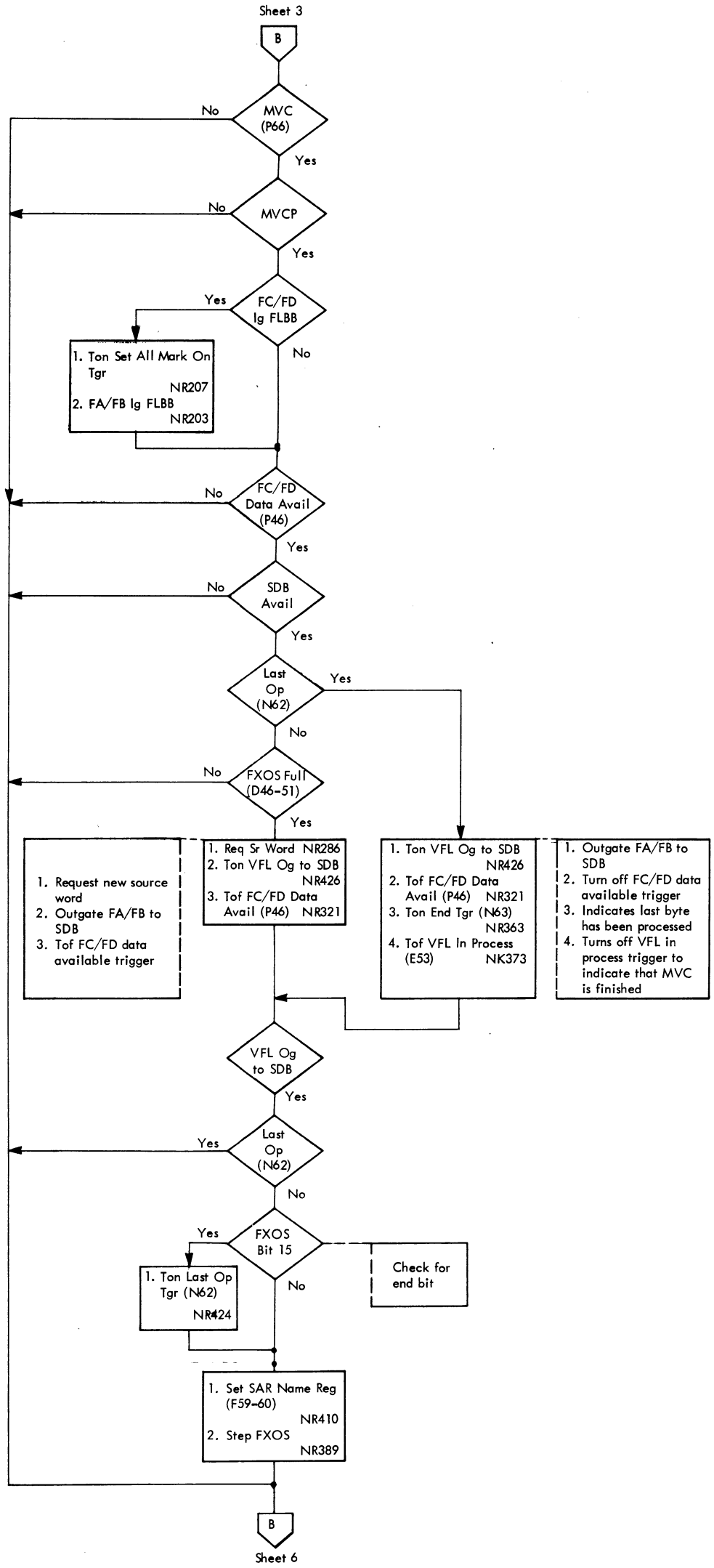


DIAGRAM 5-122. MVC INSTRUCTION (SHEET 4 OF 6)

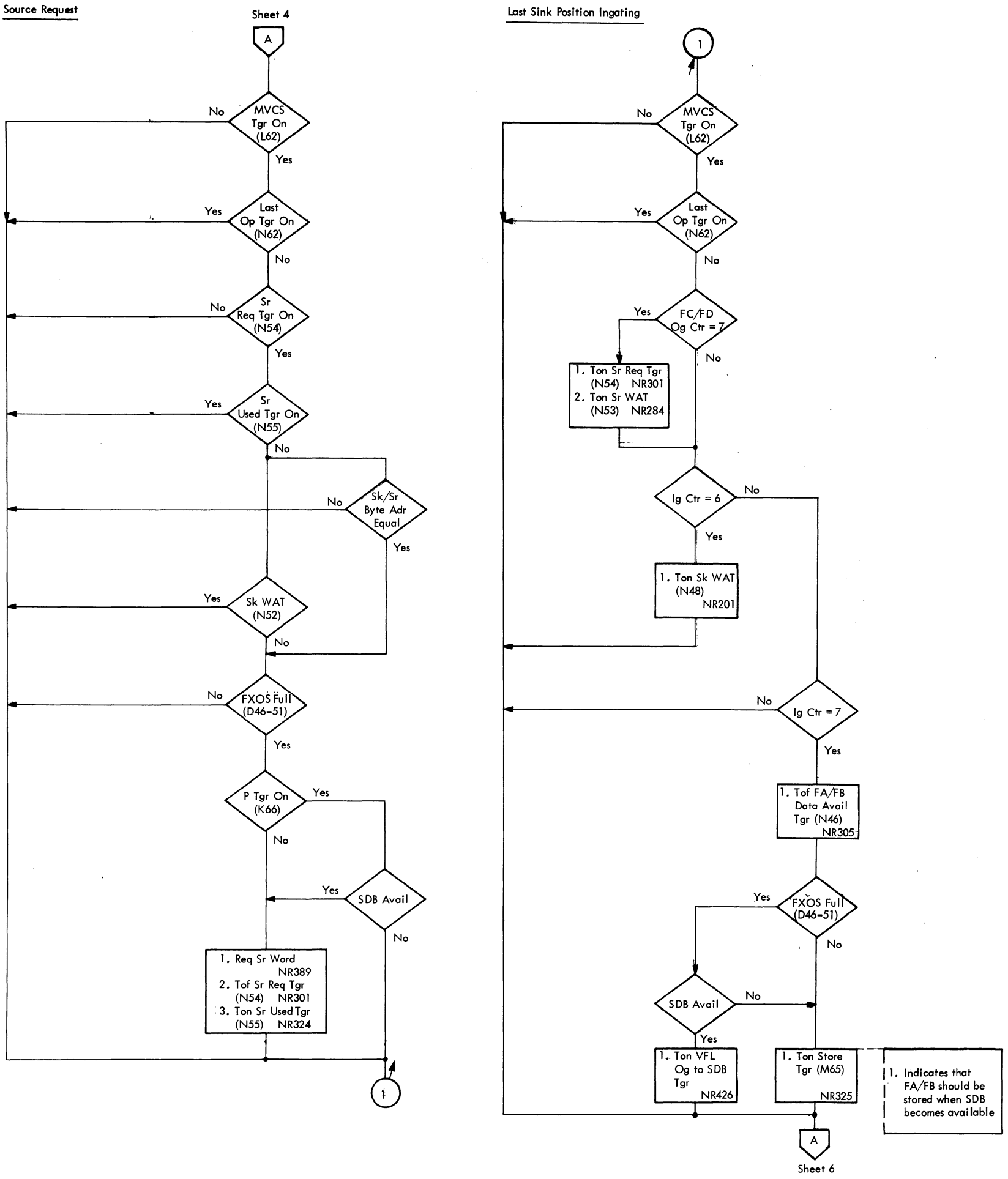


DIAGRAM 5-122. MVC INSTRUCTION (SHEET 5 OF 6)

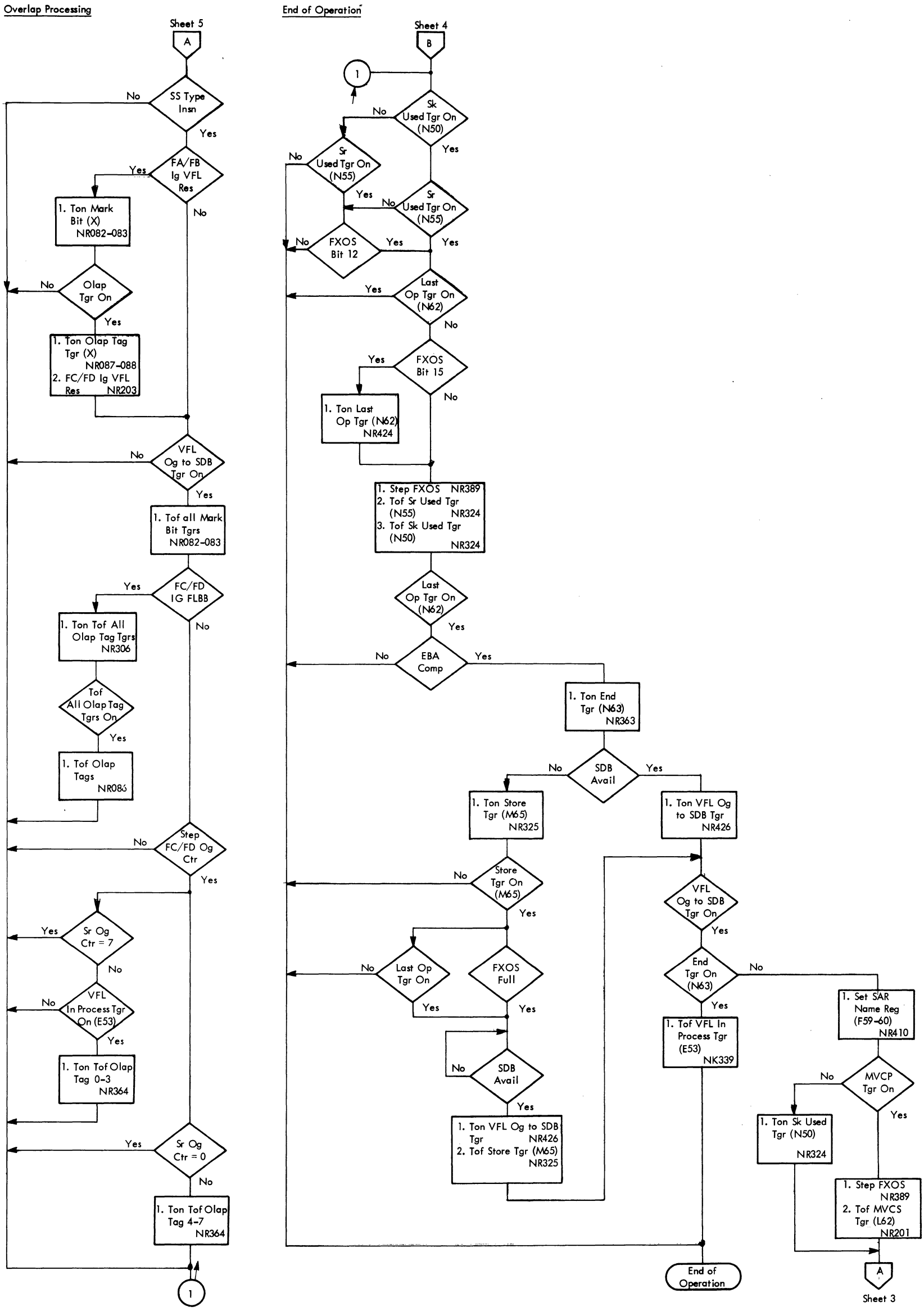
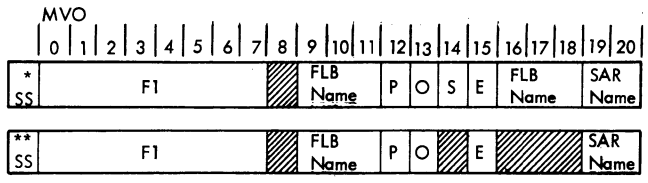


DIAGRAM 5-122. MVC INSTRUCTION (SHEET 6 OF 6)

Objectives:

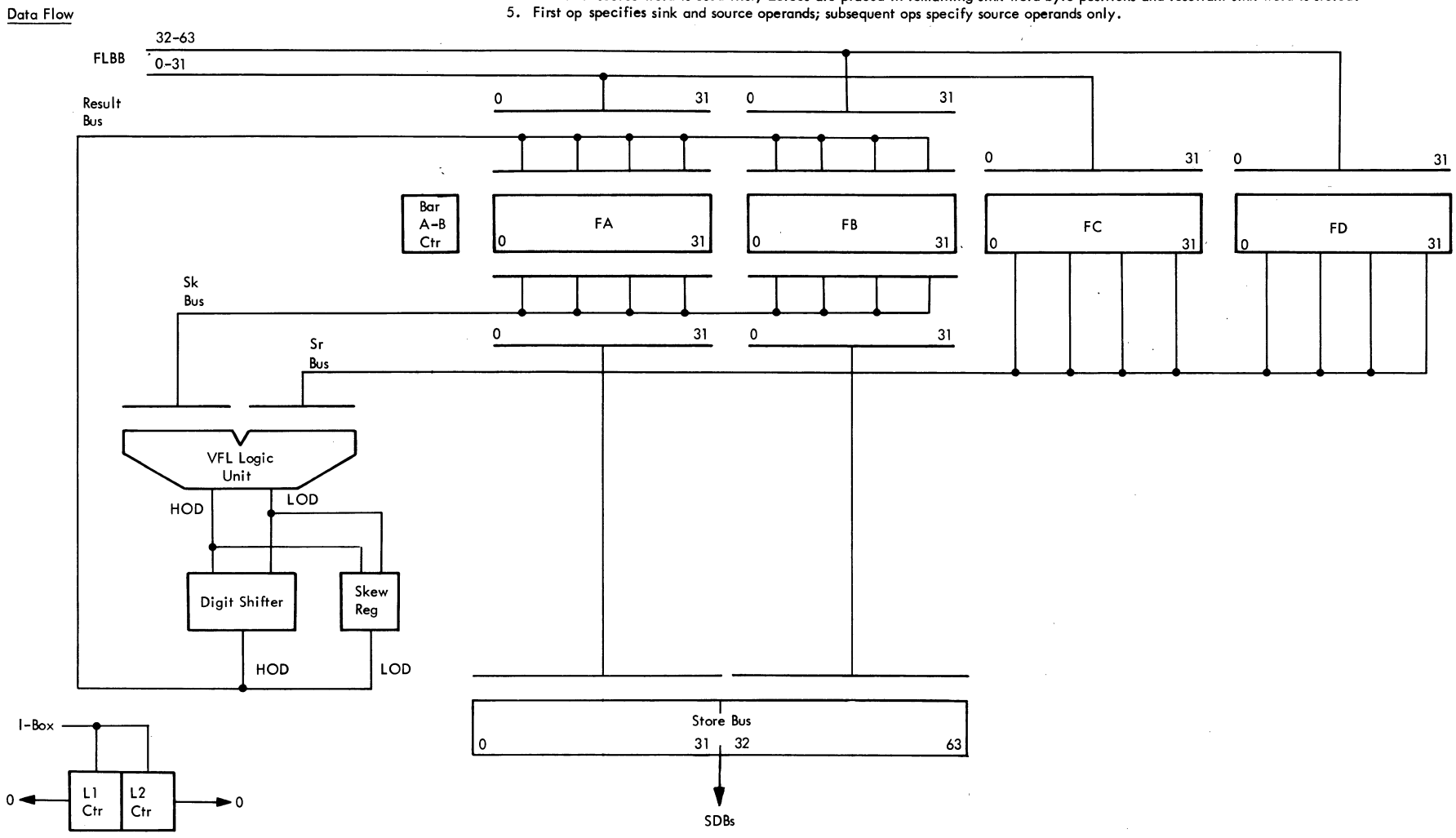
Move with Offset
MVO



* First op only ** Any op after first op

1. Instruction offsets source word one digit position toward high-order byte position.
2. First result byte: Low-order digit of sink byte is combined with high-order digit of source byte (digits retain same relative digit position).
3. Subsequent result bytes: Low-order digit of source byte is combined with high-order digit of succeeding source byte.
4. Bytes are processed until sink and source fields are used; however, if sink is used first, resultant sink word is stored and cancel mode is initiated. If source word is used first, zeroes are placed in remaining sink word byte positions and resultant sink word is stored.
5. First op specifies sink and source operands; subsequent ops specify source operands only.

Data Flow



Simplified Flow

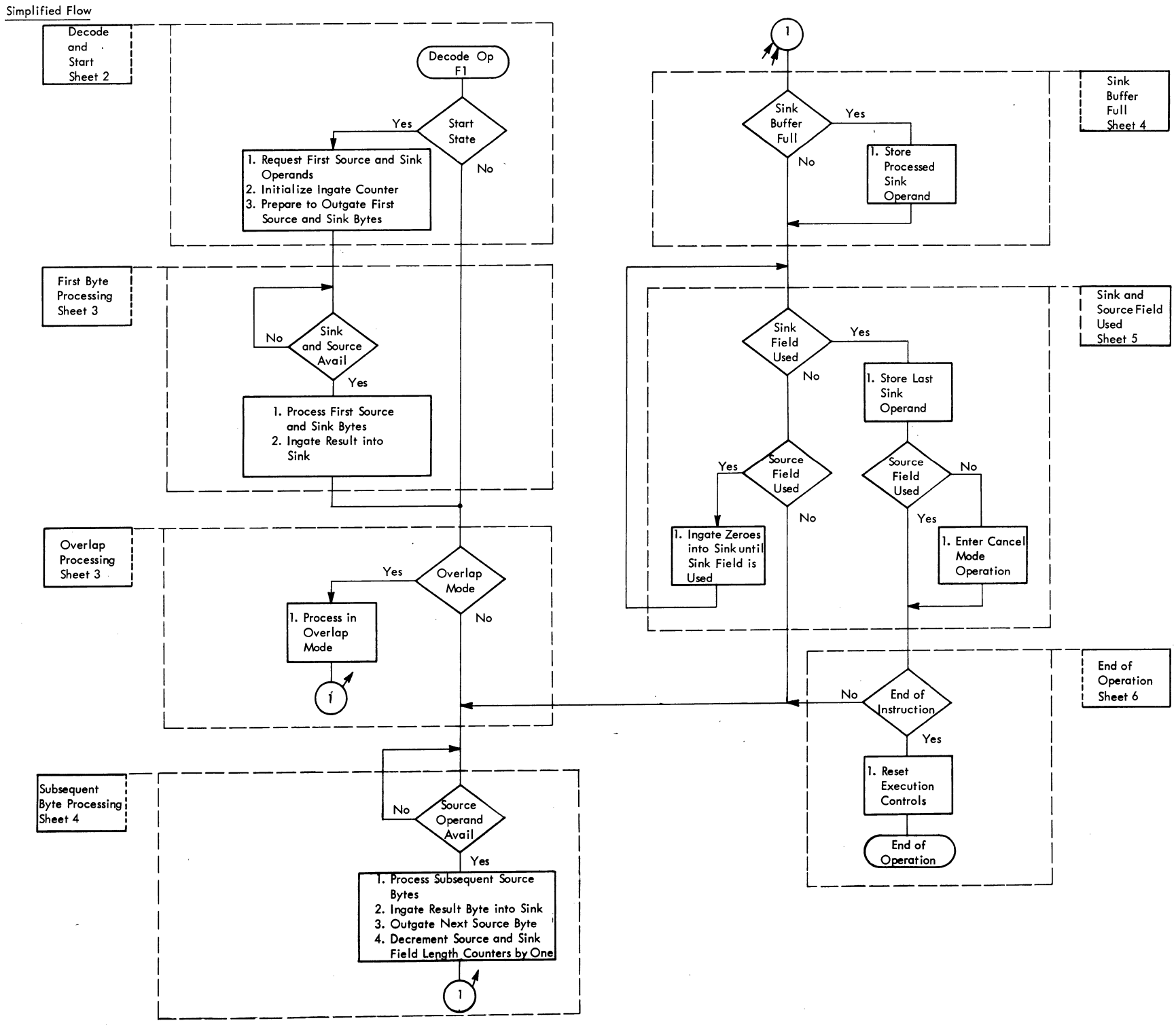


DIAGRAM 5-123. MVO INSTRUCTION (SHEET 1 OF 6)

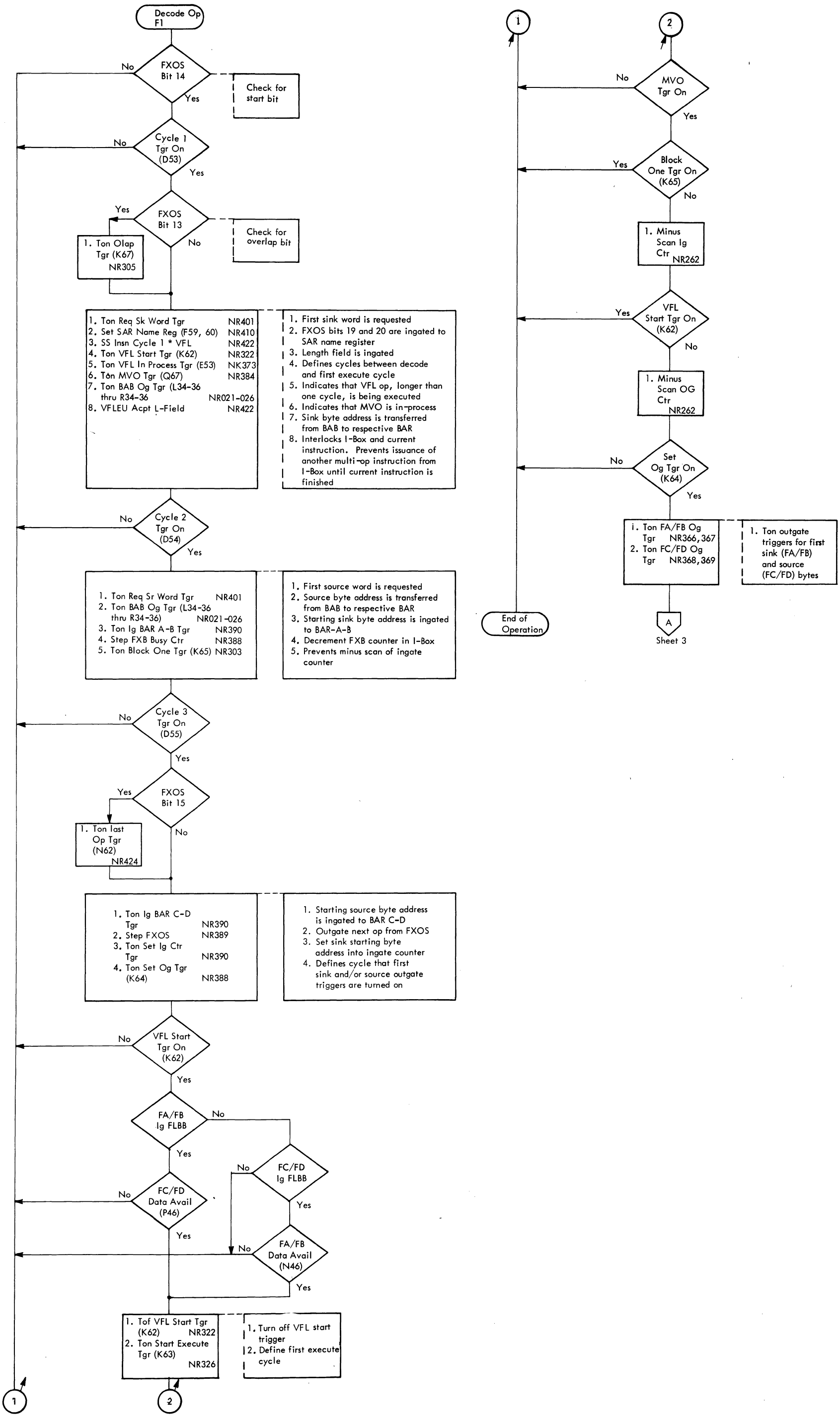


DIAGRAM 5-123. MVO INSTRUCTION (SHEET 2 OF 6)

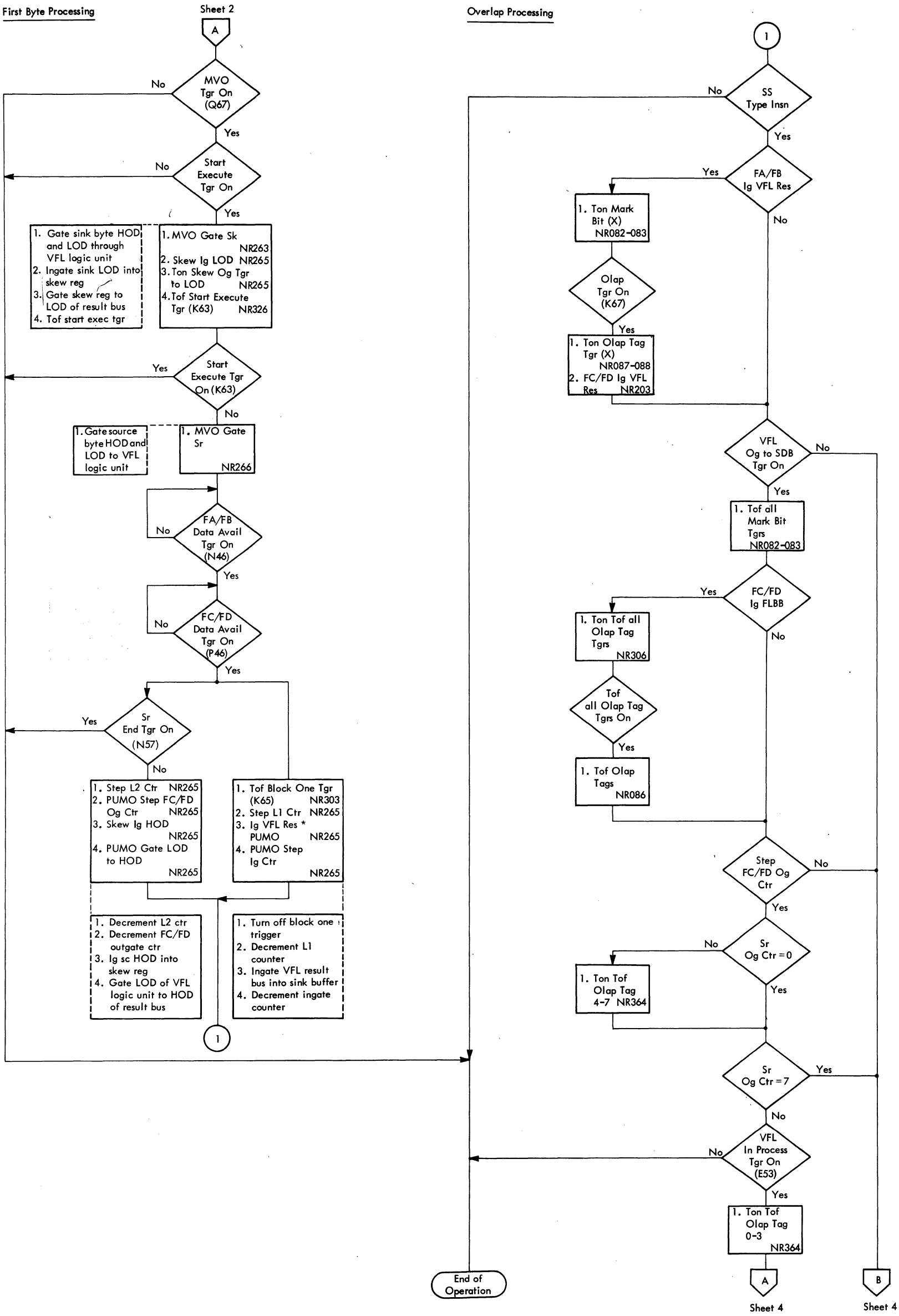


DIAGRAM 5-123. MVO INSTRUCTION (SHEET 3 OF 6)

Subsequent Byte Processing

Sink Buffer Full

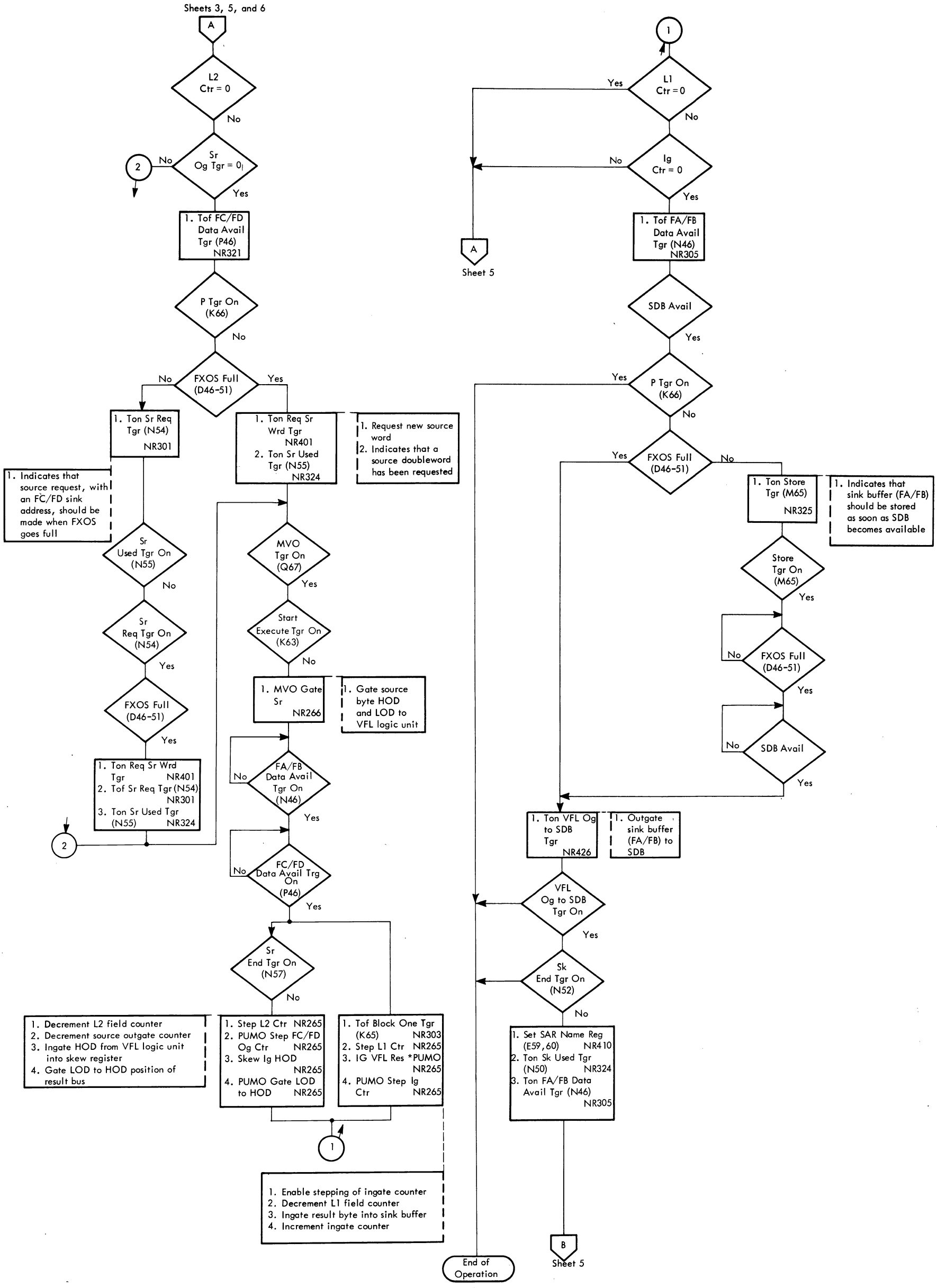


DIAGRAM 5-123. MVO INSTRUCTION (SHEET 4 OF 6)

Sink and Source Field Used

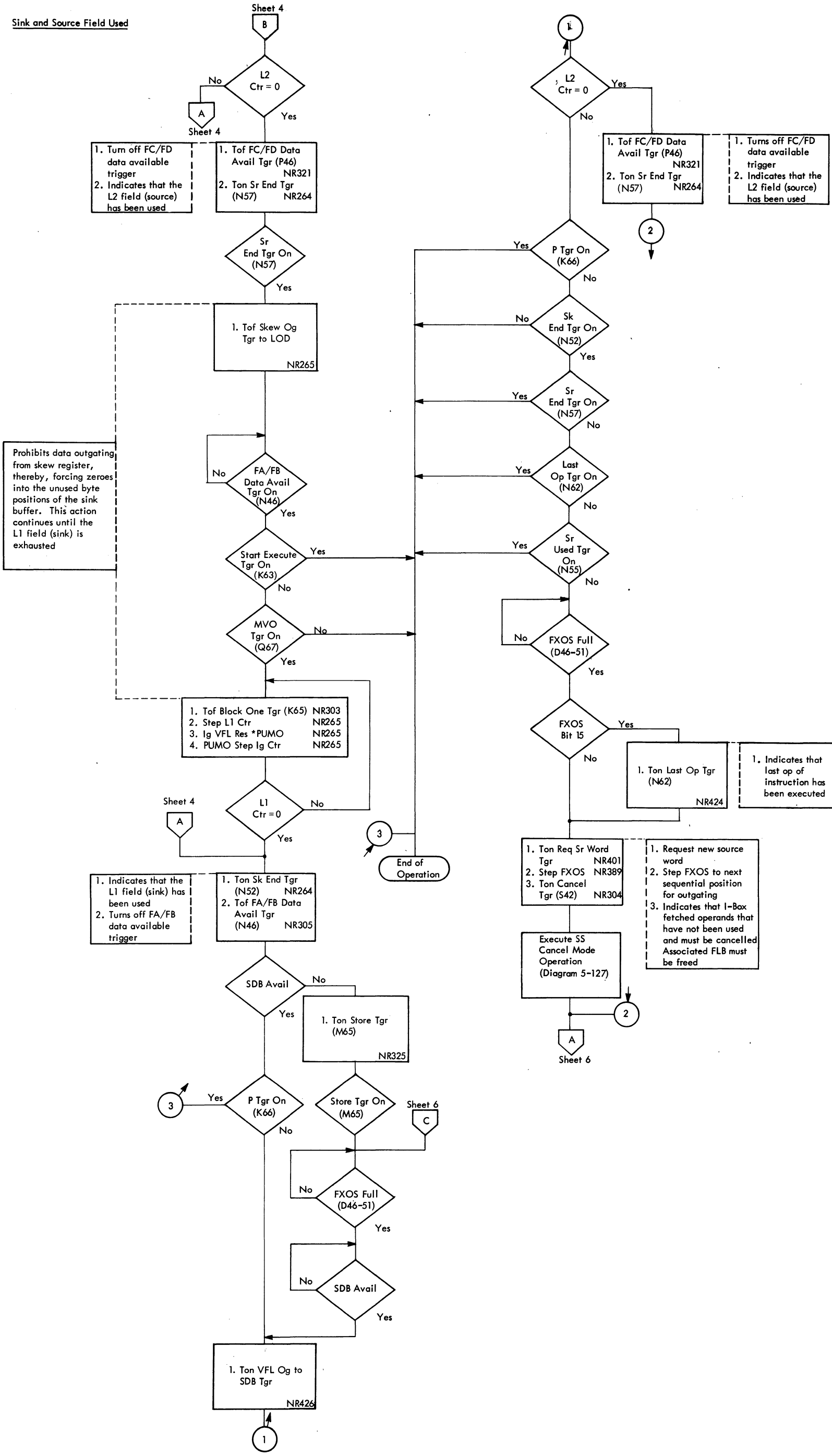


DIAGRAM 5-123. MVO INSTRUCTION (SHEET 5 OF 6)

End of Operation

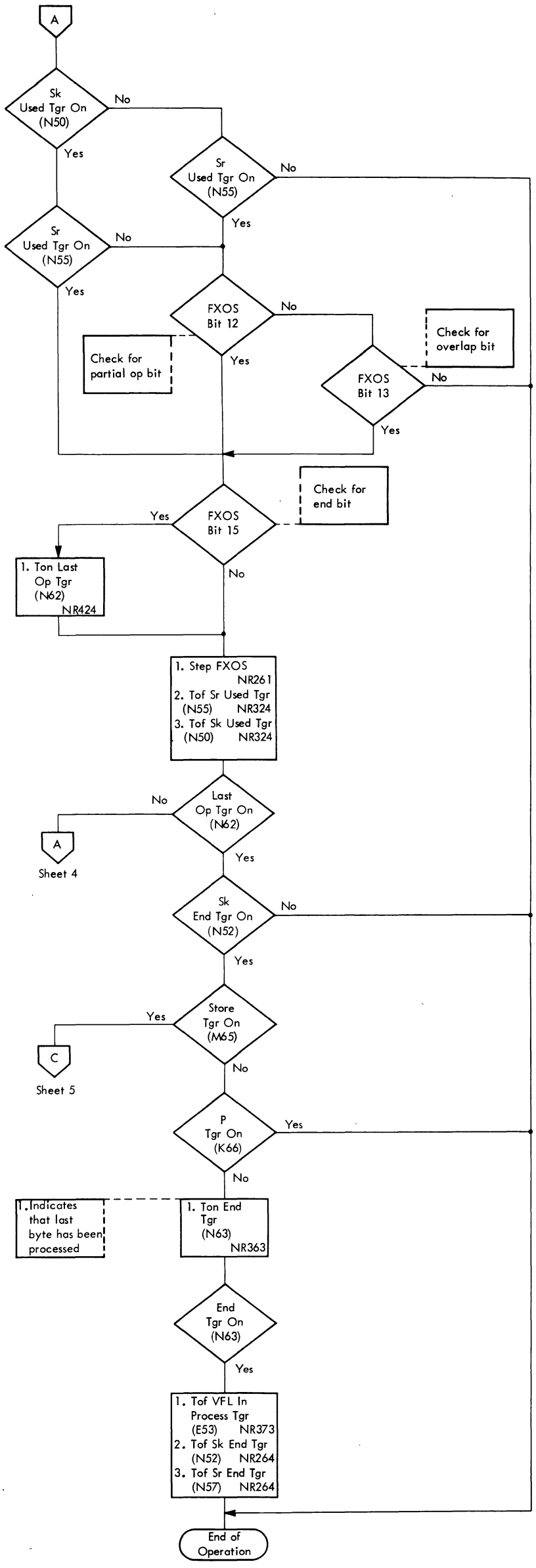


DIAGRAM 5-123. MVO INSTRUCTION (SHEET 6 OF 6)

Pack and Unpack

PACK, UNPK

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
SS	F2, F3																			
									FLB Name				P	O	S	E	Starting Byte Addr		SAR Name	

Objectives:

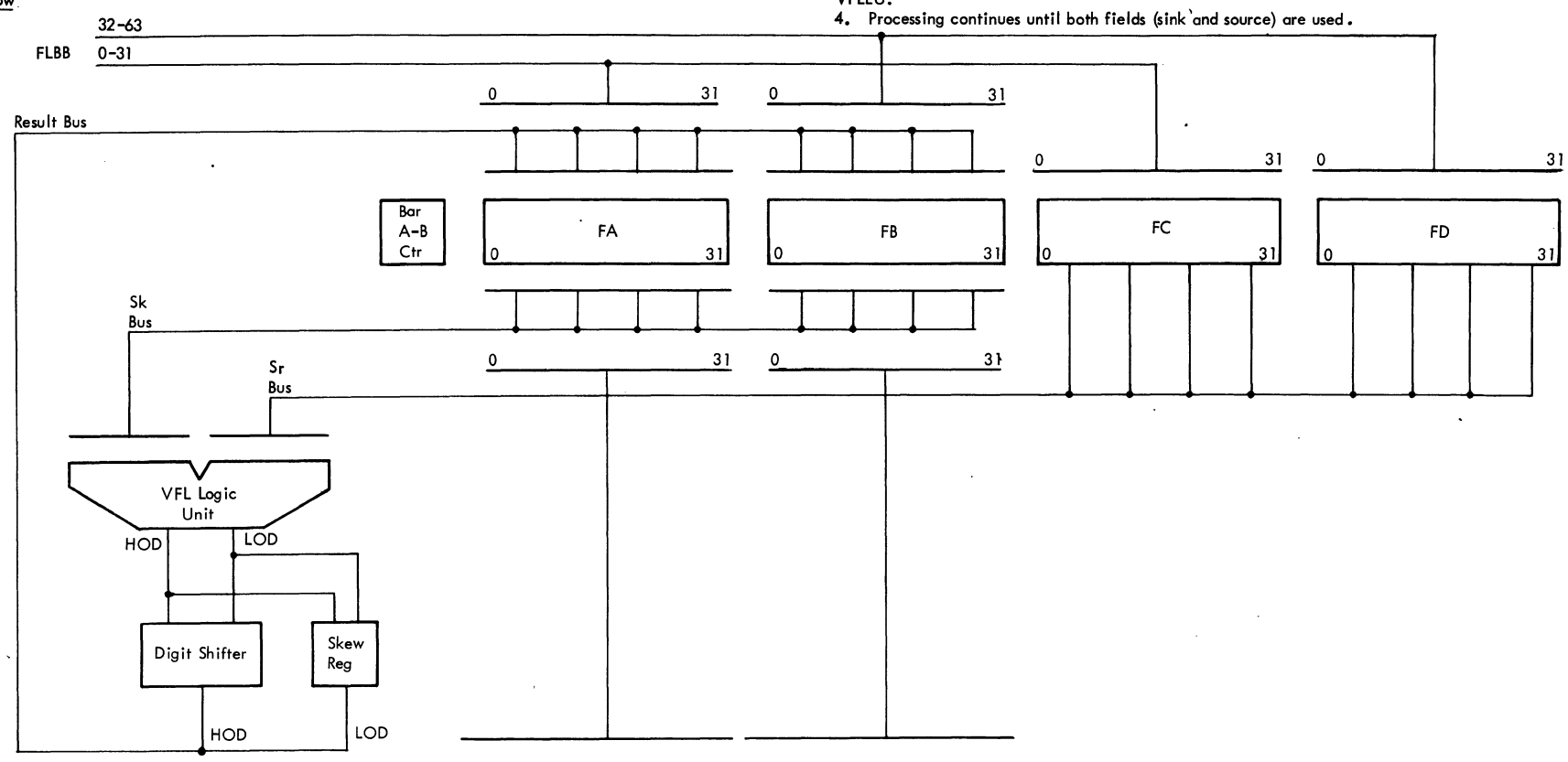
1. Source words are transferred from a specified FLB (FLB name) to source buffer (FC/FD).
2. Bytes are processed one at a time. During overlap processing result bytes are placed in sink and source buffers.
3. Processing is as follows:

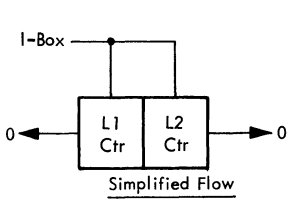
Pack - The LOD from two adjacent bytes are combined into a single, all digit result byte. HOD from these bytes (zone bits) are dropped.

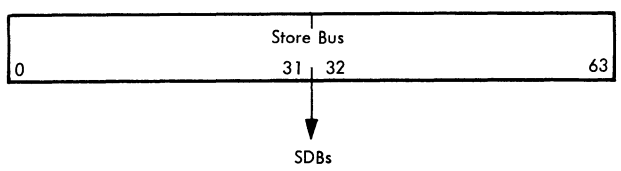
Unpack - The HOD and LOD of a byte are separated and placed in the LOD positions of two result bytes. The HOD of the result bytes (zone bits) are supplied by circuits within the VFLEU.

4. Processing continues until both fields (sink and source) are used.

Data Flow







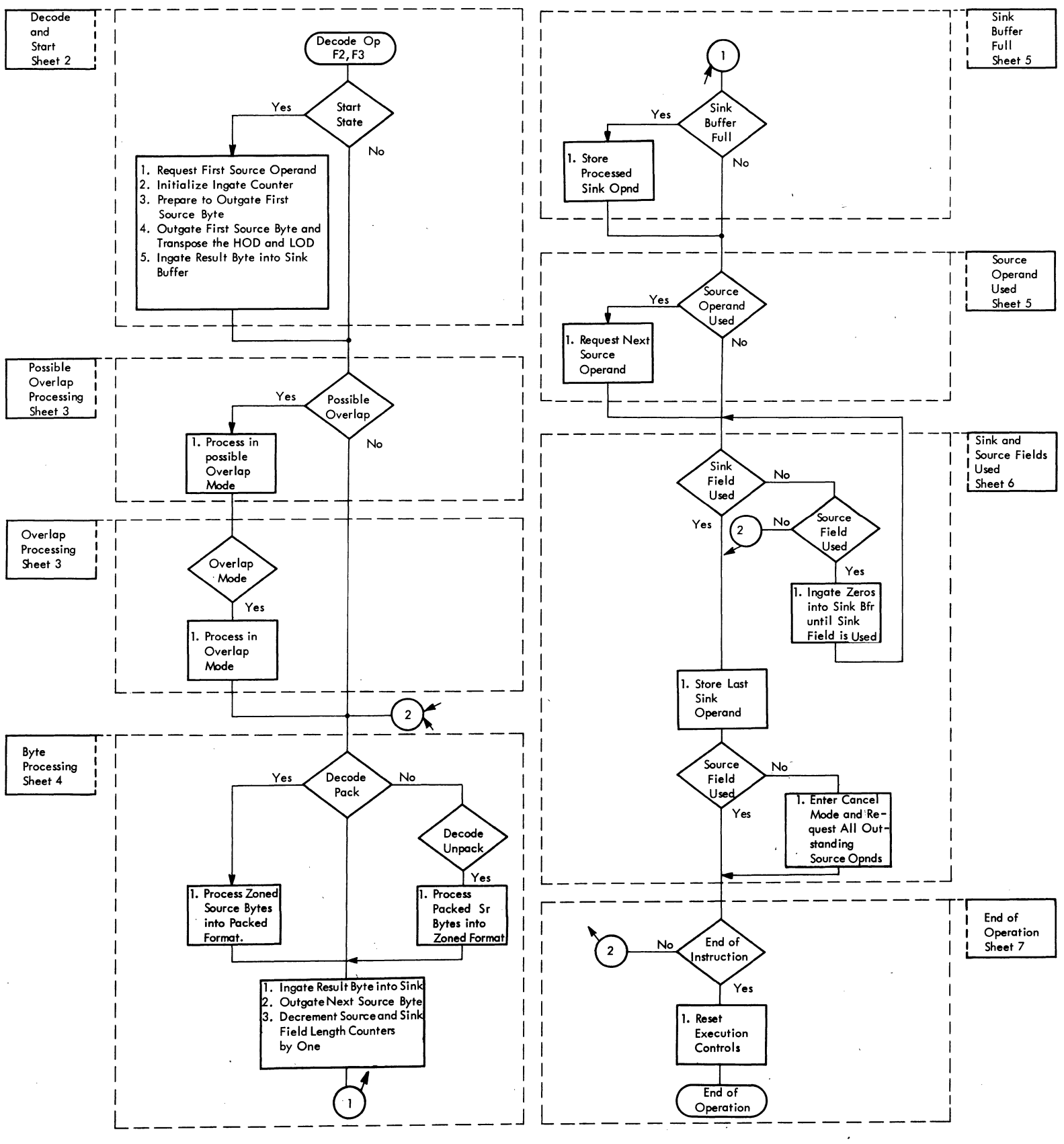


DIAGRAM 5-124. PACK AND UNPK INSTRUCTIONS (SHEET 1 OF 7)

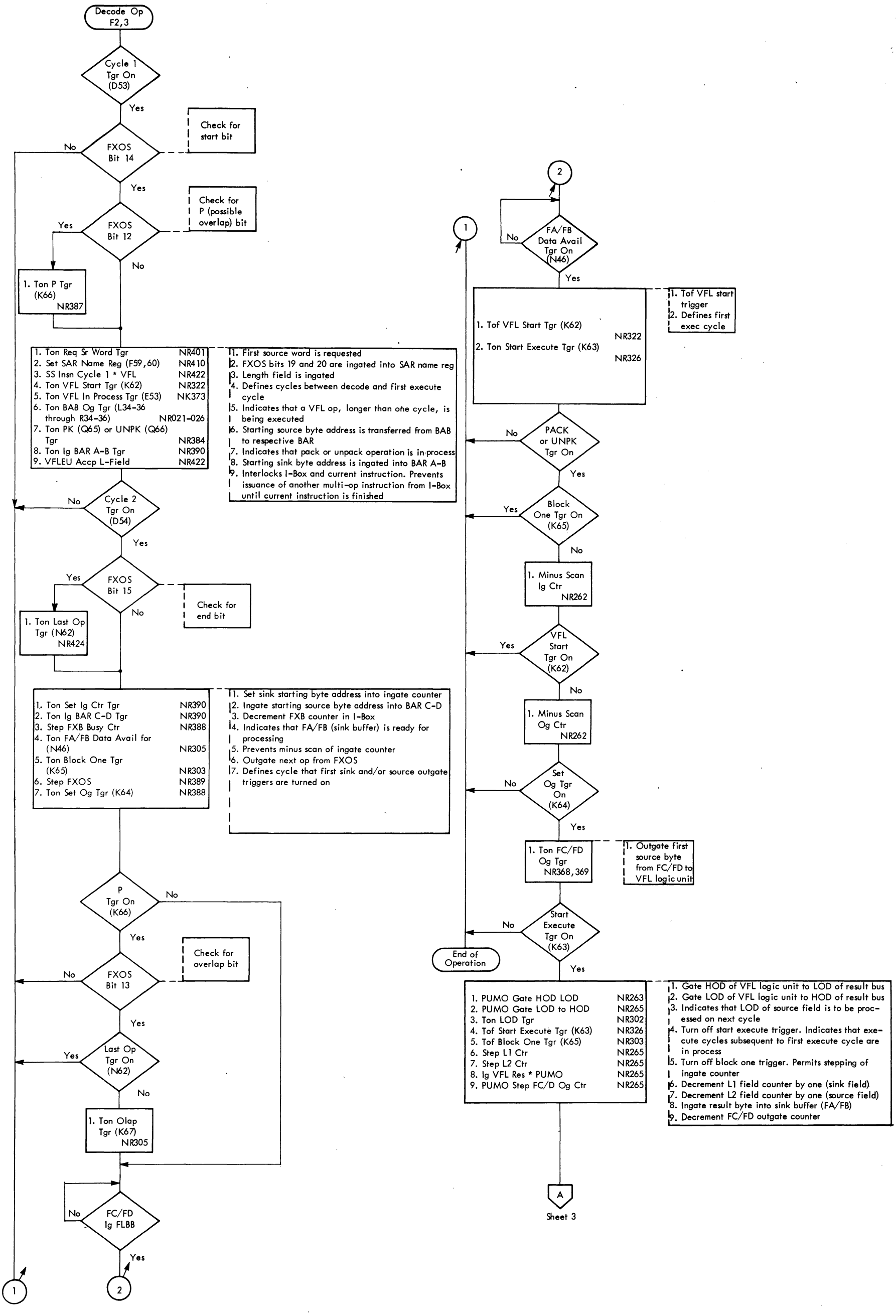


DIAGRAM 5-124. PACK AND UNPK INSTRUCTIONS (SHEET 2 OF 7)

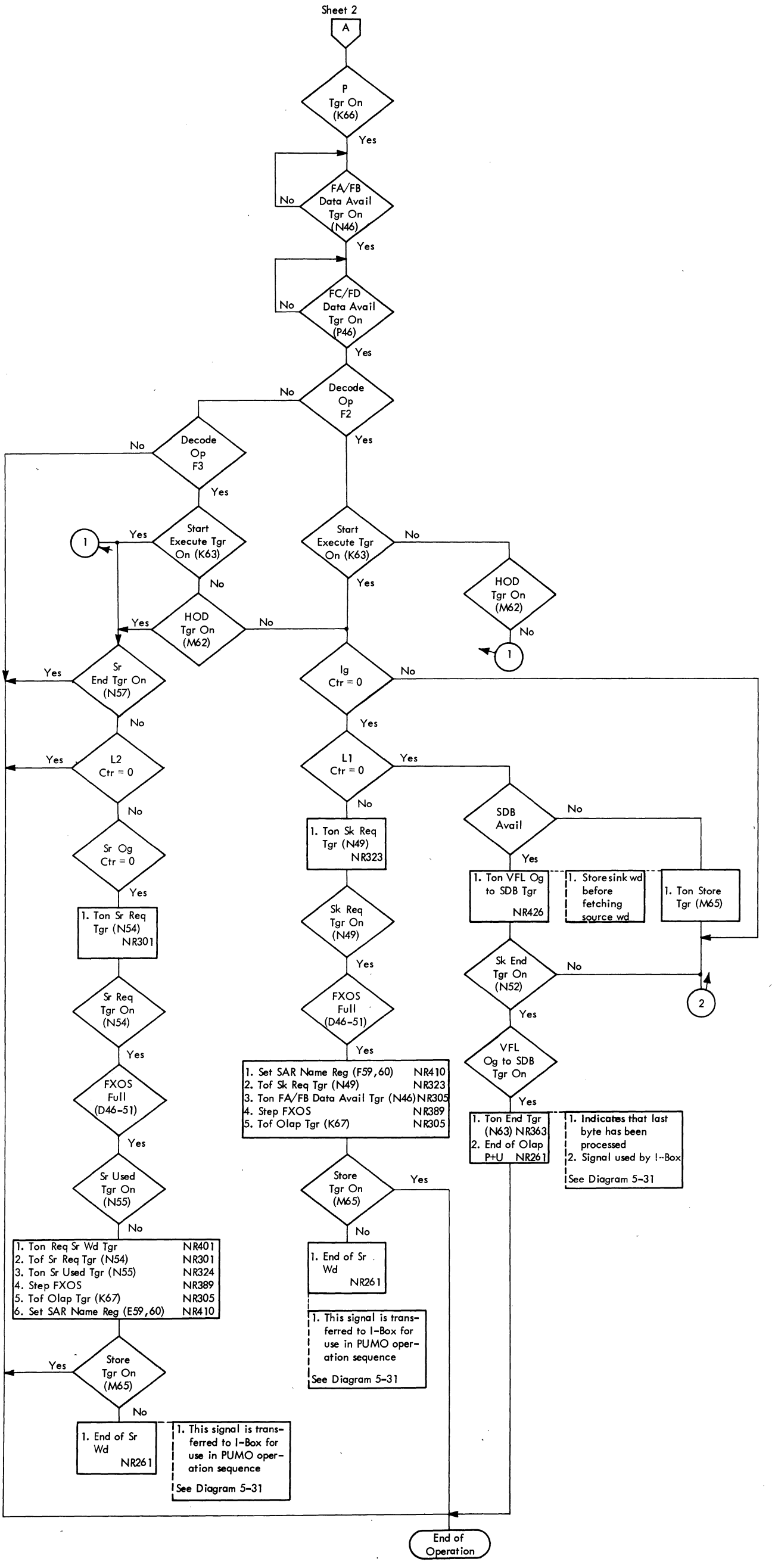


DIAGRAM 5-124. PACK AND UNPK INSTRUCTIONS (SHEET 3 OF 7)

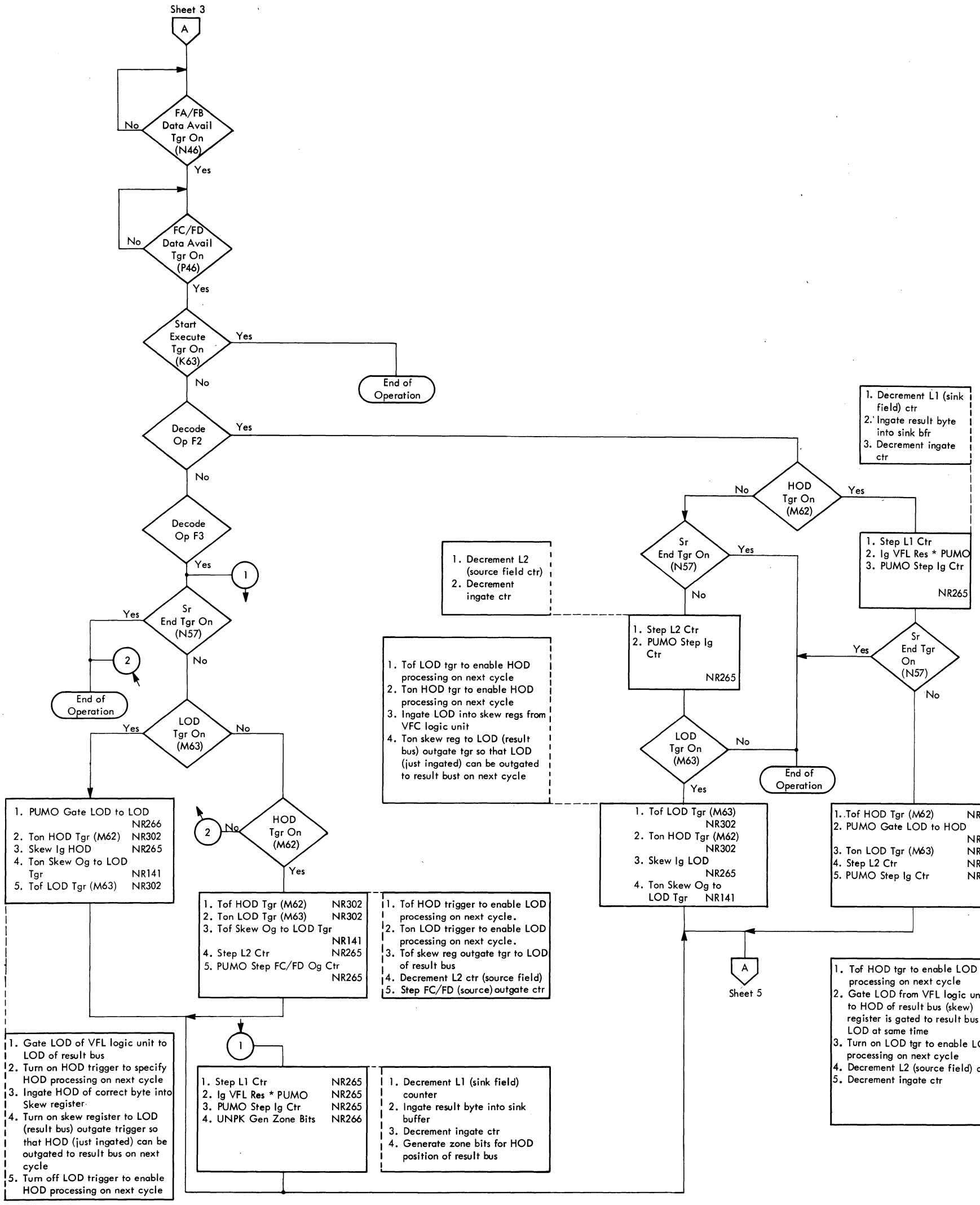


DIAGRAM 5-124. PACK AND UNPK INSTRUCTIONS (SHEET 4 OF 7)

Sink Buffer Full

Source Operand Used

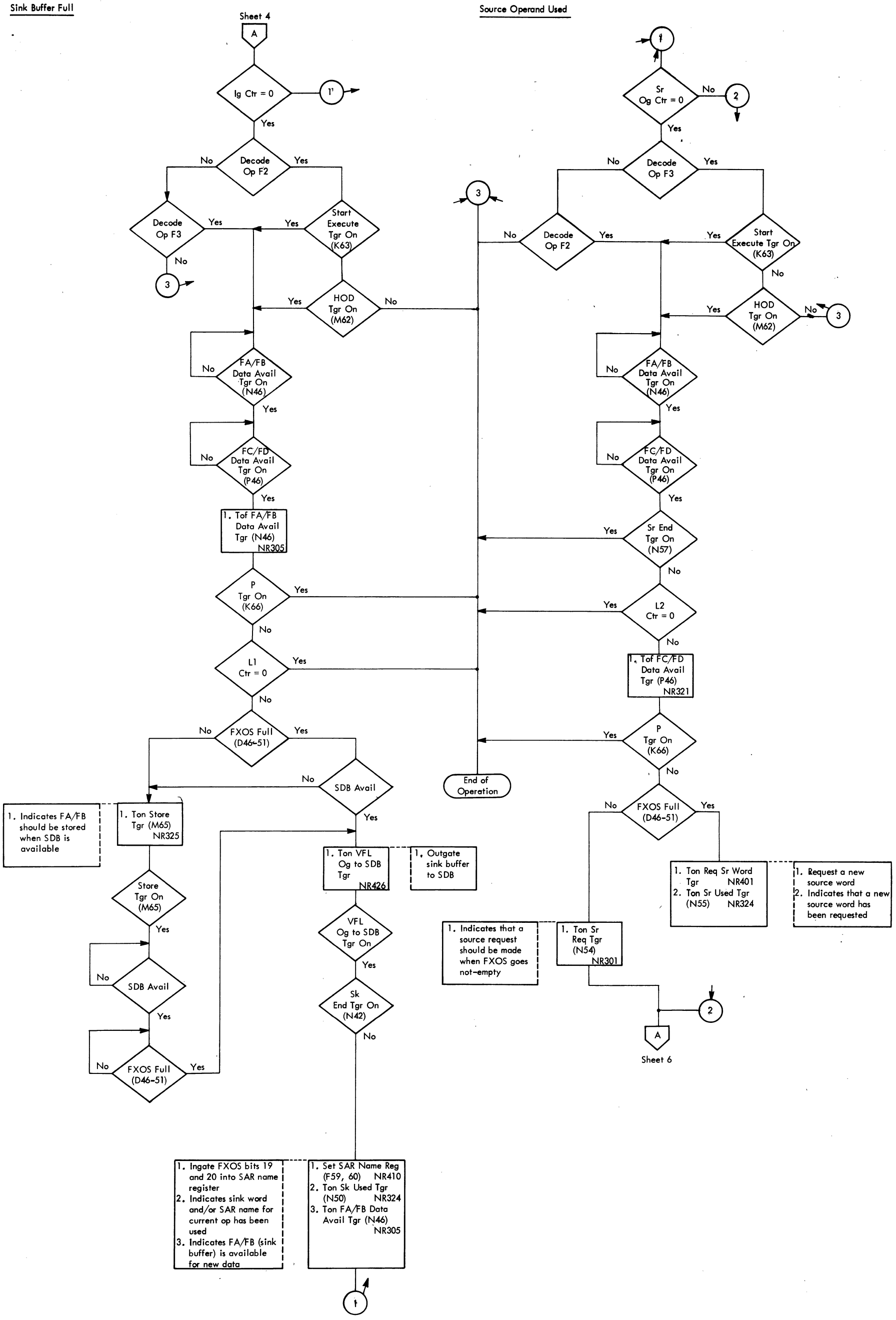


DIAGRAM 5-124. PACK AND UNPK INSTRUCTIONS (SHEET 5 OF 7)

Sink and Source Fields Used

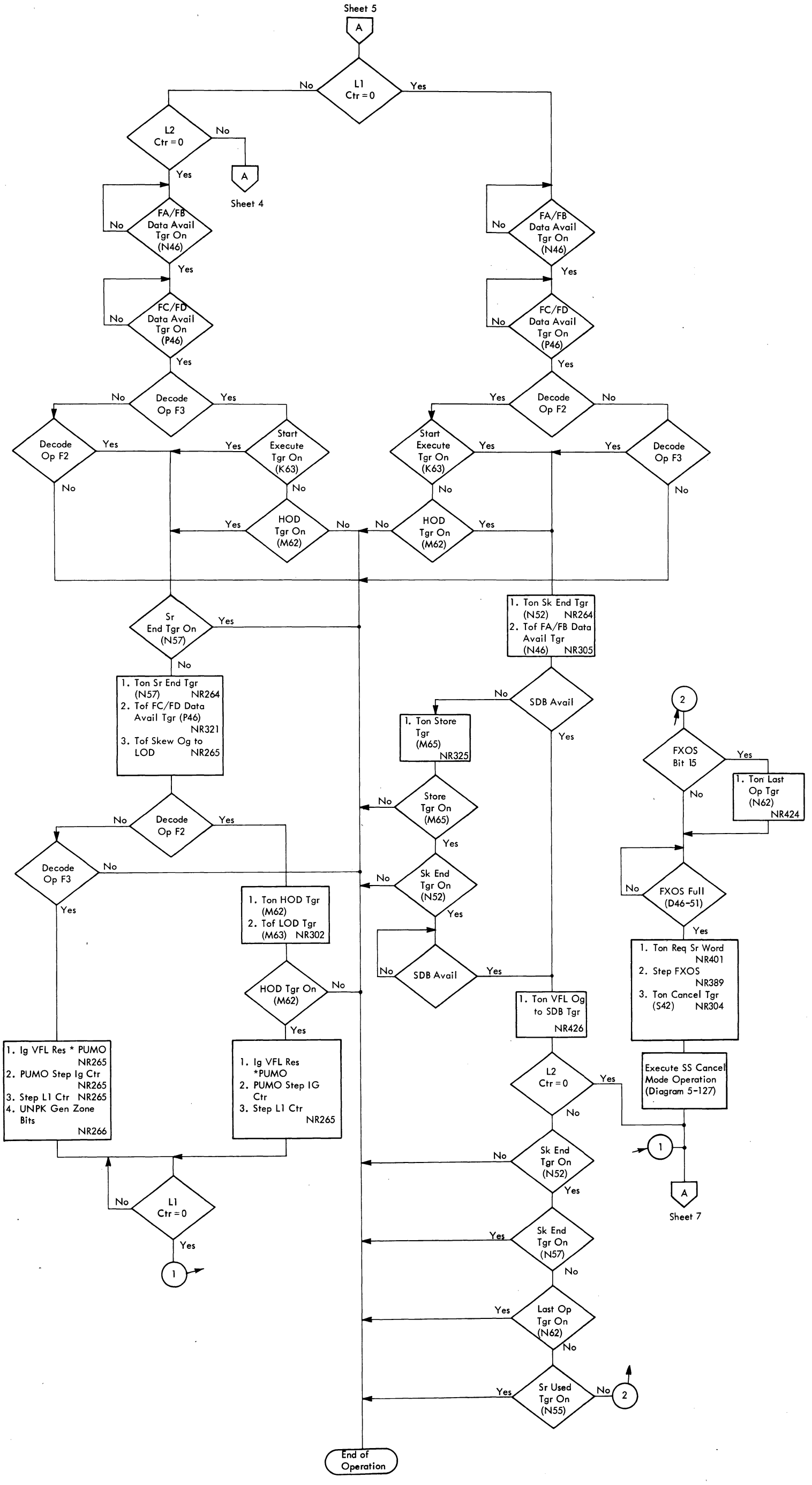


DIAGRAM 5-124. PACK AND UNPK INSTRUCTIONS (SHEET 6 OF 7)

End of Operation

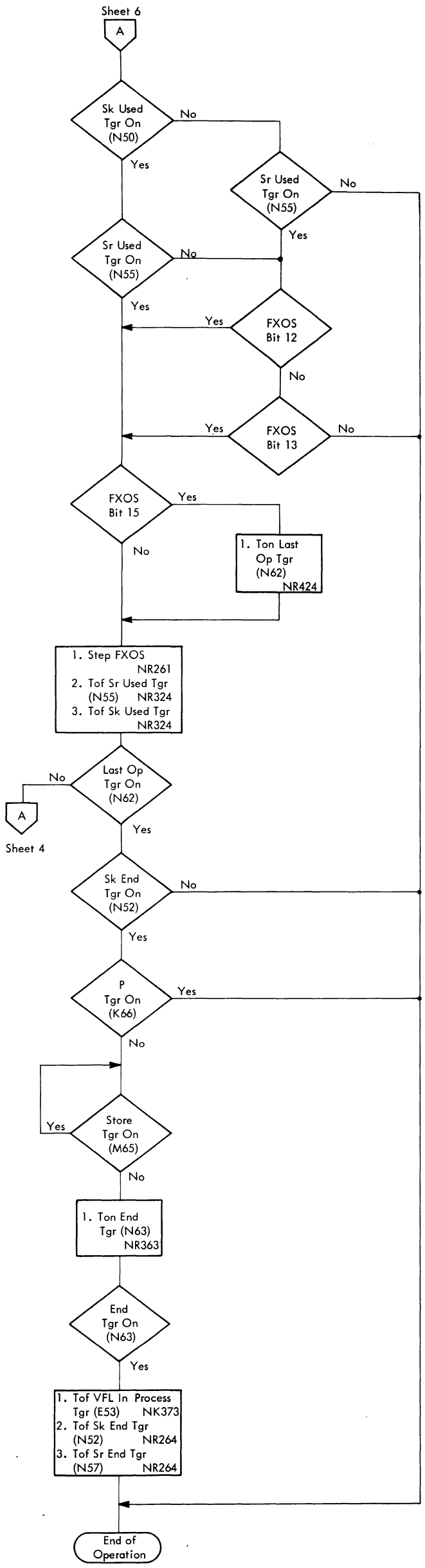
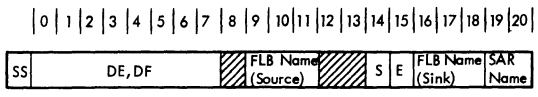


DIAGRAM 5-124. PACK AND UNPK INSTRUCTIONS (SHEET 7 OF 7)

Objectives:

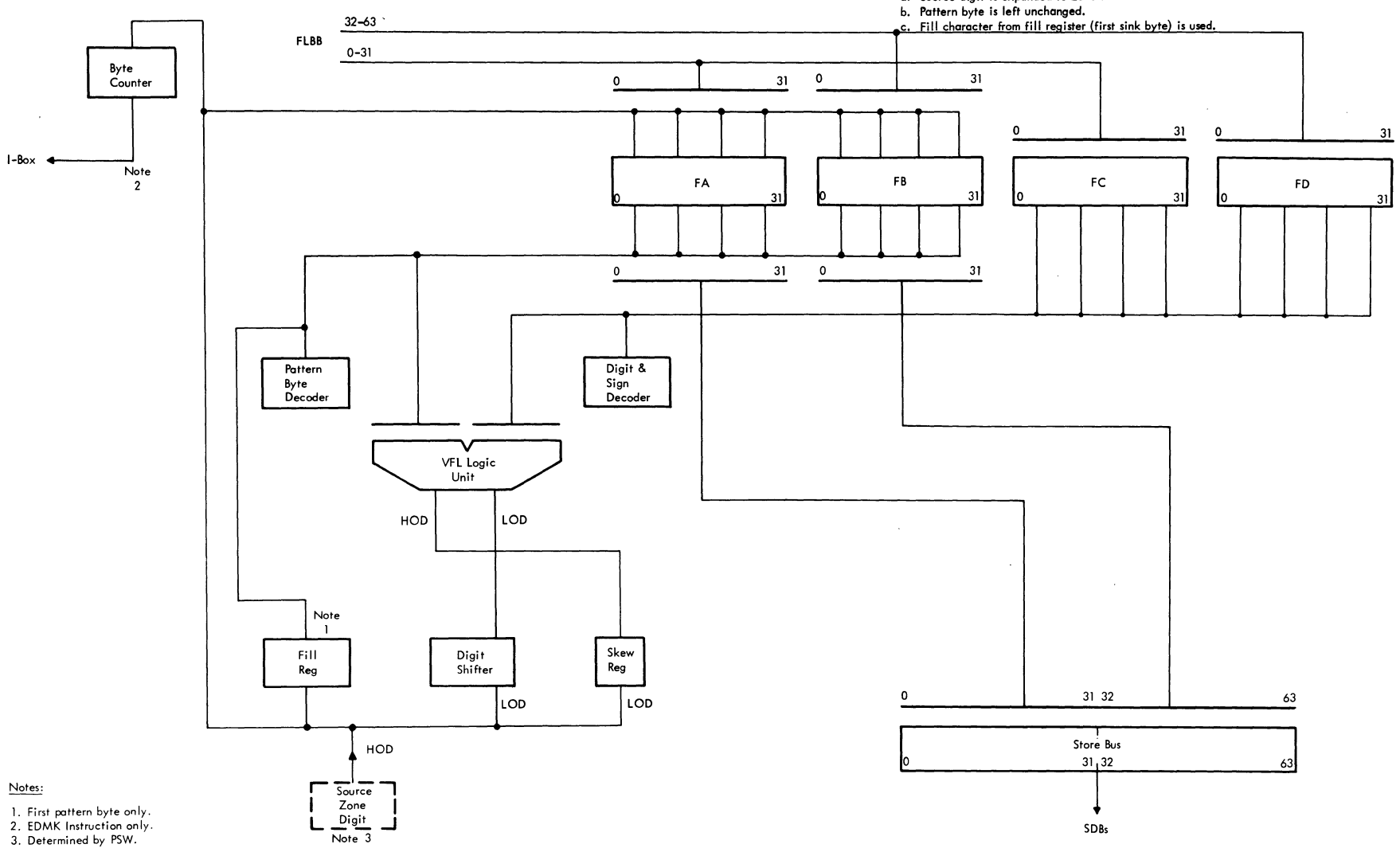
Edit, Edit and Mark

ED, EDMK



1. Source data are changed from packed to zone format and are edited under control of a pattern word (Sink).
2. Edit and Mark sends byte address of first significant result digit to I-Box.
3. During editing process:
   a. Source digit is expanded to zone format.
   b. Pattern byte is left unchanged.
   c. Fill character from fill register (first sink byte) is used.

Data Flow



Notes:

1. First pattern byte only.
2. EDMK Instruction only.
3. Determined by PSW.

Simplified Flow

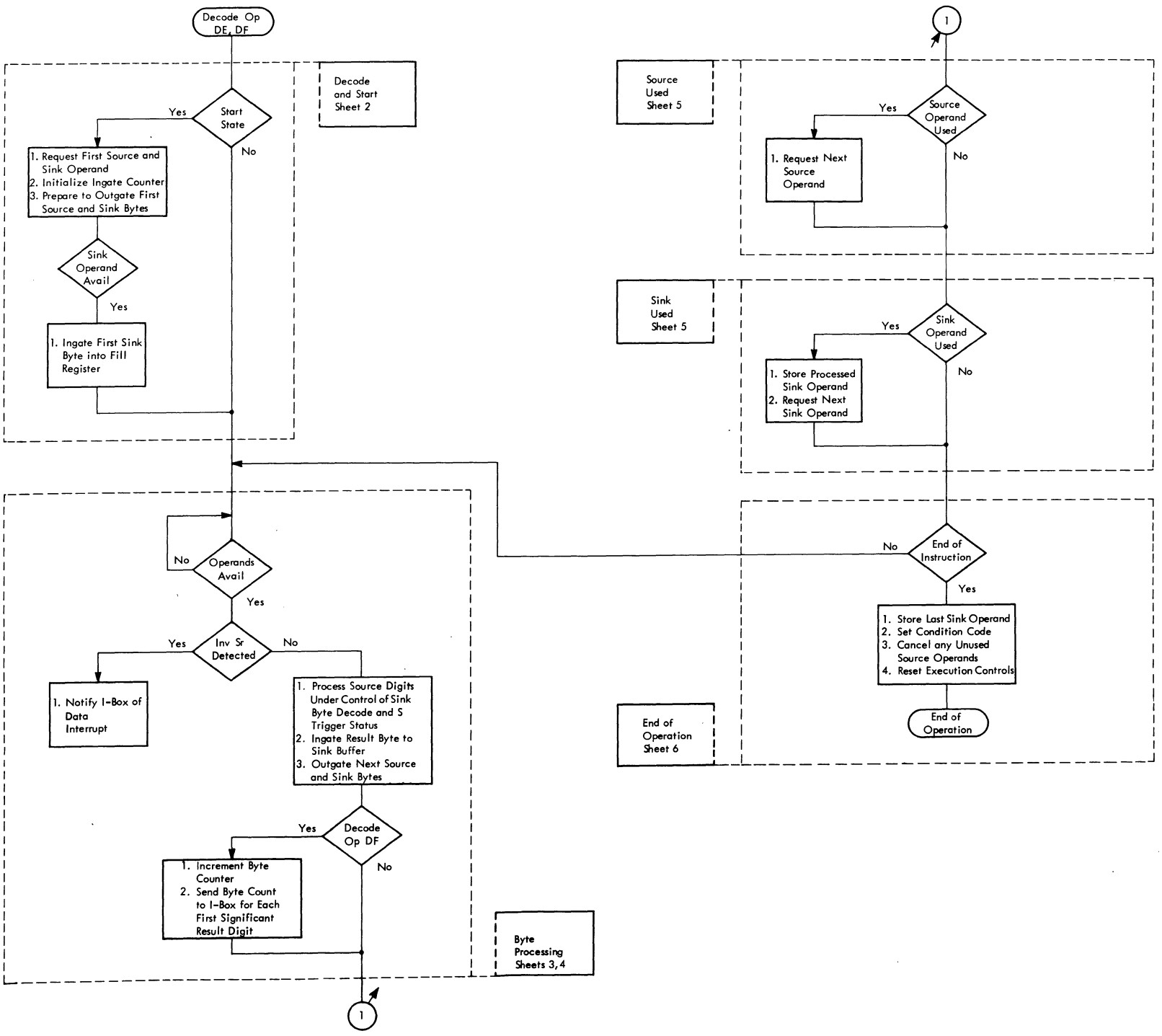


DIAGRAM 5-125. ED AND EDMK INSTRUCTIONS (SHEET 1 OF 6)

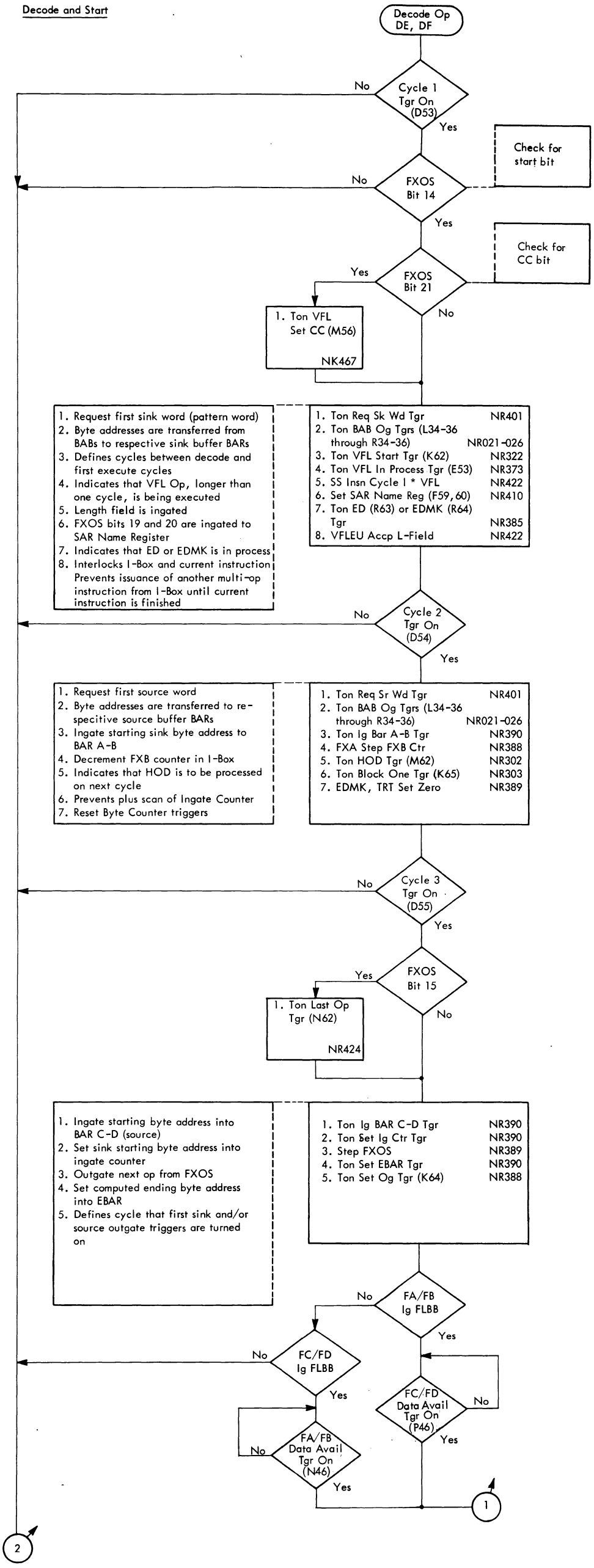


DIAGRAM 5-125. ED AND EDMK INSTRUCTIONS (SHEET 2 OF 6)

Byte Processing (Part I)

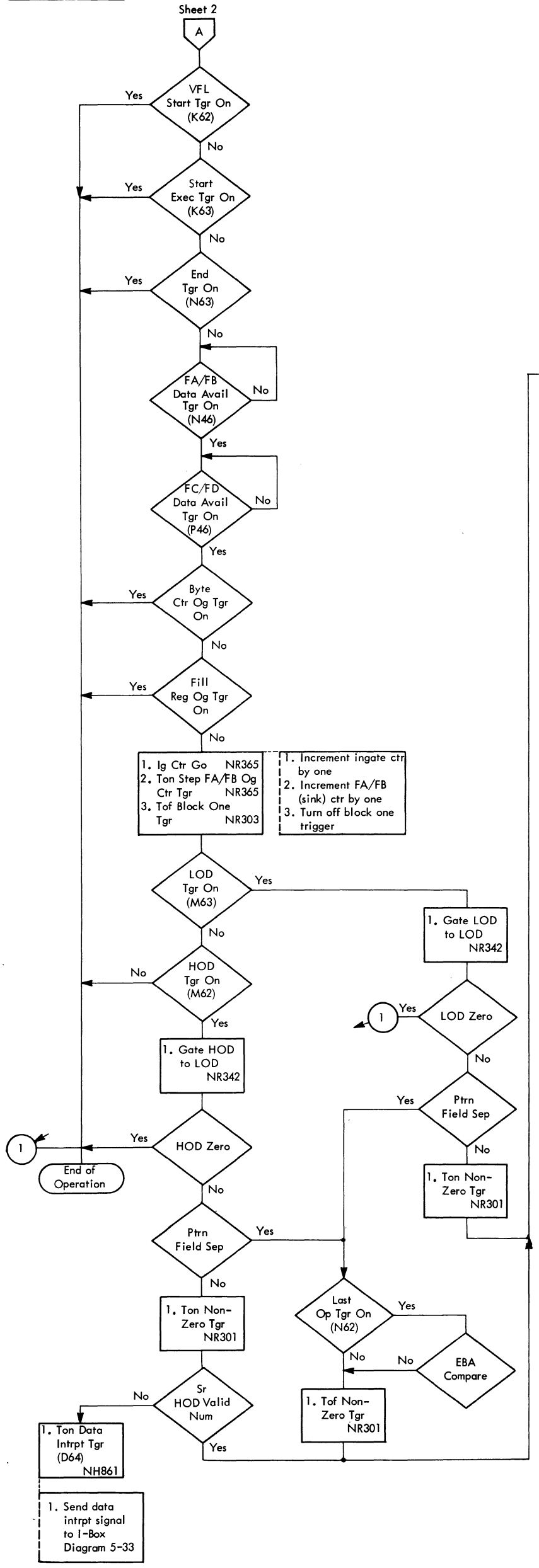


DIAGRAM 5-125. ED AND EDMK INSTRUCTIONS (SHEET 3 OF 6)

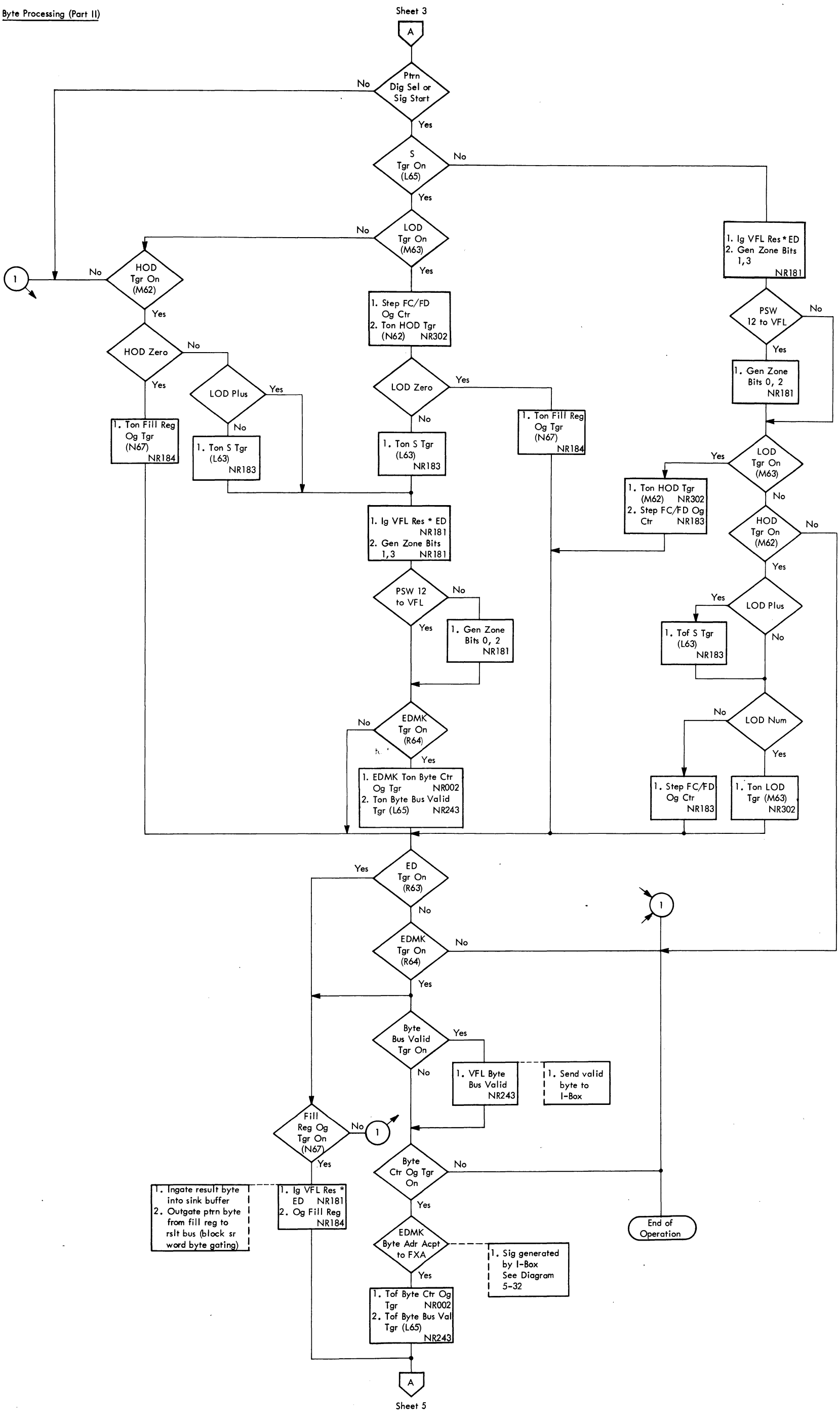


DIAGRAM 5-125. ED AND EDMK INSTRUCTIONS (SHEET 4 OF 6)

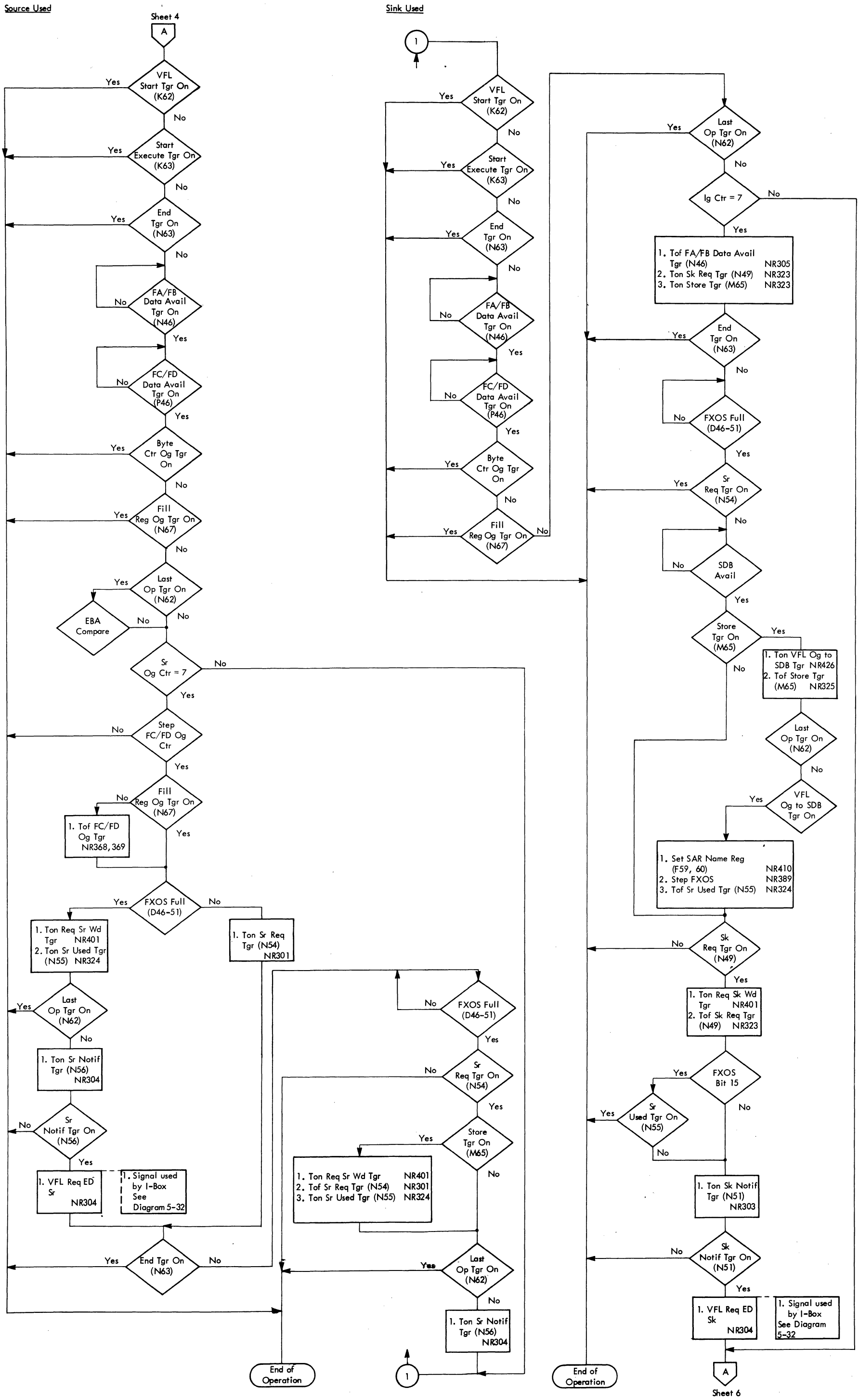


DIAGRAM 5-125. ED AND EDMK INSTRUCTIONS (SHEET 5 OF 6)

End of Operation

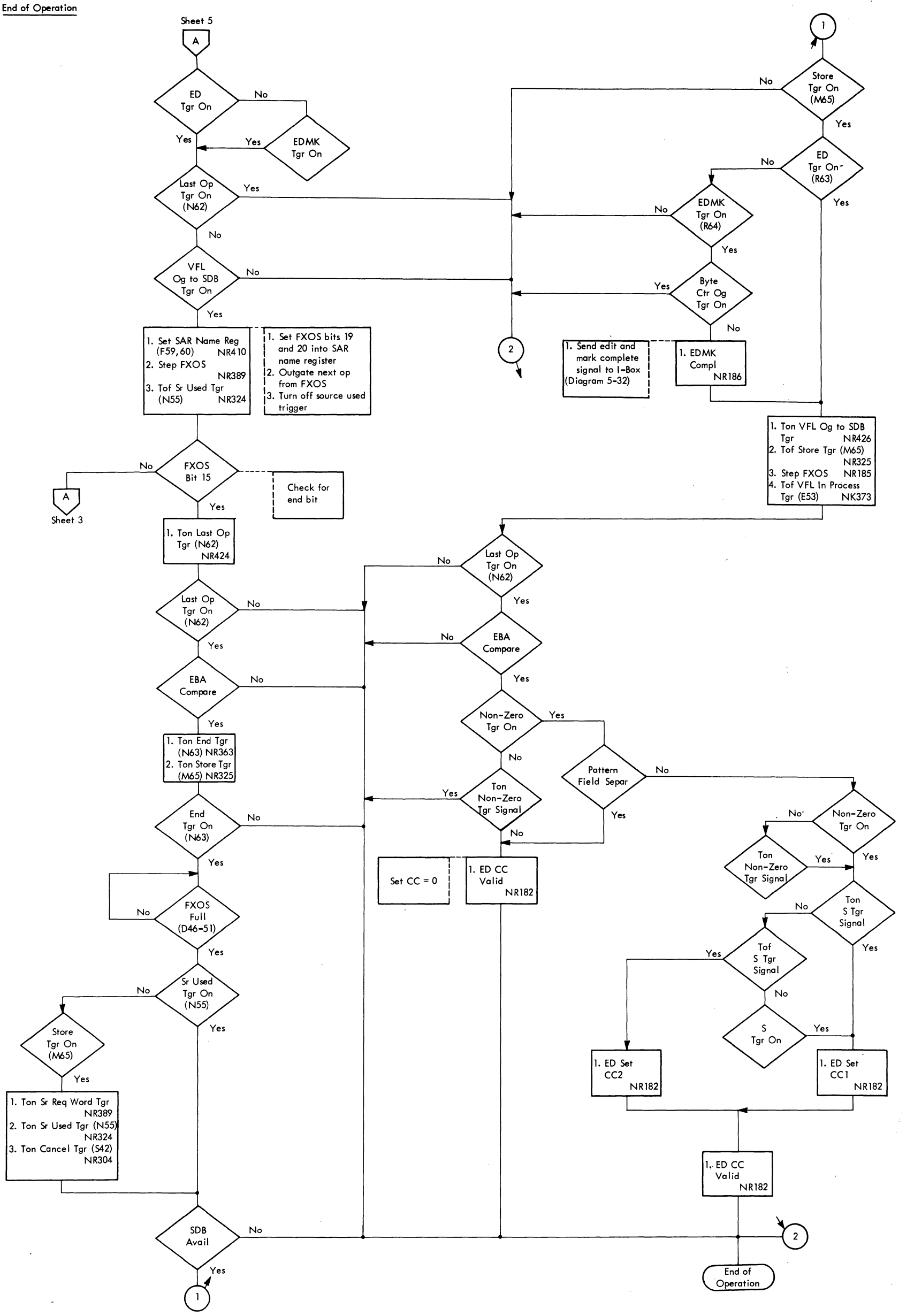


DIAGRAM 5-125. ED AND EDMK INSTRUCTIONS (SHEET 6 OF 6)

Objectives:

Translate, Translate and Test

TR, TRT

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
* SS	DC, DD												P		S		FLB Name (Argument)			SAR Name (TR Only)	

** SS	DC, DD								FLB Name (Table Word)				P ***			E	FLB Name (Argument)			SAR Name (TR Only)	

*First op only **Subsequent ops ***P bit not present in end op

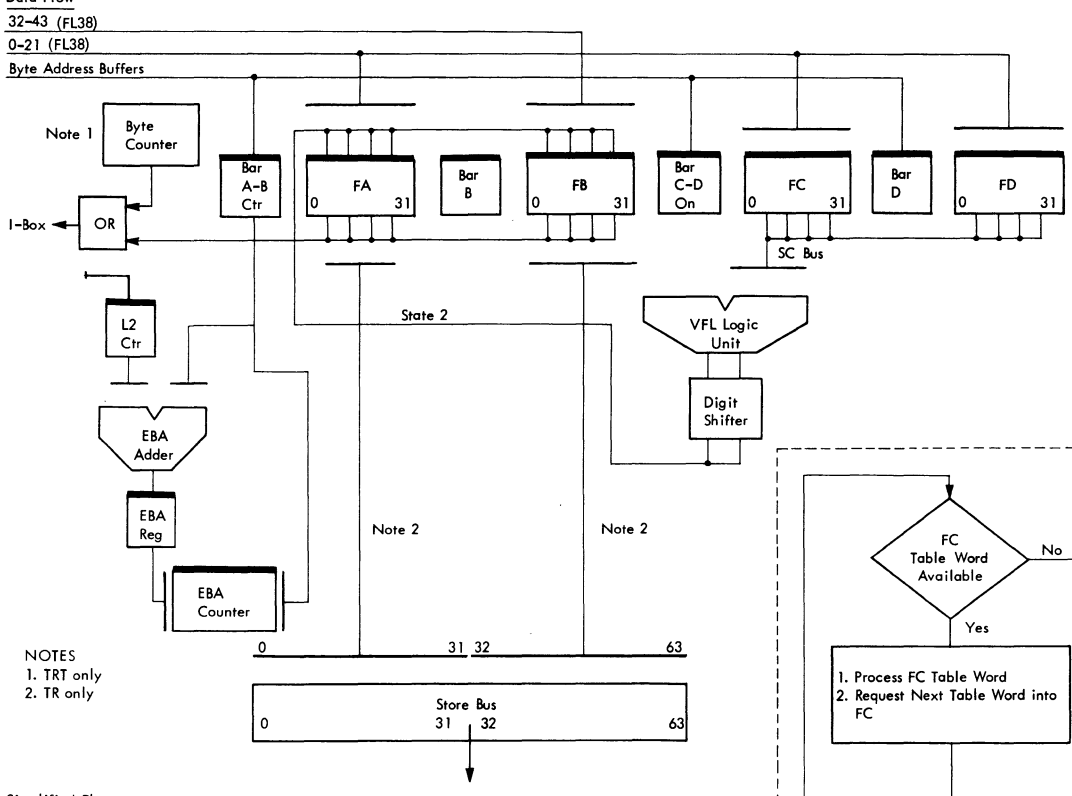
1. Data bytes from the argument word are used to reference data bytes in the table word.
2. TR operation replaces the original argument bytes with the table word bytes.
3. TRT operation checks table word bytes for zero/non-zero condition. Non-zero condition causes instruction to terminate.
4. Basic operation is as follows:
   a. Argument bytes are sent to I-Box to form address of table word containing desired table byte.
   b. Addressed table words are alternately gated to FC and FD.
   c. Desired table bytes are outgated from table words in FC and FD to VFL logic unit. In IR operation, these bytes replace the argument word. In TRT operation, these bytes are checked and the condition code is set accordingly.

Condition Code:

TR - Code remains unchanged

TRT - 0 All function bytes are zero
1 Non-zero function byte before the first operand is exhausted
2 Last function byte is non-zero

Data Flow



Simplified Flow

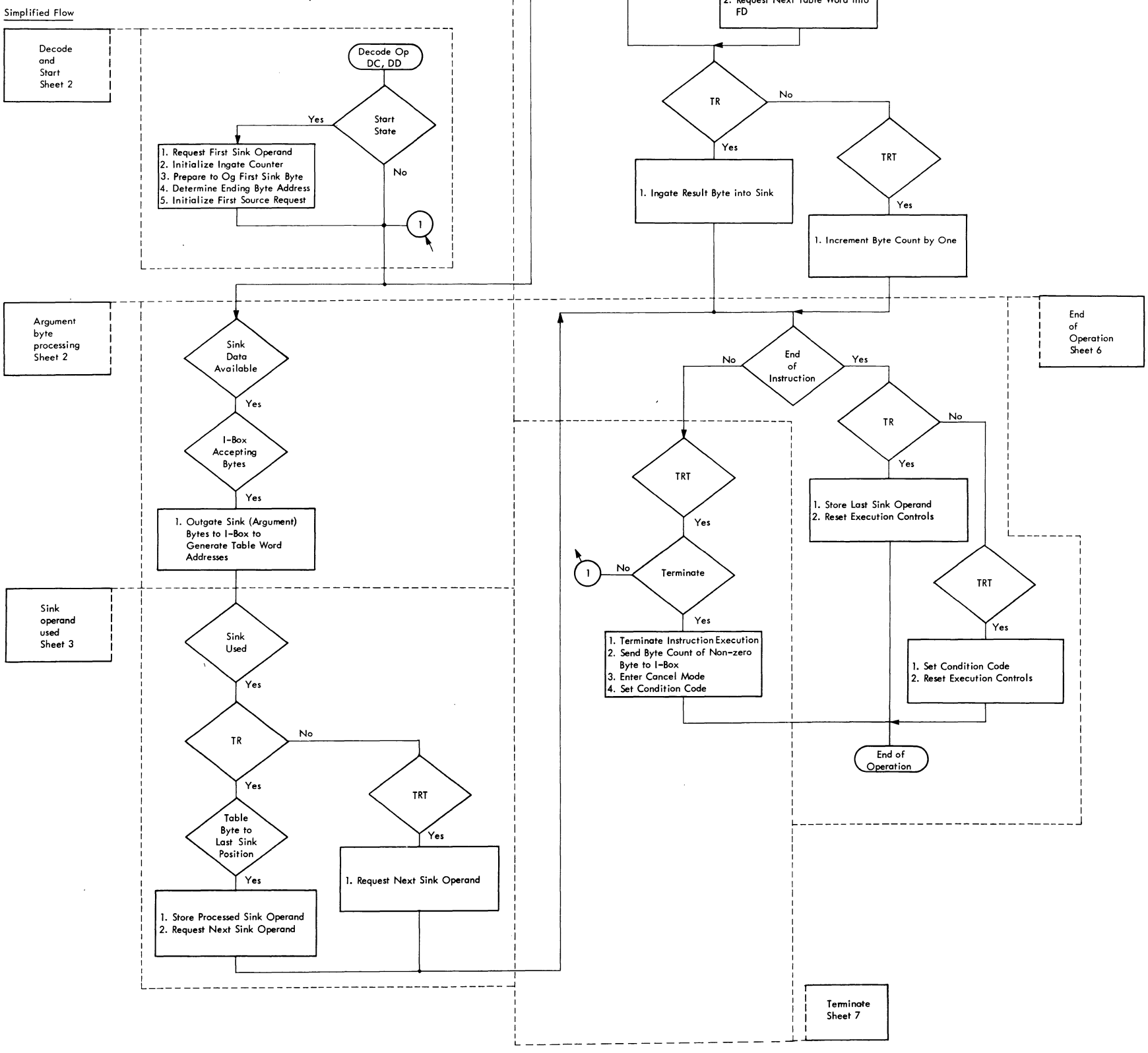


DIAGRAM 5-126. TR AND TRT INSTRUCTIONS (SHEET 1 OF 7)

Decode and Start

Decode Op DC, DD

FXOS Bit 14 — No → Argument Byte Processing (1)

Yes

Cycle 1 Tgr On (D53) — No → Any BAB Og Tgr On (L34-36-R34-36) — No → Cycle 2 Tgr On (D54) — No → Cycle 3 Tgr On (D55) — No → VFL Start Tgr On (K62) — No → FA/FB Ig FLBB — No → End of Operation

Yes

FXOS Bit 12 — Yes → 1. Ton P Tgr (K66) NR387

No

FXOS Bit 21 — Yes → 1. Ton VFL Set CC Tgr (M56) NR467

No

1. Ton Req Sk Wd NR401
2. Ton BAB Og Tgr (L34-36 Through R34-36) NR021-026
3. Ton VFL Start Tgr (K62) NR322
4. Ton VFL In Process Tgr (E53) NR373
5. Ton TR (R65) or TRT (R66) Tgr NR381
6. Set SAR Name Reg (F59, 60) NR410
7. SS Insn Cycle * VFL NR422
8. VFLEU Acpt L-Field NR422

1. Request first sink word (argument)
2. Sink byte address is outgated from BABS
3. Defines cycles between decode and first execute cycle
4. Indicates that VFL op, longer than one cycle, is being executed
5. Indicates that TR or TRT instructions are in process
6. FXOS bits 19 and 20 are ingated to SAR name register
7. Length field is ingated
8. Interlocks I-Box and current instructions; prevents issuance of another multi-op instruction from I-Box until current instruction is finished

Any BAB Og Tgr On (L34-36-R34-36)

Yes

FLB Sk Adr = $1 \cdot \overline{2}$ — No → FLB Sk Adr = $\overline{1} \cdot 2$ — Yes → 1. Ig BAR D NR423

Yes

1. Ig BAR C-D NR423

Cycle 2 Tgr On (D54)

Yes

1. Ton Ig BAR A-B Tgr NR390
2. Step FXB Busy Ctr NR388
3. EDMK, TRT Set Zero NR389

Cycle 3 Tgr On (D55)

Yes

1. Ton Set EBAR Tgr NR390
2. Ton Set Ig Ctr Tgr NR390
3. Step FXOS NR389
4. Ton Set TW Ctr Tgr NR228
5. Ton Start Req FC Tgr NR425
6. Ton Start Req FD Tgr NR425
7. Ton Use FC Tgr (P51) NR241
8. Ton FC Next Tgr (P49) NR221
9. Set FXB/FLB Reg (F56-58) NR501-503
10. Ton FA/FB Og Tgr NR366-367

1. Set computed ending byte address into EBAR
2. Set starting sink byte address into ingate counter
3. Outgate next op from FXOS
4. Set starting sink byte address into TW counter
5. Indicates that a request for FC data must be made after start as soon as FXOS goes not empty
6. Indicates that a request for FD data must be made after start as soon as FXOS goes not empty
7. Indicates that next table byte should be processed from FC
8. Indicates that next request is for FC
9. Set address of FLB containing current argument word into FXB/FLB name register
10. Turn on outgates for first argument bytes (FA/FB)

VFL Start Tgr On (K62)

Yes

FA/FB Ig FLBB

Yes

1. Ton FA/FB Data Avail Tgr (N46) NR305
2. Ton Byte Bus Valid Tgr (L65) NR243
3. Tof VFL Start Tgr (K62) NR322
4. Ton Start Execute Tgr (K63) NR326

1. Indicates that FA/FB are available for processing
2. Notifies I-Box that a Valid byte is on byte bus
3. Turns off VFL start trigger
4. Defines first execute cycle

(1)

End of Operation

Argument Byte Processing

(1)

FA/FB Ig FLBB — No → Term Tgr On (M66) — No → End Tgr On (N63) — No → Byte Bus Valid Tgr On (L65) — No → EBA Compare — No → Olap Process Tgr On (L66) — No → FA/FB Data Avail Tgr On (N46) — No → Arg Byte Accp — No → P Tgr On (K66) — No → EBA Compare — No → Olap Process Tgr On (L66) — No → End of Operation

Yes

Term Tgr On (M66) — Yes → End Tgr On (N63) — Yes → Byte Bus Valid Tgr On (L65) — Yes → P Tgr On (K66) — Yes → EBA Compare — Yes → 1. Last Arg Byte NR227

1. This signal sent to I-Box along with last argument byte in sink word (FA/FB)

FA/FB Data Avail Tgr On (N46) — Yes → Arg Byte Accp — Yes → P Tgr On (K66) — Yes → EBA Compare — Yes → Olap Process Tgr On (L66) — Yes → 1. Step FA/FB Og Ctr → A

Sheet 3

1. Ton Byte Bus Valid Tgr (L65) NR243
2. Ton FA/FB Data Avail Tgr (N46) NR305

1. Indicates that a valid argument byte is on the byte bus
2. Indicates that FA/FB data are available for processing

1. Byte Bus Valid NR243

Notifies I-Box that a valid arg byte is on the byte bus

1. Signal supplied by I-Box whenever argument bytes are being accepted by I-Box (Diagram 5-29)

DIAGRAM 5-126. TR AND TRT INSTRUCTIONS (SHEET 2 OF 7)

Sink Operand Used

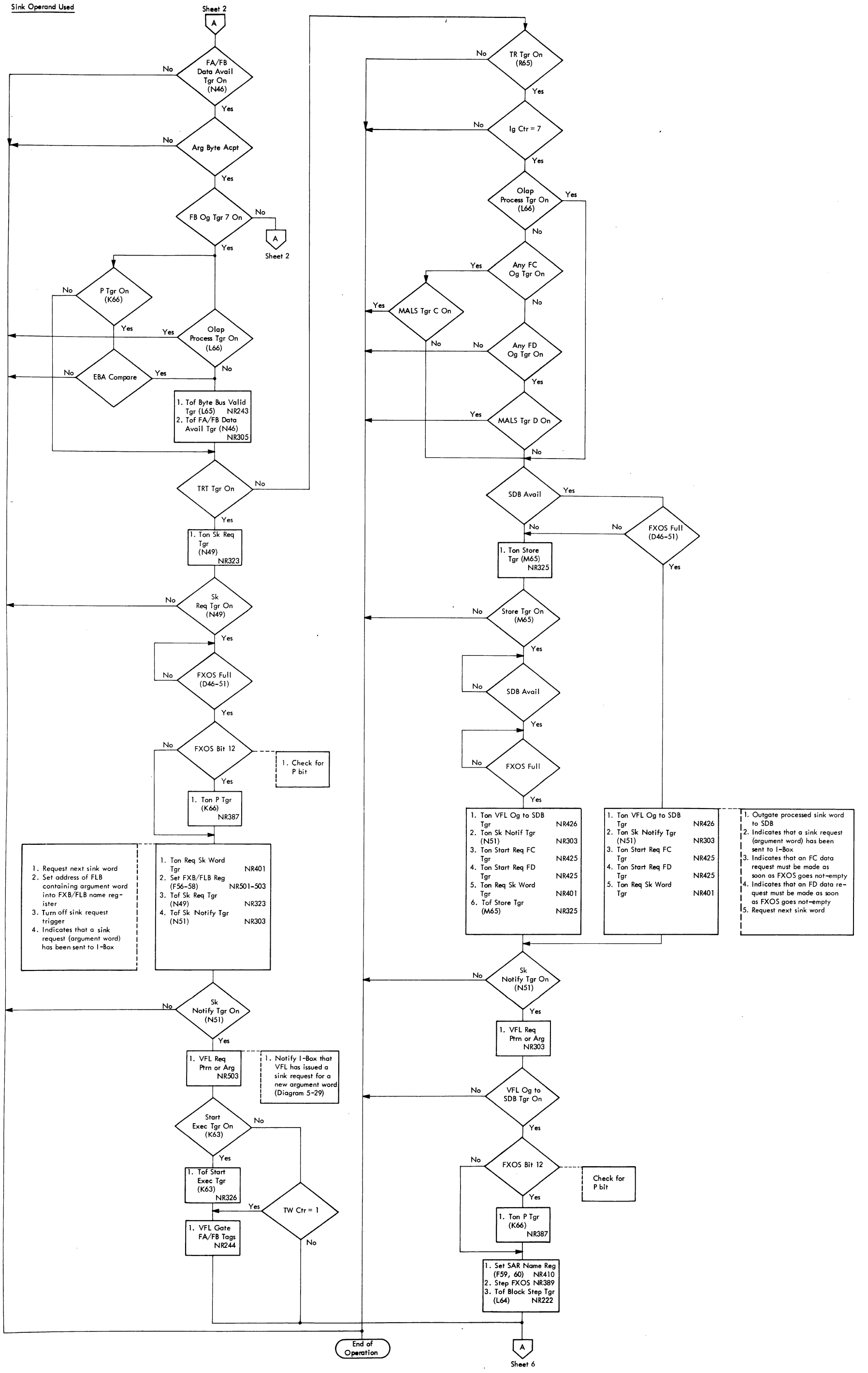


DIAGRAM 5-126. TR AND TRT INSTRUCTIONS (SHEET 3 OF 7)

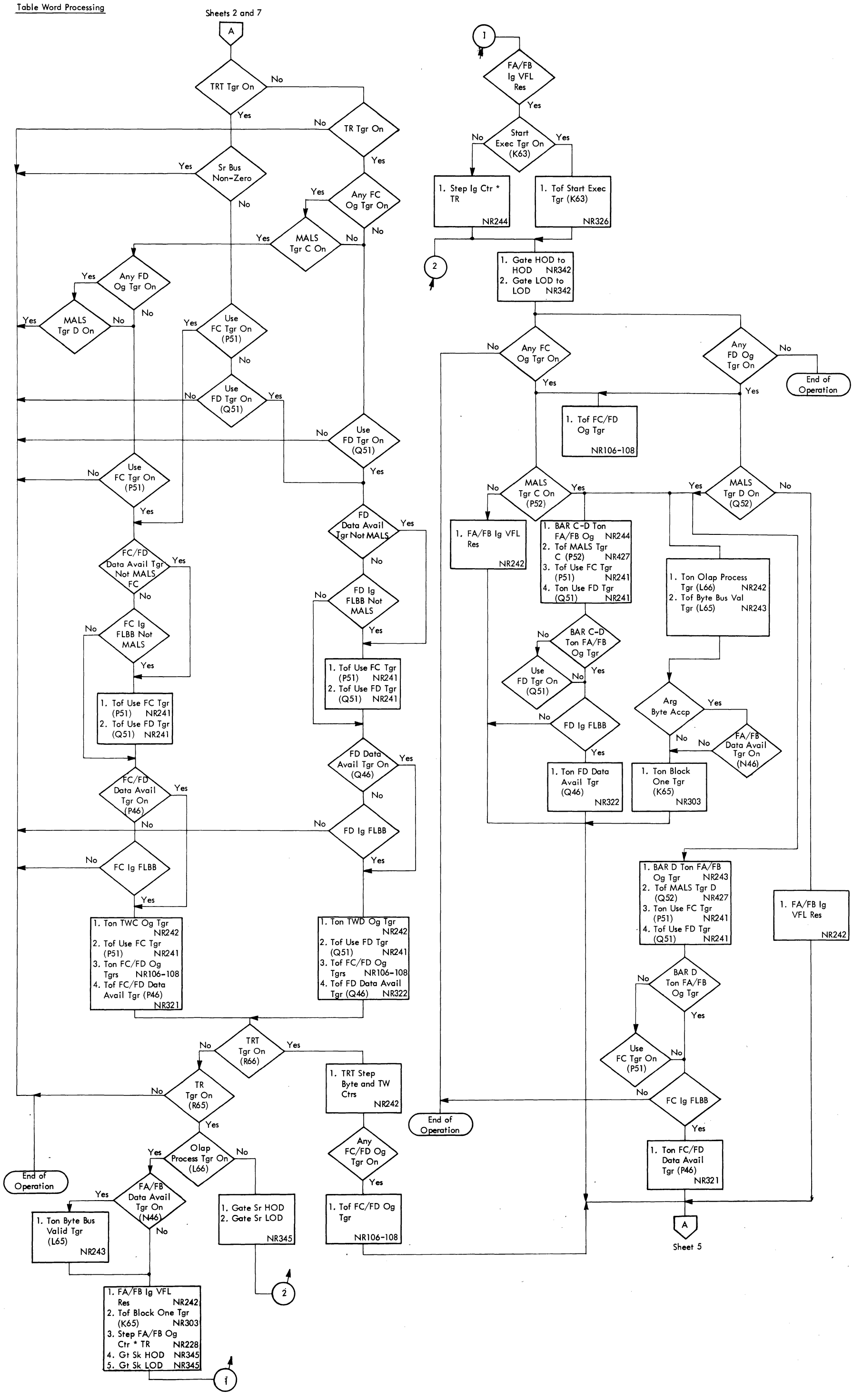


DIAGRAM 5-126. TR AND TRT INSTRUCTIONS (SHEET 4 OF 7)

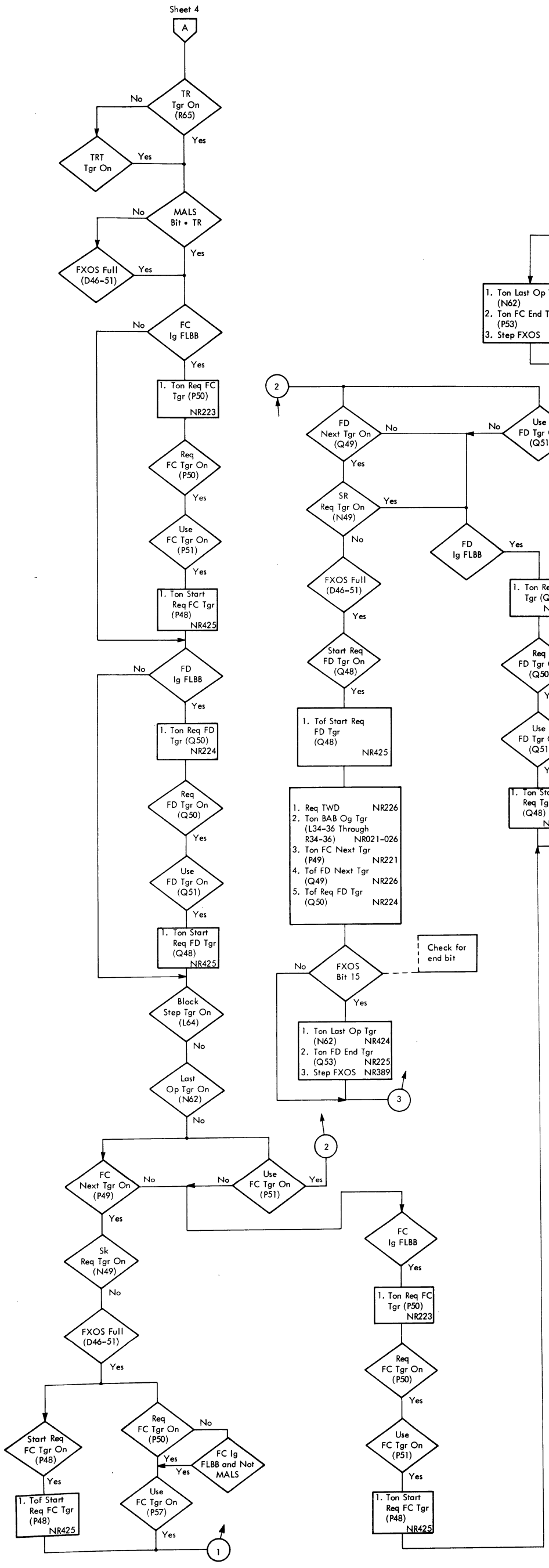


DIAGRAM 5-126. TR AND TRT INSTRUCTIONS (SHEET 5 OF 7)

End Of Operation

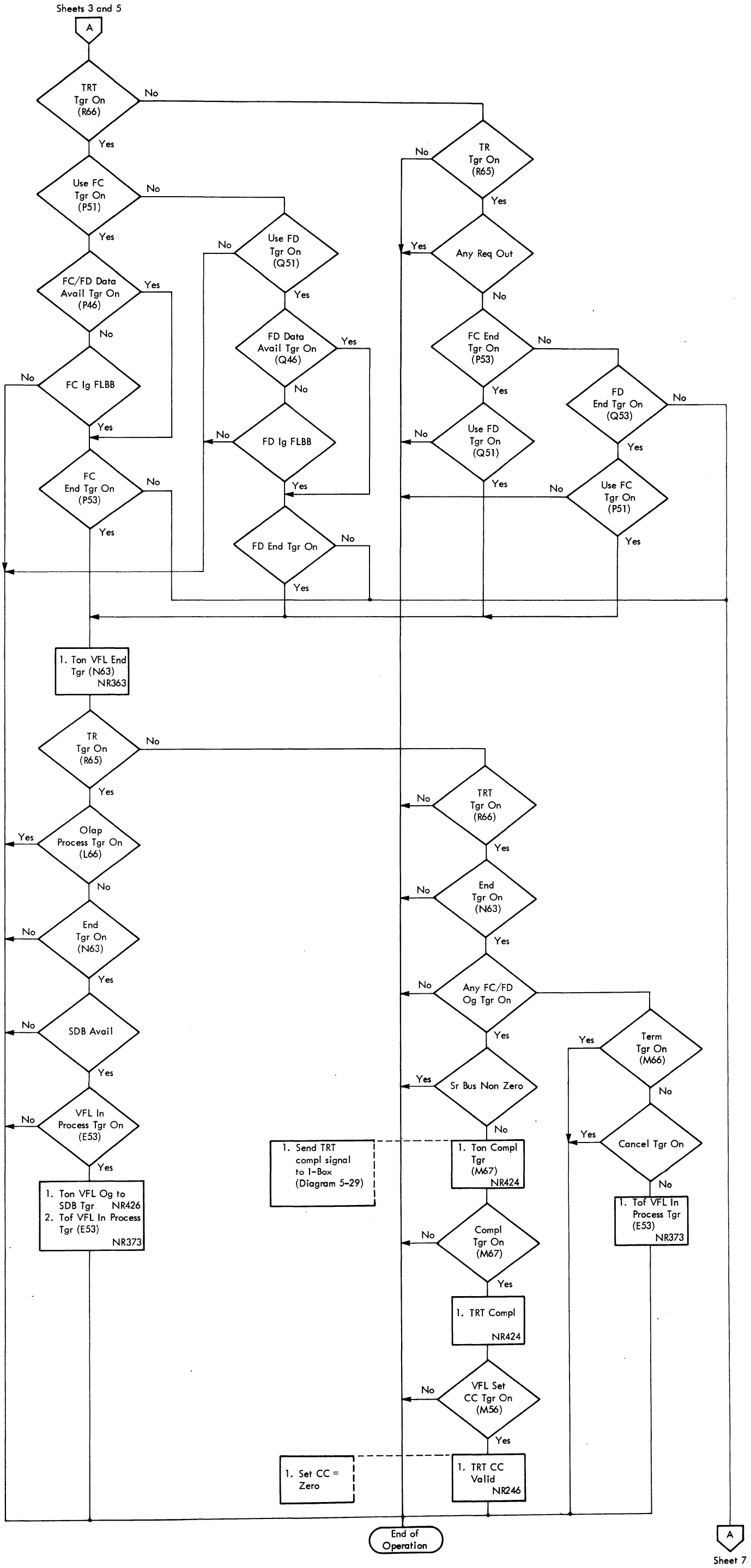


DIAGRAM 5-126. TR AND TRT INSTRUCTIONS (SHEET 6 OF 7)

TRT Terminate

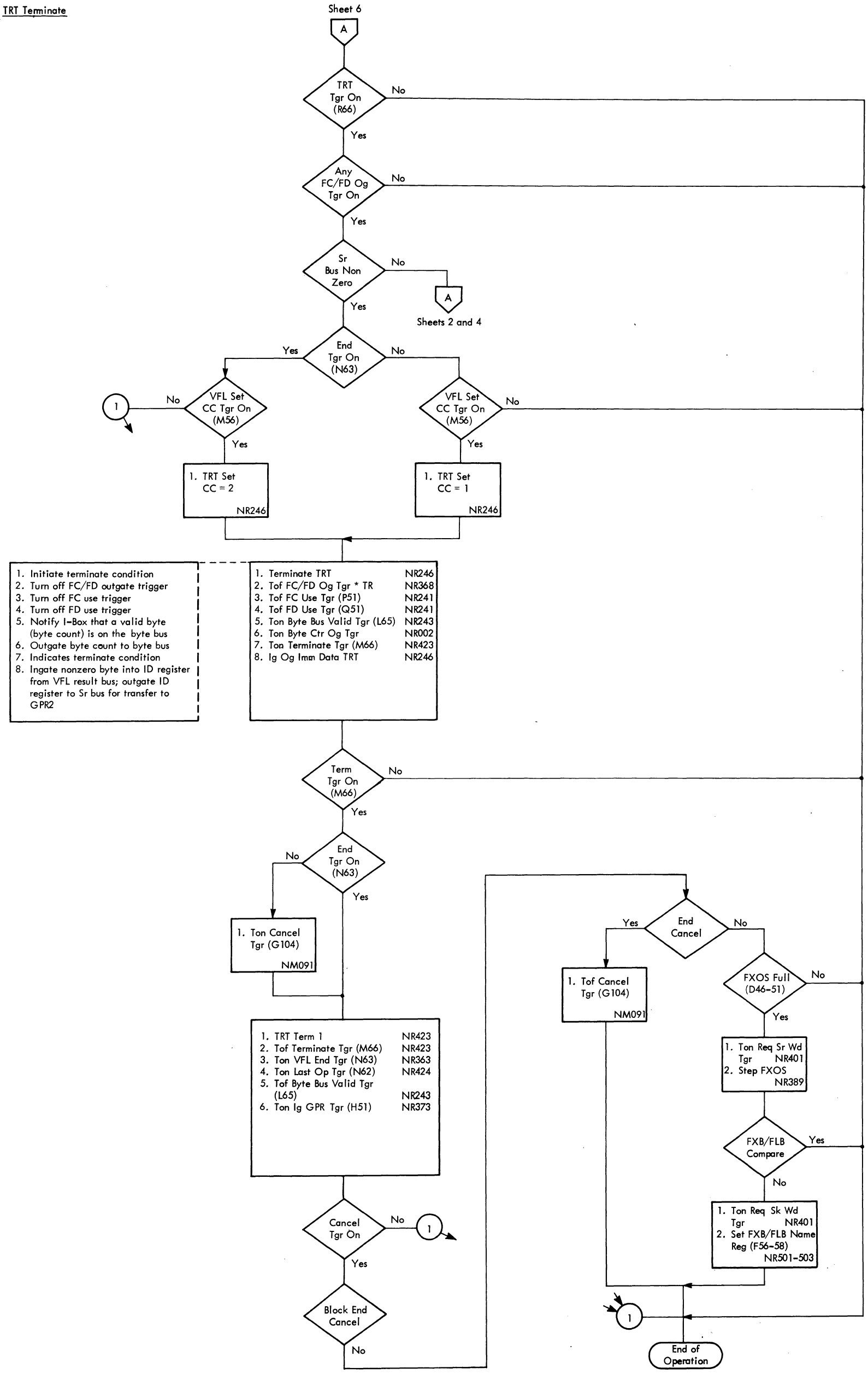
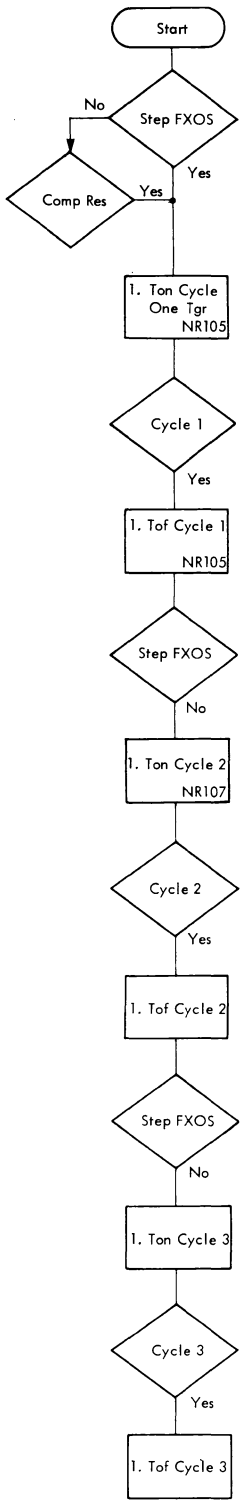


DIAGRAM 5-126. TR AND TRT INSTRUCTION (SHEET 7 OF 7)

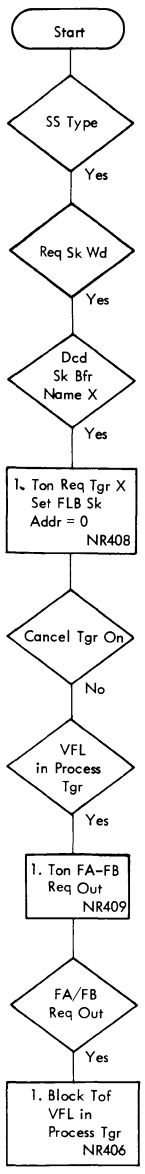
Objectives:

1. FA/FB Request sequence is executed whenever a sink word request is generated.
2. FC/FD Request sequence is executed whenever a source request or TWC/TWD (TRT instruction only) request is generated.
3. Cycle Control sequence is executed each time FXOS is stepped.
4. SDB Available sequence is used to determine which SDB is available for receiving data from the VFL area.
5. VFL Reset sequence is executed to reset various VFL control triggers.
6. Block VFL sequence prohibits VFL operations when certain conditions exist.

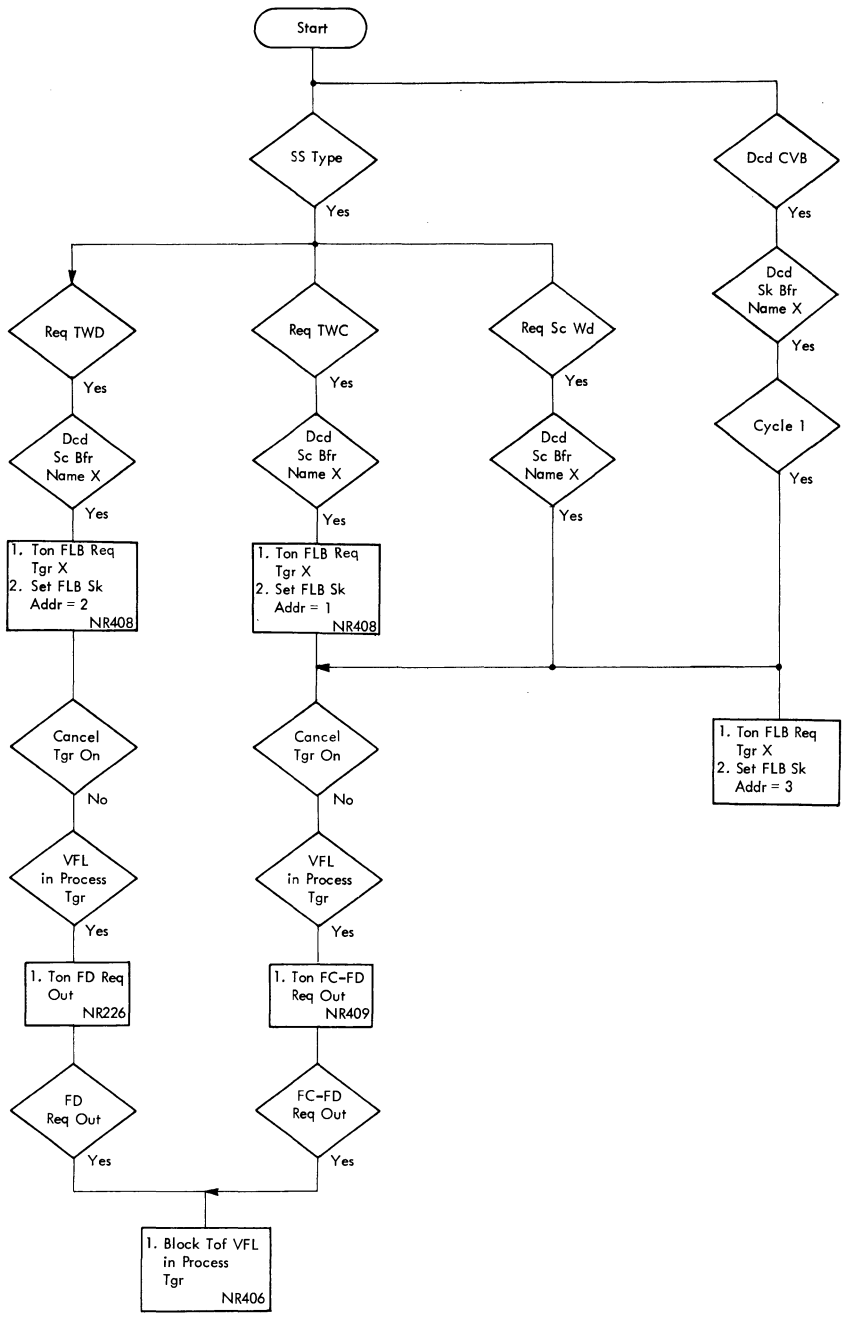
Cycle Control



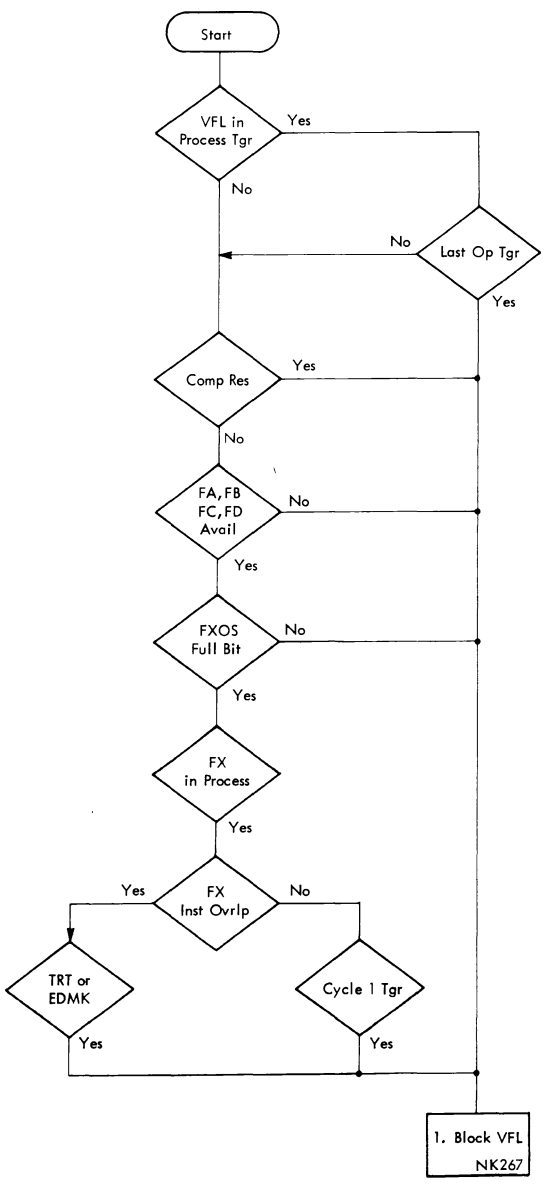
FA/FB Request



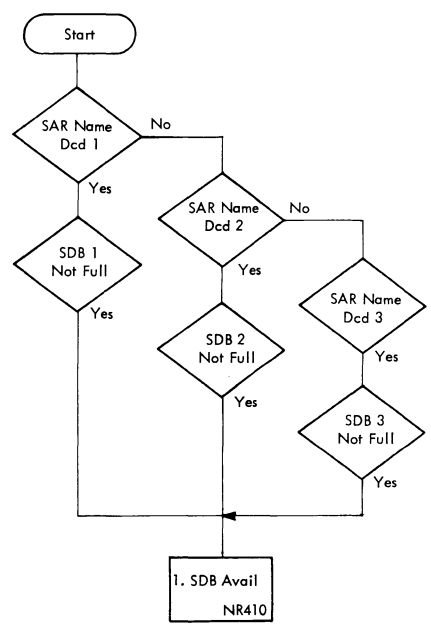
FC/FD Request



Block VFL



SDB Available



VFL Reset

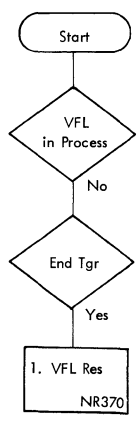


DIAGRAM 5-127. VFL CONTROLS

Objectives:

1. As each Operation is decoded from the op stack, a sequence of conditions must be checked to see if the operation will proceed or be blocked.
2. If no interfering conditions exist, the operation can proceed with decoding and processing.
3. If any block condition exists, the block op trigger is turned on and the operation must wait until the block condition is cleared before proceeding.

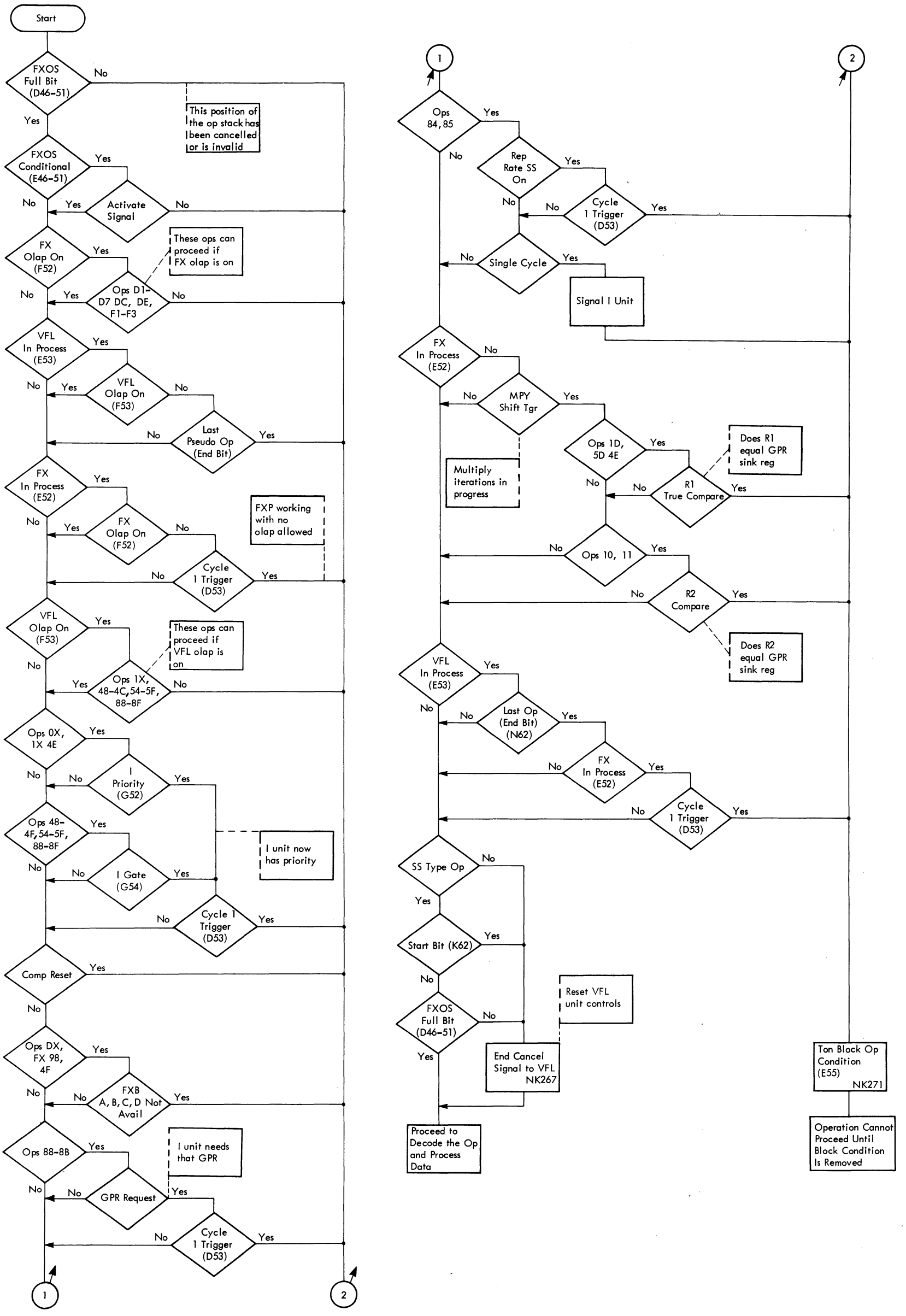


DIAGRAM 5-128. BLOCK OPERATION

Cancelled Operation Processing

Objectives:

1. A pseudo op in the FXOS can be cancelled by resetting the corresponding FXOS full bit.
2. When an operation is cancelled, all circuits set up for that operation must be restored or reset.
3. Signals are sent to I unit that all FXB's and GPR's reserved for the cancelled operation will no longer be needed by that operation.
4. FXOS steps to decode the next pseudo-op.

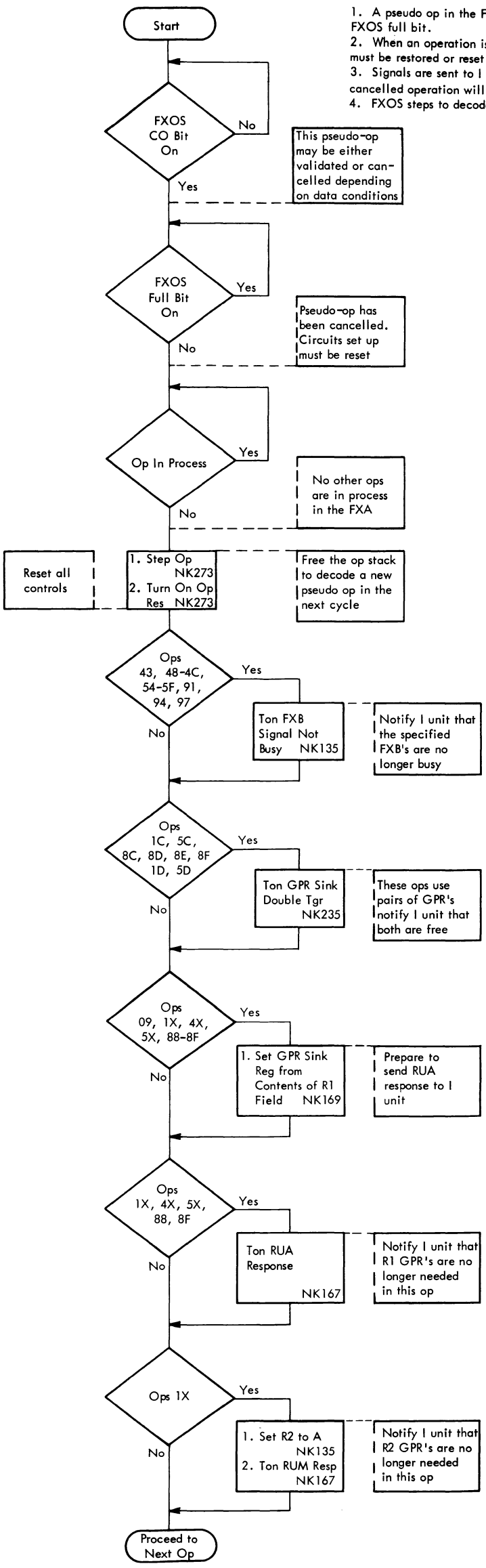
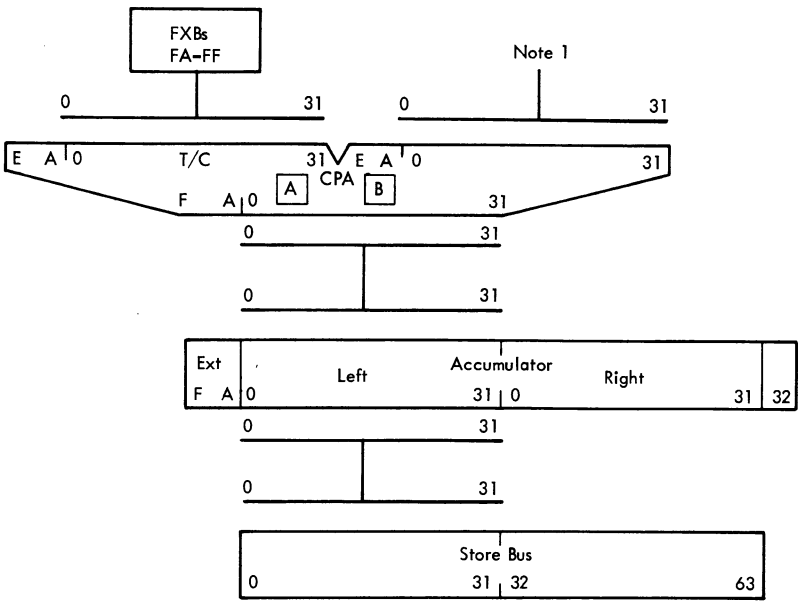
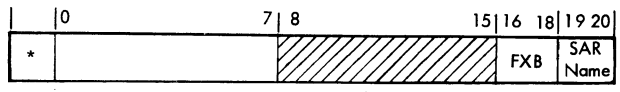


DIAGRAM 5-129. CANCELLED OPERATION PROCESSING



Notes:
1. Decrement Value

Timer Operation



* Special RR Format

Objectives:

1. Timer operation fetches a timer count word from storage location 80, decrements the count by a specified amount and stores the count word back in the same location.
2. Timer operations are issued to the fixed area at a rate of 60 per second (line frequency).
3. When the decremented count becomes negative (has passed through zero) a timer interrupt occurs to signal the program that a particular period of time has passed.

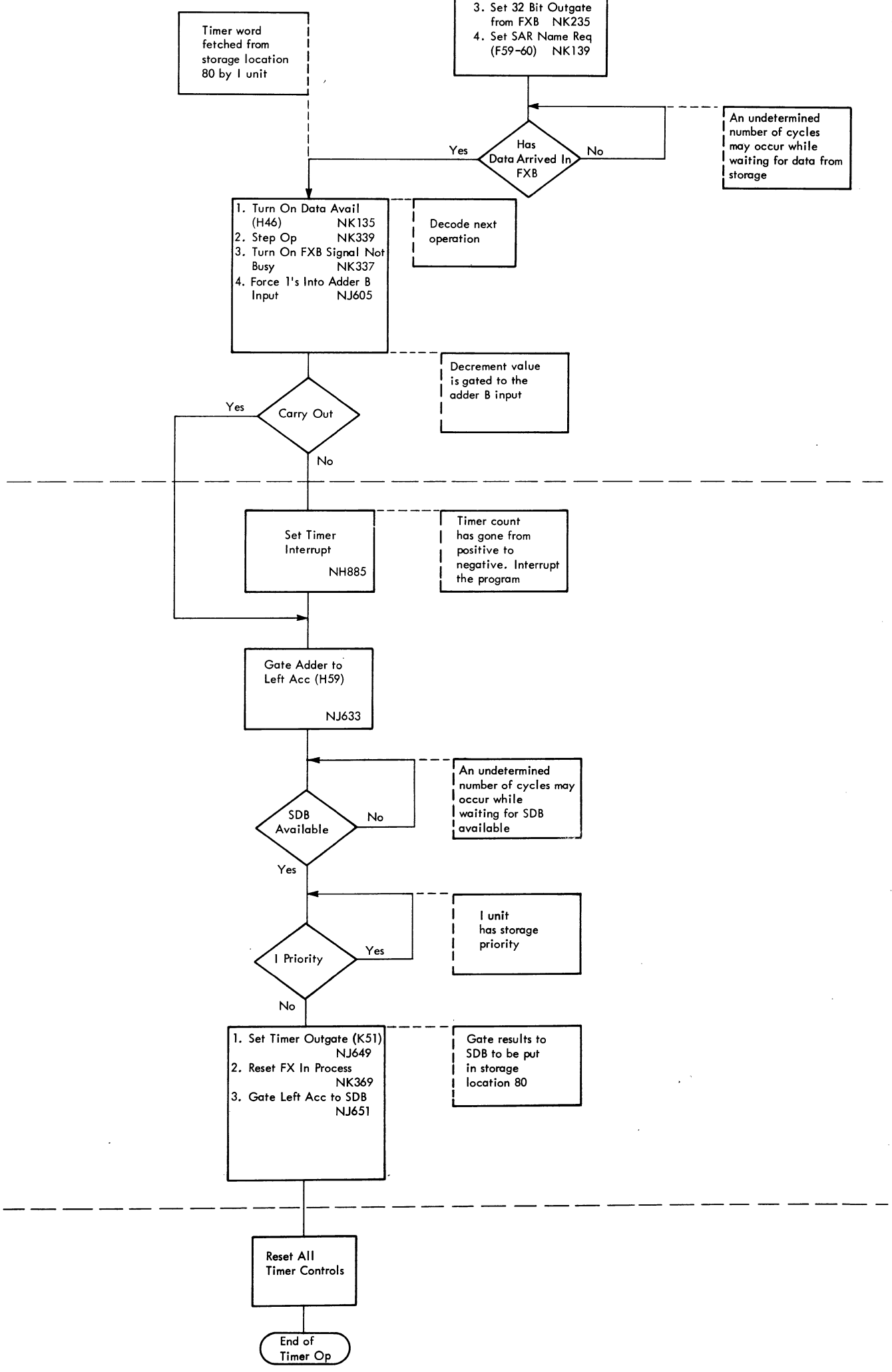


DIAGRAM 5-130. TIMER OPERATION

FLIU Ops		Decode	Data 1	Data 2 / CDB Cycle			
FAU Ops	RR	Decode	Select Data 1	Data 2	Execute 1	Execute 2	CDB Cycle
	RX	Decode	Select	Execute 1	Execute 2	CDB Cycle	
FMDU Ops	RR	Decode	Select Data 1	Data 2 *Possible Normalize	*Normalize	Execute Cycles (3-11)	CDB Cycle
	RX	Decode	Select	*Normalize	*Normalize	Execute Cycles (3-11)	CDB Cycle

*A normalize cycle occurs only for an operand that is set into a reservation station unnormalized. Execute cycles follow the select or Data 2 cycle when both operands arrive normalized.

Simplified Instruction Sequencing.

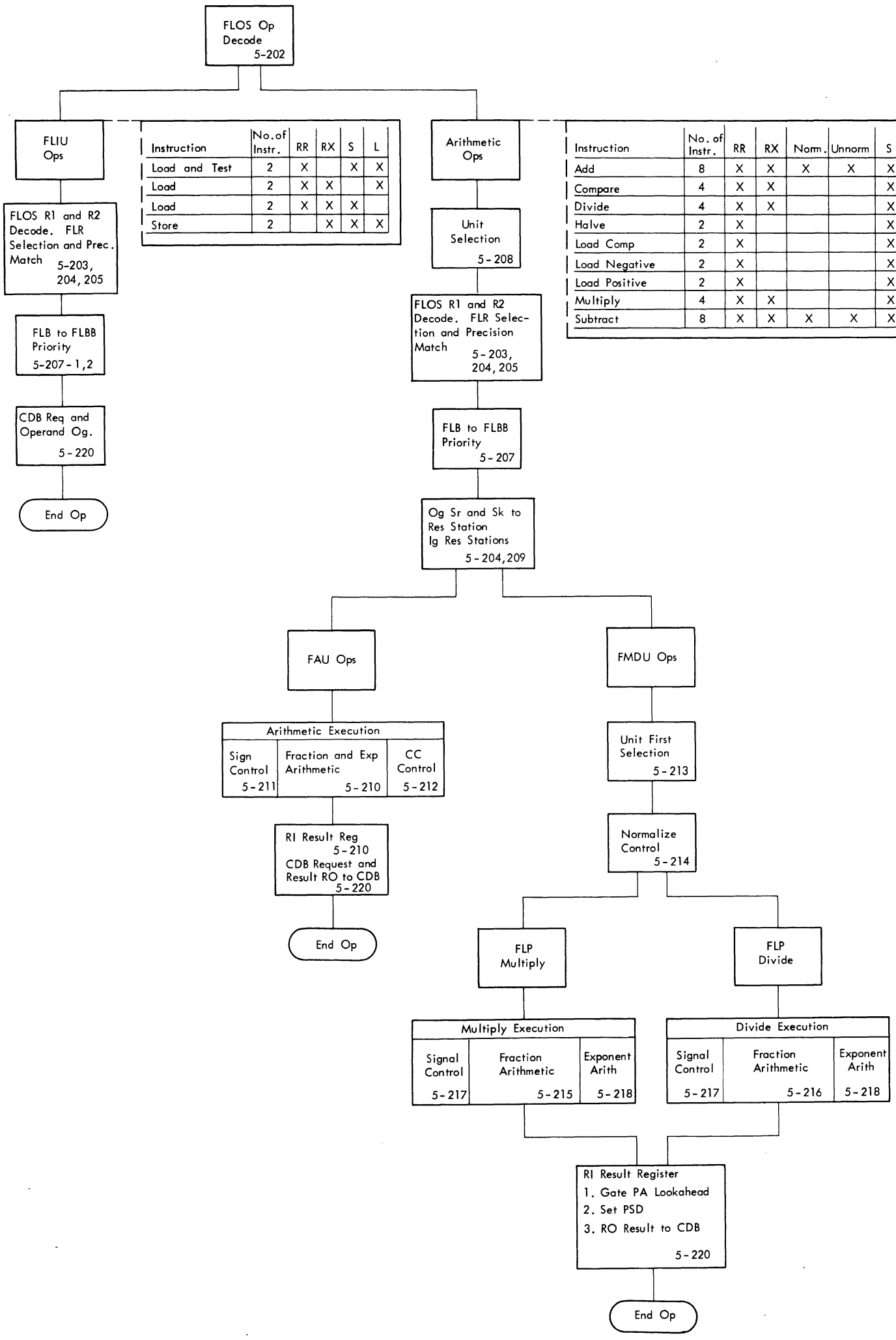


DIAGRAM 5-200. FLOATING POINT OPERATIONS

Objectives:

1. Controls ingating, storing and outgating of op codes from the I-Box for use by the arithmetic units.

2. Signals the I-Box to deliver more ops to replace those which have been processed.

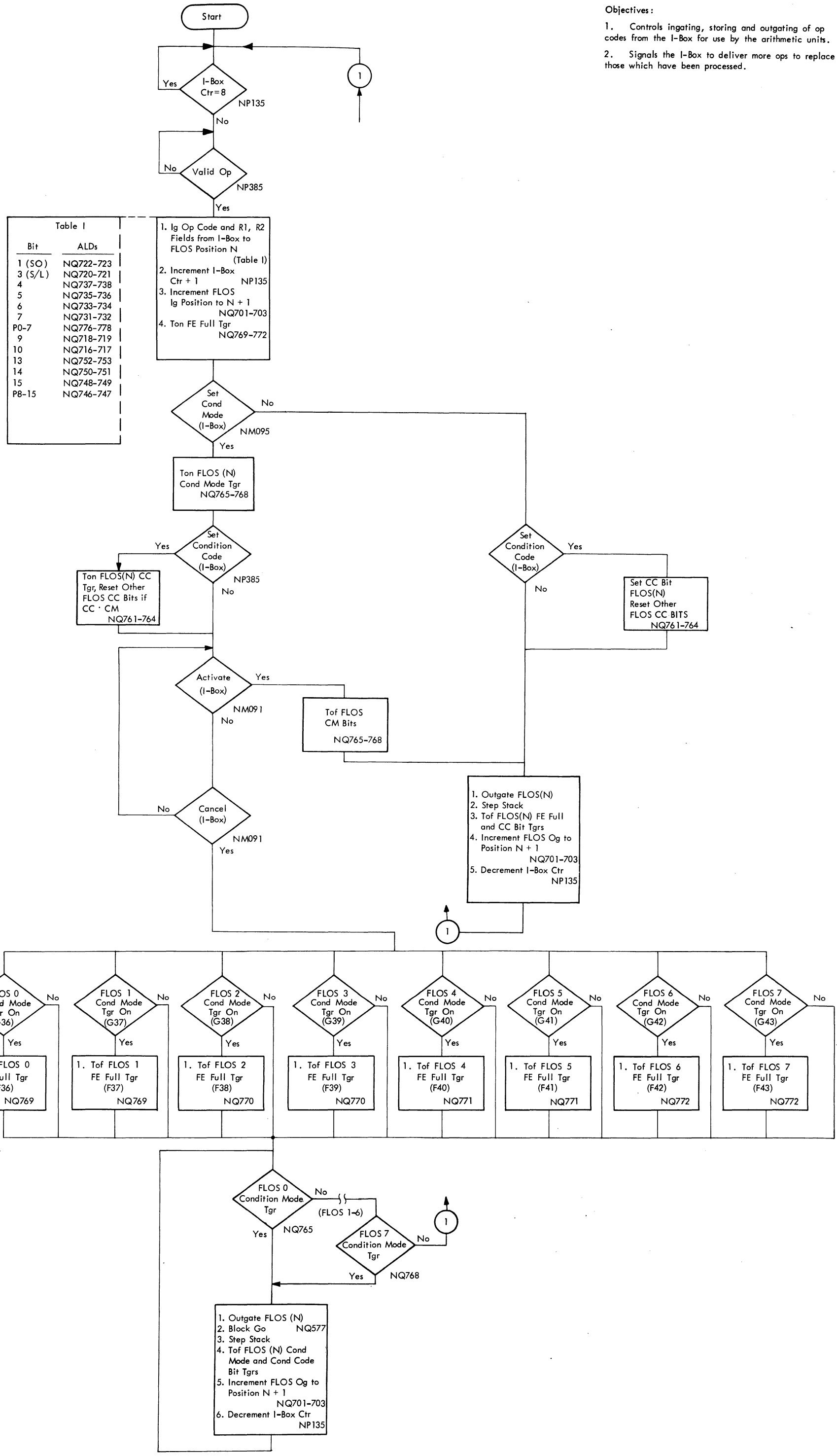


DIAGRAM 5-201. FLOS CONTROLS

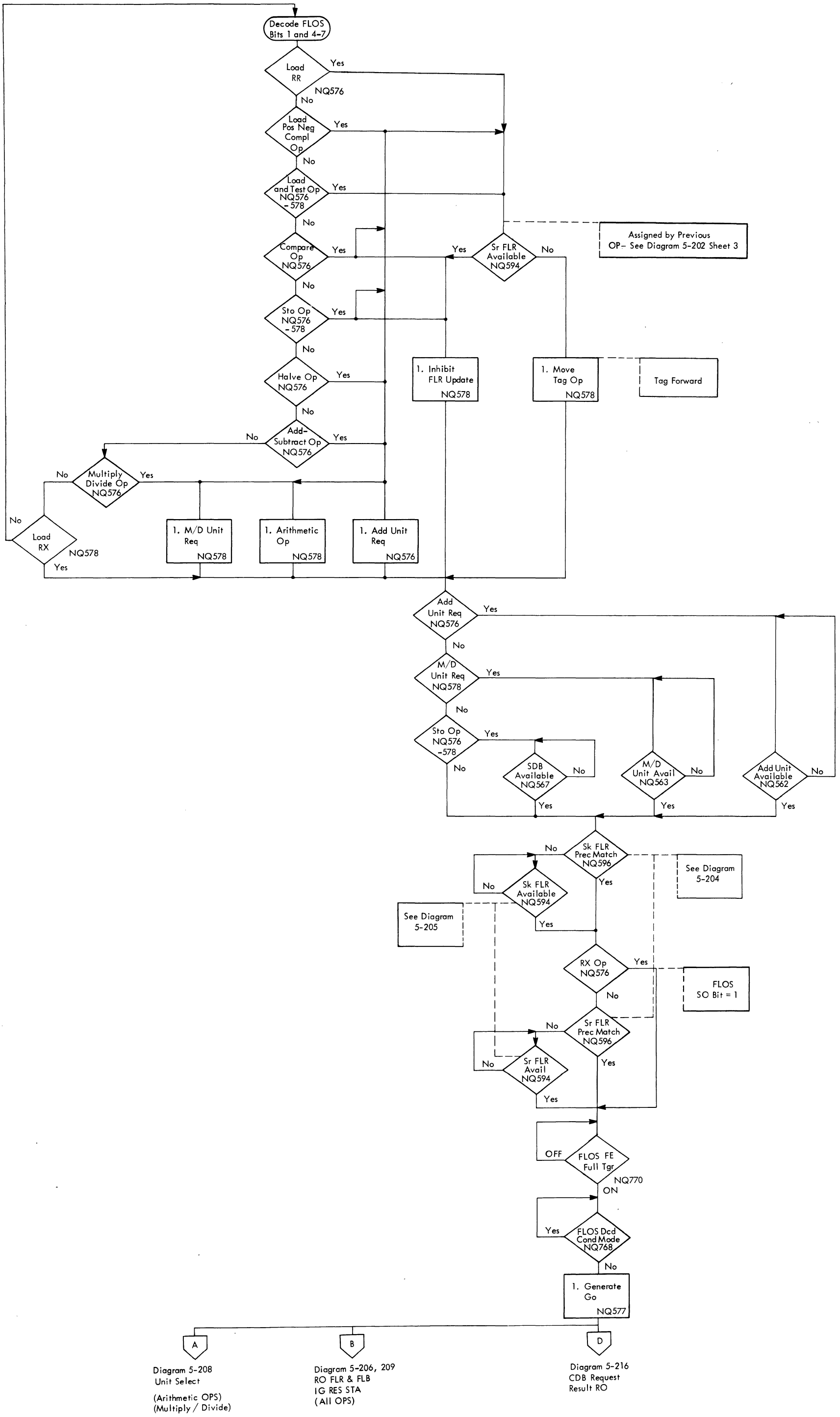


DIAGRAM 5-202. FLOS OP DECODE - GO GENERATION

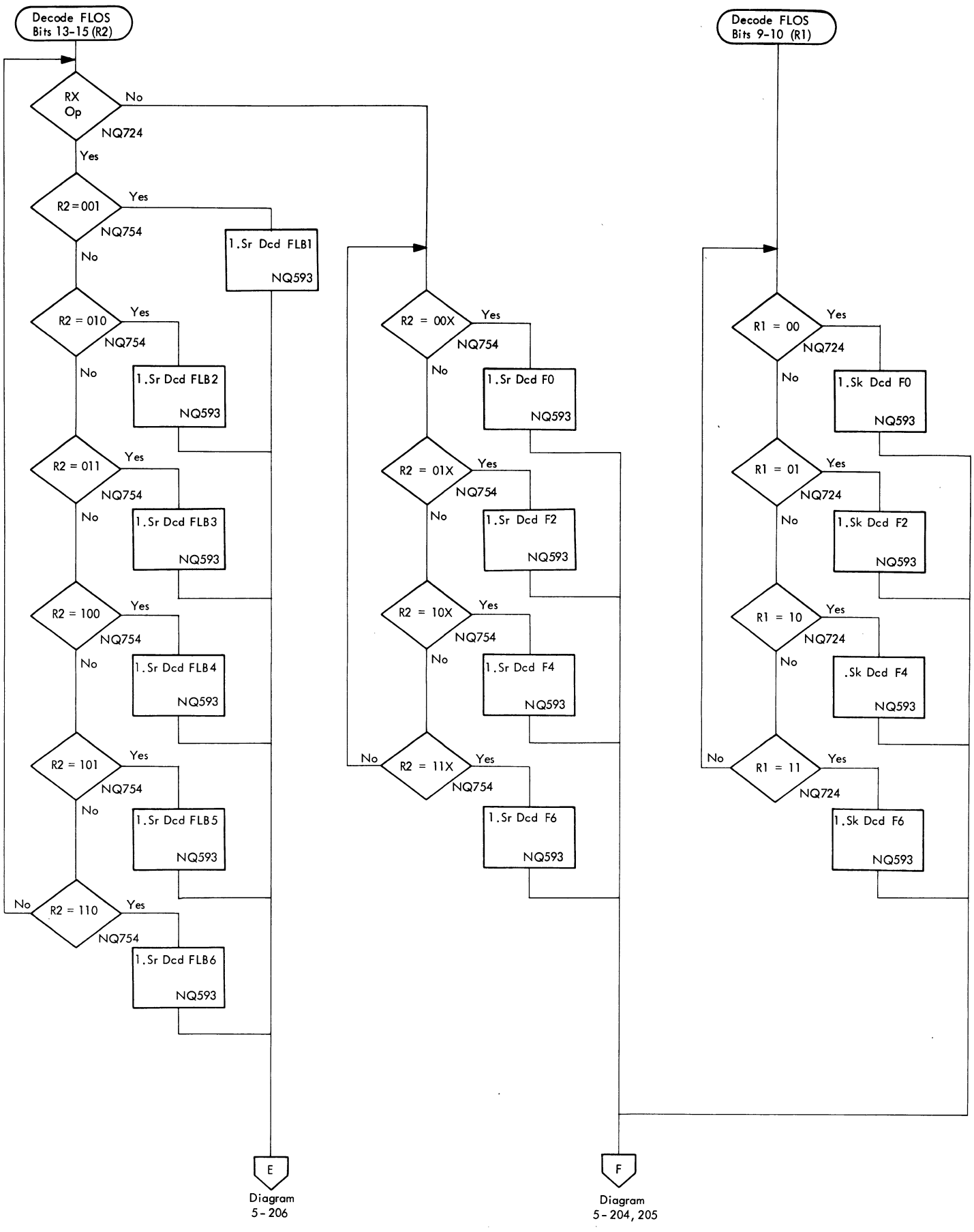


DIAGRAM 5-203. FLOS R1 AND R2 DECODE

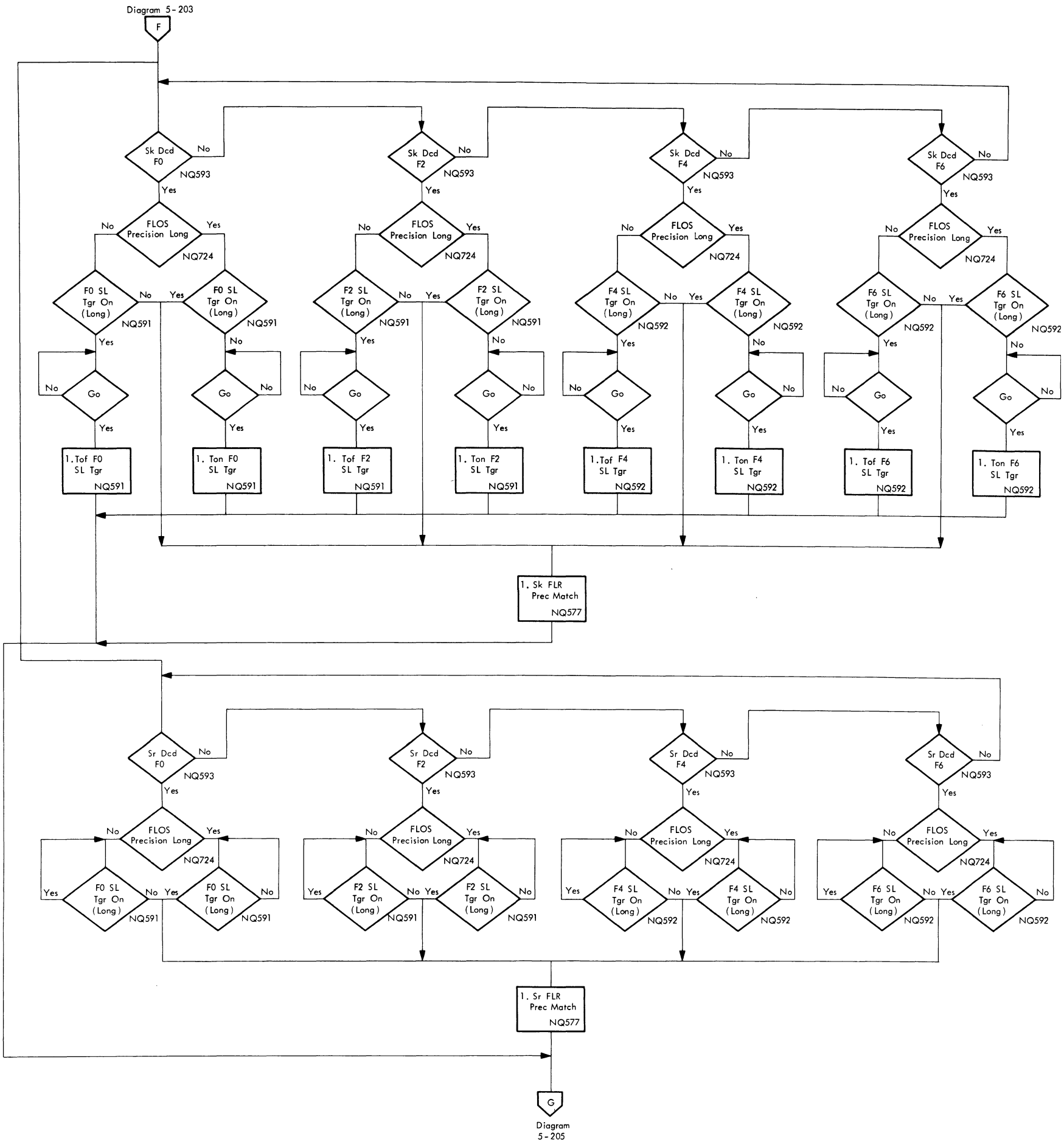


DIAGRAM 5-204. FLR PRECISION MATCH

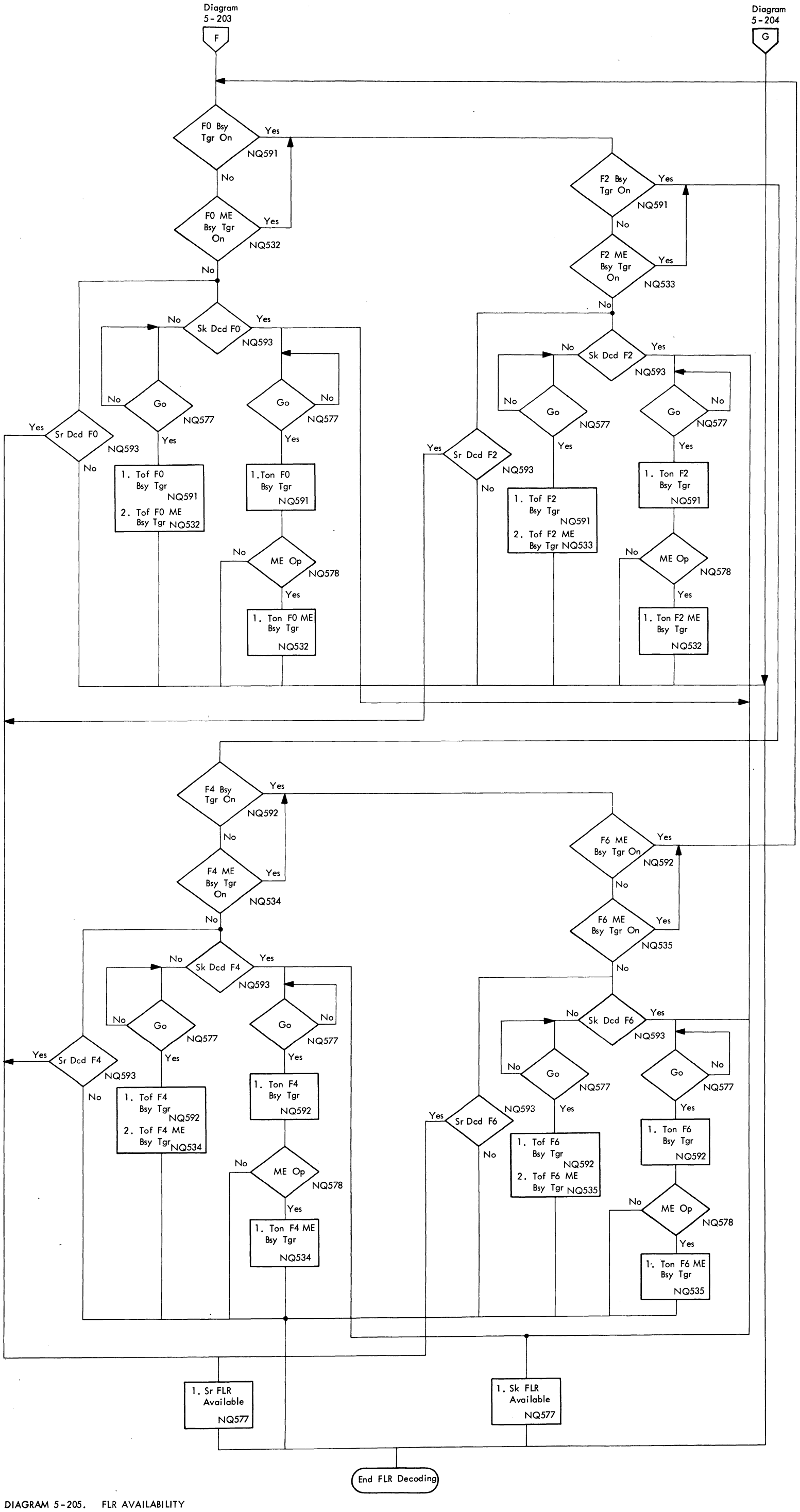


DIAGRAM 5-205. FLR AVAILABILITY

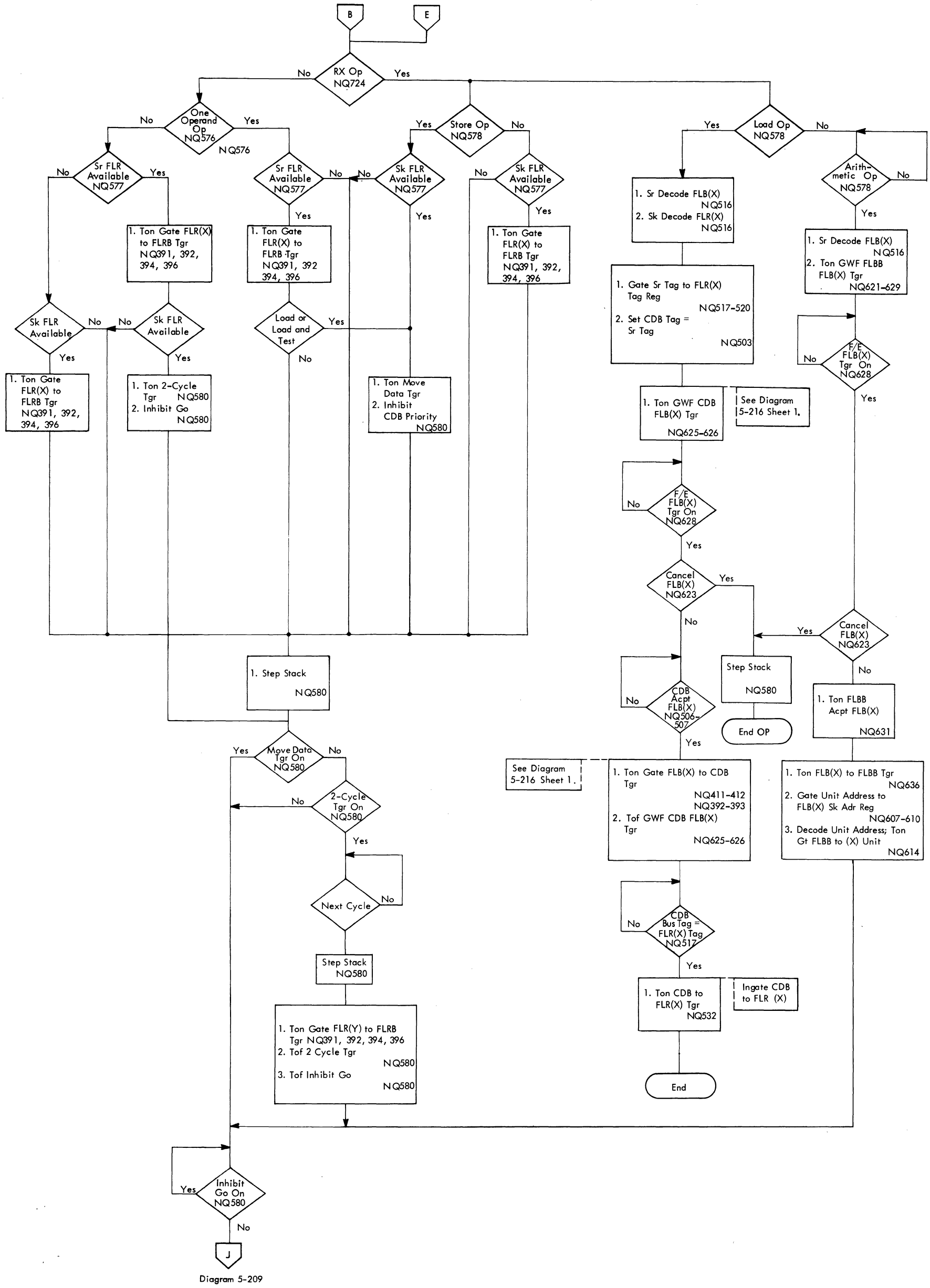


DIAGRAM 5-206. FLR OUTGATING

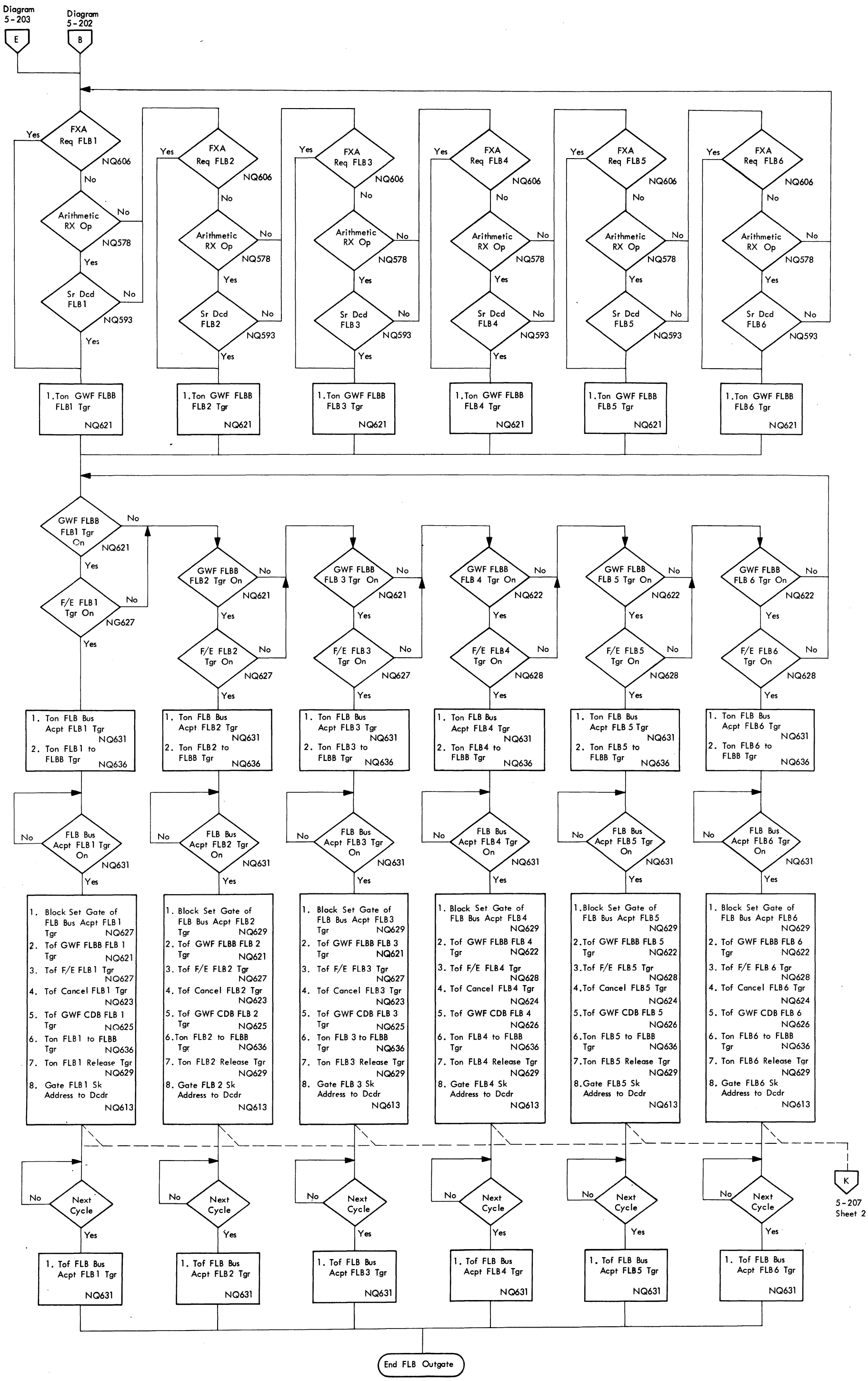


DIAGRAM 5-207. FLBB PRIORITY AND FLB OUTGATING (SHEET 1 OF 2)

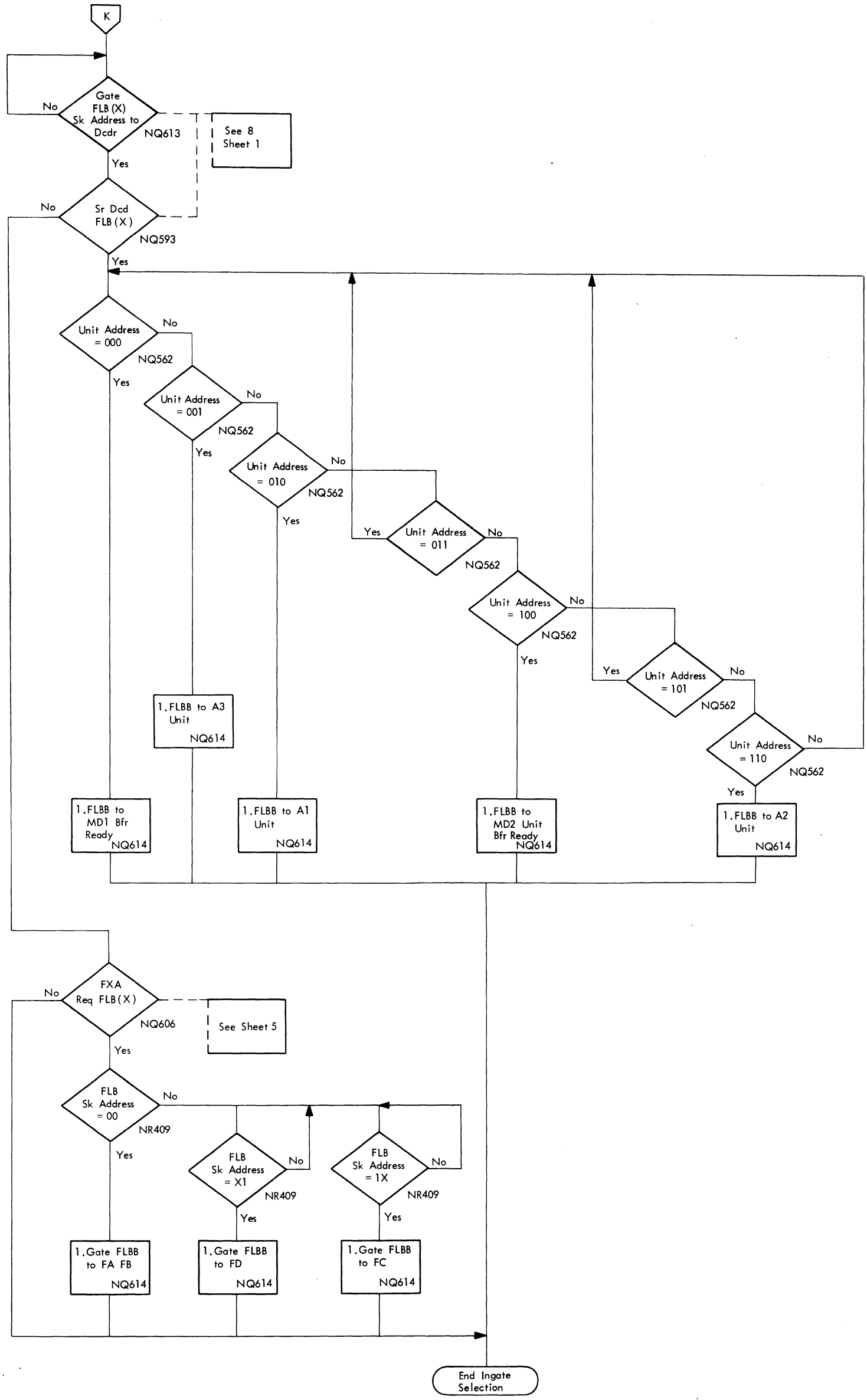


DIAGRAM 5-207. FLBB PRIORITY AND FLB OUTGATING (SHEET 2 OF 2)

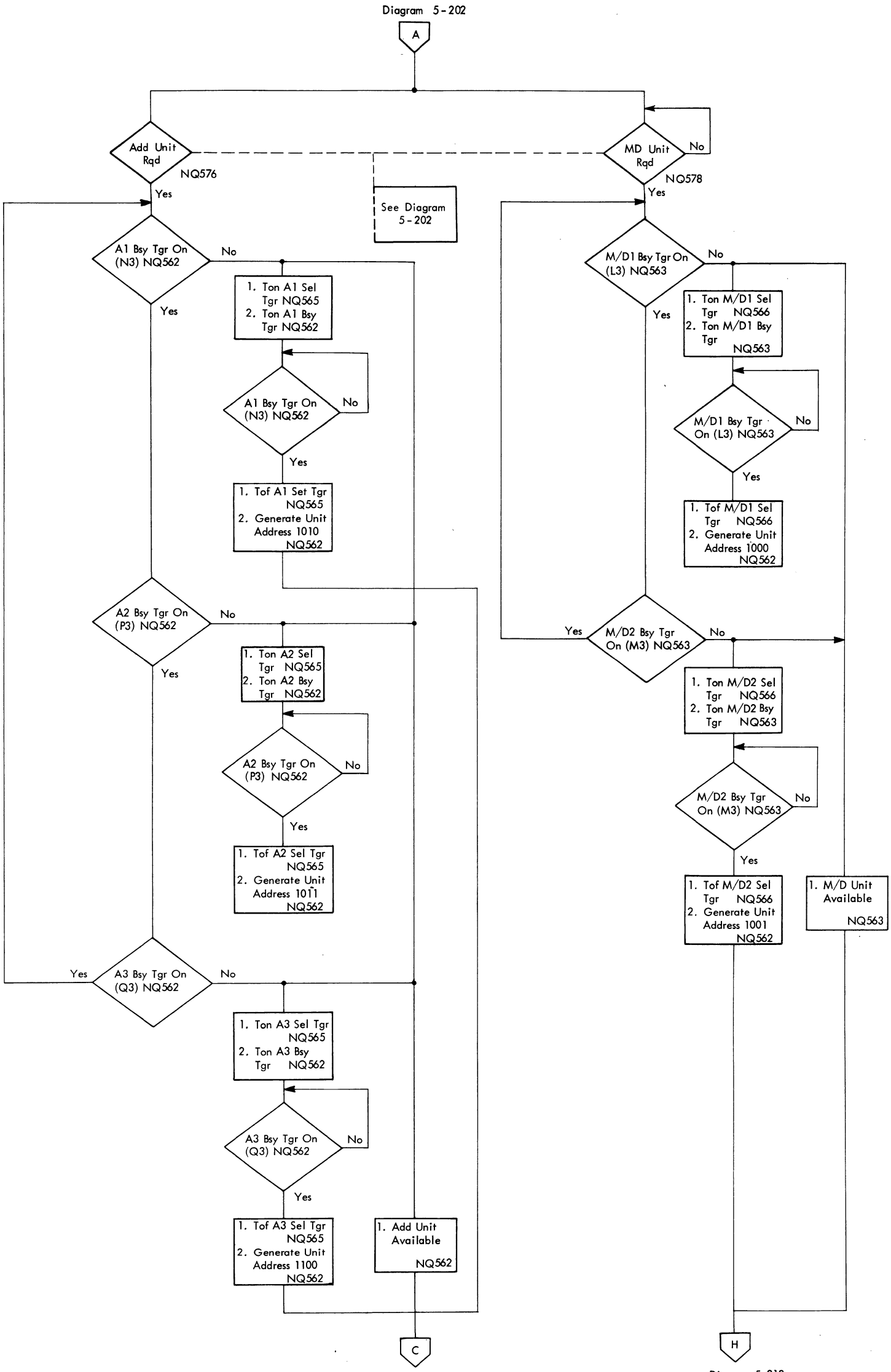


DIAGRAM 5-208. UNIT SELECTION

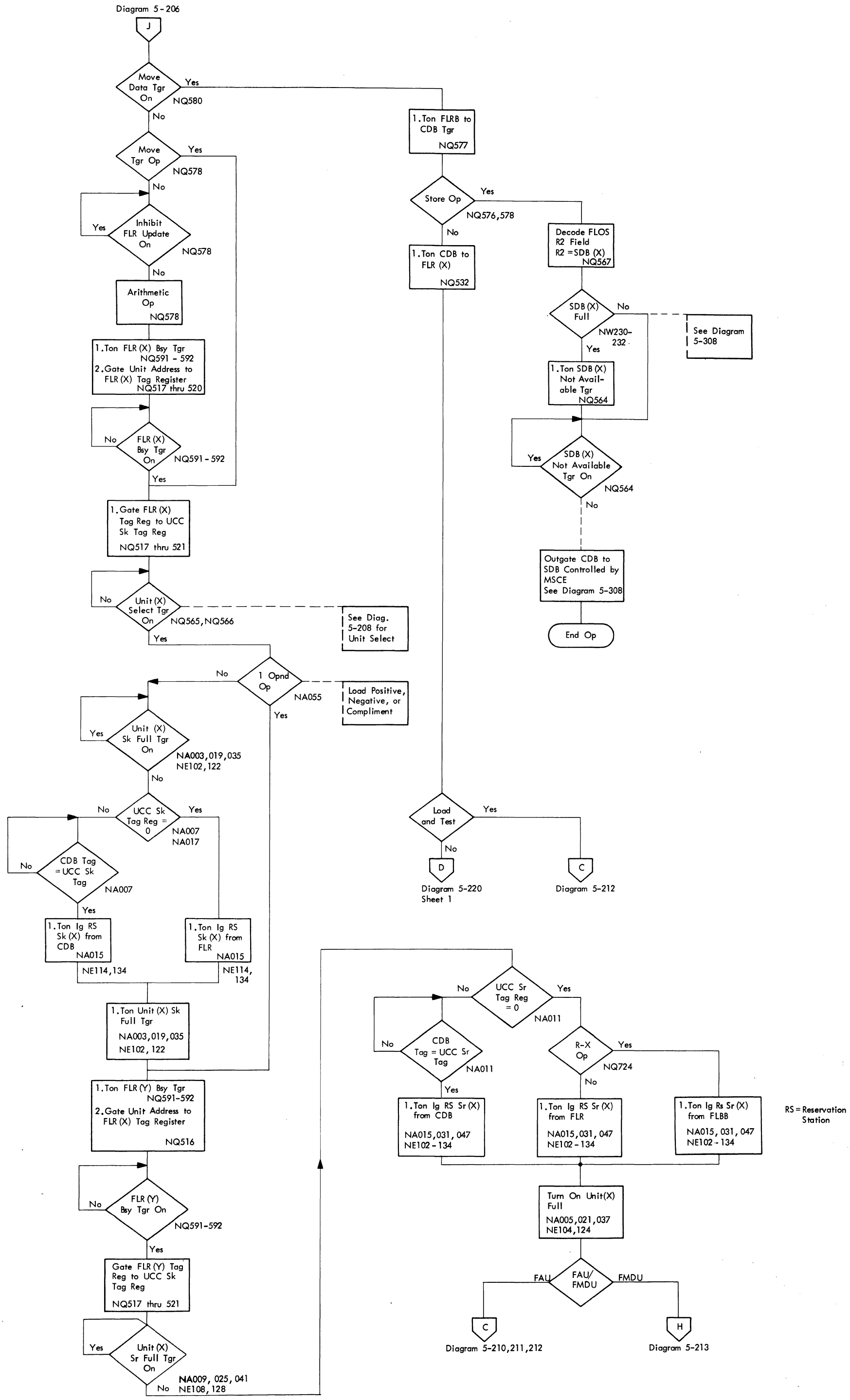


DIAGRAM 5-209. RESERVATION STATION INGATING

DIAGRAM 5-208, 209

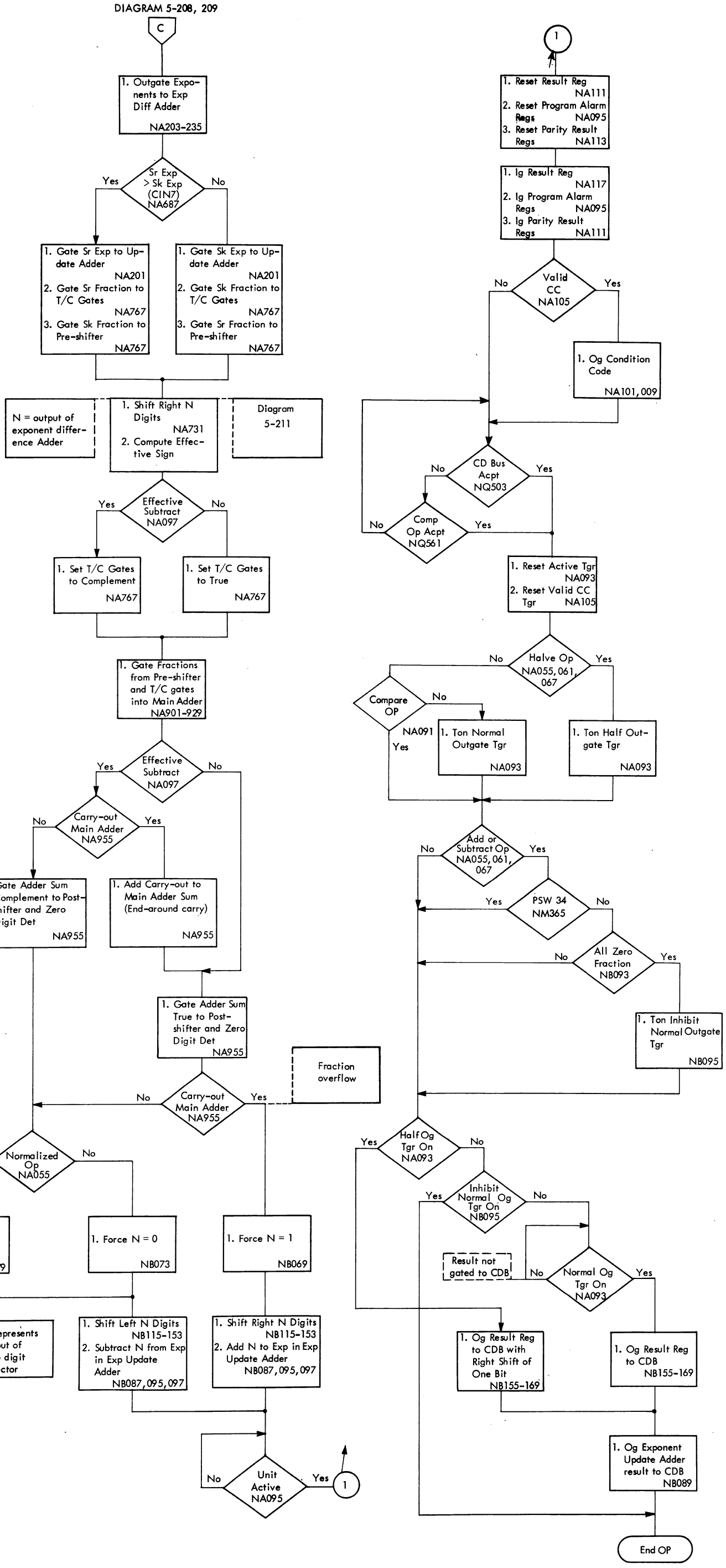


DIAGRAM 5-210. FAU EXECUTION - FRACTION AND EXPONENT

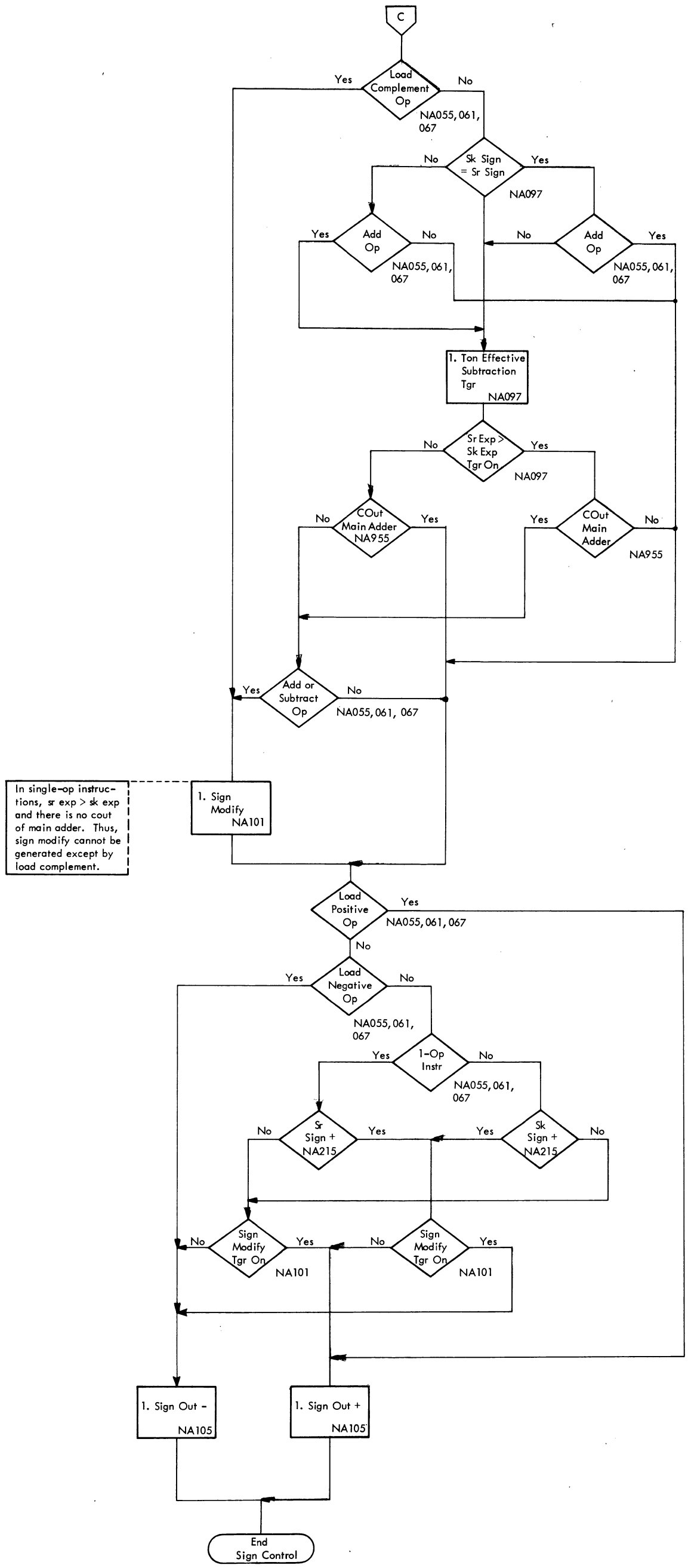


DIAGRAM 5-211. FAU SIGN CONTROL

Diagram 5-208, 209

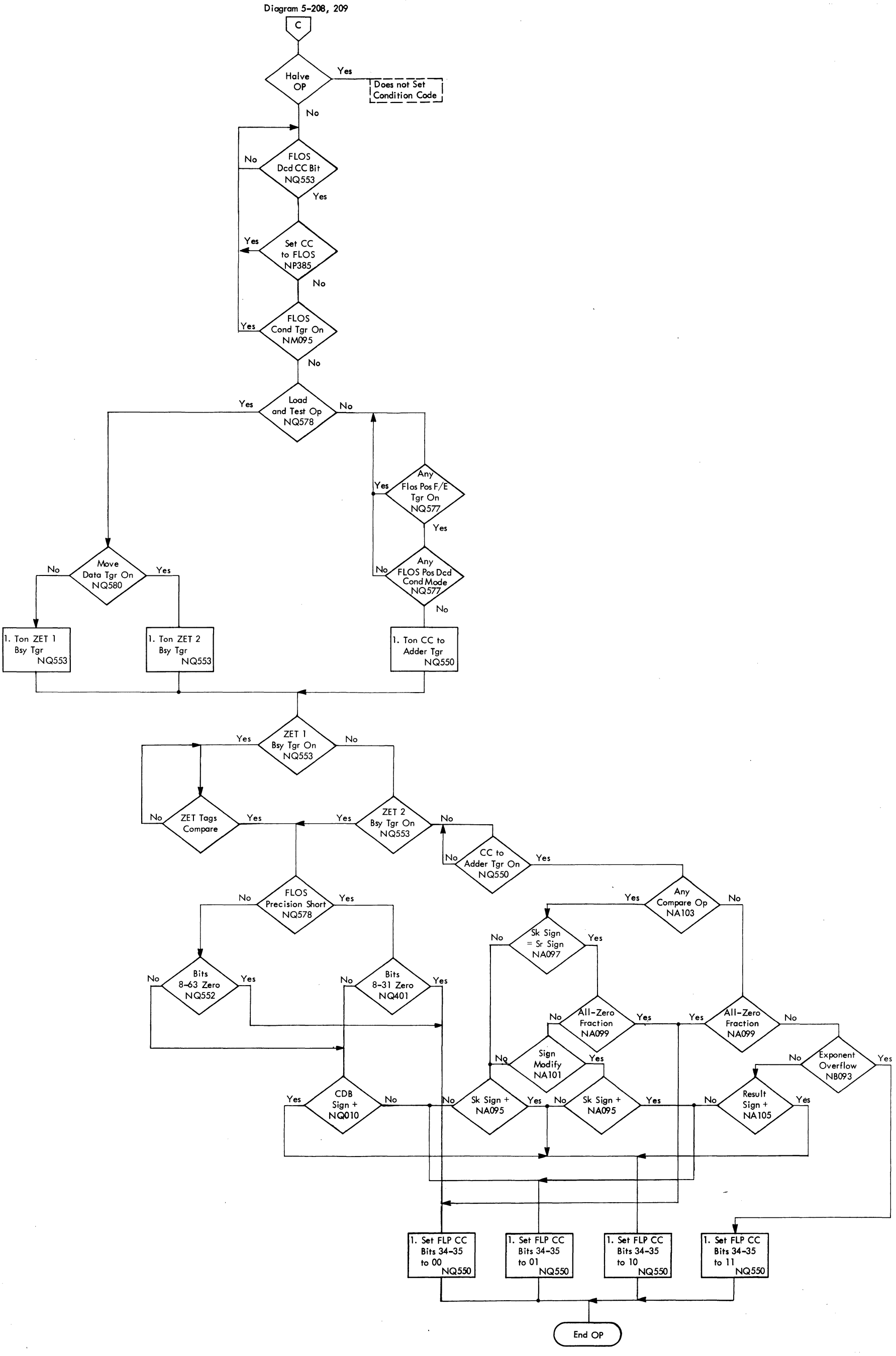


DIAGRAM 5-212. FAU AND ZET CONDITION CODE CONTROL

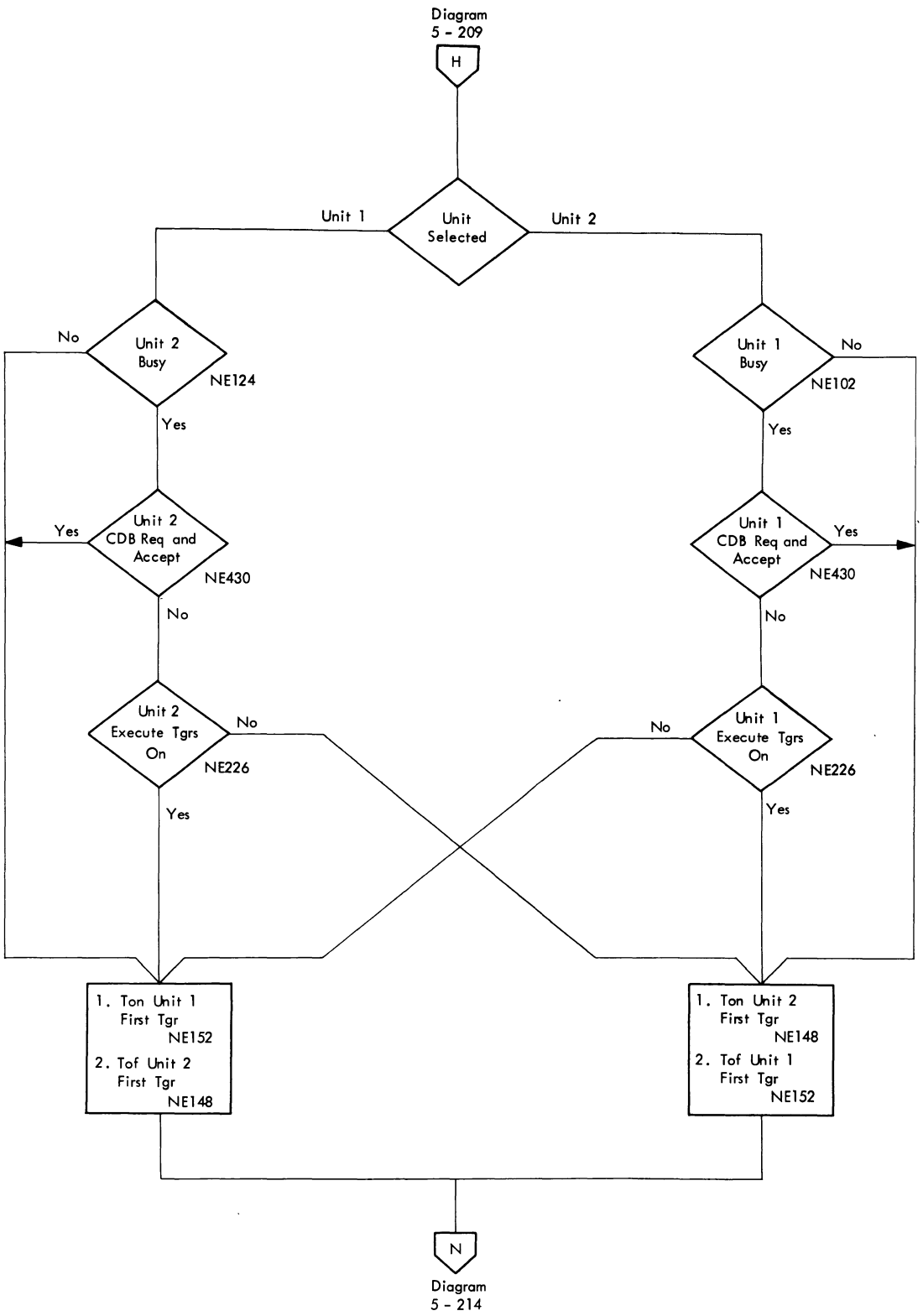


DIAGRAM 5-213. FMDU UNIT FIRST SELECTION

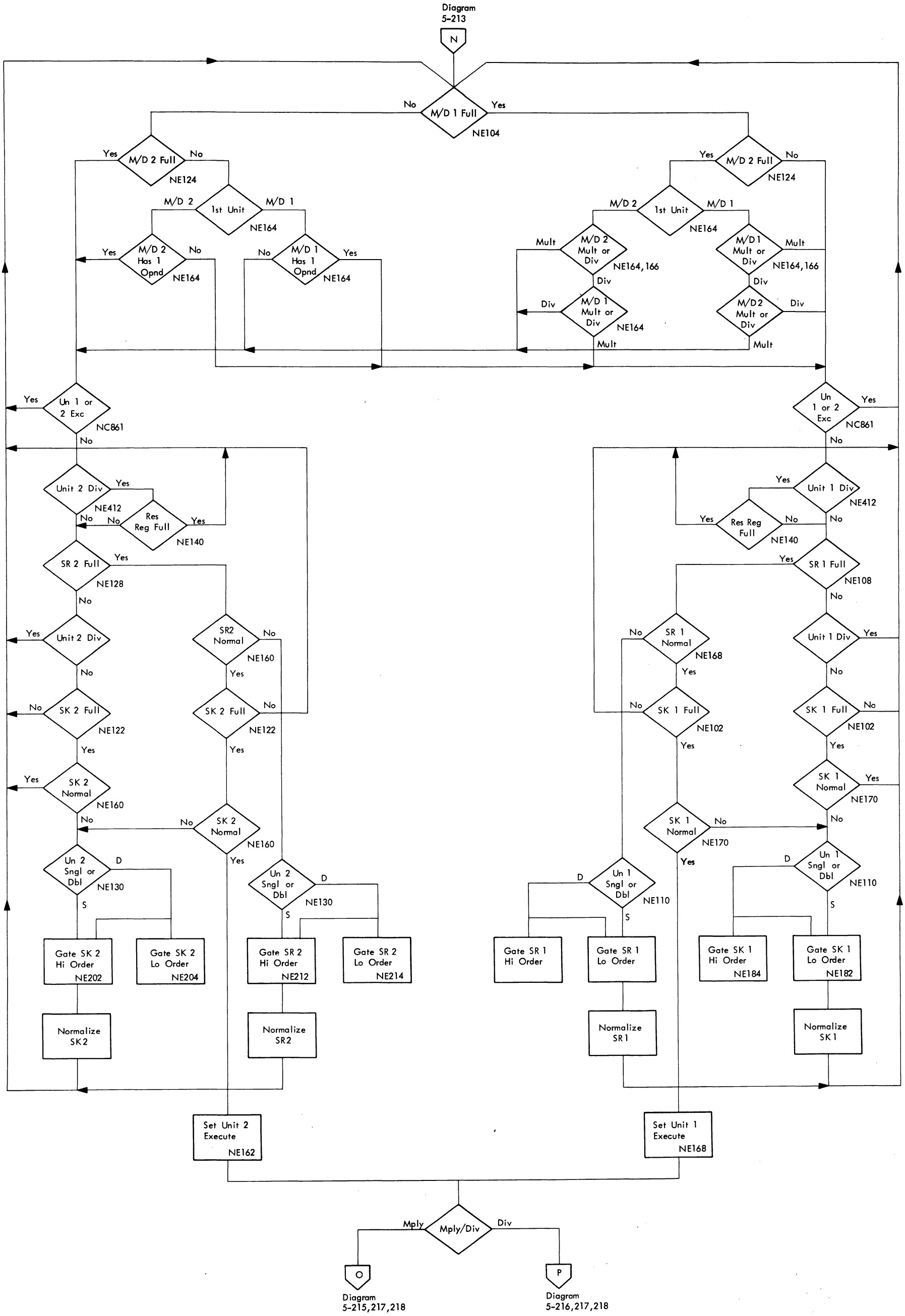


DIAGRAM 5-214. FMDU NORMALIZE CONTROL

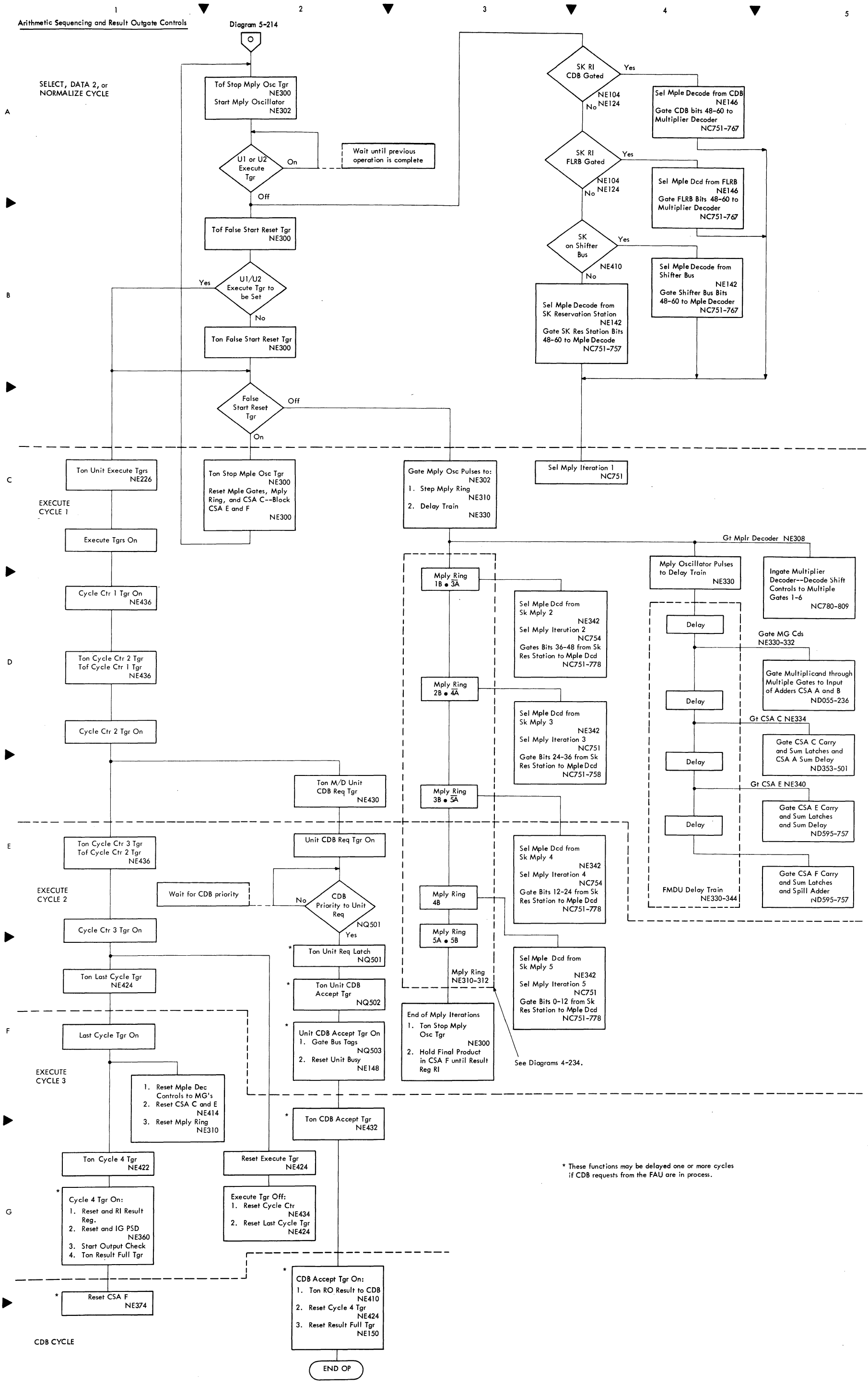
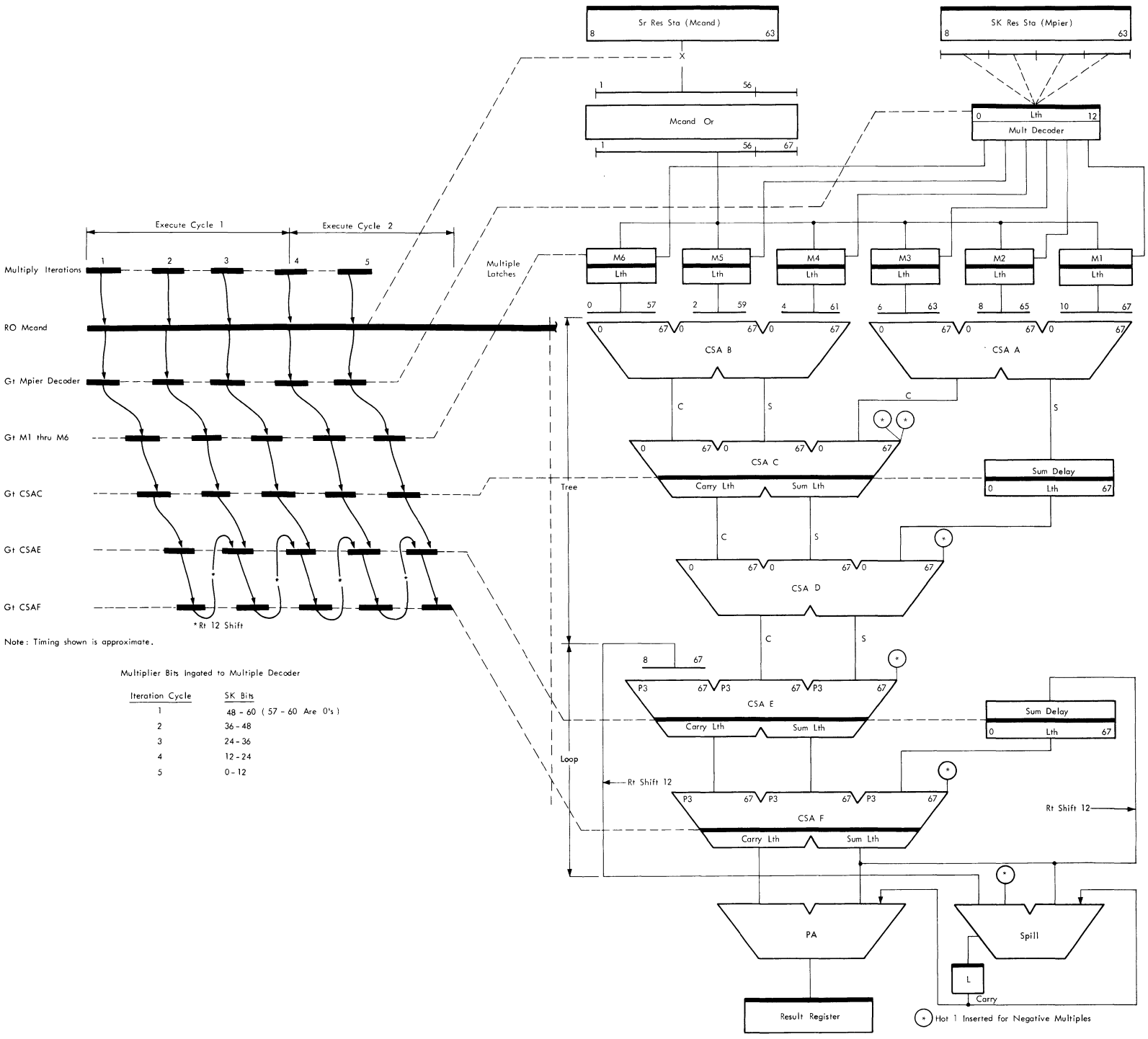


DIAGRAM 5-215. MULTIPLY EXECUTION (SHEET 1 OF 2)

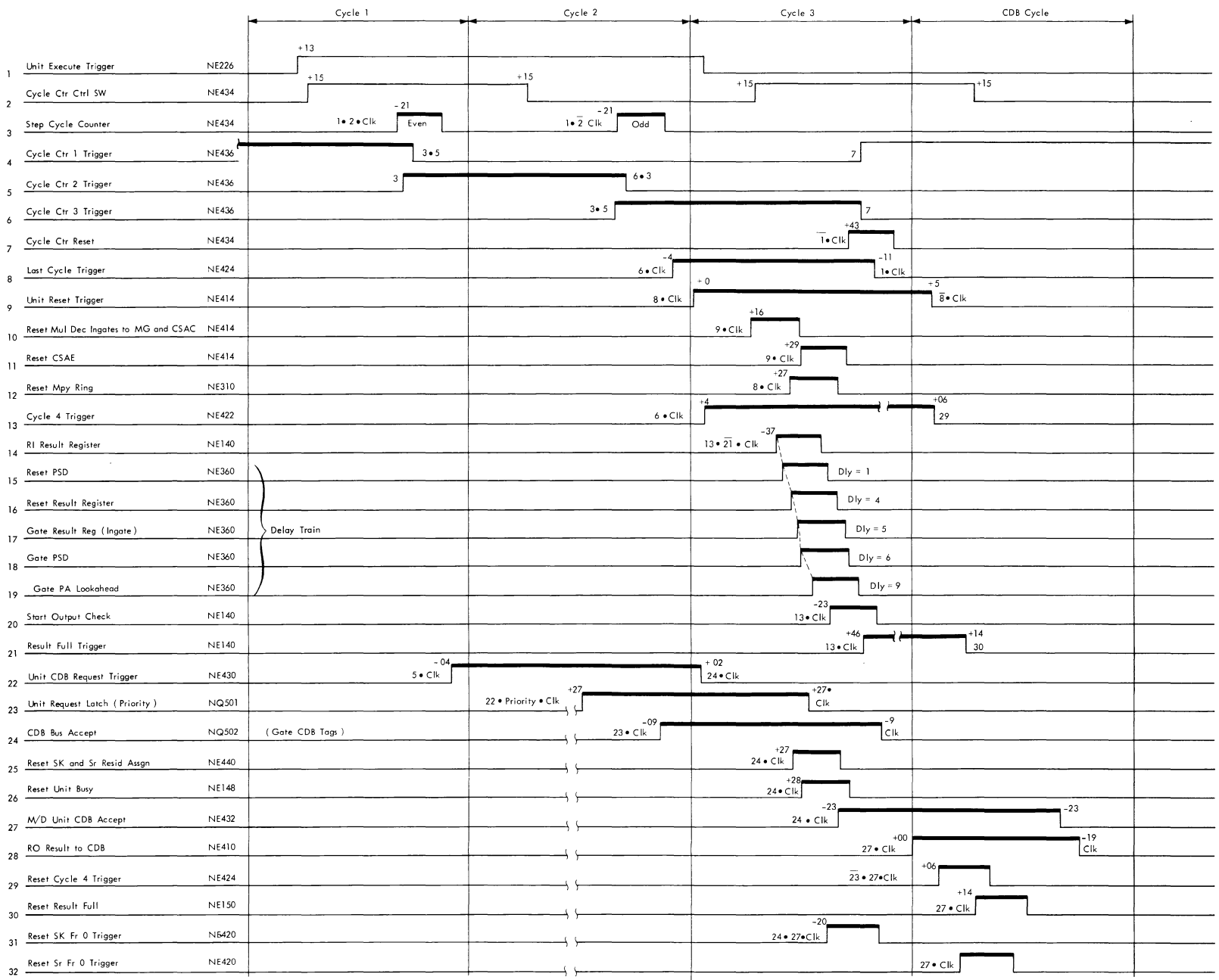


Note: Timing shown is approximate.

Multiplier Bits Ingated to Multiple Decoder

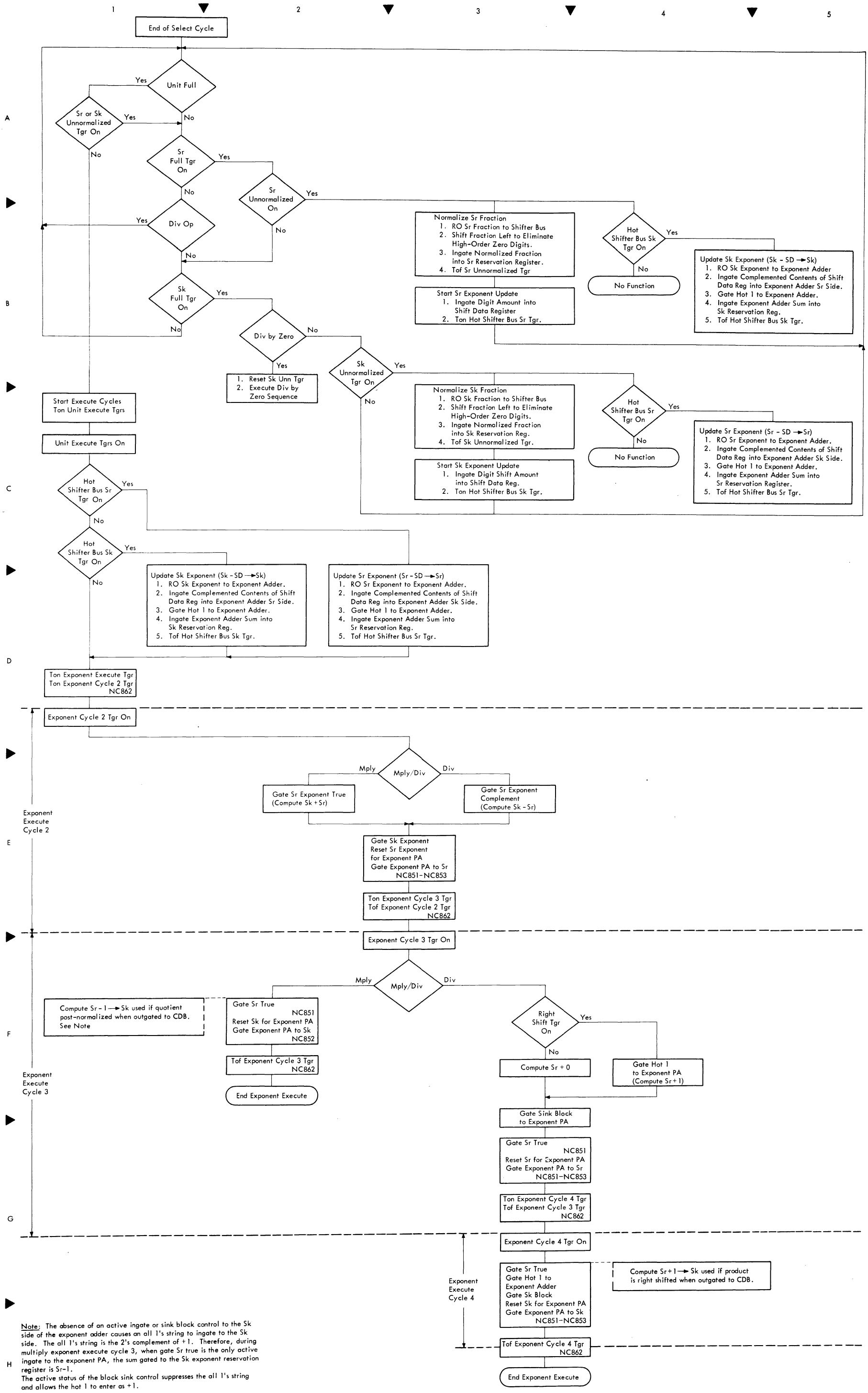
Iteration Cycle	SK Bits
1	48 - 60 (57 - 60 Are 0's)
2	36 - 48
3	24 - 36
4	12 - 24
5	0 - 12

B. Multiply Sequencing and Outgate Controls



Note: The timing shown is approximate. For each signal line, the originating clock timing is indicated; however, because of inherent circuit delays a given signal may occur several nanoseconds later.

DIAGRAM 5-215. MULTIPLY EXECUTION (SHEET 2 OF 2)



Note: The absence of an active ingate or sink block control to the Sk side of the exponent adder causes an all 1's string to ingate to the Sk side. The all 1's string is the 2's complement of +1. Therefore, during multiply exponent execute cycle 3, when gate Sr true is the only active ingate to the exponent PA, the sum gated to the Sk exponent reservation register is Sr-1.
The active status of the block sink control suppresses the all 1's string and allows the hot 1 to enter as +1.

DIAGRAM 5-218. FMDU EXPONENT EXECUTION

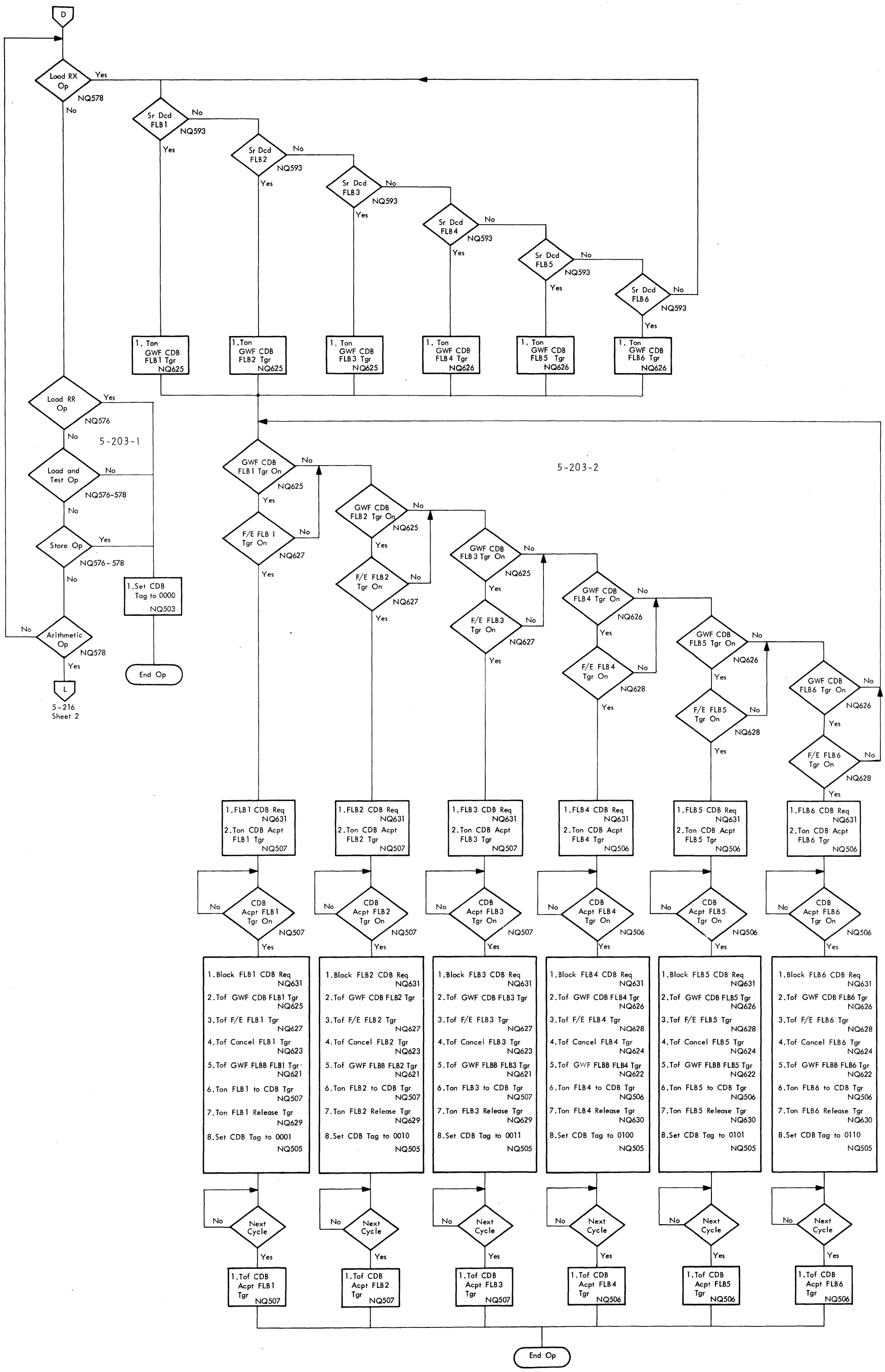


DIAGRAM 5-220. OUTGATES TO CDB (SHEET 1 OF 2)

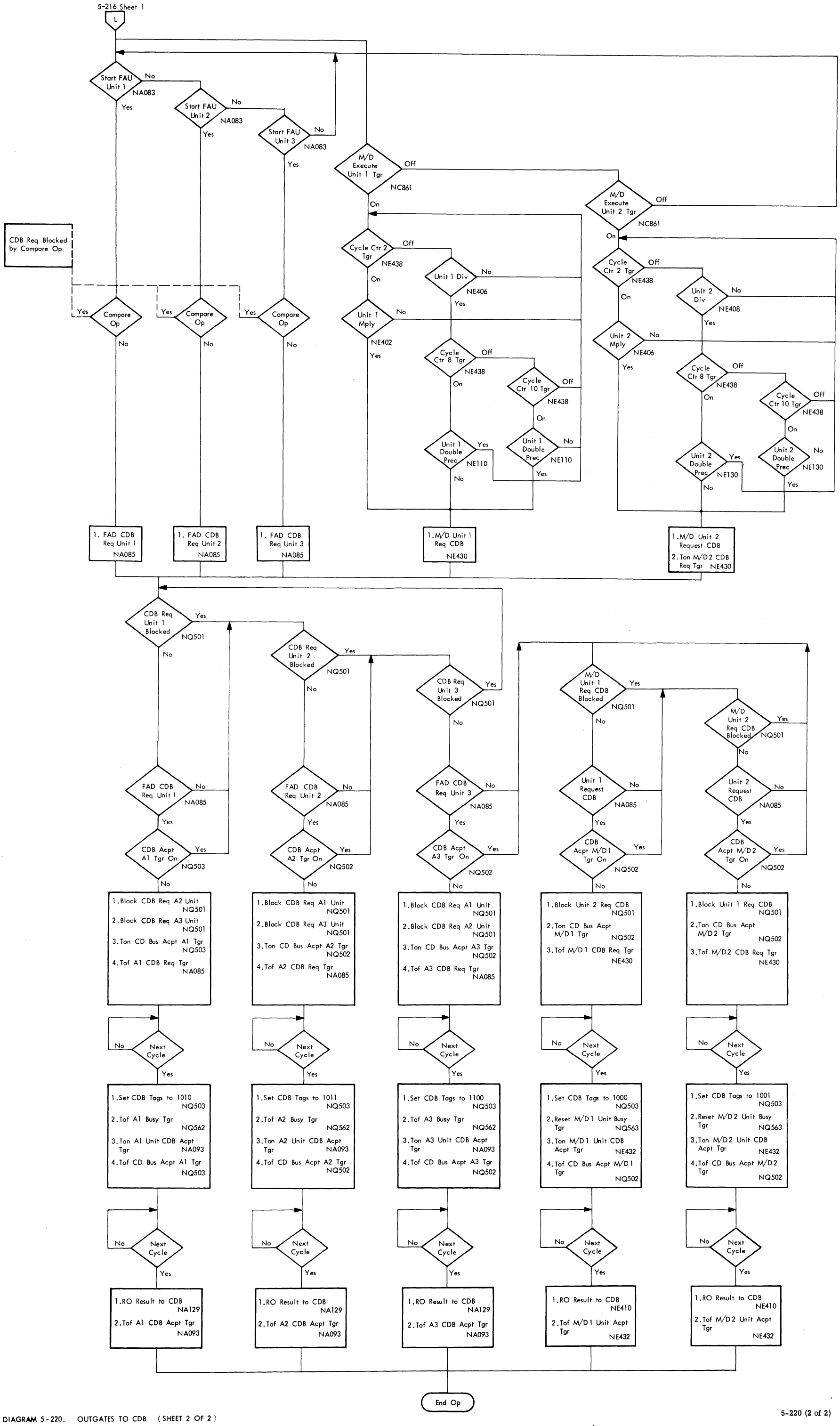
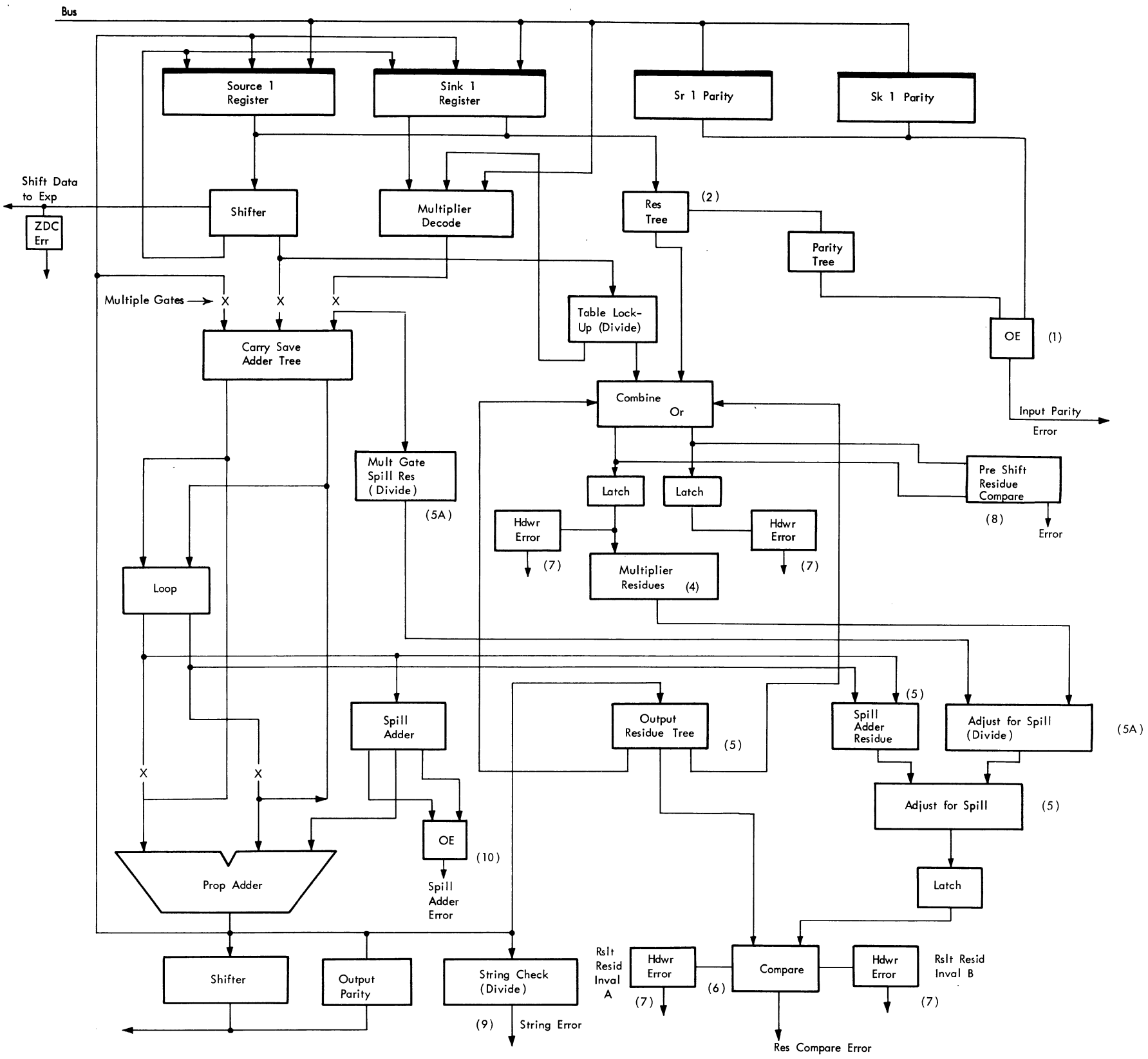


DIAGRAM 5-220. OUTGATES TO CDB (SHEET 2 OF 2)

Fraction Residue and Parity



Exponent Error Checks

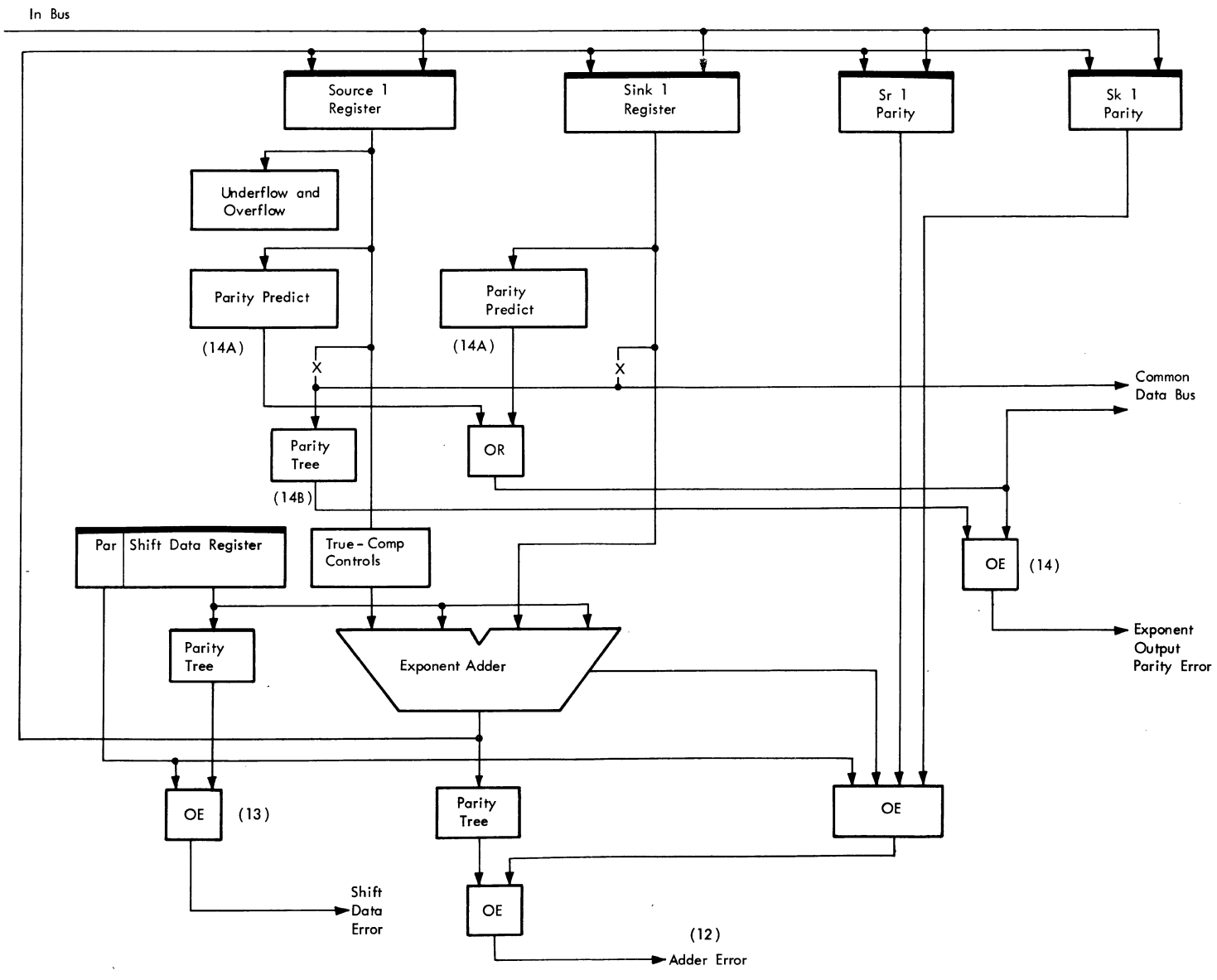
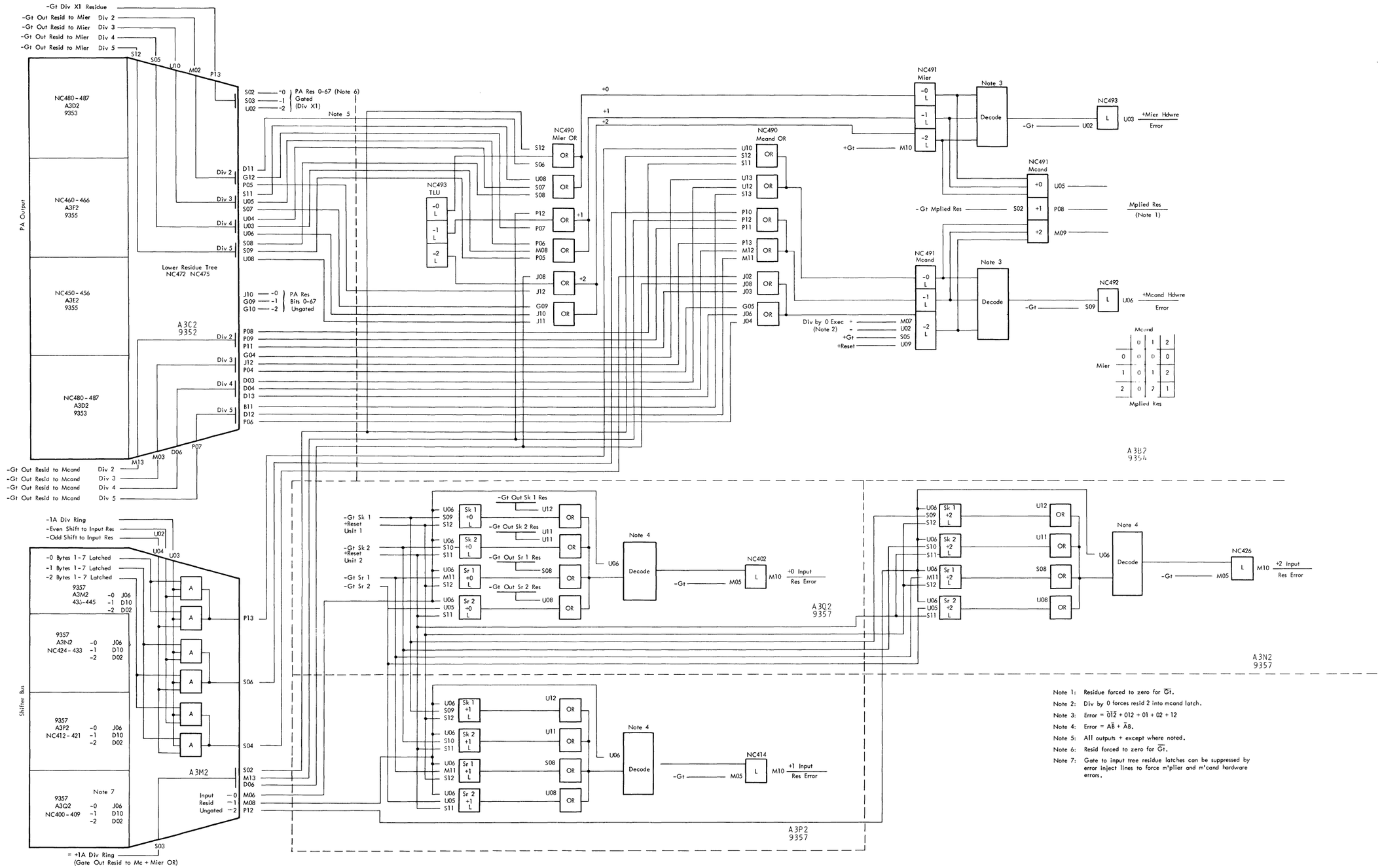


DIAGRAM 5-221. MULTIPLY/DIVIDE ERROR CHECKING

N/C	0	C2
-0	0	2
-1	1	0
-2	2	1

DIAGRAM 5-222. FLP M/D RESIDUE CHECKING (SHEET 1 OF 3)



Note 1: Residue forced to zero for $\overline{Gt}$.

Note 2: Div by 0 forces resid 2 into mcand latch.

Note 3: Error = $\bar{0}\bar{1}\bar{2} + 012 + 01 + 02 + 12$

Note 4: Error = $A\bar{B} + \bar{A}B$.

Note 5: All outputs + except where noted.

Note 6: Resid forced to zero for $\overline{Gt}$.

Note 7: Gate to input tree residue latches can be suppressed by error inject lines to force m'plier and m'cand hardware errors.

DIAGRAM 5-222. FLP M/D RESIDUE CHECKING (SHEET 2 OF 3)

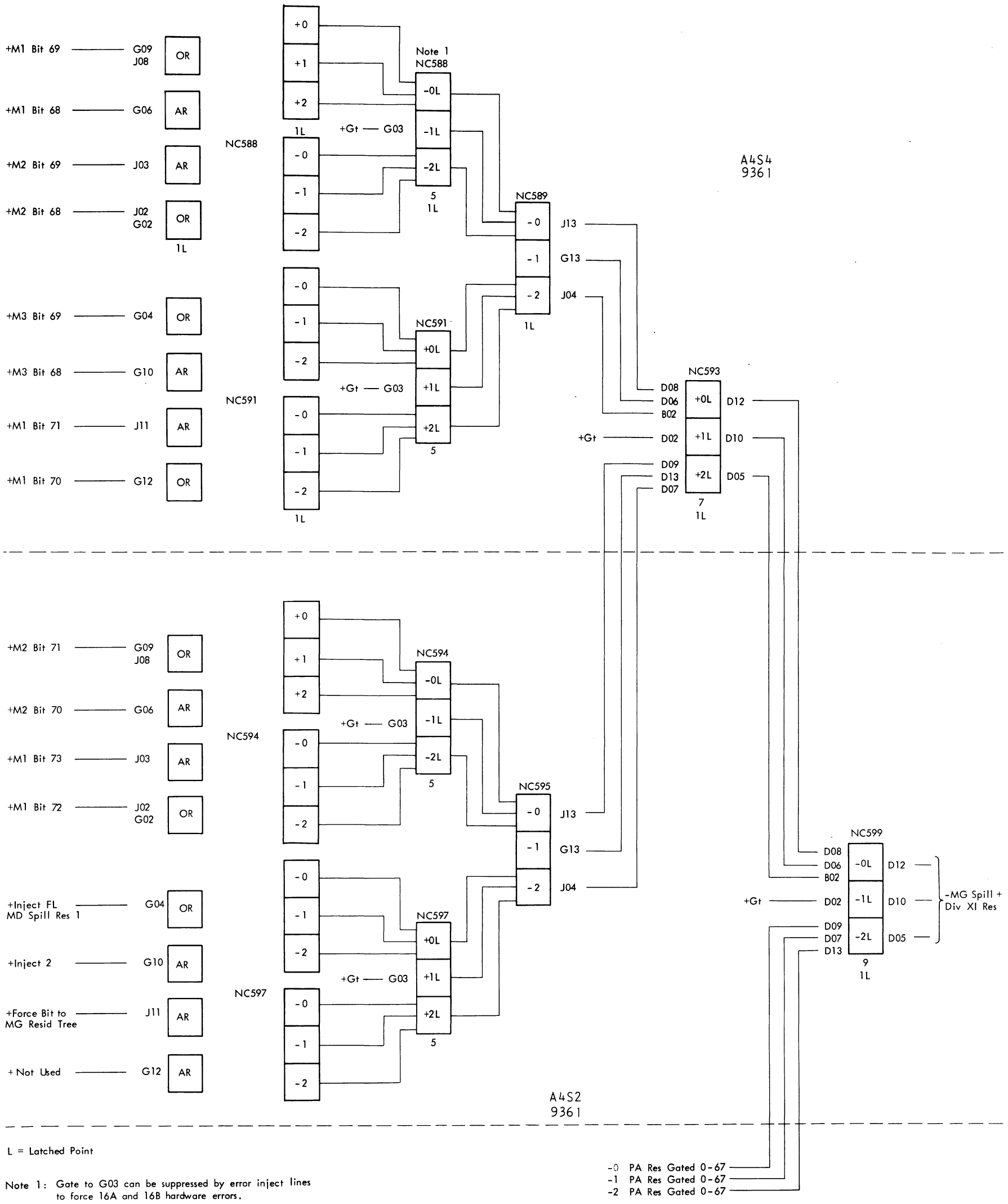


DIAGRAM 5-222. FLP M/D RESIDUE CHECKING (SHEET 3 OF 3)

CUT ALONG LINE

COMMENT SHEET

PROCESSING UNIT — VOLUME 4 — I, FXP, FLP OPERATIONS

FIELD ENGINEERING MAINTENANCE DIAGRAMS, FORM Y22-6674-0

FROM

NAME _____ OFFICE/DEPT NO. _____

CITY/STATE _____ DATE _____

To make this manual more useful to you, we want your comments: what additional information should be included in the manual; what description or figure could be clarified; what subject requires more explanation; what presentation is particularly helpful to you; and so forth.

How do you rate this manual: Excellent ____ Good ____ Fair ____ Poor ____

Suggestions from IBM Employees giving specific solutions intended for award considerations should be submitted through the IBM Suggestion Plan.

NO POSTAGE NECESSARY IF MAILED IN U.S.A.

FOLD ON TWO LINES (LOCATED ON REVERSE SIDE), STAPLE AND MAIL

CUT ALONG LINE

COMMENT SHEET

PROCESSING UNIT — VOLUME 4 — I, FXP, FLP OPERATIONS

FIELD ENGINEERING MAINTENANCE DIAGRAMS, FORM Y22-6674-0

FROM

NAME _____ OFFICE/DEPT NO. _____

CITY/STATE _____ DATE _____

To make this manual more useful to you, we want your comments: what additional information should be included in the manual; what description or figure could be clarified; what subject requires more explanation; what presentation is particularly helpful to you; and so forth.

How do you rate this manual: Excellent ____ Good ____ Fair ____ Poor ____

Suggestions from IBM Employees giving specific solutions intended for award considerations should be submitted through the IBM Suggestion Plan.

NO POSTAGE NECESSARY IF MAILED IN U.S.A.

FOLD ON TWO LINES (LOCATED ON REVERSE SIDE), STAPLE AND MAIL

fold

fold

FIRST CLASS
PERMIT NO. 419
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION

P.O. BOX 390
POUGHKEEPSIE, N.Y. 12602

ATTENTION: FE MANUALS, DEPT. B95

fold

fold

IBM

International Business Machines Corporation
Field Engineering Division
112 East Post Road, White Plains, N.Y. 10601

fold

fold

FIRST CLASS
PERMIT NO. 419
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION

P.O. BOX 390
POUGHKEEPSIE, N.Y. 12602

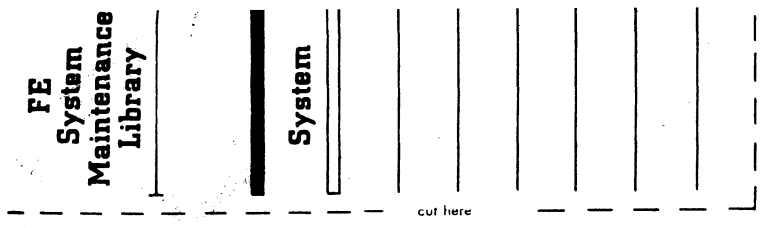
ATTENTION: FE MANUALS, DEPT. B95

fold

fold

IBM

International Business Machines Corporation
Field Engineering Division
112 East Post Road, White Plains, N.Y. 10601



Y22-6674-0

Printed in U.S.A. Y22-6674-0

IBM®

International Business Machines Corporation
Field Engineering Division
112 East Post Road, White Plains, N. Y. 10601