NORD-10/S MICROPROGRAM

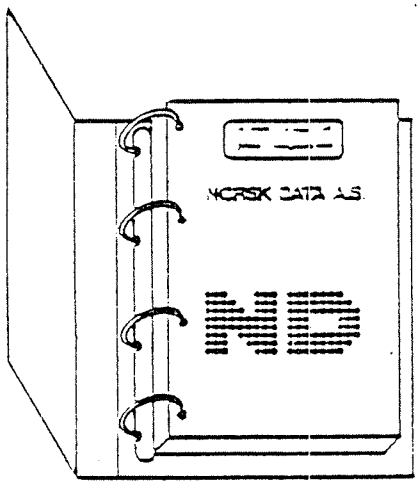
NORSK DATA A.S

NORD-10/S MICROPROGRAM

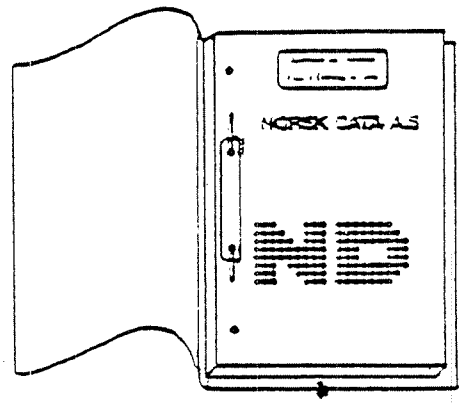
This manual is in loose leaf form for ease of updating. Old pages may be removed and new pages easily inserted if the manual is revised.

The loose leaf form also allows you to place the manual in a ring binder (A) for greater protection and convenience of use. Ring binders with 4 rings corresponding to the holes in the manual may be ordered in two widths, 30 mm and 40 mm. Use the order form below.

The manual may also be placed in a plastic cover (B). This cover is more suitable for manuals of less than 100 pages than for large manuals. Plastic covers may also be ordered below.



A Ring Binder



B Plastic Cover

Please send your order to the local ND office or (in Norway) to:

Documentation Department
Norsk Data A.S
P.O. Box 4, Lindeberg gård
Oslo 10

ORDER FORM

I would like to order

..... Ring Binders, 30 mm, at nkr 20,- per binder

..... Ring Binders, 40 mm, at nkr 25,- per binder

..... Plastic Covers at nkr 10,- per cover

Name .....
Company .....
Address .....
.....
City .....

REVISION RECORD

Revision	Notes
02/78	PRELIMINARY ISSUE

NORD-10/S Microprogram
Publication No. ND-06.010.01

NORSK DATA A.S.
Lørenveien 57, Postboks 163 Økern, Oslo 5, Norway

TABLE OF CONTENTS

\+ + +

READERS – Please Note!!!!

We frequently refer to the NORD computer in this manual as "NORD-10". However, this **does not** mean that it only applies to NORD-10 users. Please note that it **also** applies to NORD-10/S, NORD-12 and NORD-42 users. We have written "NORD-10" merely for convenience sake.

1 INTRODUCTION

The microprogram is designed to implement, in hardware, the instruction set of NORD-10, NORD-42 and NORD-10/S.

The microprocessor instruction set consists of four micro-instructions.

This manual describes the exact format of the four different micro-instructions together with some examples of usage.

The micro-instructions are stored in a 1k x 32 bits Read Only Memory – ROM.

Chapter 7 contains a listing of the μ-program.

The ROM is logically divided into the following sections:

- μ-programmed execution of NORD's 10/S, 10, 42 instruction reportoire
- μ-programmed operator panel driver
- μ-programmed operator communication in stop mode MOPC
- μ-programmed bootstrap loader
- μ-programmed memory check

1.1 *PHILOSOPHY OF MICROPROGRAMMING*

Microprogramming is primarily an orderly and systematic means of implementing control logic. By using microprogrammed control, the CPU control section may be broken down into well-defined subsections. This approach simplifies design, documentation and testing.

Flexibility is an advantage of microprogramming: new instructions may be added without changing hardware design or test methods. Alternate instruction sets are available: at present one of two floating point forms may be ordered; 32 bit or 48 bit.

1.2 *MICRO PROCESSOR INSTRUCTION SET*

The microprocessor instruction set consists of four instructions. These are ARITHMETIC, INTERBLOCK, JUMP and LOOP. This chapter deals with the exact format of these four instructions together with examples on how they may be used.

The operation code is contained in bits 30 and 31 in the Read Only Memory – ROM.

ROM 31	ROM 30	Instruction
0	0	ARITHMETIC
0	1	INTERBLOCK
1	0	JUMP
1	1	LOOP

Refer to Figure 1.1. The format shown applies to Read Only Memory and not Microinstruction Register – MIR. The two are not necessarily identical, due to the function of the OR logic.

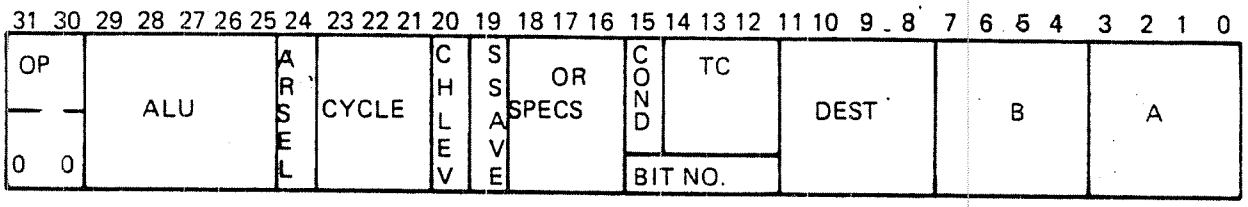
The four instructions will be described in the following figure.

1.3 *MICROPROGRAM CONTROL*

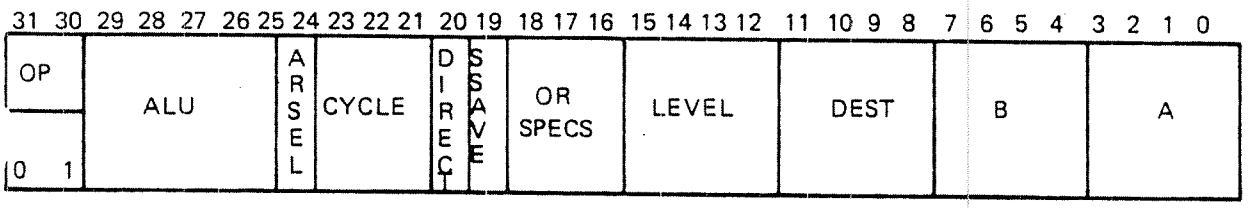
The CPU control logic transforms the content of the instruction register (IR) into a sequence of actions on CPU registers, memory and/or I/O system. These actions are controlled by a set of control signals to registers, selectors, arithmetic elements, memory, I/O system, etc. In a non-microprogrammed machine, these signals are derived directly from the instruction register and a large and complicated Time Counter/Cycle Counter. This type of control logic is not easily structured and is difficult to describe and understand.

A block diagram of the transformation from machine instructions (IR) into a sequence of microinstructions is shown in Figure 1.2. Each NORD machine instruction is executed by a sequence of one or more micro-instructions, a microprogram routine.

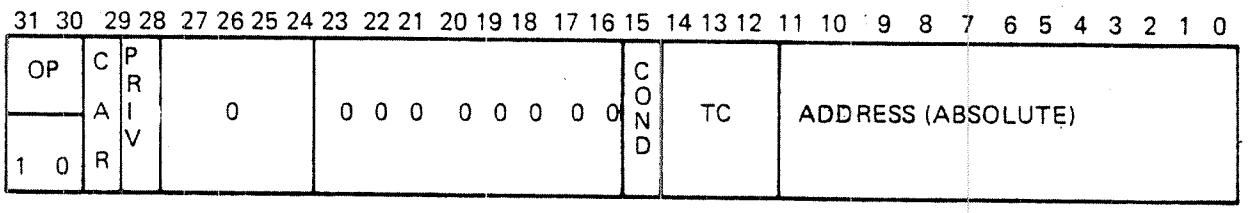
ARITHMETIC:



INTERBLOCK:



JUMP:



LOOP:

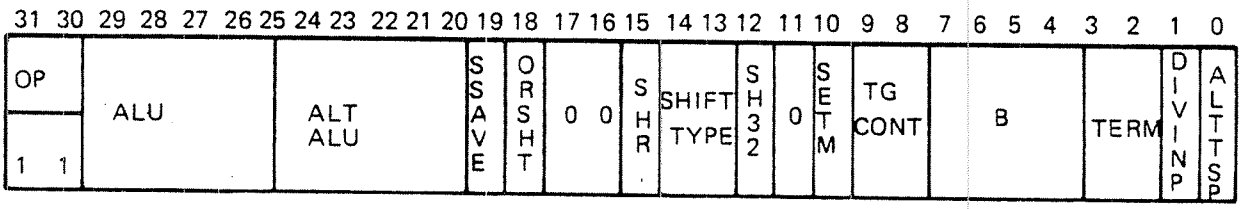


Figure 1.1: *Micro Instruction's Bit Assignment*

The microprogram entry point is generated from the machine instruction operation code by hardware. This corresponds to the instruction decoding in a non-microprogrammed machine. The microprogram is controlled by a microprogram counter, which points to the next microinstruction to be executed from the Read Only microprogram memory (ROM). Branching may be done by the microinstruction JUMP. The microinstruction counter may be read, thus providing a simple subroutine capability. ROM word length is 32 bits. ROM content is clocked into the microinstruction register, MIR, at the end of each microinstruction. The microword contains information to establish the setting of the control lines for each cycle.

Since the microword (32 bits) is not sufficient to establish the setting of all the control lines, the microword is divided into four groups or instructions, given by ROM bits 31 and 30. The remainder of the 30 bits are in some of the instructions divided into fields having the same meaning in different instructions.

The microinstruction format is tailored to the CPU structure, while keeping the target instruction set (NORD-10) in mind in order to maintain execution efficiency. Many control signals are taken directly from MIR outputs, while others are derived by simple logic from MIR bits and a small Time Counter.

1.4 *ENTRY POINT GENERATOR*

Refer to Figure 1.2 for the following discussion.

A microprogram terminates by fetching the next machine instruction to be executed. The instruction is placed in the instruction register (IR). The Entry Point Generator (EPG) will then generate a unique address (Entry Point) based on the content of the instruction register (IR). This address will be clocked into the microprogram counter (MPC).

Entry points for all NORD-10 instructions which do not have sub-instruction fields, are 100_8, 102_8, . . ., 172_8 for operation codes 0, 1, . . ., 35_8, respectively; i.e., the Entry Point equals 100 + (operation code) · 2.

Example:

LDA – opcode (bits 11, 12, 13, 14 and 15) = 01001 = 11_8

Entry point = $100 + 11_8 : 2 = 122_8$

In location 122_8 in ROM, resides the first (of two) microinstructions which constitutes the microprogram for the LDA instruction.

A spacing of two locations between the Entry Points is chosen, due to the fact that most of these microprograms occupy two locations of ROM. This applies to the instructions:

LDA, LDX, LDT,
STA, STX, STT, STZ,
JMP, JPL.

Some other instructions, such as FSB, FAB, FMU, FDV, STF, etc., require more than two locations, but a jump to another address in the ROM where the rest of the microprogram for relevant instruction resides is executed.

For all other instructions, the Entry Point is generated according to a spacing of 16 between the EP's for the main instruction operation code such as CJP, ROP, etc. Refer to table "EP for Instructions with Sub-instructions".

Section 3.2 gives the entry points for the NORD-10 instructions.

1.5 *THE OR LOGIC*

The instruction set for the NORD-10 may be divided into two main groups:

1. Instructions well defined by the operation code (upper 5 bits), will not require an OR logic to be implemented.

   Example:

   LDA, STA, ADD, AAA, SAA.

2. Instructions not completely defined by the operation code are defined by their subinstruction field. The subinstruction field will give additional information to the operation code.

   Example:

   - The subinstruction field of a SHIFT instruction will give information about shift direction and shift method.
   - The subinstruction field of a SKIP instruction will give information about a skip condition.

To reduce the number of Entry Points in the ROM (not having one for each combination of subinstruction field) the OR logic is introduced to take information directly from the subinstruction field in the Instruction Register (IR) to the Micro-Instruction Register (MIR). The ROM bits 16, 17 and 18 (8 combinations) decide which IR bits are to be transferred to MIR.

The subinstruction field (giving the large instruction reportare combinations) gives, by means of the OR logic, the microprocessor the necessary information through a minimum of logic.

From Figure 1.2 we can see that micro-instruction register (MIR) bits 0-15 is the output from the OR logic. Table 1.1 describes the origin of the MIR 0-15 for the 8 different OR specifications.

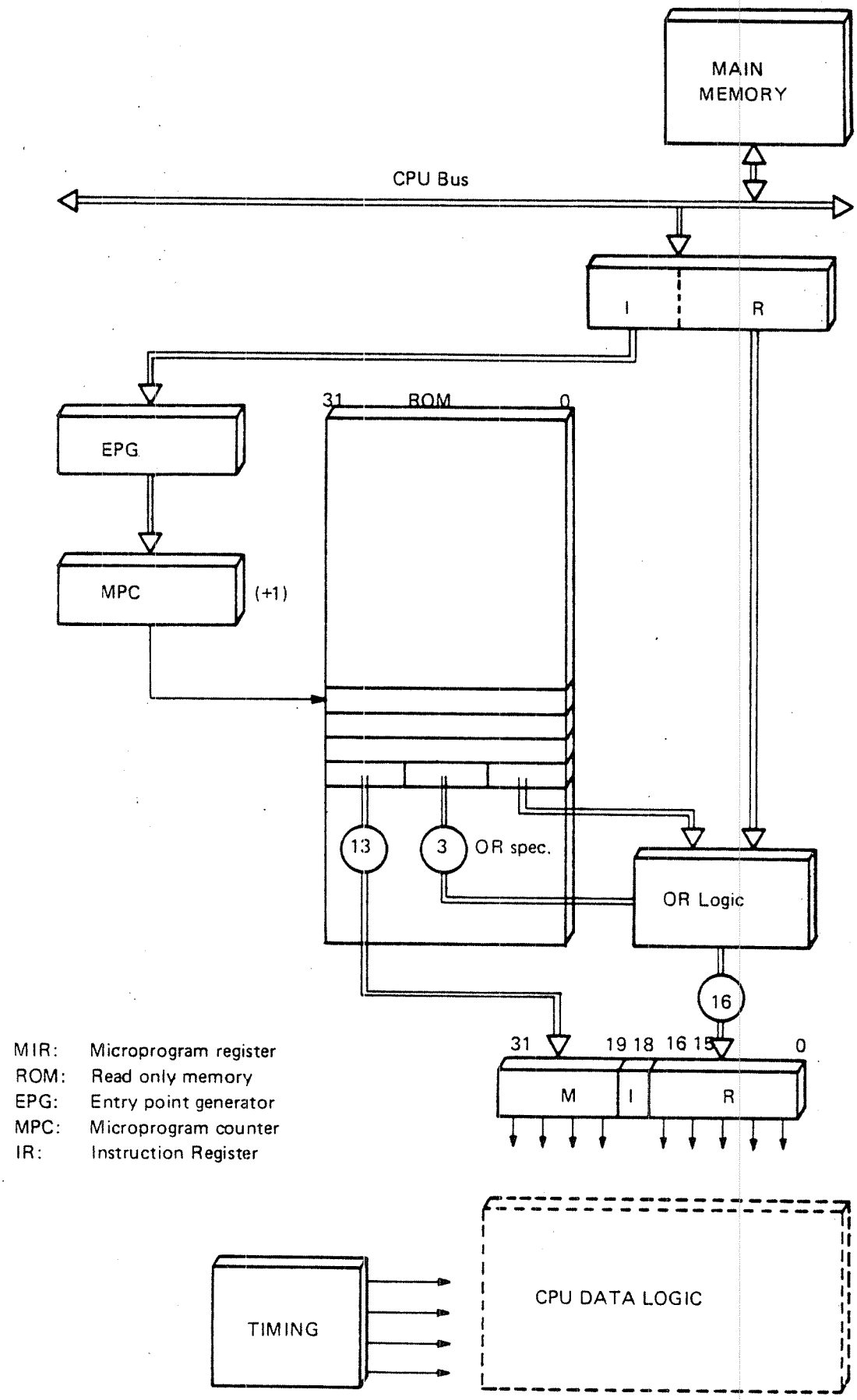


Figure 1.2: *CPU Control Section*

OR Specifications				MIR 0-15																Comments
No.	Name	MNE	Instr.	15	14	13	12	11	10	9	8	7	6		4	3	2	1	0	
0	NO OR	–	–	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0	-
1	OR for BOP without Dest.	ORBWO	ARIT	I6	I5	I4	I3	R11	R10	R9	R8	R7	R6	R5	R4	0	I2	I1	I0	If I0-2 ≠ 0
			ARIT	I6	I5	I4	I3	R11	R10	R9	R8	R7	R6	R5	R4	1	0	0	0	If I0-2 = 0
			INTB	I6	I5	I4	I3	R11	R10	R9	R8	R7	R6	R5	R4	0	I2	I1	I0	If I0-2 ≠ 0
			INTB	I6	I5	I4	I3	R11	R10	R9	R8	R7	R6	R5	R4	1	0	0	0	If I0-2 = 0
			INTB	I6	I5	I4	I3	R11	R10	R9	R8	R7	R6	R5	R4	1	1	1	0	If I0-2 = 2
3	OR for BOP with Dest.	ORBW	ARIT	I6	I5	I4	I3	0	I2	I1	I0	R7	R6	R5	R4	0	I2	I1	I0	If I0-2 ≠ 0
			ARIT	I6	I5	I4	I3	1	0	0	0	R7	R6	R5	R4	1	0	0	0	If I0-2 = 0
			INTB	I6	I5	I4	I3	0	I2	I1	I0	R7	R6	R5	R4	R3	R2	R1	R0	If I0-2 ≠ 0
			INTB	I6	I5	I4	I3	1	1	1	0	R7	R6	R5	R4	R3	R2	R1	R0	If I0-2 = 0
			INTB	I6	I5	I4	I3	1	1	1	0	R7	R6	R5	R4	R3	R2	R1	R0	If I0-2 = 2
4	OR for SKP, SHT	ORSHT	ARIT	R15	I10	I9	I8	R11	R10	R9	R8	R7	I2	I1	I0	0	I5	I4	I3	If not COND.
			ARIT	R15	I10	I9	I8	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0	If COND.
			LOOP	I5	I10	I9	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0	–
5	OR for ROP	ORROP	ARIT	R15	R14	R13	R12	0	I2	I1	I0	R7	I2	I1	I0	0	I5	I4	I3	If I6 = 0
			ARIT	R15	R14	R13	R12	0	I2	I1	I0	R7	0	0	0	0	I5	I4	I3	If I6 = 1
6	OR for SWAP cycle 2	ORSW2	ARIT	R15	R14	R13	R12	0	I2	I1	I0	R7	I2	I1	I0	0	I5	I4	I3	If not COND.
			ARIT	R15	R14	R13	R12	0	I2	I1	I0	R7	R6	R5	R4	R3	R2	R1	R0	If COND.
7	OR for SWAP cycle 3	ORSW3	ARIT	R15	R14	R13	R12	0	I5	I4	I3	R7	I2	I1	I0	0	I5	I4	I3	–

ARIT = ARITHM
INTB = INTERBLOCK

Ixx = IRxx
Rxx = ROMxx

COND. = CONDITIONAL

Table 1.1: *OR Specifications*

2 MICROINSTRUCTION DESCRIPTION

2.1 *THE ARITHMETIC MICROINSTRUCTION*

This is the most frequently used μ-instruction in the microprogram. It is used to perform arithmetical as well as logical operations. Refer to Table 2.1.

This μ-instruction is used to set up memory communication (cycle specifications).

For communication with the internal registers three special cases of the instruction exist. Those cases are illustrated in Figures 2.3, 2.4, 2.5, and 2.6.

If bit 15 in an ARITHMETIC instruction is set, the execution of the microinstruction is dependent on the result of a specified test.

The CARM instruction is a conditional ARITHMETIC instruction using the most significant unit.

For a CARM instruction, the arithmetical or logical operation specified will not be executed if the result of the specified test is false. Any cycle specification will, however, be executed.

Note: It is the result of a previous arithmetical or logical operation using the most significant unit, which is tested.

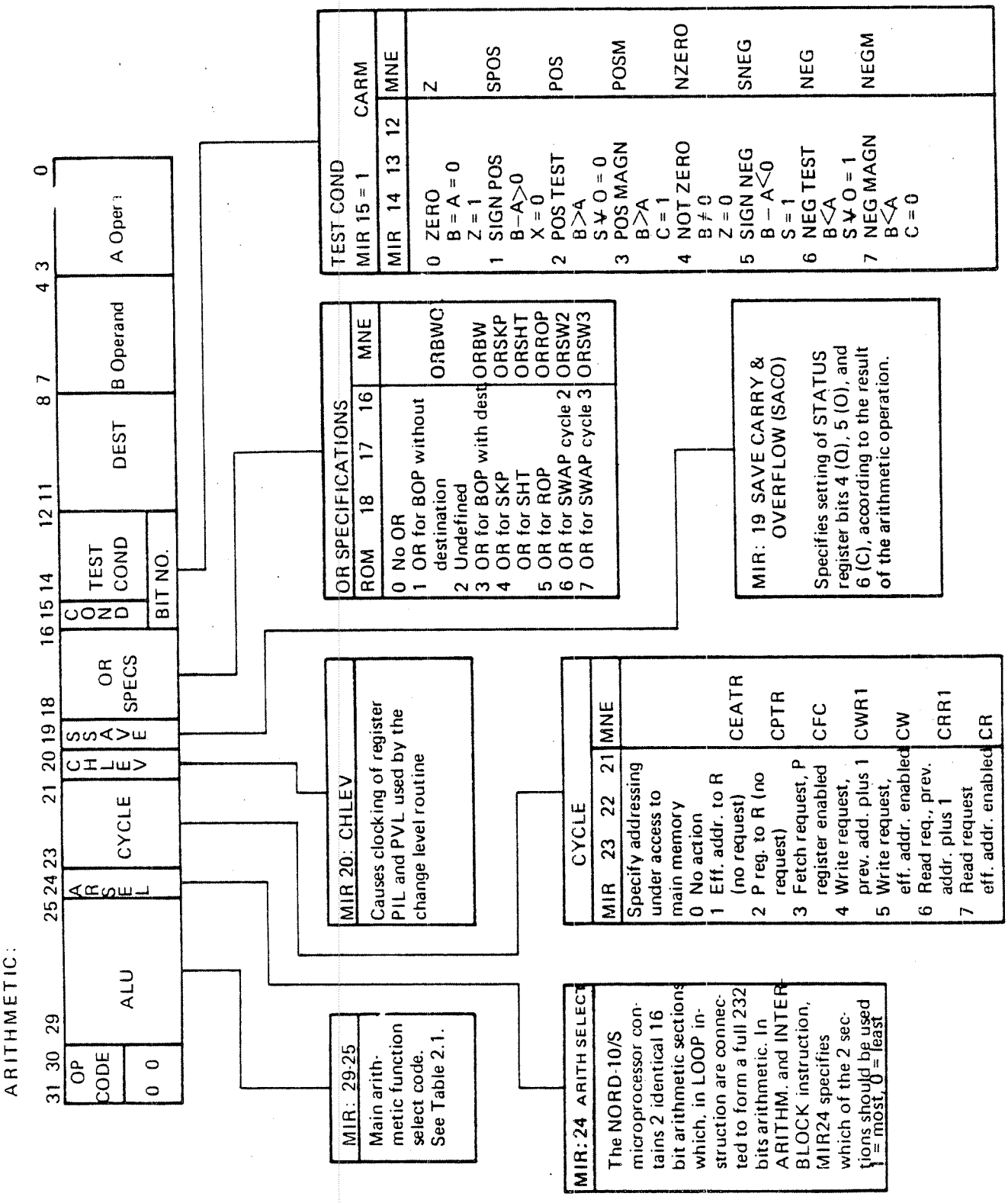


Figure 2.1: *Arithmetic µinstruction – Bit Assignment I*

MIR				LOGICAL OPERATION MIR 29 = 1		ARITHMETIC OPERATIONS MIR 29 = 0	
28	27	26	25	Function	Mne	Function	Mne
0	0	0	0	$\overline{B}$	BDIRC	B–1	BM1
0	0	0	1	$\overline{B \cdot A}$	ANDC		
0	0	1	0	$\overline{B} + A$	ORCB	B–A–1	BMAM1
0	0	1	1	LOGICAL 1	ONE	B	BD1
0	1	0	0	$\overline{B + A}$	ORC		
0	1	0	1	$\overline{A}$	ADIRC	B + A + carry	PLUS ADDC
0	1	1	0	B ∀ A	EXORC	(B–A–1) + carry	BMAM1 ADDC
0	1	1	1	$B + \overline{A}$	ORCA		
1	0	0	0	$\overline{B} \cdot A$	ANCB		
1	0	0	1	B ∀ A	EXOR	B + A	PLUS
1	0	1	0	A	ADIR		
1	0	1	1	B + A	OR		
1	1	0	0	LOGICAL 0	ZERO		
1	1	0	1	$B \cdot \overline{A}$	ANDCA	B + A + 1	PLUS ADD1
1	1	1	0	B · A	AND	B–A	BMINA
1	1	1	1	B	BDIR	B + 1	BDI ADD1

Table 2.1: *Function Select Codes*

ARITHMETIC:

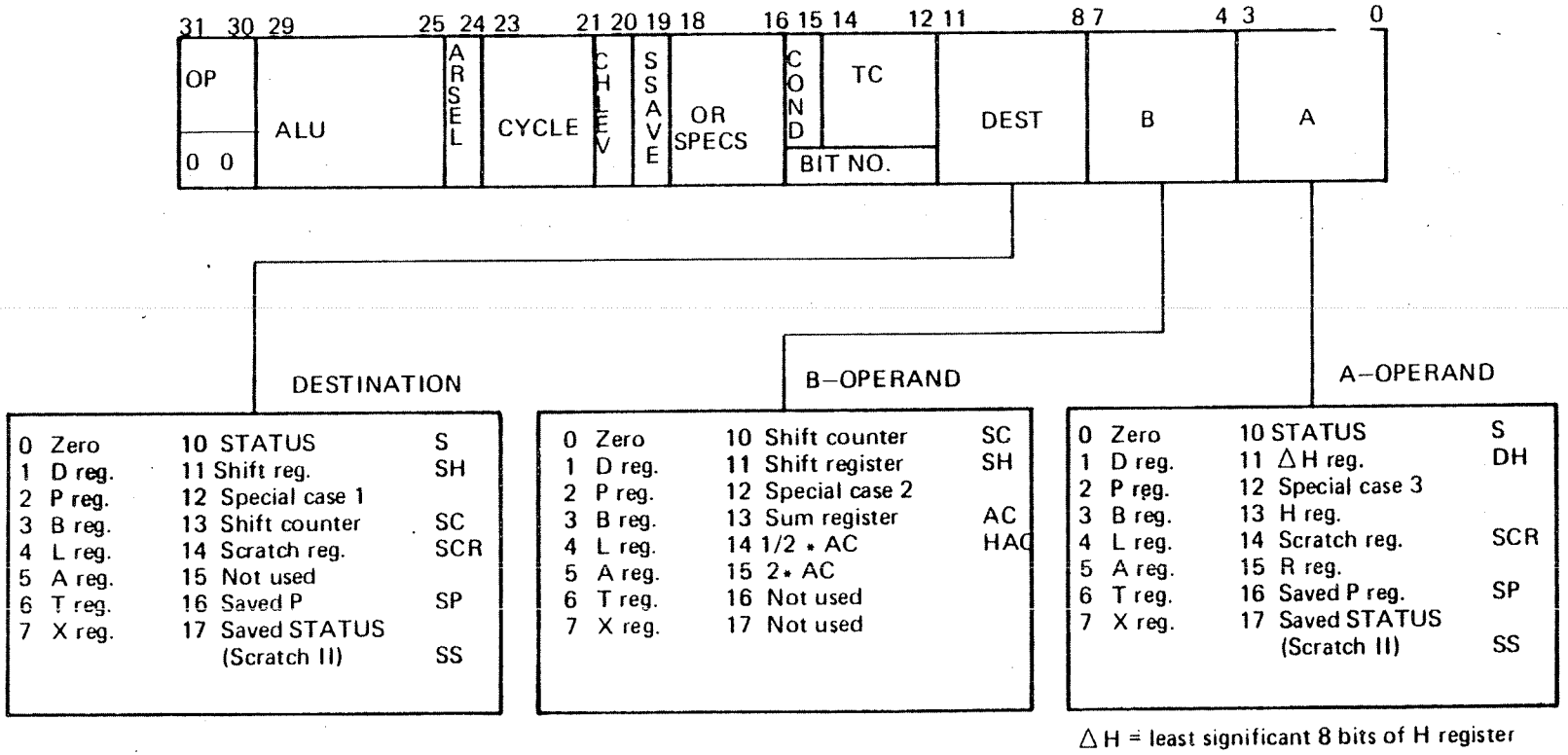
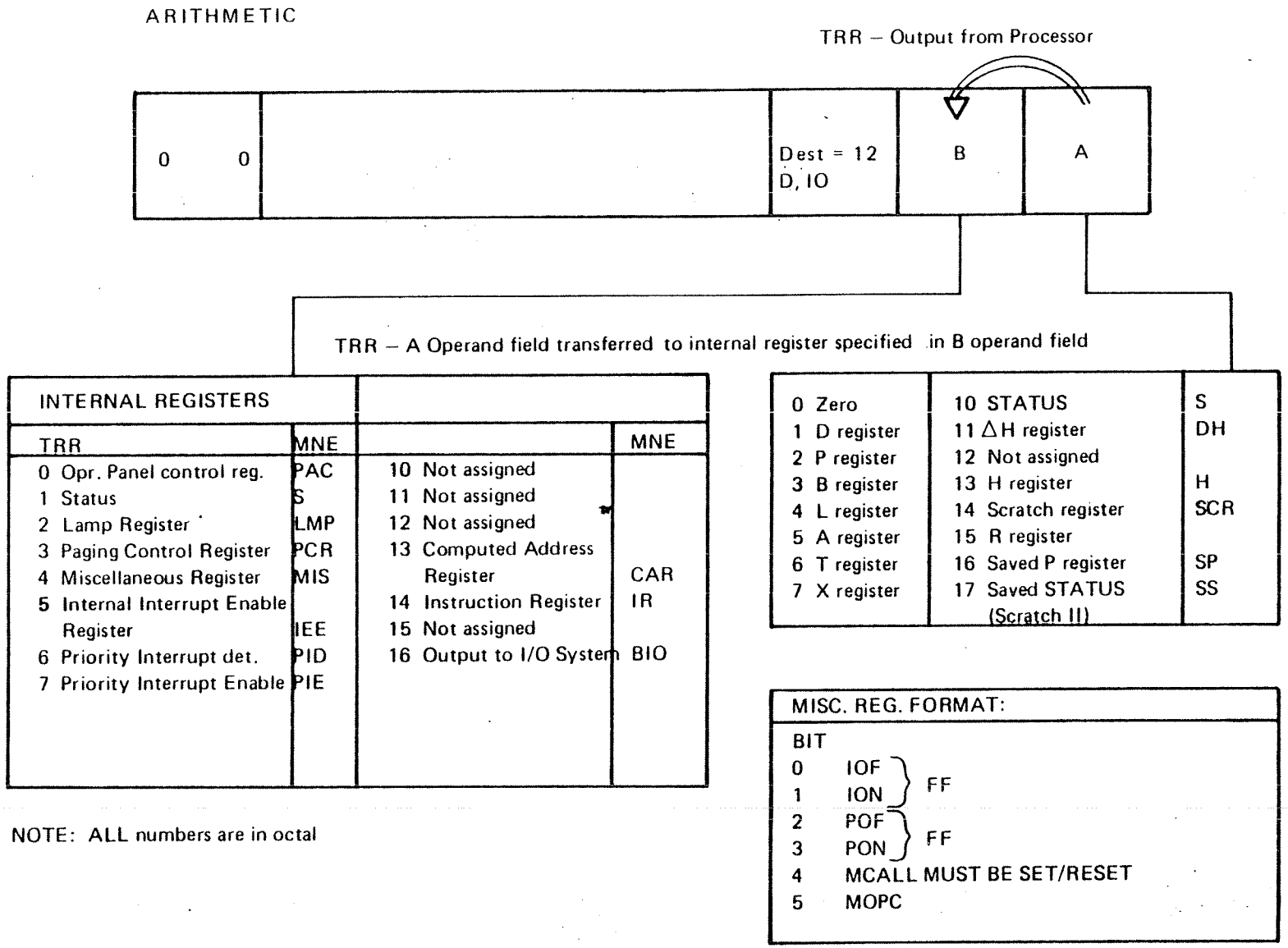
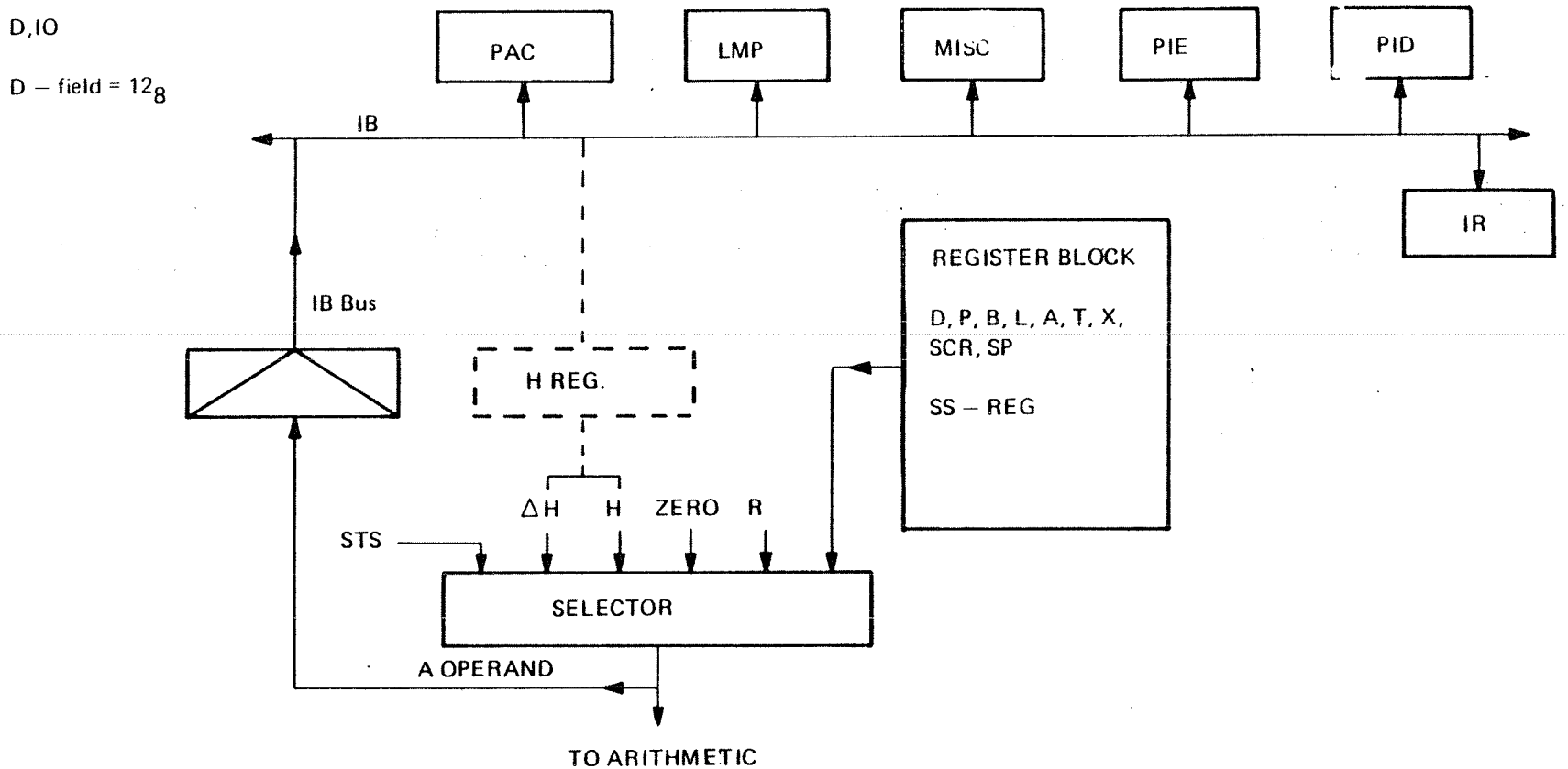


Figure 2.2. *Arithmetic μ- instruction – Bit Assignment II*



NOTE: ALL numbers are in octal

Figure 2.3: *Special Case 1*

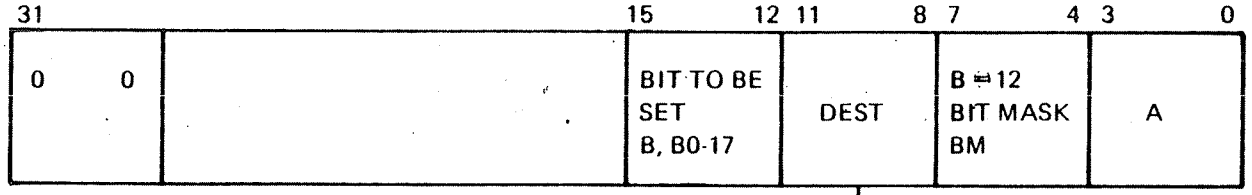


EXAMPLE:

```
ARM   A, SCR   B, PID   D, IO   % SCR → PID
ARM   A, Z     B, PIE   D, IO   % O → PIE
ARM   A, SCR   B, PAC   D, IO   % SCR → PAC
ARM   A, P     B, CAR   D, IO   % P → CAR (IR)
ARM   A, SCR   B, LMP   D, IO   % SCR → Lamp Register
                                Special Case: D = 12
```

Figure 2.4: *Special Case 1 – Illustration*

ARITHMETIC



B OPERAND = BIT MASK

BIT 12-15 DEFINED BIT NO. (Position) TO BE SET IN SELECTED DESTINATION REGISTER OR THE AC REGISTER

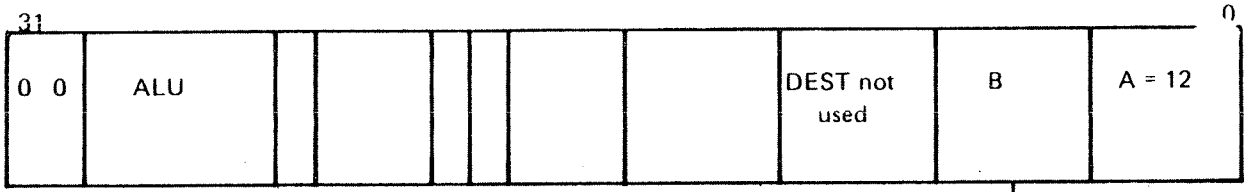
				MNE
0	None	10_8	STATUS	S
1	D register	11	Shift register	SH
2	P register	12	Illegal	
3	B register	13	Shift counter	SC
4	L register	14	Scratch register	SCR
5	A register	15	Not used	
6	T register	16	Saved P	SP
7	X register	17	Saved STATUS (Scratch II)	SS

EXAMPLE:

```
LOGM BDIR B, B4 D, SCR      % 20 (BIT 4 = 1) → SCR
ARM CR BM1 B, B5 D, SS      % 40 (BIT 5 = 1) – 1 → SS ) : 37 → SS
```

Figure 2.5: *Special Case 2*

ARITHMETIC



TRA = B operation field transferred to H register

NOTE:
All numbers are in octal. On Memory Control the TRXX signals are in decimal.

INTERNAL REGISTERS			
TRA	MNE		MNE
0 Oper. panel status register	PAS	10 Not used	
1 Status Register	S	11 Decoded PIL: causes CPU to STOP if interrupt is off	DPIL (PIM)
2 Oper. Switch Register	OPR		
3 Paging Status Register	PGS	12 Auto load description	ALD
4 Previous level	PVL	13 Memory error status register	PES
5 Internal interrupt code	IIC	14 Micro program counter	MPC
6 Priority interrupt detect	PID	15 Memory error address register	PEA
7 Priority interrupt enable register	PIE	16 Input to H from I/O system	BIO
		17 IR0-3 as B operand used by TRA/TRR instruction	BIR3

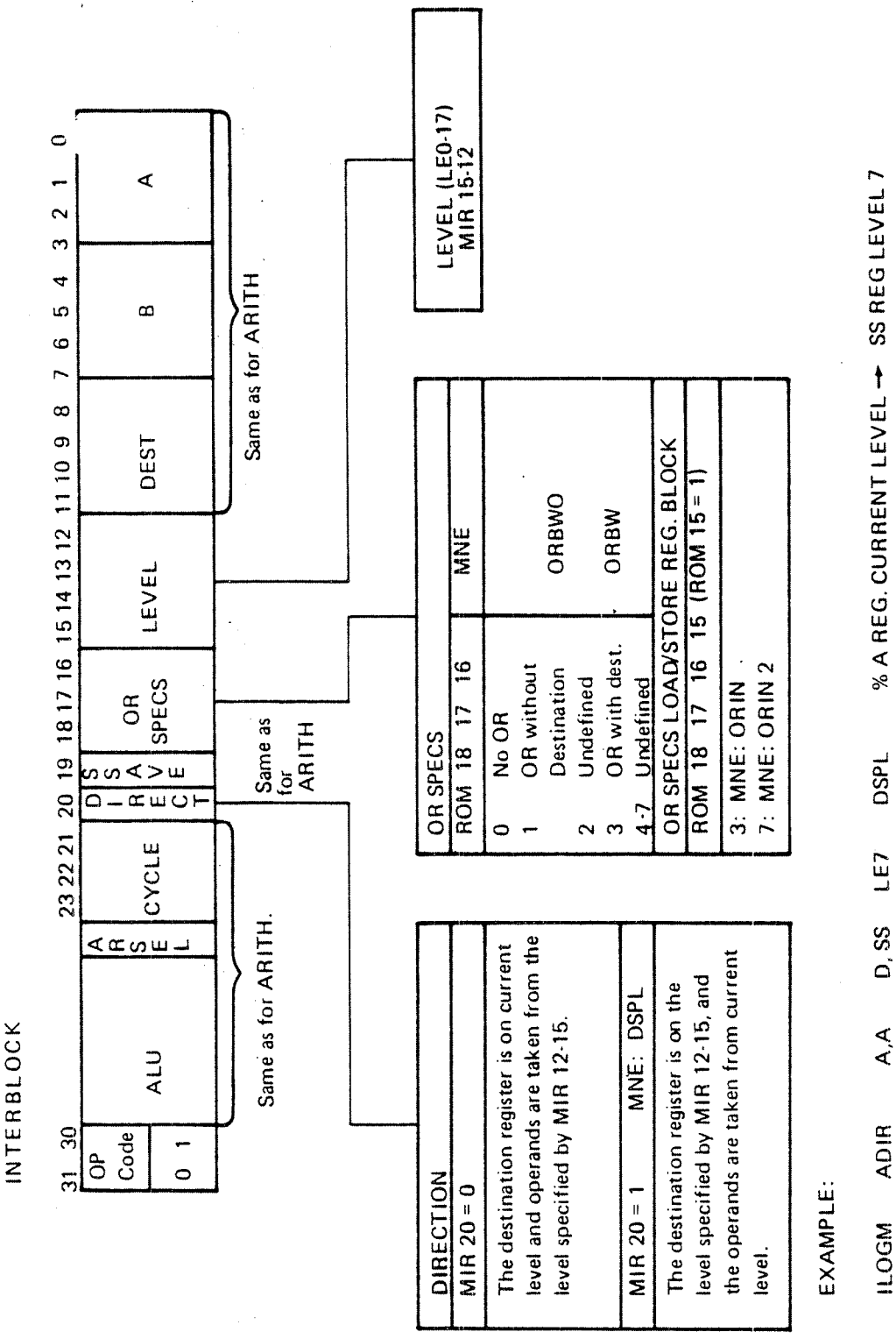
EXAMPLE:

```
LOGM    A, IO    BIO      % I/O BUS → H REG.
LOGM    A, IO    B, PES   % PES → H REG.
ARM     A, IO    B, PIM   % PIM (Decoded PIL) → H
ARM     A, IO    B, PAS   % PANEL STATUS → H
```

Figure 2.6: *Special Case 3*

2.2 *THE INTERBLOCK MICROINSTRUCTION*

Refer to Figure 2.7. The INTERBLOCK microinstruction is used for interlevel communication, i.e., for implementing the IRR and IRW instruction. The Interblock instruction is also used by the microprogram for saving and returning of information from scratch registers on different levels. Bit 20 is used for defining the direction of communication as described in Figure 2.7. The two levels to communicate between is always the current level, as specified by PIL – Current Program Level indicator, and the level specified by bits 12 to 15.



EXAMPLE:

```
ILOGM   ADIR    A,A     D, SS   LE7     DSPL    % A REG. CURRENT LEVEL → SS REG LEVEL 7
ILOGM   ADIR    D, A    A, SS   LE7             % SS LEVEL 7 → A REG. CURRENT LEVEL
```

Figure 2.7: *Interblock μ-instruction – Bit Assignment*

2.3 *THE JUMP MICROINSTRUCTION*

Refer to Figure 3.9. The JUMP instruction may be divided into:

Undonditional JUMP	– JMP
Conditional JUMP	– CJMP
Privileged Instruction JUMP	– JMP PRIV
Computed Address Register JUMP	– JMP, CAR

The JMP instruction takes bits 0 - 11 as an *absolute* address.

The CJMP takes bits 0 - 11 as an an absolute jump address if the specified condition is TRUE. If the specified condition is FALSE, the instruction following the CJMP will be executed.

The JMP PRIV instruction is used to generate Privileged Instruction Internal Interrupt (bit 6 in IIC if the privileged instruction executed is on Ring 0 or Ring 1). IOX, IOT and IDENT are decoded separately on 1058 Interrupt Control.

The JMP, CAR instruction is used during subroutine handling and is analogue to the EXIT machine instruction in that the absolute jump address is taken from Computed Address Register – CAR, which contains the main program return address.

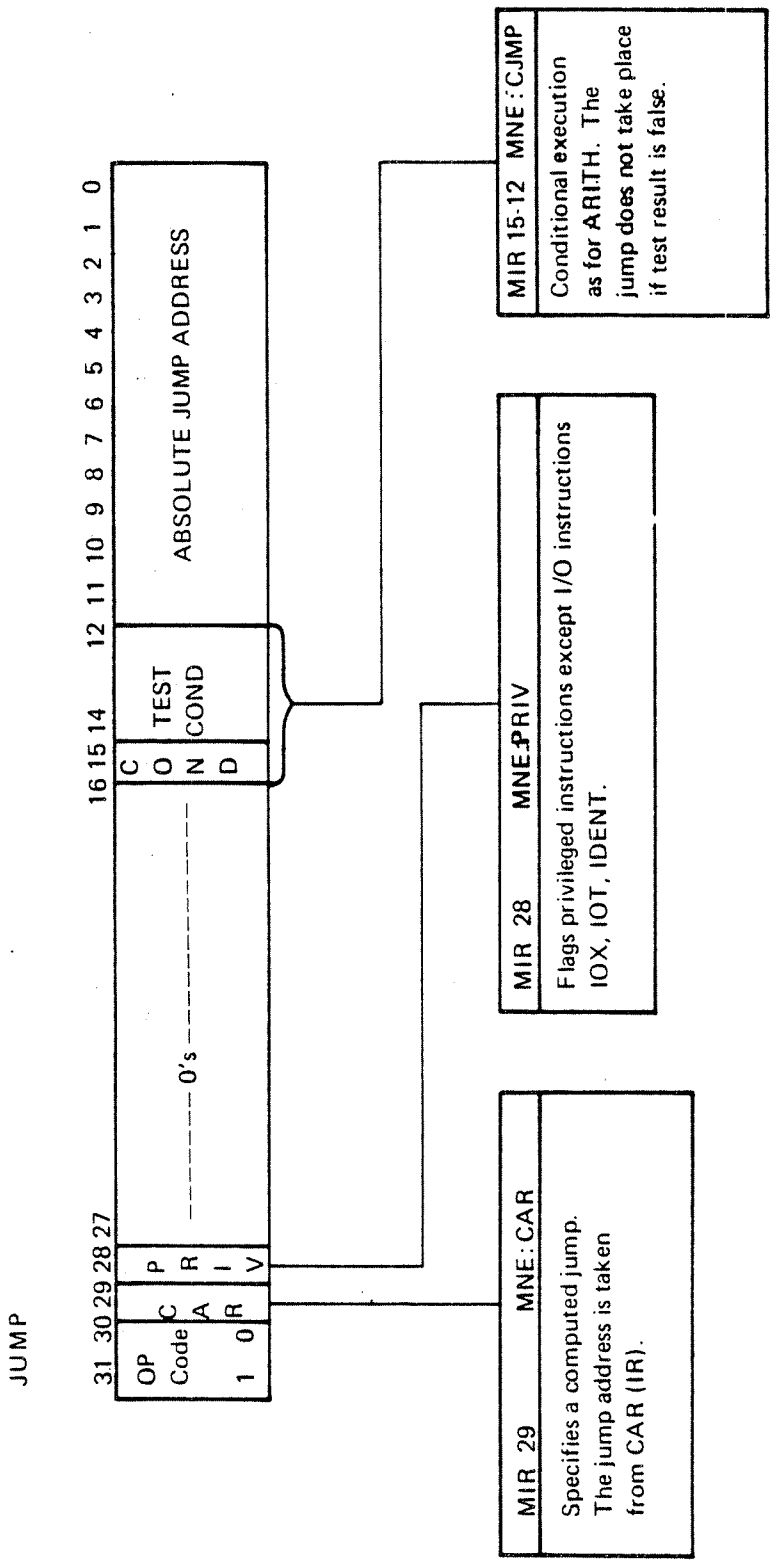


Figure 2.8: *Jump μ-instruction — Bit Assignment*

2.4 *THE LOOP MICROINSTRUCTION*

Refer to Figure 2.10. The LOOP instruction is used in all SHIFT, MULTIPLY and DIVIDE operations. When execution of a LOOP instruction is started, the instruction will repeateadly be executed until a specified terminating condition occurs. In other words, the LOOP instruction will remain in the Microinstruction Register and no incrementing of the MPC (Microprogram Counter) will take place before the terminating condition is met.

During shift operations, for instance, the LOOP instruction will be executed as many times as the number of shifts specified.

The LOOP instruction may specify any arithmetical or logical operation as for the ARITHMETIC instruction. However, the LOOP instruction may specifiy operations on both ALU's in the same instruction (depending on bit 0). The LOOP instruction may, thus, effectively operate a 32 bits ALU. This is the case for all floating and double precision instructions.

Refer to Figure 2.10. Bit 0 controls the Alternative Arithmetic function select. If bit 0 = 0, the alternative function select will be used by both ALU's if the most significant arithmetic module's shift register bit 15 (SH_{31}) = 1. This is used by the multiply routines.

When bit 0 = 1, the alternative function select will be used if least significant arithmetic module's shift register bit 0 (SH_0) = 1. This is used by the divide routines.

LOOP

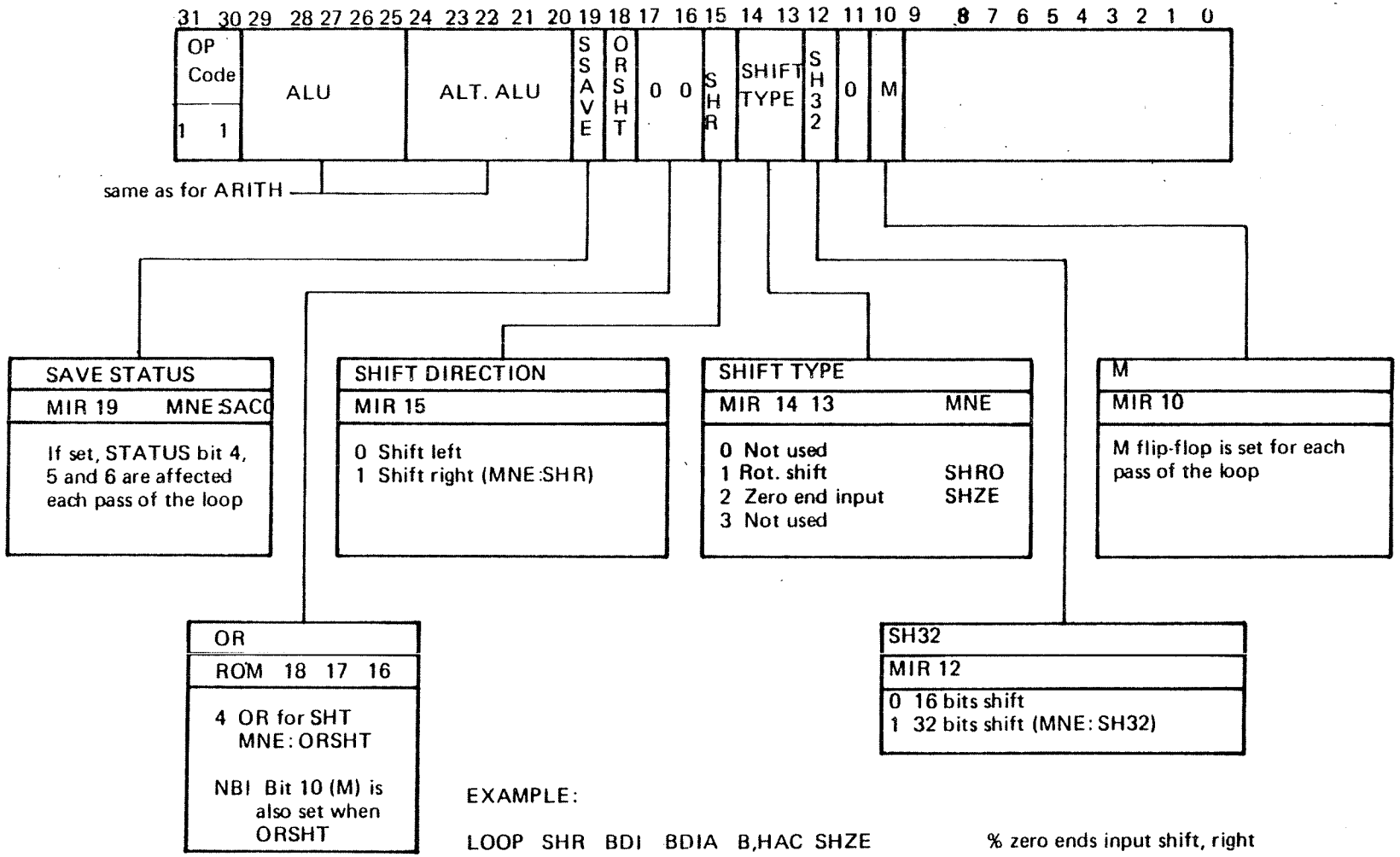


Figure 2.9: *Loop μ-instruction — Bit Assignment I*

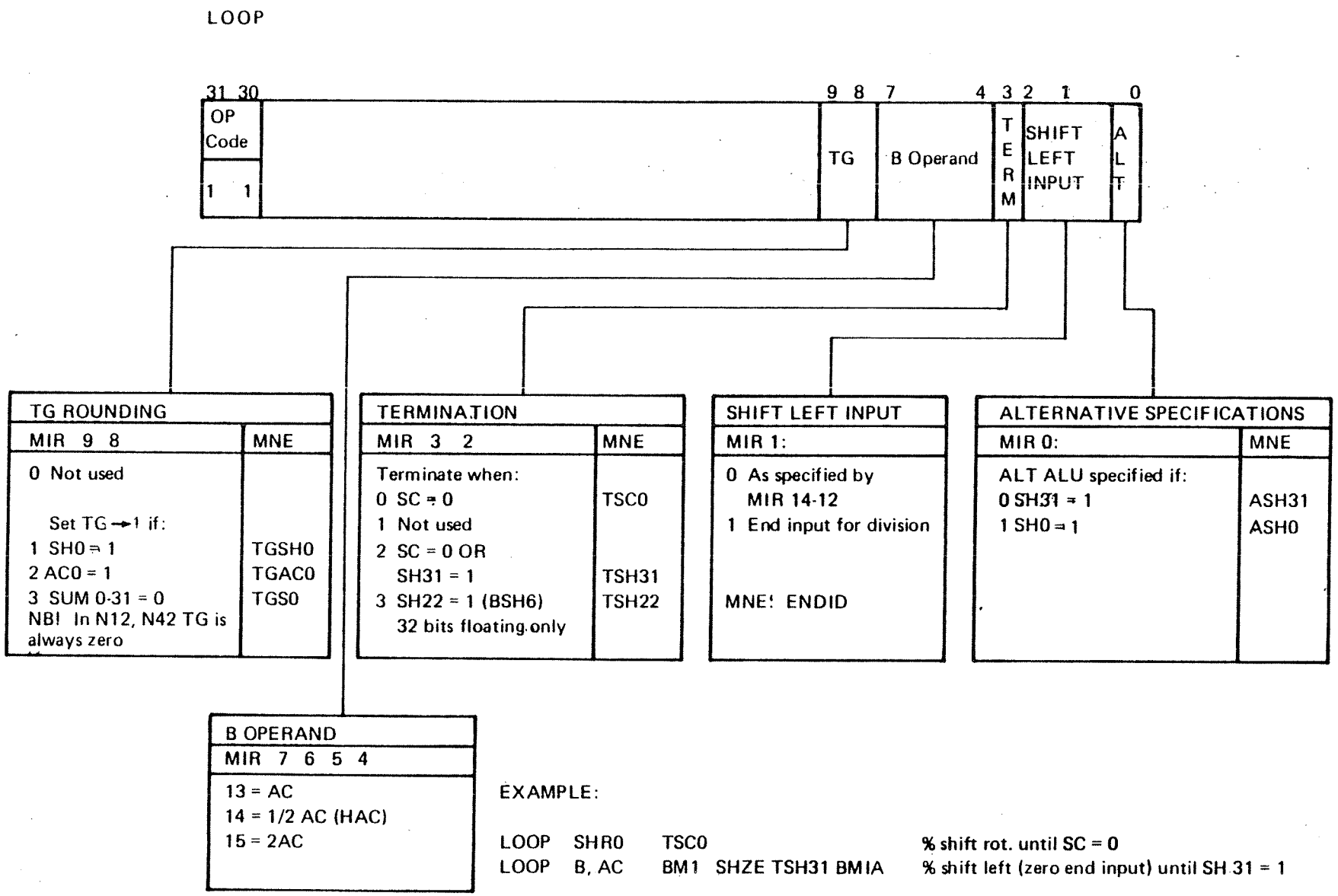


Figure 2.10: *Loop μ-instruction – Bit Assignment II*

3 THE MICROPROGRAM

3.1 *MICMAC – MICRO MAC MNEMONIC TABLE*

A,A	A register as A-operand
A, B	B register as A-operand
A,D	D register as A-operand
AC	Temporary Sum or Accumulator Register
ADD1	Forced Carry Input
ADDC	Add Carry Input
A,DH	Lower 8 bits of H with sign extension as A operand
ADIR	A-operand Direct throught Arithmetic = A
ADIRC	A-operand Direct Complemented = $\overline{A}$
A,H	H register as A-operand
A,IO	Special Case. Internal Register specified as B-operand to H register
A,L	L register as A-operand
ALD	Automatic Load Descriptor
AND	Logical AMD = $A \cdot B$
ANDC	AND compliment – NAND = $\overline{A \cdot B}$
ANDCA	$\overline{A} \cdot B$
ANDCB	$A \cdot \overline{B}$
A,P	CP register as A-operand
A,R	R register as A-operand
ARL	Arithmetic operation least significant unit
ARM	Arithmetic operation most significant unit

A,S	STATUS as A-operand
A,SCR	SCRATCH register as A-operand
ASH0	Alternative ALU specs. if SH0 = 1
ASH31	Alternative ALU specs. if SH31 = 1
A,SP	SAVED P as A-operand
A,SS	SAVED STATUS as A-operand
A,T	T register as A-operand
A,X	X register as A-operand
A,Z	Zero as A-operand
B,A	A register as B-operand
B,AC	AC register (Temporary SUM or ACCUMULATOR) as B-operand
B,2AC	2 · AC as B-operand
B,ALD	ALD – Automatic Load Descriptor as B-operand
B,B	B register as B-operand
B,B0-17	Bit number is one in Bit Mask as B-operand
B,CAR	Computed Address Register as B-operand
B,D	D register as B-operand
BDI	B-operand direct during arithmetic operation
BDIA	B-operation direct alternative ALU Specs.
BDIR	B-operand direct during logical operation
B,HAC	1/2 AC as B-operand
BIO	I/O bus as source when A,IO: I/O bus as dest. when D,IO
B,IR	IR as B-operand
BIR	IR as B-operand
BIR3	IR0-3 as B-operand

B,IR3	IR0-3 as B-operand
B,L	L register as B-operand
B,LMP	Lamp register as B operand
BM	B-operand = BIT MASK
BM1	B-operand minus 1
BM1A	B-operand minus 1, alternative ALU specs.
BMAA	B-operand – A-operand alternative ALU spec.
BMAM1	B-operand – A-operand –1 (B–A–1)
BMINA	B-operand – A-operand (B–A)
B,MIS	Miscellaneous register as B-operand
BMISC	Miscellaneous register as B-operand
B,MPC	Micro Program Counter as B-operand
B,OPR	Panel Switch Register as B operand
B,P	CP register as B-operand
B,PAC	Panel Control as B-operand
B,PAS	Panel Status as B-operand
B,PES	Memory Error Status register as B-operand
B,PID	Priority Interrupt Detect as B-operand
B,PIE	PIE as B-operand
B,PIM	Decoded PIL as B-operand
B,SC	Shift Counter as B-operand
B,SH	Shift register as B-operand
B,T	T register as B-operand
B,X	X-register as B-operand
B,Z	Zero as B-operand

CALL	Jump to subroutine
,CAR	Jump address from CAR (Computed Address Register)
CARL	Conditional Arithmetic least significant unit
CARM	Conditional Arithmetic most significant unit
CEATR	Cycle 1, Effective address to R — no request
CFC	Cycle 3, Fetch
CHLEV	Change program level
CJMP	Conditional jump
CLOGM	Conditional logical operation most significant unit
CO17	Conditional bit set and condition 7
COND	Condition bit set bit 15 in ROM
CPTR	Cycle 2, Current P register to R register
CR	Cycle 7, Read contents of effective address
CRR1	Cycle 6, Read contents of effective address + 1
CW	Cycle 5, Write into effective location
CWR1	Cycle 4, Write into effective location + 1
DH	Lower 8 bits of H sign extended
DNO	Device number
D,A	A register as destination register
D,B	B-register as destination
D,D	D register as destination
D,IO	Special case, A-operand transferred to internal registers
D,L	L register as destination
D,P	CP register as destination

D,S	Status register as destination
D,SS	Saved Status as destination
D,SCR	Scratch register as destination
D,SH	Shift register as destination
D,SP	Saved P as destination
DSPL	Destination register on level specified by bits 12-15 in ROM
D,T	T register as destination
D,X	X register as destination
ENDID	End input for division
EXOR	Exclusive OR
EXORC	Exclusive OR complement
GREM	Greater Magnitude
HAC	1/2 · AC
IARM	Interblock arithmetic operation most significant unit
IR	Instruction Register
IR3	Instruction Register bit 0-3. Register number in IR0-3 as destination register
ILOGM	Interblock logical operation most significant unit
JMP	Jump
LE0-17	Level number specified
LMP	Lamp register
LOGL	Logical operation least significant unit
LOGM	Logical operation most significant unit
LOOP	Loop instruction

MIS	Miscellaneous register
MPC	Micro Program Counter
NEG	Test for negative
NEGM	Test for negative magnitude
NZERO	Test for not zero
OR	Inclusive OR
ORBW	OR for bit operations with destination
ORBWO	OR for bit operation without destination
ORC	OR complement $\overline{(A + B)}$
ORCA	$\overline{A} + B$
ORCAR	OR with CAR
ORCB	$A + \overline{B}$
ORIN	OR for Interblock
ORIN2	OR for Interblock
ORROP	OR for Register Operations
ORSHT	OR for Shift
ORSKP	OR for SKP
ORSW2	OR for SWAP cycle 2
ORSW3	OR for SWAP cycle 3
PAC	Panel Control Register
PAS	Panel Status Register
PCR	Paging Control Register
PES	Memory Error Status Register
PIM	Decoded PIL
PLUS	A-operand + B-operand

PLUSA	A-operand + B-operand alternative specs.
POS	Test for positive
POSM	Test for positive magnitude
PRIV	Privileged instructions
S	STATUS – Register
SACO	Save Carry and Overflow
SC	Shift Counter
SCR	Scratch Register
SH	Shift Register
SH32	32 bits shift
SHAR	Arithmetic Shift
SHLI	Link end input
SHR	Shift right
SHRO	Rotational Shift
SHZE	Zero end input
SNEG	Sign negative
SPOS	Sign positive
SS	Saved Status Register
TGAC0	TG = 1 if AC0 = 1
TGS0	TG = 1 if SUM0-31 = 0
TGSH0	TG = 1 if SH0=1
TSC0	Terminate when Shift Counter = 0
TSH31	Terminate when Shift Counter = 0 or SH_{31} = 1
TSH22	Terminate when SH22 = 1 (32 bits floating)
Z	Test for zero

3.2 *NORD-10 INSTRUCTIONS AND THEIR CORRESPONDING ENTRY-POINTS*

AAA	: 352	IRW	: 256	RCLR	: 230
AAB	: 350	JAF	: 306	RDCR	: 231
AAT	: 354	JAN	: 302	RDIV	: 207
AAX	: 356	JAP	: 300	REXO	: 224 - 225
AAD	: 130	JAZ	: 304	RINC	: 232
AND	: 134	JMP	: 152	RMPY	: 205
BANC	: 374	JNC	: 312	RORA	: 226 - 227
BAND	: 375	JPC	: 310	RSUB	: 233
BLDA	: 373	JPL	: 156	SAA	: 342
BLDC	: 372	JXN	: 316	SAB	: 340
BORA	: 377	JXZ	: 314	SAD	: 274
BORC	: 376	LBYT	: 211	SAT	: 344
BSET	: 360 - 363	LDA	: 122	SAX	: 346
BSKP	: 364 - 367	LDD	: 112	SBYT	: 213
BSTA	: 371	LDF	: 116	SHA	: 270
BSTC	: 370	LDT	: 124	SHD	: 264
COPY	: 230	LDX	: 126	SHT	: 260
DNZ	: 250	LRB	: 252	SKP	: 200
EXIT	: 230	MCL	: 240	SRB	: 252
EXR	: 203	MIN	: 120	STA	: 102
FAD	: 140	MIX3	: 215	STD	: 110
FDV	: 146	MON	: 254	STF	: 114
FMU	: 144	MPY	: 150	STT	: 104
FSB	: 142	MST	: 240	STX	: 106
IDENT	: 217	NLZ	: 246	STZ	: 100
IOF	: 242	ORA	: 136	SUB	: 132
ION	: 242	POF	: 242	SWAP	: 220 - 221
IOT	: 170	PON	: 242	TRA	: 240
IOX	: 172	RADD	: 230 - 237	TRR	: 240
IRR	: 256	RAND	: 222 - 223	WAIT	: 244

3.2.1 *Special Entry Points*

Entry Point: (ADR)	
0	Entry point for STOP mode. It is automatically entered if the STOP signal is on during a fetch cycle (pushing the STOP button or executing a WAIT instruction).
1	Entry point for MASTER CLEAR. (Pushing the master clear button or power turn-on.)
400	Entry point for program interrupt, both internal and external.
1000	Entry point for operator's panel interrupt. It is entered with 3 milli-seconds interval when a general register or memory is displayed on the operator's panel.
1400	Entry point for coincident operator's panel and program interrupt.
1657	μprogram memory check.

3.3 *LABELS REFERENCED IN NORD-10 MICROPROGRAM*

ACT	: 1756	IEXA	: 1425	OUTCH	: 1735
ACT1	: 1766	IEXA1	: 1424	PANINC	: 1226
ADDF	: 472	IEXAM	: 1421	PANT1	: 1256
ASS8	: 1176	INCH	: 1716	PANT2	: 1257
BANCC	: 1026	INV	: 553	PANTT	: 1236
BANDC	: 1023	IOTC	: 402	POSDV	: 650
BANK	: 1373	IOXR	: 1632	PRLF	: 1430
BIN	: 1645	IRD	: 1365	PUTGC	: 657
BINL	: 1532	KONE2	: 154	QUM	: 1134
BLDAC	: 1020	KONE3	: 1037	RDEP	: 1476
BLDCC	: 1015	LDBC	: 64	RDIVC	: 677
BONE1	: 1022	LDDC	: 336	REAC	: 1077
BORAC	: 1031	LDFC	: 335	REDEP	: 1403
BORCC	: 1034	LEFT	: 74	REGDP	: 1412
BSBAC	: 1002	LEFTB	: 437	RESTA	: 1304
BSBSH	: 361	LOAD	: 1503	RETPA	: 1462
BSKC	: 420	MAS1	: 1602	RETU	: 1224
BSKCC	: 414	MAS2	: 1622	RETU1	: 1223
BSOC	: 416	MASS	: 1601	RETU5	: 1434
BSTAC	: 1011	MCLS	: 331	REX	: 1443
BSTCC	: 1005	MCLS	: 1774	REXAM	: 1321
BSZC	: 422	MCRY1	: 616	RLOOP	: 1201
CHCR	: 1343	MEXM	: 1456	RPANT	: 1717
CIIP	: 1041	MINC	: 41	RPDEP	: 1404
CLC	: 1441	MLOOP	: 1767	RSTRT	: 1576
CRY1	: 501	MM0	: 1661	SADC	: 175
DE0	: 1516	MM1	: 1662	SEEK	: 1535
DEPP	: 1467	MM2	: 1663	SETAD	: 1446
DNZC	: 2	MM3	: 1664	SIKI	: 1536
DOLET	: 1505	MM4	: 1674	SLRB	: 727
DOLL	: 1501	MM00	: 1660	SRB	: 16
EASS8	: 1220	MM41	: 1677	STBC	: 424
EQUAL	: 536	MONC	: 654	STDC	: 166
ERDP	: 1326	MOPC	: 1054	STFC	: 165
ERR	: 1712	MOPCM	: 1043	STFP	: 1307
ETSGN	: 1504	MOPCR	: 1060	STLP	: 1554
EXAM	: 1273	MPYC	: 753	STORB	: 444
EXECC	: 160	MPYDC	: 661	STPR	: 1310
EXRO	: 1300	NDEP	: 1360	STSP	: 1305
EXTN	: 1574	NECHP	: 1160	SUBF	: 525
FADC	: 454	NED1	: 505	SUBF2	: 524
FAFSC	: 456	NLZC	: 52	SUBF3	: 543
FDVC	: 621	NOINV	: 556	SWPC	: 47
FDVO	: 566	NORMA2	: 521	SWPCC	: 46
FETC5	: 341	NRDP	: 1406	TGTN	: 645
FETCH	: 101	NRDP1	: 1410	TRRS	: 327
FETCZ	: 343	NYFAF	: 1075	TTGN	: 613
FMUC	: 572	OUT1	: 1736	TTMMC	: 320
FSBC	: 451	OUT2	: 1743	WAITC	: 772
GETR	: 353	OUT3	: 1754	ZIR6	: 410
IEX	: 1436	OUT8	: 1146	ZTAD	: 567

3.4 *THE MICROPROGRAM LISTING*

```
0000                    JMP 1402                                %STOP
0001                    JMP 1401                                %MASTER CLEAR

0002 %ROUTINE TO CONVERT FROM FLOATING NUMBER IN T,A,D-REG,
0002 %TO INTEGER NUMBER IN A-REG
0002 %D-REG AND T-REG IS SET TO ZERO
                                              ADDRESS:
0002        DNZC,   ARM PLUS A,DH B,T D,A                       %T + ΔH → A
0003                ARM PLUS D,SS B,B4 A,A                      %A + 20 → SS
0004                LOGM AND B,B16 A,SS                         %TEST BIT 16 in SS-REG
0005                CJMP ZERO ZTAD              - 567           %JMP IF OVERFLOW
0006                LOGM AND D,D B,B16 A,A                      %TEST BIT 16 IN A-REG
0007                CJMP NZERO FETCZ            - 343           %JMP IF OVERFLOW
0010                LOGM ADIR A,A D,SC                          %SET SHIFTCOUNT
0011                LOOP SHR SHZE TSC0                          %SHIFT RIGHT TO SC = 0
0012                LOGM BDIR B,SH D,A                          %SH → A
0013                LOGM BDIR B,T                               %TEST SIGN
0014                CARM SNEG BMINA A,A B,Z D,A                 %INVERT IF SIGN NEG
0015                LOGM CFC ADIR A,Z D,T                       %0 → T, FETCH
0016
0016        SRB,    ARM BM1 B,X D,P CPTR                        %STORE BLOCK, X -1 → CP
0017                LOGM ADIR A,SP D,P                          %UNSAVE CP
0020                ILOGM ADIR ORIN2 A,SP                       %READ SP SPES, LEV
0021                LOGM CWR1 A,SP                              %STORE SP
0022                ILOGM ADIR ORIN2 A,X                        %READ X
0023                ARM CWR1 A,SP
0024                ILOGM ADIR ORIN2 A,T                        %READ T
0025                ARM CWR1 A,SP
0026                ILOGM ADIR ORIN2 A,A                        %READ A
0027                ARM CWR1 A,SP
0030                ILOGM ADIR A,D ORIN2                        %READ D
0031                ARM CWR1 A,SP
0032                ILOGM ADIR ORIN2 A,L                        %READ L
```

```
0033                ARM CWR1 A,SP
0034                ILOGM ADIR ORIN2 A,S                    %READ STATUS
0035                ARM CWR1 A,SP
0036                ILOGM ADIR ORIN2 A,B                    %READ B
0037                LOGM CWR1 A,SP
0040                ARM CFC                                 %FETCH'
0041
0041
0041        MINC,   LOGM ADIR A,R D,P                       %R → CP
0042                ARM BM1 B,P D,P                         %CP −1 → CP
0043                ARM CPTR PLUS ADD1 A,H B,Z D,SCR        %CP → R  H + 1 → SCR
0044                LOGL CWR1 BDIR A,SCR B,SH D,P           %SCR → (R + 1), SH(L) → CP
0045                CARM ZERO CFC PLUS ADD1 A,P B,Z D,P     %CP + 1 → CP
0046        SWPCC,  LOGM ADIRC ORSW3                        %ZERO:CP + 1 → CP
0047        SWPC,   LOGM ADIR ORSKP D,SP                    %REG → REG  (ONE'COMPL)
0050                ARM ORSW3 BD1
0051                LOGM CFC ORSW2 COND CO17 ADIR A,SP      %
0052
0052
0052

0052  %ROUTINE TO CONVERT FROM INTEGER TO FLCATING NUMBER
0052  %INTEGER IN A-REG, FLOATING IN T, A, D-REG

0052
0052        NLZC,   LOGM ADIR A,A D,SH                      %A → SH
0053                CJMP ZERO ZTAD              567         %RESULT ZERO IF A = 0
0054                ARL B,B16 PLUS A,DH D,T                 %40000 + H (0−7) → T, T → AC(L)
0055                CJMP POS *3
0056                ARL B, B17 PLUS A,T D,T                 %SET SIGN BIT, T → AC(L)
0057                ARM B,Z BMINA A,A D,SH                  %−A → SH IF A < 0
0060                ARM BDIR B,B4 D,SC                      %20 → SC
0061                LOOP B,AC BM1 SHZE TSH31 BM1A           %SHIFT LEFT TO SH31 = 1
0062                LOGL BDIR B,AC D,T                      %AC → T
0063                LOGM BDIR B,SH D,A CFC                  %SH → A, FETCH
0064
0064        LDBC,   LOGM ADIR A,X                           %X → AC
0065                ARM BM1 B,HAC D,SH                      %1/2*AC −1 → SH
```

```
0066                ARM CPTR PLUS A,T B,SH D,P          %T + SH → CP, CP → R
0067                LOGM CRR1 ADIR A,SP D,P             R + 1) → H, SP → CP
0070                LOGM AND A,X B,B0           74      %TEST LEFT RIGHT
0071                CJMP ZERO LEFT
0072                ARM BM1 B,B10 D,SH                  %377 → SH
0073                LOGM CFC AND A,H B,SH D,A           %BYTE → A-REG, FETCH
0074    LEFT,       ARM BM1 B,B3 D,SC                   %7 → SC
0075                ARM PLUS A,H B,Z                    %H → AC
0076                LOOP SHR BDI BDIA B,HAC SHZE
0077                LOGM CFC BDIR B,HAC D,A             %BYTE → A-REG, FETCH
0100
0100
0100                ARMCW A,Z                           %STZ, ZERO → EFFECTIVE ADDRESS
0101    FETCH,      ARM CFC                             %FETCH REQUEST
0102                ARM CW A,A                          %STA, A-REG → EA
0103                ARM CFC                             %FETCH
0104                ARM CW A,T                          %STT, T-REG, → EA
0105                ARM CFC                             %FETCH
0106                ARM CW A,X                          %STX, X-REG → EA
0107                ARM CFC                             %FETCH
0110                ARM CW A,A                          %STD, A-REG → EA
0111                JMP STDC                166
0112                ARM CR                              %LDD, (EA) → H
0113                JMP LDDC                330
0114                ARM CW A,T                          %STF, T-REG → EA
0115                JMP STFC                165         %JUMP TO CONTINUE STF
0116                ARM CR                              %LDF, READ REQUEST
0117                JMP LDFC                335
0120                LOGL CR ADIR A,P D,SH               %MIN, CP → SH          READ MEMORY
0121                JMP MINC                41
0122                ARM CR                              %LDA, (EA) → H
0123                LOGM CFC ADIR A,H D,A               %H → A, FETCH REQUEST
0124                ARM CR                              %LDT, (EA) → H
0125                LOGM CFC ADIR A,H D,T               %H → T, FETCH REQUEST
```

```
0126                ARM CR                                  %LDX, (EA) → H
0127                LOGM CFC ADIR A,H D,X                   %H → X, FETCH REQUEST
0130                ARM CR                                  %ADD, (EA) → H
0131                ARM CFC PLUS SACO A,H B,A D,A           %H + A → A, FETCH
0132                ARM CR                                  %SUB, (EA) →H
0133                ARM CFC BMINA SACO A,H B,A D,A          %A − H → A, FETCH
0134                ARM CR                                  %AND, (EA) → H
0135                LOGM CFC AND A,H B,A D,A                %A · H → A, FETCH
0136                ARM CR                                  %ORA, (EA) → H
0137                LOGM CFC OR A,H B,A D,A                 %A + H → A, FETCH
0140    FAD,        LOGM ANDCB B,B1 A,S D,S                 %STARTADD. FAD, RESET "TG"
0141                JMP FADC                        454
0142    FSB,        LOGM AND CB B,B1 A,S D,S                %START FSB, RESET "TG"
0143                JMP FSBC                        451
0144    FMU,        LOGM ANDCB B,B1 A,S D,S                 %START FMU, RESET "TG"
0145                JMP FMUC                        572
0146    FDV,        LOGM ANDCB B,B1 A,S D,S                 %START FDV, RESET "TG"
0147                JMP FDVC                        621
0150    MPY,        LOGM BDIR B,B4 D,SC                     %START MPY, SET SHIFTCOUNTER
0151                JMP MPYC                        753
0152                ARM CEATR                               %JMP, EA → R
0153                LOGM CFC ADIR A,R D,P                   %R → CP, (R) → H, IR
0154    KONE2,      LOGM AND CB B,BM ORBW                   %0 → SPECIFIED BIT
0155                LOGM CFC OR B,B2 A,S D,S                %1 → K, FETCH
0156                LOGM CEATR ADIR A,P D,L                 %JPL, EA → R, CP → L
0157                LOGM CFC ADIR A,R D,P                   %R → CP, (R) → H, IR
0160    EXECC,      LOGM ADIR ORSKP D,SP                    %REG → SP
0161                LOGM EXOR A,H B,AC D,SS                 %TEST EXEC. INSTR IN REG
0162                LOGM AND CB A,SS B,SH                   %
0163                CJMP ZERO FETCZ                 343     %JMP IF EXECUT INSTR IN REG
0164                ARM A,SP B,IR D,I0                      %INSTRUC. → IR
0165    STFC,       ARM CWR1 A,A                            %A → (R + 1)
0166    STDC,       ARM CWR1 A,D                            %D → (R + 1)
0167                ARM CFC
0170    IOT,        JMP PRIV IOTC                   402     %JMP IOT CONTINUE
```

```
0171        FETC1,  LOGL CFC BDIR B,SH D,D                  %SH(L) → D, FETCH
0172        IOX,    LOGM ADIR A,A BIO D,IO                  %A –REG → IO–BUS
0173                LOGM A,IO BIO                           %IO–BUS → H–REG
0174                LOGM CFC ADIR A,H D,A                   %H–REG → A–REG FETCH
0175        SADC,   LOOP ORSHT SH32 TSC0                    %SHIFT, TERM:SC = 0
0176                LOGM BDIR B,SH D,A                      %SH → A
0177                LOGL CFC BDIR B,SH D,D                  %SH(L) → D, FETCH
0200                ARM BMINA ORSKP                         %SKP
0201                CARM ORSKP PLUS ADD1 CFC A,Z B,P D,P    %TRUE CP + 1 → CP, FETCH
0202        FETC2,  LOGL CFC BDIR B,2AC D,D                 %2AC(L) → D
0203        EXEC,   ARM BM1 B,B6 D,SH                       %EXECUTE, GENERATE MASK
0204                JMP EXECC                       160
0205        RMPY,   LOGM ADIR ORSKP D,SP                    %FIRST OPERAND → SP , REGISTER MULTIPLY
0206                JMP MPYDC                       661
0207        RDIV,   LOGM BDIR B,B4 D,SC                     %20 → SC, REGISTER DIVIDE
0210                JMP RDIVC                       677
0211        LDB,    LOGM ADIR A,P D,SP                      %LOAD BYTE, CP → SP
0212                JMP LDBC                         64
0213        STB,    LOGM ADIR A,P D,SP                      %STORE BYTE, SAVE CP REG
0214                JMP STBC                        424
0215        MPX3,   ARL BM1 B,A D,SCR                       %INSTR:(A– 1) * 3 → SCRA → AC
0216                ARL CFC PLUS A,SCR B,2AC D,X            %2 · AC + SCR → SCR
0217        IDENT   JMP IOX                         172     %IDENT
0220                JMP SWPC                     47         %SWAP
0221                JMP SWPCC                    46         %SWAP CM1
0222                LOGM CFC AND ORROP                      %RAND, D · S → D
0223                LOGM CFC ANDCA ORROP                    %RAND CM1, D · S → D
0224                LOGM CFC EXOR ORROP                     %REXO, D⊕S → D
0225                LOGM CFC EXORC ORROP                    %REXO CM1, D⊕S → D
0226                LOGM CFC OR ORROP                       %RORA, D + S → D
0227                LOGM CFC ORCA ORROP                     %OR CM1, D + S → D
0230                ARM CFC PLUS ORROP SACO                 %RADD, COPY, EXIT, RCLR, D + S → D
0231                ARM CFC BMAM1 ORROP SACO                %RADD CM1, D – S – 1 → D, RDCR
0232                ARM CFC PLUS ADD1 ORROP SACO            %RADD AD1, D + S + 1 → D, RINC
0233                ARM CFC BMINA ORROP SACO                %RADD AD1 CM1, D – S → D RSUB
0234                ARM CFC PLUS ADDC ORROP SACE            %RADD ADC, D + S + C → D
```

```
0235                ARM CFC BMAM1 ADDC ORROP SACO          %RADD ADC CM1, D – S - 1 + C → D
0236                ARM CFC PLUS ADD1 ORROP SACO           %RADD AD1 ADC, D + S + 1 → D
0237                ARM CFC BMINA ORROP SACO               %RADD AD1 ADC CM1, D – S → D
0240      MISS,     LOGM AND A,H B,B6                      %MST, MCL, TRA, TRR, TEST BIT 6
0241                JMP PRIV TTMMC          320
0242      IIPP,     JMP PRIV CIIPP          1041           %IOF, ION, PON, POF
0243                0
0244      WAIT,     ARM A, I0  B, PID                      %READ PID
0245                JMP PRIV WAITC          772
0246      NLZ,      LOGM BDIR B, Z D, D                    %START NLZ, 0 → D
0247                JMP NLZC                52
0250      DNZ,      LOGM ADIR D,SH A, A                    %START  DNZ, A → SH
0251                JMP DNZC                2
0252                LOGM ADIR A, P D,SP                    %LOAD/STORE REGISTER BLOCK, CP → SP
0253                JMP PRIV SLRB           727
0254      MON,      LOGM BDIR B, B4 D, SCR                 %MONITOR CALL, 4 → SCR
0255                JMP MONC                654
0256      PUTG,     LOGM AND A,H B, B7                     %TEST IRR, IRW
0257                JMP PRIV PUTGC          657
0260                LOGM ADIR A, H D, SC                   %SHT, H(0 – 5) → SC
0261                LOGM ADIR A,T  D, SH                   %T → SH
0262                LOOP ORSHT TSC0                        %SHIFT TO SC = 0
0263                LOGM CFC BDIR B, SH D, T               %SH → T, (R) → H, IR
0264                LOGM ADIR A, H D, SC                   %SHD, H (0 - 5) → SC
0265                LOGM ADIR A, D D, SH                   %D → SH
0266                LOOP ORSHT TSC0                        %SHIFT TO SC = 0
0267                LOGM CFC BDIR B, SH D, D               %SH → D
0270                LOGM ADIR A, H D, SC                   %SHA, H (0 - 5) → SC
0271                LOGM ADIR A, A D, SH                   %A → SH
0272                LOOP ORSHT TSC0                        %SHIFT, TERMINATION:SC = 0
0273                LOGM CFC BDIR B, SH D, A               %SH → A , (R) → H, IR
0274                LOGM ADIR A, H D, SC                   %SAD, H (0-5) → SC
0275                LOGM ADIR A, A D, SH                   %A → SH
```

```
0276                LOGL ADIR A,D D,SH          ;D → SH (LEAST SIGNIFICANT)
0277                JMP SADC                175
0300                LOGM CEATR ADIR A,A         ;JAP, EA → R, TEST A
0301                CLOGM CFC POS ADIR A,R D,P  ;(A > 0):R → CP, (R) → H, IR
0302                LOGM CEATR ADIR A,A         ;JAN, EA → R TEST A
0303                CLOGM CFC NEG ADIR A,R D,P  ;(A < 0):R → CP, (R) → H, IR
0304                LOGM CEATR ADIR A,A         ;JAZ, EA → R, TEST A
0305                CLOGM CFC ZERO ADIR A,R D,P ;A = 0:R → CP, (R) → H, IR
0306                LOGM CEATR ADIR A,A         ;JAF, EA → R, TEST A
0307                CLOGM CFC NZERO ADIR A,R D,P;(A ≠ 0):R → CP, (R) → H, IR
0310                ARM CEATR BDI ADD1 B,X D,X  ;JPC, X + 1 → X, EA → R
0311                CLOGM CFC SPOS ADIR A,R D,P ;(X > 0):R → CP, (R) → H, IR
0312                ARM CEATR BDI ADD1 B,X D,X  ;JNC, X + 1 → X, EA → R
0313                CLOGM CFC SNEG ADIR A,R D,P ;(X < 0):R → CP, (R) → H, IR
0314                LOGM CEATR ADIR A,X         ;JXZ, EA → R, TEST X
0315                CLOGM CFC ZERO ADIR A,R D,P ;(X < 0):R → CP
0316                LOGM CEATR ADIR A,X         ;JXN, EA → R, TEST X
0317                CLOGM CFC NEG ADIR A,R D,P  ;(X ≤ 0):R → CP, (R) → H, IR
0320  TTMMC,       CJMP ZERO ZIR6              410 ;JMP IF BIT 6 = 0
0321                LOGM AND A,H B,B7           ;TEST BIT 7
0322                CJMP ZERO TRRS              327 ;JMP IF BIT 7 = 0
0323  MSTS,         ARM A,IO BIR3               ;MST, REG → H
0324                LOGM OR A,H B,A D,SP        ;H + A → SP
0325                LOGM ADIR A,SP BIR3 D,I0    ;SP → REG
0326                JMP FETCH                   101
0327  TRRS,         LOGM A,A BIR3 D,IO ADIR     ;TRR, A → REG
0330                JMP FETCH                   101
0331  MCLS,         ARM A,IO BIR3               ;MCL, REG → H
0332                LOGM ANDCB A,H B,A D,SP     ;N(A), H → SP
0333                LOGM A,SP BIR3 D,IO ADIR    ;SP → REG
0334                JMP FETCH                   101
0335  LDFC,         LOGM CRR1 ADIR A,H D,T      ;H → T, (R + 1) → H
0336  LDDC,         LOGM CRR1ADIR A,H D,A       ;H → A, (R + 1) → H
0337                LOGM CFC ADIR A,H D,D       ;H → D, (R) → H, IR
```

```
0340                LOGM CFC ADIR A,DH D,B                  %H (0 - 7) → B, FETCH
0341        FETC5,  LOGL BDIR B,AC D,D CFC                  %AC (L) → D
0342                LOGM CFC ADIR A,DH D,A                  %SAA, H (0 - 7) → A, (R)  → H, IR
0343        FETCZ,  LOGM CFC OR A,S B,B3 D,S                %1 → Z, FETCH
0344                LOGM CFC ADIR A,DH D,T                  %SAT, H(0-7) → T, (R) → H, IR
0345                JMP *
0346                LOGM CFC ADIR A,DH D,X                  %SAX, H(0-7) → X, (R) → H, IR
0347        FETC3,  ARM CFC BM1 B,T D,T                     %DECR, EXPONENT, FETCH
0350                ARM CFC PLUS SACO A,DH B,B D,B          %AAB, H(0-7) + B → B (R) → H, IR
0351                0
0352                ARM CFC PLUS SACO A,DH B,A D,A          %AAA, H (0-7) + A → A, (R) → H, IR
0353        GETR,   ILOGM CFC ADIR ORBWO D,A                %REG → A, FETCH, IRR
0354                ARM CFC PLUS SACO A,DH B,T D,T          %AAT, H(0-7) + T → T, (R) → H, IR
0355        OVERF,  LOGM OR B,B5 A,S D,S CFC                %SET STATIC OVERF, FETCH
0356                ARM CFC PLUS SACO A,DH B,X D,X          %AAX, H(0-7) + X → X, (R) → H, IR
0357        FETC4,  LOGL CFC BDIR B,2AC D,D                 %2 · AC(L) → D, FETCH
0360                LOGM ANDCB B,BM ORBW CFC                %BSET ZERO
0361        BSBSH,  LOGM OR B,BM ORBW CFC                   %BSET ONE
0362                LOGM EXOR B,BM ORBW CFC                 %BSET BCM
0363                JMP BSBAC                    1002       %JMP TO BSET BAC
0364                JMP BSZC                      422       %JMP TO BSKP ZERO
0365                JMP BSOC                      416       %JMP TO BSKP ONE
0366                JMP BSKCC                    1005       %JMP BSKP K0
0367                JMP BSKC                      420       %JMP BSKP K
0370                JMP BSTCC                    1005       %JMP BSTC
0371                JMP BSTAC                    1011       %JMP BSTA
0372                JMP BLDCC                    1015       %JMP BLDC
0373                JMP BLDAC                    1020       %JMP BLDA
0374                JMP BANCC                    1026       %JMP BANC
0375                JMP BANDC                    1023       %JMP BAND
0376                JMP BORCC                    1034
0377                JMP BORAC                    1031       %JMP BORA
0400                LOGM CHLEV ADIR A,P D,SP                %INTERRUPT, SAVE CP–REG
0401                LOGM CFC ADIR A,SP D,P                  %SAVE CP AND FETCH ON NEW LEVEL
0402        IOTC,   LOGM ADIR A,A BIO D,IO                  %A–REG → IO–BUS
```

```
0403            LOGM A,IO BIO                          %IO-BUS → H-REG
0404            LOGM ADIR A,H D,A                      %H → A-REG
0405            LOGM A,IO B,PES                        %PES → H-REG
0406            LOGM ADIR A,H                          %TEST BIT 15 PES   (N-1 SKIP)
0407            CARM CFC SNEG BEI ADD1 B,P D,P         %IF MPC 15 = 1 THEN P + 1 ⟶ P
0410    ZIR6,   LOGM AND A, H B,B7                     %TEST BIT 7
0411            CJMP NZERO MCLS              1774      %JMP IF BIT 7 = 1
0412            ARM A, IO BIR 3                        %TRA, REG → H
0413            LOGM CFC ADIR A,H D,A                  %H → A, (R) → H, IR
0414    BSKCC,  LOGM AND A,S B,B2                      %TEST K
0415            CJMP NZERO BSZC               422      %JMP IF K = 1
0416    BSOC,   LOGM AND ORBWO B,BM                    %TEST BIT
0417            CARM CFC NZERO BDI ADD1 B,P D,P        %IF SPECIFIED BIT = 1:CP + 1 → CP, FETCH
0420    BSKC,   LOGM AND A,S B,B2                      %TEST K
0421            CJMP NZERO BSOC               416      %JMP IF K = 1
0422    BSZC,   LOGM AND ORBWO B,BM                    %TEST BIT
0423            CARM CFC ZERO BDI ADD1 B,P D,P         %IF SPECIFIED BIT = 0:CP + 1 → CP
0424
0424
0424    STBC,   ARM BDI B,X                            %X → AC
0425            ARM BM1 B,HAC D,SH                     %1/2*AC - 1 → SH
0426            ARM CPTR PLUS A,T B,SH D,P             %T + SH → CP, CP → R
0427            ARM PLUS A,SP B,Z D,P CRR1             %(R + 1) → H
0430            ARM CPTR PLUS A,T B,SH D,P
0431            ARM BM1 B,B10 D,SCR                    %377 → SH
0432            LOGM AND B,B0 A,X                      %TEST LEFT RIGHT
0433            CJMP ZERO LEFTB               437
0434            LOGM AND A,SCR B,A D,SS                %MASK OUT RIGHT PART
0435            LOGM ADIRC A,SCR D,SH
0436            JMP STORB                     444
0437    LEFTB,  ARM BM1 B,B3 D,SC                      %7 → SC
0440            LOGL BDIR B,A
0441            LOOP BDI BDIA B,2AC                    %SHIFT TO LEFT BYTE
0442            LOGL BDIR B,2AC D,SS
0443            LOGM ADIR A,SCR D,SH
```

```
0444        STORB,  LOGM AND A,H B,SH D,SH
0445                LOGM ADIR A,SP D, P                    %SP → CP
0446                LOGM OR A,SS B,SH D,SS                 %NEW BYTE + OLD BYTE → SS-REG
0447                LOGM CWR1 A,SS                         %STORE BYTE
0450                LOGM CFC                               %FETCH
0451
0451        FSBC,   ARM CR NM1 B,B5 D,SS                   %37 → SS
0452                LOGM EXOR B,B17  A,H D,SCR             %EXPONENT → SCR, INVERT BIT 17
0453                JMP FAFSC                  456
0454        FADC,   ARM CR BM1 B,B5 D,SS                   %37 → SS
0455                LOGM ADIR A,H D,SCR                    %H → SCR
0456        FAFSC,  LOGM ANDCB B,B17 A,T D,SH              %RESET SIGN BIT
0457                LOGM ANDCB B,B17 A,SCR D,SP            %RESET SIGN BIT
0460                ARM CRR1 BMINA B,SH A,SP D,SC          %SP - SH → SC
0461                CJMP SNEG NED1             505         %JMP IF NUMB. IN MEMORY GREATEST
0462                CJMP ZERO EQUAL            536         %JMP EXPONENTS EQUAL
0463                LOGM ANDCA B,AC A,SS                   %TEST AC <40
0464                CJMP NZERO FETCH           101         %NO MANTISS OVERLAP IF NOT ZERO
0465                LOGM CRR1 ADIR A,H D,SH                %H → SH
0466                LOGL ADIR A,H D,SH                     %H → SH(L)
0467                LOOP SHR SH32 TGSH0 SHZE               %SHIFT RIGHT TERM:SC = 0
0470                LOGM EXOR A,SCR B,T                    %TEST EQUAL SIGNS
0471                CJMP SNEG SUBF             525         %JMP IF DIFFERENT SIGN
0472        ADDF,   LOGL BDIR B,SH D,SCR                   %SH(L) → SCR
0473                ARM PLUS A,SCR B,D D,D SACO            %ADD. OF LEAST MANT.
0474                ARM PLUS ADDC A,A B,SH D,A SACO        %ADD. OF MOST. MANT.
0475                CJMP GREM CRY1             501         %JMP IF CRY
0476        NOCRY,  LOGM AND B B1 A,S                      %TEST TG        * CFC is added in N12/42
0477                CJMP ZERO FETCH            101         %ZERO:FINISHED
0500                LOGM CFC OR A,D B,B0 D,D
0501
0501        CRY1,   LOGL A,D ADIR                          %D → AC(L)
0502                LOGL BDIR B,HAC D,D                    %1/2 AC(L) → D
0503                LOGM BDIR B,HAC D,A                    %1/2 AC → A  CRY → AA
```

```
0504            ARM PLUS B,B0 A,T D,T CFC                   %T + 1 → T, FETCH
0505
0505    NED1,   LOGM ORC A,SS B,AC                          %TEST:AC > –40
0506            CJMP NZERO NORMA 2                  521     %NO MANTISSE OVERLAP IF NOT ZERO
0507            LOGL ADIR A,D D,SH
0510            LOGM ADIR A,A D,SH
0511            LOGM CRR1 ADIR A,H D,SP                     %READ MOST SIGNIF:MANT
0512            LOGM ADIR A,H D,D
0513            LOGM ADIR A,SP D,A
0514            LOOP SHR SH32 TGSH0 SHZE TSC0               %SHIFT RIGHT, TERMIN; SC = 0
0515            LOGM EXOR A, SCR B,T                        %TEST FOR EQUAL SIGN
0516            CJMP SNEG SUBF2                     524     %JMP TO SUBTRACTION IF UNEQUAL
0517            LOGM ADIR A,SCR D,T
0520            JMP ADDF                            472
0521
0521    NORMA2, LOGM CRR1ADIR A,H D,A                       %H → A READ LEAST SIGNIF MANT.
0522            LOGM ADIR A,H D,D
0523            LOGM CFC ADIR A,SCR D,T                     %EXPONENT → T–REG, FETCH
0524
0524
0524    SUBF2,  LOGM ADIR A,SCR D,T
0525    SUBF,   LOGL BDIR B,SH D,SCR
0526            LOGM BDIR B,SH D,SP                         %LEAST NUMB → REG. ON A–BUS
0527            ARM BMINA B,D A,SCR D,D SACO                %D – SCR → SCR
0530            ARM BMAM1 ADDC B,A A,SP D,SH                %A – SP → SH
0531            LOGM AND B,B1 A,S                           %TEST TG
0532            CJMP ZERO *2
0533            LOGL OR B,B0 A,D D,D                        %SET D0 → 1
0534            LOGL ADIR A,D D,SH
0535            JMP NOINV +1                        557
0536
0536
0536    EQUAL,  LOGM EXOR A,SCR B,T                         %TEST EQUAL SIGNS
0537            CJMP SNEG SUBF3                             %UNEQUAL JMP TO SUB
0540            LOGM ADIR A,H D,SH CRR1                     %H → SH, READ LEASTS. MANT
```

```
0541                LOGM ADIR A,H D,SCR
0542                JMP ADDF +1                                  473
0543
0543    SUBF3,      LOGM ADIRC A,H D,SP CRR1
0544                LOGM ADIRC A,H D,SCR
0545                ARM PLUS A,SCR B,D D,SCR SACO
0546                ARM PLUS ADDC A,SP B,A D,SH SACO
0547                CJMP NEGM INV                                553
0550                ARM PLUS ADD1 A,SCR D,SCR SACO
0551                ARM PLUS ADDC B,SH D,SH
0552                JMP NOINV                                    556
0553    INV,        LOGL EXOR B,B17 A,T D,T
0554                LOGL ADIRC A,SCR D,SCR
0555                LOGM BDIRC B,SH D,SH
0556    NOINV,      LOGL ADIR A,SCR D,SH
0557                LOGM BDIR B,B5 D,SC
0560                LOGL BDIR B,T
0561                LOOP BM1 BM1A TSH31 SH32 B,AC
0562                LOGL BDIR B,AC D,T
0563                LOGM BDIR B,SH D,A
0564                CJMP SPOS ZTAD                               567
0565                LOGL CFC BDIR B,SH D,D
0566
0566    FDVO,       LOGM OR B,B3 A,S D,S
0567    ZTAD,       LOGM ADIR A,Z D,T
0570                LOGM ADIR A,Z D,A
0571                LOGM CFC ADIR A,Z D,D
0572
0572    FMUC,       LOGM CR BDIR B,B5 D,SC
0573                ARM PLUS A,H B,T D,SP
0574                LOGM ANDCB B,B17 A,SP D,SP
0575                LOGM CRR1 EXOR B,T A,H
0576                CJMP SPOS * 2
0577                LOGM OR B,B17 A,SP D,SP
```

```
%LEAST SIGN. MANT → SCR

%H → SCR READ LEASTS. MANT
%H → SP, ONE'S COMPL
%D + SCR → SCR
%A + SP → SP
%INVERT IF CARRY = 0
%SCR + 1 → SCR
%SH + CARRY → SH

%INVERT SIGN
%ONE'S COMPL. OF SCR → SCR
%ONE'S COMP. OF SH → SH
%SCR → SH
%40 → SC
%T → AC(L)
%SHIFT LEFT TO SH31 = CRY
%EXPONENT → T-REG
%SH → A
%RESULT ZERO
%SH(L) → D, FETCH

%1 → Z
%0 → T
%0 → A
%0 → D, FETCH

%40 → SC, READ EXPO.
%H + T → SP
%RESET BIT 17 IN SP
%TEST SIGN

%SET BIT 17 IF NEG. RESULT
```

```
0600                LOGM CRR1 ADIR A,H D,SH          %H → SH, READ LEAST S. M.
0601                LOGM EXOR B,B16 A,SP D,T         %INVERT BIAS BIT
0602                LOGL ADIR A,H D,SH
0603                LOGL BDIR A,D B,Z                %D → A-LATCH, 0 → AC(L)
0604                LOGM BDIR A,A B,Z
0605                LOOP BDI PLUSA B,HAC SH32 ASH0
                    SHR TGAC0 SACO                   %MULTIPLY LOOP
0606                CJMP GREM MCRY1          616     %JMP IF CARRY
0607                CJMP ZERO ZTAD           567     %JMP IF ZERO
0610                LOGL BDIR B,AC D,D               %AC(L) → D
0611                LOGM BDIR B,AC D,A               %AC → A
0612                ARM BM1 B,T D,T                  %T - 1 → T
0613    TTGN,       LOGM*AND B,B1 A,S                %TEST TG        * CFC is added in N12/42
0614                CJMP ZERO FETCH          101     %JMP IF TG = 0
0615                LOGM CFC OR B,B0 A,D D,D         %SET BIT 0 → 1
0616    MCRY1,      LOGL BDIR B,HAC D,D              %1/2 AC(L) → D
0617                LOGM BDIR B,HAC D,A              %1/2 AC → A
0620                JMP TTGN                 613
0621    FDVC,       LOGM CR BDIR B,B5 D,SC           %40 → SC, READ EXPO.
0622                LOGL BDIR B,B0 D,SH              %SET SH(L) → 1, DIV ALGOR.
0623                ARM BMINA B,T A,H D,SP           %T - H → SP
0624                LOGM ANDCB B,B17 A,SP D,SP       %RESET BIT 17 IN SP
0625                LOGM CRR1 EXOR A,H B,T           %TEST SIGN, READ MOST S.M.
0626                CJMP SPOS *2
0627                LOGM OR B,B17 A,SP D,SP          %SET BIT 17 → 1, NEGATIVE
0630                LOGM CRR1 ADIR A,H D,SCR         %H → SCR, READ LEAST S.M.
0631                CJMP SPOS FDVO           566     %OVERF. IF ZERO
0632                LOGM EXOR B,B16 A,SP D,T         %INVERT BIAS BIT
0633                LOGL BDIR B,D                    %D → AC(L)
0634                ARM BDI B,A                      %A → AC
0635                CJMP SPOS ZTAD           567     %FINISHED IF ZERO OR NOT NORM.
0636                LOGL BDIR B,HAC A,H              %1/2 AC(L) → AC(L), H → A-LATCH
0637                LOGM BDIR B,HAC A SCR
0640    FDVLO,      LOOP PLUS BMAA B,2AC SACO SH32
                    ASH0 ENDID TGS0                  %FDV LOOP
```

```
0641                LOGM BDIR B,SH D,A                  %SH → A
0642                CJMP SPOS POSDV          650        %JMP TO SHIFT
0643                ARM BDI ADD1 B,T  D,T               %T + 1 → T
0644                LOGL BDIR B,SH D,D
0645        TGTN2,  LOGM*AND B,B1 A,S                   %TEST TG        * CFC is added in N12/42
0646                CJMP NZERO FETCH         101
0647                LOGM CFC OR B,B0 A, D D,D           %SET BIT 0 → 1
0650        POSDV,  LOGL BDIR B,SH D,D                  %SH(L) → D
0651                LOGM BDIR B,2AC D,A                 %2 · AC → A
0652                LOGL BDIR B,2AC D,D                 %2·AC(L) → D
0653                JMP TGTN2                645
0654
0654
0654
0654
0654
0654    % MONITOR CALL CONTINUES
0654        MONC,   ARM A,SCR BMISC  D,IO               %SCR →MISC, BIT 4 MISC   → 1
0655                ARM A, Z BMISC D,IO                 %0 → MISC
0656                ILOGM ADIR CFC A, DH D,T  LE16 DSPL %Δ H → T, LEV14, FETCH
0657
0657        PUTGC,  CJMP NZERO GETR          353        %JMP IF IRR
0660                ILOGM CFC BDIR ORBW B,A DSPL        %A → REG
0661
0661
0661

0661    %DOUBLE PRESISON MULTIPLY
0661    % OPERANDS IN TWO GENERAL·REGISTERS
0661    %RETURN WITH ANSWER IN A (MOST SIGN.) AND ·D-REG

0661
0661        MPYDC,  LOGM BDIR ORSKP D,A                 %SECOND OPERAND → A
0662                LOGL EXOR A,SP B,A D,SS             %SIGN → SS
0663                CARM NEG BMINA A,A B, Z D,A         %INVERT A—REG
0664                LOGM ADIR A,SP D,SH
0665                CARM NEG BMINA B, Z A,SP D,SH       %INVERT IF NEG
0666                LOGM BDIR B,B4 D,SC                 %16 → SC
```

```
0667                LOGL BDIR A,A B,Z
0670                LOOP BDI PLUSA B,2AC ASH31
0671                LOGM BDIR B,AC D,A                  %MOST SIGN. PART → A-REG
0672                LOGM ADIR A,SS                      %TEST SIGN
0673                CJMP SPOS FETC5             341
0674                LOGL BDIR B,AC D,D                  %AC(L) → D
0675                ARM BMINA SACO B,Z A,D D,D          %-D →D
0676                ARM CFC BMAM1 ADDC B,Z A,A D,A      %-A → A
0677
0677
0677
0677    %INTEGER DIVISON
0677    %DIVIDEND IN A-D-REGISTER, DIVISOR IN SPESYFIED REGISTER
0677    %RESULT IN A-REGISTER, REST IN D-REGISTER
0677
0677        RDIVC,  LOGM ORSKP ADIR D,SCR               %DIVISOR → SCR
0700                CARL NEG BMINA A,SCR B, Z  D,SCR    %INVERT SCR
0701                LOGM EXOR A,A B,AC D,SS             %SIGN → SS17
0702                LOGM BDIR B,A D,SP                  %A → SP
0703                CJMP POS *3
0704                ARM B,Z BMINA A,D D,D SACO          %INVERT D-REG
0705                ARM B,Z BMAMI A,A D,A ADDC          %INVERT A-REG
0706                ARM BMINA B,A A,SCR D,SH            %A - SCR → SH
0707                CJMP  POSM FETCZ            343
0710                LOGL B,Z BDIR D,SH                  %0 → SH(L)
0711                LOGL B,D  BDIR A,Z                  %D → AC, 0 → A (LATCH)
0712                LOGM BDIR B,SH A,SCR                %SH → AC, SCR → A (LATCH)
0713                LOOP B,2AC PLUS BMAA SH32 ASH0
                    ENDID TSC0
0714                LOGM B,AC BDIR D,D                  %AC → D, REST
0715                LOGL BDIR B,SH D,A                  %RESULT → A
0716                LOGM B,B0 AND A,A
0717                CARM PLUS B,D A,SCR D,D ZERO        %SCR + D → D IF A (0) = 0
0720                LOGM A,SP ADIR                      %TEST SIGN OF DIVIDEND
```

```
0721                CARM BMINA A,D B,Z D,D NEG          %INVERT D IF NEG
0722                LOGM A,SS ADIR                      %SIGN TEST
0723                CARM NEG BMINA A,A B,Z D,A          %COMPL. IF NEG
0724                LOGM EXOR B,A A,SS                  %OVERFLOW TEST
0725                CJMP SNEG FETCZ          343
0726                ARM CFC
0727
0727
0727
0727        SLRB,   LOGM AND B, B7  A,H                 %TEST LOAD, STORE BLOCK
0730                CJMP ZERO SRB            16
0731
0731
0731
0731        LRB,    ARM CPTR BM1 B,X D,P                %  X - 1 → CP, CP → R
0732                LOGM CRR1 ADIR A,SP D,P             %SP → CP
0733                LOGM ADIR A,H D,SH                  %CP → SH
0734                ILOGM BDIR B,SH D,SP ORIN DSPL CRR1 %LOAD SP
0735                LOGM ADIR A,H D,SH                  %X → SH
0736                ILOGM BDIR B,SH D,X ORIN DSPL CRR1  %LOAD X
0737                LOGM ADIR A,H D,SH                  %T → SH
0740                ILOGM BDIR B,SH D,T ORIN DSPL CRR1  %LOAD T
0741                LOGM ADIR A,H D,SH                  %A → SH
0742                ILOGM BDIR  B,SH D,A ORIN DSPL CRR1 %LOAD A
0743                LOGM ADIR A,H D,SH                  %D → SH
0744                ILOGM BDIR B,SH D,D ORIN DSPL CRR1  %LOAD D
0745                LOGM ADIR A,H D,SH                  %L → SH
0746                ILOGM BDIR B,SH D,L ORIN DSPL CRR1  %LOAD L
0747                LOGM ADIR A,H D,SH                  %S → SH
0750                ILOGM BDIR B,SH D,S ORIN DSPL CRR1  %LOAD S
0751                LOGM ADIR A,H D,SH                  %B → SH
0752                ILOGM BDIR B,SH D,B ORIN DSPL CFC   %LOAD B, FETCH
0753
0753    %MULTIPLY CONTINUED
0753    %ONE OPERAND IN A-REG
0753    %SECOND OPERAND,  CONTENT OF EFFECTIVE ADDRESS
0753
```

```
0753        MPYC,   LOGM ADIR A,A D,SCR CR          %A → SCR, READ SEC. OPE.
0754                CARM NEG BMINA A,A B, Z D,SCR   %  VERT IF NEG
0755                LOGM ANDCB B,B4 A,S D,S         %RESET DYNAMIC OVERFL.
0756                LOGM ADIR A,H D,SH              %SECOND OPER. → SH
0757                CARM NEG BMINA B, Z A,H D,SH    %INVERT IF NEG
0760                LOGM BDIR A, Z B, Z             %ZERO → AC AND A–LATCH
0761                LOGL BDIR A,SCR B, Z            %ZERO → AC(L) AND A → A–LATCH
0762                LOOP BDI PLUSA B, 2AC ASH31     %MULTIPLY LOOP
0763                LOGM BDIR B, 2AC                %TEST OVERF
0764                CJMP ZERO *3                    %JMP IF NOT OVERFL.
0765                LOGM OR B,B4 A,S D,S            %SET STATIC OVERFL
0766                LOGM OR B,B5 A,S D,S            %SET DYNAMIC OVERFL
0767                LOGM EXOR A,H B,A               %TEST SIGN
0770                LOGL BDIR B,AC D,A              %RESULT → A
0771                CARM NEG CFC BMINA A,A B, Z D,A %INVERT IF NEG.
0772
0772
0772
0772        WAITC,  LOGM ADIR A,H D,SH              %PID → SH
0773                ARM A,IO B,PIM                  %DECODED PIL → H
0774                LOGM AND CA A,H B,SH D,SCR      %CLEAR BIT IN PID
0775                ARM A,SCR B, PID D,IO           %SET PID
0776                JMP FETCH           101
0777                0
1000
1000
1000
1000        PANIN,  LOGM ADIR A, P D,SP             %ENTRY PANEL INTERRUPT
1001                JMP PANINC          1226
1002
1002
1002
1002
1002
1002
1002
1002        BSBAC,  LOGM AND A,S B,B2               %TEST K
```

```
1003                CJMP NZERO BSBSH
1004                LOGM CFC ANDCB B,BM ORBW
1005        BSTCC,  LOGM AND A,S B,B2
1006                CJMP NZERO  KONE2
1007                LOGM OR B,BM ORBW
1010                LOGM CFC OR B,B2 A,S D,S
1011        BSTAC,  LOGM AND A,S B,B2
1012                CJMP NZERO KONE3
1013                LOGM ANDCB B,BM ORBW
1014                LOGM CFC ANDCB B,B2 A,S D,S
1015        BLDCC,  LOGM AND ORBWO B,BM
1016                CJMP NZERO BONE1
1017                LOGM CFC OR B,B2 A,S D,S
1020        BLDAC,  LOGM AND ORBWO B,BM
1021                CJMP NZERO BLDCC +2
1022        BONE1,  LOGM CFC ANDCB B,B2 A,S D,S
1023        BANDC,  LOGM AND ORBWO B,BM
1024                CJMP NZERO FETCH
1025                LOGM CFC ANDCB B,B2 A,S D,S
1026        BANCC,  LOGM AND ORBWO B,BM
1027                CJMP NZERO BANDC +2
1030                LOGM CFC
1031        BORAC,  LOGM AND ORBWO B,BM
1032                CJMP NZERO BORCC + 2
1033                LOGM CFC
1034        BORCC,  LOGM AND ORBWO B,BM
1035                CJMP  NZERO BORAC +2
1036                LOGM CFC OR B,B2 A,S D,S
1037
1037
1037        KONE3,  LOGM OR B,BM ORBW
```

```
361
        %0 → BIT, FETCH
        %TEST K
154     %JMP IF K = 1
        %1 →  SPECIFIED BIT
        %1 → K, FETCH
        %TEST K
1037    %JMP IF K = 1
        %0 →  SPECIFIED BIT
        %0 → K, FETCH
        %TEST BIT
1022    %JMP IF BIT = 1
        %1 → K, FETCH
        %TEST BIT
1017    %JMP IF B = 1
        %0 → K, FETCH
        %TEST BIT
101
        %0 → K, FETCH
        %TEST BIT
1025    %JMP IF BIT = 1
        %K → K, FETCH
        %TEST BIT
1036    %JMP IF BIT = 1
        %K → K, FETCH
        %TEST BIT
1033    %JMP IF BIT = 1
        %1 → K, FETCH


        %1 → SPECIFIED BIT
```

```
1040                LOGM CFC ANDCB B,B2 A,S D,S          %^ → K, FETCH
1041
1041
1041
1041        CIIPP,  ARM A,H B,MIS D,IO                   %ION, IOF, PON, POF
1042                ARM CFC
1043
1043
1043
1043    % MOPC
1043    % OPERATORS COMMUNICATION FOR NORD-10

1043        MOPCM,  ARM A,Z B,PIE D,IO                   %MASTER CLEAR, RESET PIE
1044                ARM CHLEV                            %CHANGE LEVEL
0145                LOGM OR A,Z B,B2 D,SCR               %SET BIT 2 IN SCR → 1
1046                ARM A,SCR B,PCR D,IO                 %SET BANKNUMBER → 0
1047                ARM BDI ADD1 B,B2 D,SCR              %5 → SCR
1050                ARM A,SCR B,MIS D,IO                 %RESET PAGING AND INTERRUPT
1051                LOGM BDIR B,B5 D,P                   %BIT5 → CP
1052                ARM A,P B,MIS D,IO                   %SET MOPC–BIT, BIT 5 MISC
1053                JMP MOPCR                    1060
1054        MOPC,   LOGM ADIR A,P D,SP                   %CP → SP
1055                LOGM BDIR B,B5 D,P                   %BIT5 → CP
1056                ARM A,P B,MIS D,IO                   %BIT–5 → MISC
1057                ILOGM ADIR A,A D,SS LE11 DSPL        %SAVE A-REG
1060
1060        MOPCR,  LOGM BDIR B,B15 D,SCR                %BIT TO RESET LOAD LIGHT
1061                LOGM A,SCR B,PAC D,IO
1062                LOGM BDIR B,B5 D,SCR
1063                LOGM BDIR B,B4 D,SH
1064                IARM PLUS A,SCR B,SH D,SCR LE10 DSPL %60 → SCR LEV10
1065                LOGM BDIR B,B3 D,SCR
1066                ARM PLUS A,SCR B,B1 D,SH
1067                ILOGM BDIR B,SH D,SCR LE6 DSPL       %12 →  · SCR LEV 6
```

```
1070                LOGM ADIR A, Z D, SS                    %MOPC STATUS, 0 ↦ (SS LEV0)
1071                ILOGM ADIR A, Z D, SS LE12 DSPL         %CLEAR EXAMINE ADDRESS
1072                LOGM BDIR D, SCR B, B6                  %100 → SCR
1073                ARM PLUS B, B7 A, SCR D, SCR            %SET BIT 7
1074                ILOGM ADIR A, SCR D, SS LE17 DSPL       %300 → (SS LEV17) CONSOLE DEV NR.
1075
1075        NYFAF,  ARM A, IO B, MPC                        %SAVE RETUR ADD.
1076                JMP ACT                         1756
1077        REAC,   ARM A. IO B, MPC                        %SAVE MPC
1100                JMP ASS8                        1176

1101    %TEST TERMINATING CHARACTER IN A-REG
1101    %OCTAL NUMBER IN SH IF BIT14 SS IS ONE

1101                LOGM ADIR A, A
1102                JMP TO REAC IF ZERO             1077    %IGNORE BLANK'S
1103                LOGM BDIR B, B6 D, SCR                  %100 → SCR
1104                ARM BMINA A, SCR B, A D, A              %100–A → A
1105                CJMP ZERO MOPCM                 1043    %CHARACTER "100" READ
1106                LOGM BDIR B, B3 D, SCR
1107                ARM BMAMI B, A A, SCR
1110                CJMP ZERO IEX                   1436    %111, CHARACTER "1"
1111                ARM BMAM1 B, AC A, SCR D, SCR
1112                CJMP ZERO REX                   1443    %122, CHARACTER "R"
1113                ARM PLUS B, B4 A, SCR                   %102, B
1114                CJMP ZERO BANK                  1373
1115                ARM PLUS A, A B, B4 D, A ADD1           %A + 20 + 1 → A
1116                CJMP ZERO EXAM                  1273    %CHARACTER"/"
1117                ARM PLUS A, A  B, B2 ADD1 D, A          %TEST, 52*
1120                CJMP ZERO CLC                   1441
1121                ARM PLUS A, A B, B2 D, A                %A + 10 → A
1122                CJMP ZERO ETSGN                 1504    %CHAR. "&"
1123                ARM PLUS A, A B, B1 D, A                %A + 2 → A
1124                CJMP ZERO DOLL                  1501    %CHAR. "$"
1125                ARM PLUS A, A B, B1 ADD1 D, A           %A + 3 → A
1126                CJMP ZERO STPR                  1310    %CHAR. "!"
1127                ARM BDI ADD1 D, A B, A
```

```
1130                CJMP ZERO REAC                          1077    %SPACE
1131                ARM PLUS A, A B, B4 D, A                        . . + . 20 → A
1132                ARM PLUS A, A B, B1 ADD1                        %A + 3 → A
1133                CJMP ZERO CHCR                          1343    %CARRIAGE RET.
1134
1134        QUM,    LOGM BDIR D, SCR B, B6
1135                ARM PLUS B, B7 A, SCR D, SCR                    %300 → SCR
1136                ILOGM ADIR A, SCR D, SS LE17 DSPL               %300 → SS(LEV17) CONSOLE DEVICE
1137                ARM BM1 B, B6 D, A                              %77 → A
1140                LOGM ANDCB B, B17 D, SS                         %RESET LOAD BIT
1141                ARM A, IO B, MPC                                %SAVE MPC
1142                JMP OUTCH                               1735    %PRINT "?"
1143                LOGM BDIR B, B14 D, SCR                         %SET BIT 14
1144                ARM A, SCR B, PAC  D, IO                        %SET ERROR INDICATOR
1145                JMP MOPCR                               1060    %READ NEXT CHAR.
1146
1146    %ROUTINE TO PRINT OCTAL NUMBERS
1146    %PRINTS ON OPCOMDEV
1146    %NUMBER IN SH-MOST SIGN.
1146        OUT8,   IARM PLUS ADD1 A, H LE16 D, SCR DSPL            %SAVE RETUR
1147                LOGM BDIR B, B0 D, SC
1150                LOOP SHRO TSC0                                  %SHIFT LEFT SHRO 16
1151                LOGM BDIR B, SH D, A
1152                LOGM AND B, B0 A, A D, A                        %FIRST OCTAL NUMB → A-REG
1153                IARM PLUS A, SCR B, AC D, A LE10                %ADD 60
1154                ARM A, IO B, MPC                                %SAVE MPC
1155                JMP OUTCH                               1735    %PRINT FIRST DIGIT
1156                ARM PLUS A, Z ADD1 B, B2 D, SCR                 %5 → SCR
1157                IARM BMINA D, SCR B, Z A, SCR LE12 DSPL         %-SCR → SCR (LEV12)
1160        NECHP,  ARM BM1 B, B2 D, SC                             %3 → SC
1161                LOOP SHRO TSC0                                  %SHIFT 3 SHRO LEFT
1162                ARM BM1 B, B3 D, SCR                            %7 → SCR
1163                LOGM AND B, SH A, SCR                           %SH · 7 → AC
1164                IARM PLUS A, SCR B, AC D, A LE10                %AC + 60 → AC
1165                ARM A, IO B, MPC                                %SAVE MPC
```

```
1166                    JMP OUTCH                                   1735    %PRINT ONE DIGIT
1167                    ILOGM ADIR D,SCR A,SCR LE12
1170                    IARM PLUS ADD1 D, SCR B, Z A,SCR LE12
                        DSPL                                        %COUNT
1171                    CJMP NZERO NECHP                            1160
1172                    LOGM BDIR D, A B, B5                        %40 → A
1173                    LOGM ADIR B, MPC A, IO
1174                    CALL OUTCH                                  1735    %PRINT SPACE
1175                    JMP RETU1                                   1223
1176
1176
1176
1176
1176  %THIS ROUTINE READS AN OCTAL NUMBER
1176  %RETURN :OCTAL NUMBER IN SH(MOSTS)
1176  %TERMINATING CHARACTER IN A-REG
1176
1176            ASS8,   IARM PLUS ADD1 A,H B, Z D,SCR LE16
                        DSPL                                        %SAVE RETUR
1177                    LOGM ADIR A, Z D,SH                         %0 → SH
1200                    LOGM ANDCB B, B14 A,SS D,SS                 %RESET OCTAL FLAG
1201            RLOOP,  ARM A, IO B, MPC                            %SAVE RETUR
1202                    JMP INCH                                    1716    %READ ONE CHARACTER
1203                    LOGM ANDCB D, P B, B7 A, A                  %RESET PARITY
1204                    IARM BMINA D, A B, AC A,SCR LE10            %AC – 60 → A
1205                    CJMP SNEG EASS8                             1220    %JMP IF NOT OCT. DIGIT
1206                    LOGM BDIR B, B3 D,SCR                       %10 → SCR
1207                    ARM BMINA B, A A,SCR D, A                   %A–10 → A
1210                    CJMP SPOS EASS8                             1220    %JMP IF NOT OCT. DIGIT
1211                    ARM BM1 B, B3 D, A                          %7 → A
1212            LOGM    LOGM AND A, P B, A D,SCR                    %CP · A → SCR
1213                    ARM BM1 B, B2 D,SC                          %3 → SC
1214                    LOOP TSC0                                   %SHIFT 3 LEFT
1215                    LOGM OR A,SCR B,SH D,SH                     %NEW OCTAL DIGIT → SH
1216                    LOGM OR A,SS B, B14 D,SS                    %SET OCTAL FLAG
1217                    JMP RLOOP                                   1201    %READ NEXT CHAR.
```

```
1220
1220        EASS8,   LOGM ADIR D,A A,P                    ;TERMINATION CHAR. FROM P
1221                 IARM BMINA B,AC A,SCR LE6            ;TEST FOR LF
1222                 JMP TO RLOOP IF ZERO          1201   ;LINE FEED IGNORE
1223        RETU1,   ILOGM ADIR A,SCR D,P LE16            ;RETUR ADD. TO SCR
1224        RETU,    LOGM A ,P B,CAR D,IO                 ;SCR → CAR
1225                 JMP,CAR
1226
1226        PANINC,  LOGM BDIR B,B5 D,P                   ;BIT 5 SET IN CP
1227                 ARM A,P B,MIS D,IO                   ;SET MOPC–BIT 5 IN MISC
1230                 ARM A,IO B,OPR                       ;READ OPR
1231                 ILOGM ADIR D,SS A,H LE12 DSPL        ;SAVE OPR
1232                 ARM A,IO B,PAS                       ;READ PANEL STATUS
1233                 LOGM ADIR A,H D,P
1234                 LOGM BDIR B,B6 D,SS                  ;SET STATUS FOR PANEL INTERRUPT
1235                 JMP PANT2                     1257
1236
1236
1236
1236
1236
1236
1236
1236
1236  %THIS ROUTINE TESTS PANEL BUTTONS  AND
1236
1236        PANTT,   LOGM AND B,B17 A,SS                  ;TEST LOAD
1237                 CJMP NZERO RPANT              1717   ;JMP TO INCH
1240                 ILOGM BDIR LE2 B,SH D,SS DSPL        ;SAVE SH–REG
1241                 ARM A,IO B,PAS                       ;PANEL STATUS → H
1242                 LOGM ADIR A,H D,P                    ;SAVE PAS
1243                 ILOGM ADIR A,SS D,A LE11             ;UNSAVE A
1244                 LOGM AND B,B14 A,H                   ;TEST SET ADDRESS
```

```
1245                CJMP NZERO SETAD
1246                LOGM AND B,B12 A,H
1247                CJMP NZERO RESTA
1250                LOGM AND B,B13 A,H
1251                CJMP NZERO DEPP
1252                LOGM AND B,B11 A,H
1253                CJMP NZERO STFP
1254                LOGM AND B,B10 A,H
1255                CJMP NZERO LOAD
1256                JMP PANT1
1256
1256
1256        PANT1,  ILOGM BDIR B,SH D,SS LE2 DSPL
1257        PANT2,  LOGM AND B,B15 A,P
1260                CJMP NZERO RETPA
1261                ARM BM1 B,B7 D,SH
1262                LOGM  AND B,SH A,P D,SCR
1263                ARM A,SCR B,PAC D,IO
1264                LOGM AND B,B7 A,P
1265                CJMP NZERO MEXM
1266                ARM A,P B,CAR D,IO
1267                ARM
1270                ILOGM ADIR ORBWO D,SCR
1271                ARM A,SCR B,LMP D,IO
1272                JMP RETPA
1273
1273
1273
1273    %JMP TO THIS ROUTINE WHEN "/" IS TYPED
1273
1273        EXAM,   LOGM OR B,B10 A,SS D,SS
1274                LOGM AND B,B7 A,SS
1275                CJMP NZERO REXAM
1276                LOGM ANDCB B,B16 A,SS D,SS
1277                ILOGM BDIR B,SH D,SS LE12 DSPL
```

```
1446
                %TEST RESTART
1304            %JMP TO RESTART
                %TEST DEPOSIT
1467            %DEPOSIT, ADDR, IN SS LE12
                %TEST CONTINUE
1307            %START PROG. IN MAIN MEM.
                %TEST LOAD
1503            %LOAD FROM LOAD DEV.
1256            %JMP TO REGEX. OR MEM. EX


                %SAVE SH
                %TEST NOOP
1462

                %RESET BIT IN PAS
                %SET PAC
                %TEST REG. OR MEM. EXAM.
1456
                %SET CAR

                %READ REG
                %SET LAMP DISP.
1462


                %SET EXAM BIT
                %TEST REGISTER EXAM
1321            %JMP REG EXAM IF NOT ZERO
                %RESET REG DEP
                %SAVE ADDRESS
```

```
1300        EXRO,   ILOGM ADIR A,SS D,P LE12                %ADDRESS → CP
1301                ARM CPTR BM1 B,P D,P                    %CP – 1 → CP
1302                ARM CRR1
1303                JMP IEXA1                       1424
1304
1304
1304  %START,  <OCTALN.>!
1304  %START ADDRESS IN SH IF OCTALNUMBER WRITTEN ELSE IN
1304  %SP (CONTINUE)
1304
1304        RESTA,  LOGM BDIR B,B4 D,SP                     %SET RESTART ADDRESS
1305        STSP,   ARM A,Z B,PIE D,IO                      %RESET PIE
1306                ARM CHLEV                               %CHANGE LEVEL
1307        STFP,   LOGM ADIR A,Z D,SS                      %0 → SS,  0 → OCTAL FLAG
1310        STPR,   LOGM AND B,B14 S,SS D,SS                %PREPARE OCTAL TEST
1311                ILOGM ADIR A,SS D,A LE11                %UNSAVE A–REG
1312                ARM A,Z B,MIS D,IO                      %0 → MISC
1313                ILOGM ADIR A,SS LE0                     %TEST OCTAL FLAG
1314                CLOGM NZERO BDIR B,SH D,SP              %START ADDRESS → SP
1315                LOGM BDIR B,B15 D,SCR                   %SET BIT FOR RESET LOAD
1316                LOGM OR B,B11 A,SCR D,SCR               %SET CONTINUE BIT
1317                ARM A,SCR B,PAC D,IO                    %SET BITS TO PANEL CONT.
1320                LOGM ADIR A,SP  D,P CFC                 %START ADDRESS → CP, START
1321
1321
1321
1321  %REGISTER EXAMIN. JMP FROM EXAM
1321  %TESTS ON BIT 15 IN STATUS(SS) AND JUMPS
1321  %TO IEXAM IF "INTERNAL REG"
1321
1321        REXAM,  LOGM ANDCB B,B7 A,SS                    %RESET REG EXAM
1322                LOGM OR B,B16 A,SS D,SS                 %SET REG DEP
1323                ILOGM BDIR B,SH D,SS LE7 DSPL           %SAVE REG NUMBER
1324                LOGM AND A,SS B,B15                     %TEST INTERNAL REG
```

```
1325                CJMP NZERO IEXAM                    1421  %"INTERNAL REG" IF NOT ZERO
1326        ERDP,   ILOGM ADIR A,SS D,SH LE7                  %UNSAVE REG NUMBR
1327                ARM BM1 B,B3 D,SCR                        %7 → SCR
1330                LOGM AND A,SCR B,SH D,SCR                 %MASK REG NUMB
1331                ILOGM ADIR A,SS D,SH LE14                 %READ LEVEL
1332                ARM BM1 B,B2 D,SC                         %3 → SC
1333                LOOP TSC0                                 %SHIFT REG NUMB
1334                LOGM OR A,SCR B,SH D,SCR                  %GENERATE REG AND LEVEL
1335                ARM A,SCR B,CAR D,IO
1336                ILOGM ADIR A,SS D,A LE11                  %UNSAVE A
1337                LOGM AND B,B11 A,SS                       %TEST DEPOSIT
1340                CJMP NZERO REDEP                  1403    %RETUR TO DEPOSIT
1341                ILOGM ADIR ORBWO D,SH                     %READ REGISTER
1342                JMP IEXA                          1425
1343
1343
1343
1343  %DEPOSIT ROUTINE, ENTERED AFTER CARRIAGE RETURN
1343
1343        CHCR,   ARM A,IO B,MPC                            %SAVE RETUR
1344                JMP PRLF                          1430    %PRINT LF
1345                LOGM ANDCB B,B7 A,SS D,SS                 %RESET BIT FOR RI (INT-REG)
1346                LOGM AND B,B16 A,SS                       %TEST REG DEPOSIT
1347                CJMP NZERO REGDP                  1412    %JMP TO REG DEPOSIT IF NOT ZERO
1350                LOGM AND B,B14 A,SS                       %TEST OCTAL NUMB. WRITTEN
1351                CJMP ZERO NDEP                    1360
1352                LOGM  AND B,B10 A,SS
1353                JMP TO QUM IF ZERO                1134    % / FLAG NOT SET
1354        DEPM,   ILOGM ADIR A,SS D,P LE12                  %READ ADDRESS
1355                ARM CPTR BM1 B,P D,P                      %CP - 1 → CP, CP → R
1356                LOGM BDIR B,SH D,SCR                      %DATA TO A-BUS REG.
1357                ARM CWR1 A,SCR                            %WRITE SCR
1360        NDEP,   LOGM AND B,B10 A,SS                       %TEST EXAM BIT
1361                CJMP ZERO REAC                            %NO DEPOSIT
1362                ILOGM ADIR D,P A,SS LE12          1077
1363                IARM BDI ADD1 D,SS B,P LE12 DSPL          %INCREMENT ADDRESS
1364                JMP EXRO                          1300
```

```
1365
1365
1365
1365  %INTERNAL REGISTER DEPOSIT
1365
1365        IRD,    ILOGM ADIR A,SS D,SCR LE17          %READ REG NUMB.
1366                ARM A, SCR B,CAR D,IO               %SET REG. NUMBER IN CAR
1367                LOGM BDIR B,SH D,SCR                %NUMBER WRITTEN IN SCR
1370                ARM ADIR A,SCR B,IR3 D,IO           %DEPOSIT IN SPESY. REG
1371                LOGM ANDCB A,SS B,B15 D,SS          %RESET INTERNAL REG BIT
1372                JMP NRDP                    1406
1373
1373
1373        BANK,   LOGM  BDIR B,SH D,SCR
1374                LOGM OR B,B2 A,SCR D,SCR            %SET BIT 2  IN SCR → 1
1375                ARM A,SCR B,PCR D,IO                %SET BANKNUMBER IN PCR
1376                LOGM ANDCB B,B7 A,SS D,SS           %RESET BIT FOR R- I
1377                JMP NRDP1                   1410
1400
1400
1400  %*=*=*=*
1400
1400
1400                JMP 400                             %INTERRUPT AND PANEL INT
1401                JMP MOPCM                   1043    %MASTER CLEAR
1402                JMP MOPC                    1054    %STOP
1403
1403
1403  %*=*=*=*
1403
1403
1403        REDEP,  ILOGM ADIR A,SCR D,SH LE1           %READ NUMBER
1404        RPDEP,  ILOGM BDIR ORBW DSPL B,SH           %DEPOSIT IN REG
1405                ILOGM ADIR A,A D,SS LE11 DSPL       %SAVE A-REG
1406        NRDP,   LOGM ANDCB B,B11 A,SS D,SS          %RESET DEPOSIT BIT
1407                LOGM ANDCB A,SS B,B16 D,SS          %RESTORE REGEX BIT
```

```
1410        NRDP1,  LOGM ANDCB A,SS B,B10 D,SS          %RESET / BIT
1411                JMP TO REAC                 1077
1412
1412
1412
1412
1412    %ROUTINE FOR REGISTER DEPOSIT
1412
1412        REGDP,  LOGM AND B,B14 A,SS                 %TEST OCTAL NUMBER WRITTEN
1413                CJMP ZERO NRDP              1406    %NO DEPOSIT
1414                LOGM AND B,B15 A,SS                 %TEST "INTERNAL REG"
1415                CJMP NZERO IRD              1365    %INTERNAL REGISTER DEPOSIT
1416                ILOGM BDIR B,SH D,SCR LE1 DSPL      %SAVE NUMBER
1417                LOGM OR B,B11 A,SS D,SS             %SET BIT FOR DEPOSIT
1420                JMP ERDP                    1326    %JMP TO GEN, REGNR. AND LEVEL
1421
1421
1421
1421    %"INTERNAL REG" EXAMIN
1421
1421        IEXAM,  LOGM BDIR B,SH D,SCR                %REG. NUMBER → SCR
1422                ARM A,SCR B,CAR D,IO                %SET REG. NUMBER → CAR
1423                ARM A,IO B,IR 3                     %READ REG CONTENT
1424        IEXA1,  LOGM ADIR A,H D,SH
1425        IEXA,   ARM A,IO B,MPC                      %SAVE RETUR
1426                JMP OUT8                    1146    %PRINT NUMBER
1427                JMP REAC                    1077    %READ NEXT INP
1430
1430
1430
1430        PRLF,   IARM PLUS A,H B,Z ADD1 D,SCR LE5
                    DSPL                            %SAVE RETUR
```

```
1431                ILOGM ADIR A,SCR LE6  D,A          %12 → A
1432                ARM A,IO B,MPC                     % SAVE RETUR
1433                JMP OUTCH                  1735    %PRINT LF
1434        RETU5,  ILOGM ADIR A,SCR D,P LE5           %RETUR ADR → CP
1435                JMP RETU                   1224
1436
1436
1436
1436
1436
1436    %CHARACTER "I" WRITTEN
1436
1436        IEX,    LOGM OR B,B15 A,SS D,SS            %SET FLAG FOR I
1437                LOGM OR B,B7 A,SS D,SS             %SET FLAG FOR R + I
1440                JMP NRDP1                  1410
1441
1441        CLC,    ILOGM ADIR A,SS D,SH LE12          %READ CURRENT LOC. COUNTER
1442                JMP IEXA                   1425
1443
1443    %CHARACTER "R" WRITTEN
1443
1443        REX,    ILOGM BDIR B,SH D,SS LE14 DSPL     %SAVE LEVEL
1444                LOGM ANDCB B,B15 A,SS D,SS         %RESET BIT FOR I
1445                JMP IEX + 1                1437
1446
1446
1446        SETAD,  ARM A,IO B,OPR                     %READ SWITCH REGISTER
1447                ILOGM ADIR A,H D,SS LE12 DSPL      %SAVE ADDRESS
1450                LOGM OR B,B0 A,P D,SH              %SET BIT 0 (BIT 2 PCR)
1451                LOGM BDIR B,B1 D,SC                %SET SHIFT COUNTER
1452                LOOP SHRO TSC0                     %SHIFT BANK NUMBER
1453                LOGM BDIR B,SH D,SCR
```

```
1454                ARM A,SCR B, PCR D,IO                    %SET BANKNUMBER
1455                JMP REAC                          1077
1456
1456
1456    MEXM,       ILOGM ADIR A,SS D,P LE12                 %READ ADDRESS
1457                ARM BM1 B, P D,P CPTR
1460                ARM CRR1
1461                ARM A,H B,LMP D,IO                       %SET DATA
1462    RETPA,      ILOGM ADIR A,SS D,SH LE2                 %UNSAVE SH
1463                LOGM AND B,B6 A,SS                       %TEST PANEL INTERRUPT
1464                CJMP ZERO RPANT                   1717   %RETUR TO INCH
1465                ARM A, Z B,MIS D,IO                      %RESET MOPC BIT
1466                LOGM CFC ADIR A,SP D,P                   %SP → CP, FETCH
1467
1467
1467
1467
1467
1467
1467    DEPP,       ARM A,IO B,OPR                           %READ OPR
1470                LOGM AND B,B7 A,P                        %TEST REG. OR MEM. DEPOSIT E
1471                CJMP ZERO RDEP                    1476
1472                ILOGM ADIR A,SS   D,P LE12               %READ ADDRESS
1473                ARM BM1 B,P D,P CPTR
1474                ARM CWR1 A,H                             %DEPOSITE IN MEM.
1475                JMP REAC                          1077
1476
1476    RDEP,       ARM A,P B,CAR D,IO                       %SET REG. N. AND LEVEL → CAR
1477                LOGM ADIR A,H D,SH                       %DATA → SH
1500                JMP RPDEP                         1476
1501
1501
```

```
1501  %ENTRY IN MOPC AFTER $,& OR LOAD COMMAND
1501  %(SH1)= ALD-NUMBER, SPECIFIED ON CONSOLE DEVICE
1501  %IF NO ALD  NUMBER IS SPECIFIED, THE DEFAULT NUMBER IS READ FROM ALD
1501  %ALD NUMBER FORMAT
1501  %BIT 15 : EXTN, EXTENDED LOAD FUNCTION
1501  %BIT 14 : RESTART
1501  %BIT 13 : MASS STORAGE LOAD
1501  %BIT 12 : OTAL LOAD
1501
1501
1501        DOLL,   LOGM BDIR D, A B, B14                    %10000 → A, AFTER $
1502                JMP DOLET                          1505
1503        LOAD,   LOGM BDIR B, B17 D, SS                   %SET LOAD–FLAG, RESET OCTAL READ FLAG
1504
1504        ETSGN,  LOGM BDIR D, A B, Z                      %0 → A, AFTER & OR LOAD
1505
1505        DOLET,  LOGM ADIR B, PAS A, IO                   %READ PANEL
1506                ARM BM1 D, P B, B10                      %377 → P
1507                LOGM AND D, P B, P A, H
1510                LOGM OR D, P B, B10 A, P                 %SET LOAD BIT IN PANEL CONTROL
1511                LOGM ADIR D, IO B, PAC A, P
1512                LOGM AND B, B14 A, SS                    %TEST IF OCTAL NUMBER READ
1513                JMP TO DE0 IF NZERO                15!6  %JMP OCTAL READ
1514                LOGM ADIR B, ALD A, IO                   %GET ALD
1515                LOGM ADIR D, SH A, H
1516        DE0,    LOGM OR D, A A, A   B, SH                %OR IN B14 IF $ WRITTEN
1517                LOGM AND B, B17 A, A
1520                JMP TO EXTN IF NZERO               1574  %EXTENDED LOAD FUNCTION
1521
1521                LOGM AND B, B16 A, A                     %RESTART FUNCTION?
1522                JMP TO RSTRT IF NZERO              1576  %YES
1523                LOGM AND B, B15 A, A
1524                JMP TO MASS IF NZERO               1601  %MASS STORAGE LOAD
1525
1525        DE1,    ILOGM ADIR D, SS A, A LE17 DSPL          %DEV. NR. → SS LEV17
```

```
1526                LOGM BDIR D,SS  B,B17                   %SET LOAD FLAG
1527                LOGM OR D,SS B,B10 A,SS                 %SET EXAMINE BIT IN SS
1530                LOGM AND B,B14 A,A
1531                JMP TO NYFAF IF NZERO       1075        %OCTAL LOAD IF A = 010000
1532                JMP TO BINL                 1532
1532
1532
1532
1532
1532
1532  %BINARY LOAD
1532  %WILL PICK UP NECESSZRY PARAMETERS FROM A STANDARD (OCTAL)
1532  %BOOTSTRAP AND LOAD THE BINARY INFORMATION INTO CORE
1532  %IMMEDIATELY FOLLOWING THE CHECKSUM THERE IS AN ACTION
1532  %COMMAND. IF TERMINATING
1532  %CHARACTER IS A BLANK, THEN THE LOADED PROGRAM IS STARTED IN
1532  %THE START ADDRESS FOUND IN THE BOOTSTRAP.  OTHERWISE THE COMMAND
1532  %IS ACTED UPON BY MOPC
1532
1532
1532
1532        BINL,   LOGM  ADIR B,MPC  A,IO
1533                CALL ACT                    1756        %ACTIVATE DEVICE
1534                LOGM ADIR D,SH A, Z                     %0 → SH
1535
1535        SEEK,   LOGM BDIR D,SP B,SH                     %SH → SP (POSSIBLE START ADDRESS)
1536
1536        SIKI,   LOGM ADIR B, MPC A, IO
1537                CALL ASS8                   1176
1540                LOGM AND B, B14 A, SS                   %OCTAL NUMBER READ?
1541                JMP TO SIKI IF ZERO         1536        %NO
1542                ARM BMINA B, B5 A, A                    % A – 40 → AC
1543                ARM PLUS ADD1 B, AC A, Z                %AC + 1 → AC
1544                JMP TO SEEK IF NZERO        1535        %SEEK FOR '1' AS TERMINATOR
1545                LOGM ADIR B, MPC A, IO
1546                CALL BIN                    1645
```

```
1547                ARM BM1 D,P B,SH CPTR                   %SH - 1 → CP, CP → R, BLOCK START
1550                LOGM ADIR B,MPC A,IO
1551                CALL BIN                        1645
1552                LOGM BDIR D,T B,SH                      %SH → T, WORD COUNT
1553                LOGM ADIR D,D A,Z                       %0 → D
1554
1554        STLP,   LOGM ADIR B,MPC A,IO                    %STORE LOOP
1555                CALL BIN                        1645
1556                LOGM BDIR D,A B,SH                      %SH → A, BINARY WORD
1557                ARM PLUS D,D B,D A,A                    %ACCUMULATE CHECKSUM
1560                ARM BM1 D,T B,T CWR1 A,A                %T - 1 → T, STORE A IN (R) + 1
1561                JMP TO STLP IF NZERO            1554
1562                LOGM ADIR B,MPC A,IO
1563                CALL BIN                        1645    %READ CHECKSUM
1564                LOGM EXOR B,SH A,D
1565                JMP TO QUM IF NZERO             1134    %CHECKSUM ERROR
1566                LOGM BDIR D,SS B,B17                    %SET LOAD BIT
1567                LOGM ADIR B,MPC A,IO
1570                CALL ASS8                       1176    %READ ACTION CODE
1571                LOGM ADIR A,A
1572                JMP TO STSP IF ZERO             1305    %START PROGRAM (TERMINATOR = BLANK)
1573                JMP TO REAC+2                   1101
1574
1574
1574
1574
1574
1574
1574    %EXTN
1574    %EXTENDED LOAD FUNCTION
1574    %WHEN THE EXTN-BIT (BIT15) IN A IS SET, A MICRO-JUMP TO THE ADDRESS
1574    %FOUND IN A BIT0-10 IS PERFORMED
1574    %NO FURTHER DECODING OF ALD TAKES PLACE
1574
1574
1574        EXTN,   LOGM ADIR D,IO B,CAR A,A
1575                JMP, CAR
```

```
1576
1576
1576  %RSTRT
1576  %RESTART FUNCTION (NOT TO BE CONFUSED WITH THE RESTART BUTTON)
1576  %WHEN THE RSTRT-BIT (BIT 14) OF A  IS SET, THE CPU IS STARTED
1576  %IN 4* (THE ADDRESS FOUND IN BIT0-13 IN A ). BIT 0-1 IN THE
1576  %RESULTING ADDRESS IS 0
1576
1576       RSTRT,   ARM PLUS D, A B, A A, A                    %*2
1577                ARM PLUS D, SP  B, A A, A                  %*4
1600                JMP TO STSP                          1305  %START PROGRAM
1601
1601
1601
1601  %MASS
1601  %MASS STORAGE LOAD
1601  %MASS WILL LOAD 1K WORDS FROM MASS STORAGE ADDRESS 0 INTO MEMORY
1601  %FROM MEMORY ADDRESS0
1601  %(LOWEST)  MASS STORAGE REGISTER ADDRESS IS FOUND IN BIT 0-10 IN ALD
1601  %THE ACTUAL MASS STORAGE MUST CONFORM WITH THE DRUM/DISC
1601  %PROGRAMMING SPECIFICATION.
1601
1601       MASS,    ARM PLUS D, P B, B0 A, A CPTR              %DEV NR(DNO) + 1 → CP, CP → R
1602
1602       MAS1,    LOGM ADIR D, A A, Z                        %0 → A
1603                LOGM ADIR D, SCR A, R                      %DNO + 1
1604                LOGM ADIR B, MPC A, IO
1605                CALL IOXR                            1632  %0 → CORE ADDRESS
1606                ARM PLUS D, SCR B, B1 A, R                 %DNO + 3
1607                LOGM ADIR B, MPC A, IO
1610                CALL IOXR                            1632  %0 → BLOCK ADDRESS
```

```
1611                LOGM BDIR D, A B, B12                    ?000 → A
1612                ARM PLUS D,SCR B, B2 A,SCR               %DNO + 7
1613                LOGM ADIR B, MPC A, IO
1614                CALL IOXR                          1632  %2000 → WORD COUNTER
1615                LOGM BDIR D, A B, B2                     %4 → A
1616                ARM PLUS D,SCR B, B2 A, R                %DNO + 5
1617                LOGM ADIR B, MPC A, IO
1620                CALL IOXR                          1632  %ACTIVATE DEVICE
1621                ARM PLUS  ADD1 D,SCR B, B1 A, R          %DNO + 4
1622
1622        MAS2,   LOGM ADIR B, MPC A, IO
1623                CALL IOXR                          1632
1624                LOGM AND B, B2 A, A
1625                JMP TO MAS2 IF NZERO               1622  %DEVICE ACTIVE
1626                LOGM ADIR D,SP A, Z                      %START IN MEMORY ADDRESS 0
1627                LOGM AND B, B4 A, A                      %ERROR?
1630                JMP TO STSP IF ZERO                1305  %OK
1631                JMP MAS1                           1601  %ERROR' TRY AGAIN
1632
1632
1632
1632
1632  %IOX
1632  %IOX EXECUTES A GENERAL IOX OPERATION
1632  %CALL WITH : IOX_INSTRUCTION OR DEVICE-NUMBER IN SCR0
1632  %  DATA IN A0
1632  %RETURN WITH : DATA IN A0  '  IOXR INSTRUCTION IN SCR0
1632  %
1632
1632        IOXR,   ARM PLUS D, P B, B0 A, H                 %SAVE RETURN = CALL + 1
1633                ARM BM1 B, B13                           %3777 → AC
1634                LOGM AND D,SCR B, AC A,SCR               %STRIP OFF DEVICE NUMBER
1635                ARM BM1 B, B13                           %3777 → AC
1636                LOGM ORCB D,SCR B, AC A,SCR              %17400 + SCR → SCR
1637                LOGM ANDCB D,SCR B, B14 A,SCR            %167777 · SCR → SCR
```

```
1640                LOGM ADIR D, IO B, CAR A, SCR          %SET UP CAR
1641                LOGM ADIR D, IO B, IO A, A             %OUT -
1642                LOGM ADIR B, IO A, IO                  %- IN -
1643                LOGM ADIR D, A A, H                    % - TO A-REG
1644                JMP RETU                        122-4  %STANDARD RETURN
1645
1645
1645
1645
1645  %BIN
1645  %BIN READS A 16BIT WORD IN TWO BYTES
1645  %CALL WITH  : DEVICE-NO IN SS17
1645  %RETURNS WITH : DATA IN SH1
1645
1645
1645        BIN,    IARM PLUS ADD1 D, SCR B, Z A, H LE5 DSPL  %SAVE RETURN
1645
1646        BIN1,   LOGM ADIR B, MPC A, IO
1647                CALL INCH                       1716   %READ 1, BYTE
1650                LOGM ADIR D, SH A, A
1651                LOGM BDIR D, SC B, B3                  %8 → SC
1652                LOOP                                   %SHIFT 8 LEFT
1653
1653        BIN2    LOGM ADIR B, MPC A, IO
1654                CALL INCH                       1716   %READ 2, BYTE
1655                LOGM OR D, SH B, SH A, A               %JOIN
1656                JMP RETU 5                      1434   %STANDARD RETURN
1657
1657
1657  %MAIN MEMORY CHECK
1657  %MICRO-PROGRAM, VERSION C
1657  %NJL 6/6/73
```

```
1657   %
1657   %TESTS MEMORY FROM ADDRESS SPECIFIED IN B-REG AND UP TO
1657   %ADDRESS SPECIFIED IN X-REG
1657   %
1657   %ERROR INDICATION:
1657   %      T=FAILING BITS          [0R6 ]
1657   %      P=FAILING ADDRESS       [0R2 ]
1657   %      D=ERROR PATTERN         [0R1 ]
1657   %      L=TEST PATTERN          [0R4 ]
1657   %      B=START ADDRESS         [0R3 ]
1657   %      X=STOP ADDRESS          [0R7 ]
1657   %
1657   %IF NO ERROR IS FOUND T = 000000
1657   %
1657   %PATTERNS USED :
1657   %
1657   %      000001
1657   %      000002
1657   %      000004
1657   %      ......
1657   %      ......
1657   %      100000
1657
1657   %      ADDRESS STORED IN ADDRESS (THIS PATTERN IS RUN 16 TIMES)
1657   %
1657   %      AND THEIR COMPLEMENTS
1657   %
1657   %USE OF REGISTERS IN THIS PROGRAM :
1657   %
1657   %      A = 000000  USE T AS TEST DATA
1657   %      A = 177777  USE ADDRESS AS TEST DATA
1657   %
1657   %      SCR = 000000  WRITE TEST DATA INTO CORE
1657   %      SCR = 177777  READ TEST DATA FROM CORE
1657   %
1657   %      D = 000000 USE TEST DATA AS IS
1657   %      D = 177777 USE COMPLEMENT OF TEST DATA
1657   %
1657   %      L =          TEST DATA
```

```
1657
1657    MMCC,   LOGM ADIR D, A A, Z                 %0 → SS                 INIT PHASE
1660
1660    MM00,   LOGM ADIR D, D A, Z                 %0 → D                  INIT POLARITY
1661
1661    MM0,    LOGM BDIR D, T B, B0                %1 → T                  INIT DATA
1662
1662    MM1,    LOGM ADIR D, SCR A, Z               %0 → SCR                INIT STORE
1663
1663    MM2,    ARM BM1 D, P B, B CPTR              %(B) – 1 → CP, CP → R   INIT ADDRESS
1664
1664    MM3,    ARM PLUS ADD1 D, L B, Z A, R        %(R) + 1 → L            MAKE DATA
1665            LOGM ADIR A, A
1666            LOGM ADIR D, L A, T IF ZERO         %(T) → L IF (A) = 0
1667            LOGM EXOR D, L B, D A, L            %(L) → L IF (D) = 177777
1670            LOGM ADIR A, SCR
1671            JMP TO MM4 IF NZERO         1674
1672            ARM A, L CWR1                       %LOAD IF (SCR) = 0
1673            JMP TO MM41                         %STORE L IN (R) + 1
1674                                        1677
1674    MM4,    ARM CRR1                            %((R) + 1) → H
1675            LOGM EXOR B, L A, H
1676            JMP TO ERR IF NZERO         1712    %ERROR DETECTED
1677    MM41,   ARM BMINA B, X A, R                 %TEST FOR LAST LOCATION
1700            JMP TO MM3 IF NZERO         1664    %CONTINUE
1701            LOGM ADIRC D, SCR A, SCR            % SWITCH LOAD/STORE
1702            JMP TO MM2 IF NZERO         1663
1703            ARM PLUS D, T B, T A, T             %T + T → T              SWITCH DATA
1704            JMP TO MM1 IF NZERO         1662
```

```
1705        MM45,   LOGM ADIRC D,D A,D                      /o                  SWITCH POLARITY
1706                JMP TO MM0 IF NZERO             1661    %DO COMPLEMENT
1707
1707        MM5,    LOGM ADIRC D,A A,A                      %                   SWITCH PHASE
1710                JMP TO MM00 IF NZERO            1661    %CONTINUE WITH ADDRESS IN ADDRESS
1711                JMP TO MOPC                     1054    %END OF TEST
1712
1712        ERR,    LOGM ADIR D,D  A,H                      %DATA READ
1713                LOGM ADIR D,SP A,R                      %FAILING ADDRESS R → SP
1714                LOGM EXOR D,T B,L A,D                   %FAILING BITS → T
1715                JMP QUM                         1134    %MOPC ERROR ENTRY-POINT
1716
1716
1716
1716    %INCH
1716    %INPUT CHARACTER (BYTE) FROM DEVICE FOUND IN  (SS17)
1716    %RETURNS CHARACTER IN A0
1716    %***  SCR0 IS DESTROYED  ***
1716    %IF LOAD-BIT IS SET (BIT17 IN SS0) THEN THERE WILL BE NO ECHOING
1716    %ON DEVICE-NUMBER (SS17) +4
1716
1716        INCH,   IARM PLUS ADD1 D,SCR B, Z A,H LE15 DSPL
1717
1717        RPANT,  ILOGM ADIR D,SCR A,SS LE17              %(RETURN FROM PANTT) DN0 → SCR
1720                ARM PLUS D,SCR  B,B1 A,SCR              %DN0 + 2 → SCR
1721                LOGM ADIR B, MPC A, IO
1722                CALL IOXR                       1632    %READ STATUS
1723                LOGM AND B,B3 A,A                       %READY FOR TRANSFER ?
1724                JMP TO PANTT IF ZERO            1236    %NO, TEST PANEL
1725                ILOGM ADIR D,SCR A,SS LE17              %YES, DN0 → SCR
1726                LOGM ADIR B, MPC A, IO
1727                CALL IOXR                       1632    %READ CHARACTER
```

```
1730                ILOGM ADIR D,SCR A,A DSPL LE13          %SAVE A
1731                LOGM ADIR B,MPC A,IO
1732                CALL ACT                        1756    %ACTIVATE DEVICE
1733                ILOGM ADIR D,A A,SCR LE13               %UNSAVE A0
1734                JMP OUT1                        1736    %FOR ECHO-ING
1735
1735
1735  %OUTCH
1735  %WRITES A CHARACTER ON DEVICE (SS17) +4
1735  %*** SCR0 IS DESTROYED ***
1735
1735        OUTCH,  IARM PLUS ADD1 D,SCR B,Z A,H LE15 DSPL
1736
1736        OUT1,   LOGM AND B,B17 A,SS
1737                JMP TO OUT3 IF NZERO            1754    %NO OUTPUT IF LOAD MODE
1740                ILOGM ADIR D,SCR A,A LE13 DSPL          %SAVE A0
1741                IARM PLUS ADD1 D,SCR B,Z A,SS LE17      %DN0 + 1 → SCR
1742                ARM PLUS ADD1 D,SCR B,B2 A,SCR          %DN0 + 6 → SCR
1743
1743        OUT2,   LOGM ADIR B,MPC  A,IO
1744                CALL IOXR                       1632    %READ STATUS
1745                LOGM AND B,B3 A,A                       %READY FOR TRANSFER?
1746                JMP TO OUT2 IF ZERO             1743    %NO
1747                LOGM ADIR A,SCR                         %SCR → AC
1750                ARM BM1 D,SCR B,AC                      %DN0 + 5 → SCR
1751                ILOGM ADIR D,A A,SCR LE13               %UNSAVE A0
1752                LOGM ADIR B,MPC A,IO
1753                CALL IOXR                       1632    %WRITE CHARACTER
1754
1754        OUT3,   ILOGM ADIR D,P A,SCR LE15
1755                JMP RETU                        1224    %STANDARD RETURN
1756
1756
1756  %ACTIVATES DEVICE
1756
1756        ACT,    IARM PLUS ADD1 D,SCR B,Z A,SS LE17      %DN0 + 1 → SCR
```

```
1757                ARM PLUS D,SCR B,B1 A,SCR          %DN0 + 3 → SCR
1760                ARM PLUS ADD1 A.Z B,B2 D,A         ~< → A
1761                LOGM AND B,BI7 A,SS                %LOAD MODE ?
1762                JMP TO ACT1 IF NZERO          1766 % YES. DO NOT SET PARI.  ETC.
1763                ARM PLUS A,A B, B13 D,A            % 4005 → A
1764                LOGM OR D,A B,B17 A,A
1765                LOGM OR D,A B,B16 A,A              %144005 → A
1766
1766        ACT1,   CALL IOXR                          %(RETURN ADDRESS ALREADY IN H-REG)
1767
1767
1767
1767
1767    %THIS PROGRAM STORES THE L-REGISTER IN THE LOCATION CONTAINED
1767    %IN B-REGISTER.  IT THEN LOADS THE SAME ADDRESS INTO THE D-REGISTER
1767    %AND FORMS EXCLUSIVE OR OF L- AND D-REGISTER IN THE T-REGISTERS
1767    %THIS LOOP CAN ONLY BE TERMINATED BY PUSHING MASTER CLEAR.
1767
1767        MLOOP,  ARM BM1 D,P B,B CPTR               %B - 1 → CP, CP → R
1770
1770        MLP1,   LOGM ADIR A,L CWR1 IF ZERO         %STORE L (ACTIVATE CONDITION)
1771                LOGM ADIR D,D A,H CPTR             %H → D, CP→ R
1772                LOGM EXOR D,T B,L A,D CRR1         %L ∨ D → T, LOAD H
1773                JMP MLOOP                 1767
1774
1774
1774        MCLS,   ARM A,IO BIR3                      %MCL, REG → H
1775                LOGM ANDCB A,H B,A D,SP            %N(A), H → SP
1776                LOGM A,SP BIR3 D,IO ADIR           %REG → SP
1777                JMP FETCH                 101
2000
2000
2000        )LINE
```

– we make bits for the future

NORSK DATA A.S BOX 4 LINDEBERG GARD OSLO 10 NORWAY PHONE 30 90 30 TELEX 18661