NORD - 1
REFERENCE MANUAL

NORD - 1
REFERENCE MANUAL

NORD - 1 REFERENCE MANUAL

Complete instruction repertoire

Date: January 1968

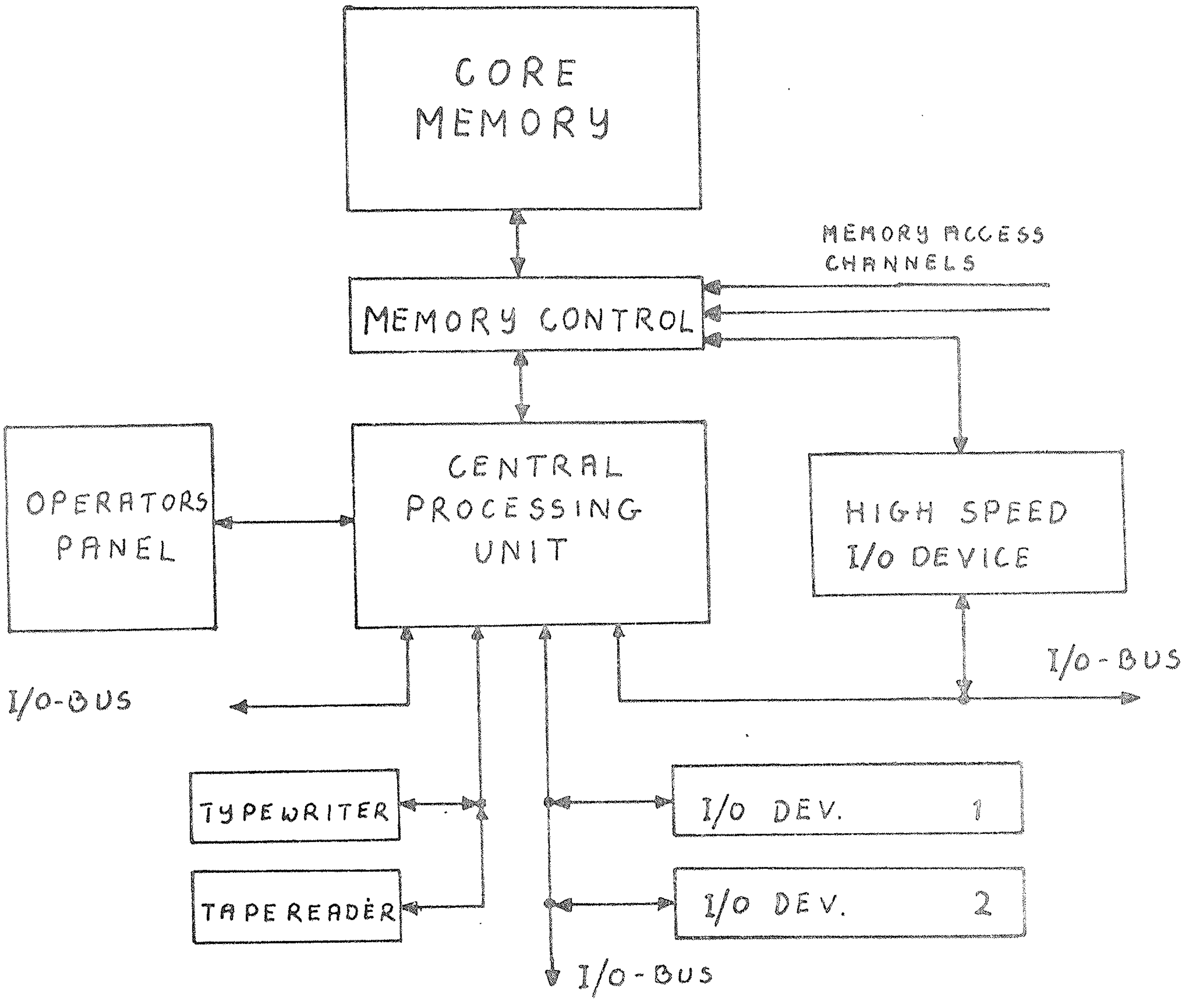
Contents

NORD-1 REFERANCE MANUAL

1. INTRODUCTION

NORD-1 is a general purpose stored program digital computer developed, manufactured and marketed by A/S NORDATA, Norwegian Data Electronics, Oslo.

The main characteristic of the NORD-1 computer is its extremely broad instruction repertoire which also includes floating point arithmetic, its unique interrupt system for real-time and multiprogramming systems, and its flexible communication with peripheral equipment.



NORD-1 COMPUTER SYSTEM

2 SYSTEM ORGANIZATION

2.1 Core memory

The main storage device is a coincident current ferrite core memory. The memory size varies from 4096 words to 65536 words. Word length is 16 bits plus parity bit.

The central processing unit operates asynchronous to the memory timing control and the computer therefore may accept memories of different speed. The fastest memory speed which may be efficiently utilized by the central processing unit is 1 microsec. cycletime.

2.2 Memory control

Each memory block has its own memory control. This memory control permits direct access from 4 different devices to the memory block. The priority between the devices will be fixed (wired in priority). One of the devices is the central processing unit, usually at lowest priority. Together with one CPU, three data channels may have access to each memory block. The data channels are usually connected to such devices as disc storage, magnetic tape storage, line printers or other input-output devices with high data transfer rate. When the data channels are operating memory cycles are stolen from the program running, for each data channel transfer of a 16 bit word one memory cycle is stolen. With a 1 us cycle time core store the maximum total data channel transfer rate is 16.000.000 bits/sec. The memory control is designed for multiprocessor systems. Two or more central processing units may be connected to one or more memory blocks.

2.3 Central processing unit

The central processing unit, CPU, controls the execution of the instructions and the input-output system. Basically the CPU consists of a register block, control flip-flops and an arithmetic and control unit.

2.3.1 Register block.

The register block consists of 8 general registers, 4 bus memory registers and 2 priority interrupt control registers. The CPU registers are 16 bit high-speed, integrated circuit registers.

The 8 general registers are:

R-register:	Address register, this register is not accessible by program.

A-register:	This is the main register for arithmetic and logical operations directly to the memory. This register is also used for input-output communication.
D-register:	This register is an extension to the A-register in double precision or floating point operations. It may be connected to the A-register during double length shifts.
T-register:	Temporary register. In floating point instructions it is used to hold the exponent part.
L-register:	Link register. The return address after a subroutine jump is contained in this register.
X-register:	Index register. In connection with indirect addressing it causes post-indexing.
B-register:	Base register or second index register. In connection with indirect addressing it causes pre-indexing.
P-register:	Program counter, address of current instruction. This register is controlled automatically in the normal sequencing or branching mode. But it is also fully programcontrolled and its content may be transferred to or from other registers.

Besides from the R- and P-register all registers are fully programcontrolled and may be used for other purposes than those described here.

Two instructions, ROP and SKP, may specify a register whose content is always zero.

2.3.2 Control flip-flops.

Six control flip-flops are accessible by program.

These six flip-flops are:

C	: Carry flip-flop. The carry flip-flop is dynamic and affected by the instructions ADD, SUB, RADD, RSUB, COPY, AAA, AAT, AAX, AAB.
Q	: Dynamic overflow flip-flop. It is affected by the instructions ADD, SUB, RADD, RSUB, COPY, AAA, AAT, AAX, AAB.
O	: Static overflow flip-flop. This flip-flop remains set after an overflow condition until it is reset by program. It is affected by the instructions ADD, SUB, RADD, RSUB, AAA, AAT, AAX, AAB.

Z	: Floating point overflow flip-flop. This flip-flop is static and remains set until it is reset by program. The Z flip-flop may be internally connected to an interrupt level such that an error message routine may be triggered. It is affected by the instruction FDV, if division by zero is tried.
K	: One bit accumulator. This flip-flop is used in the BOP, bit operation, instruction to store temporary one-bit data.
M	: Multi shift link flip flop. This flip-flop is used as temporary storage for vacated bits in shift instructions in order to ease the shifting of multiple precision words.

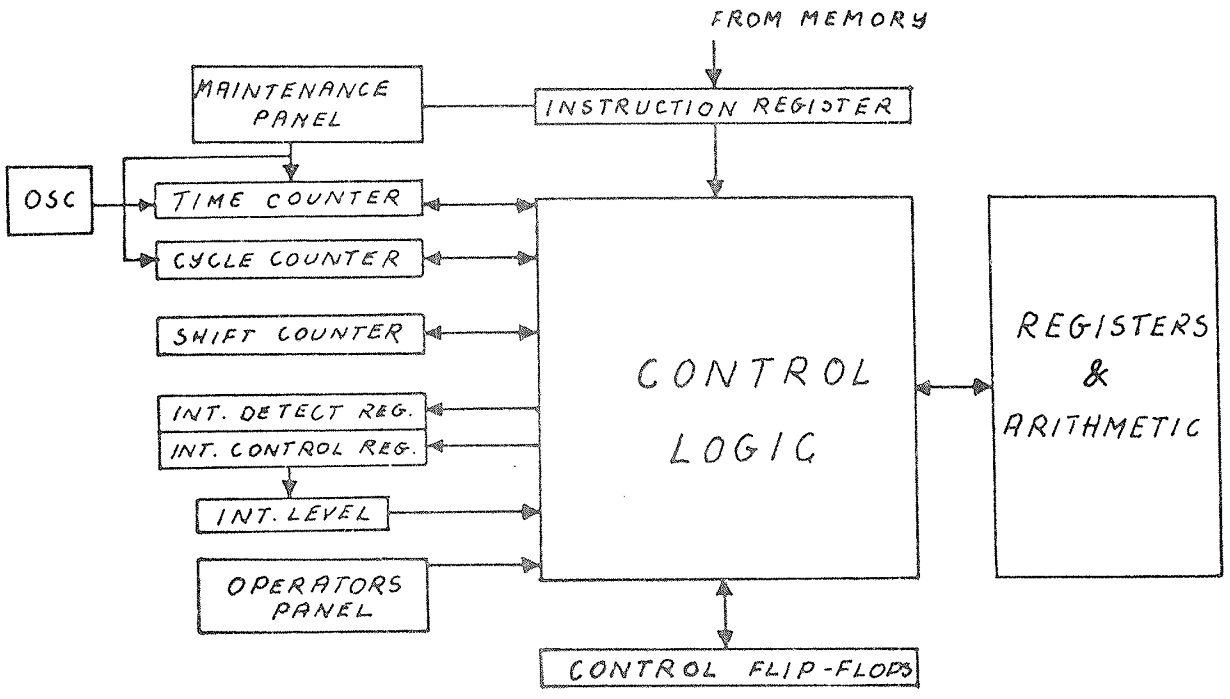
These six flip-flops are fully programcontrolled either by means of the BOP instruction of by the TRA or TRR sub-instructions where all flip-flops may be transferred to and from the A-register.

It is only the automatic affection in connection with carry and overflow that are described here.

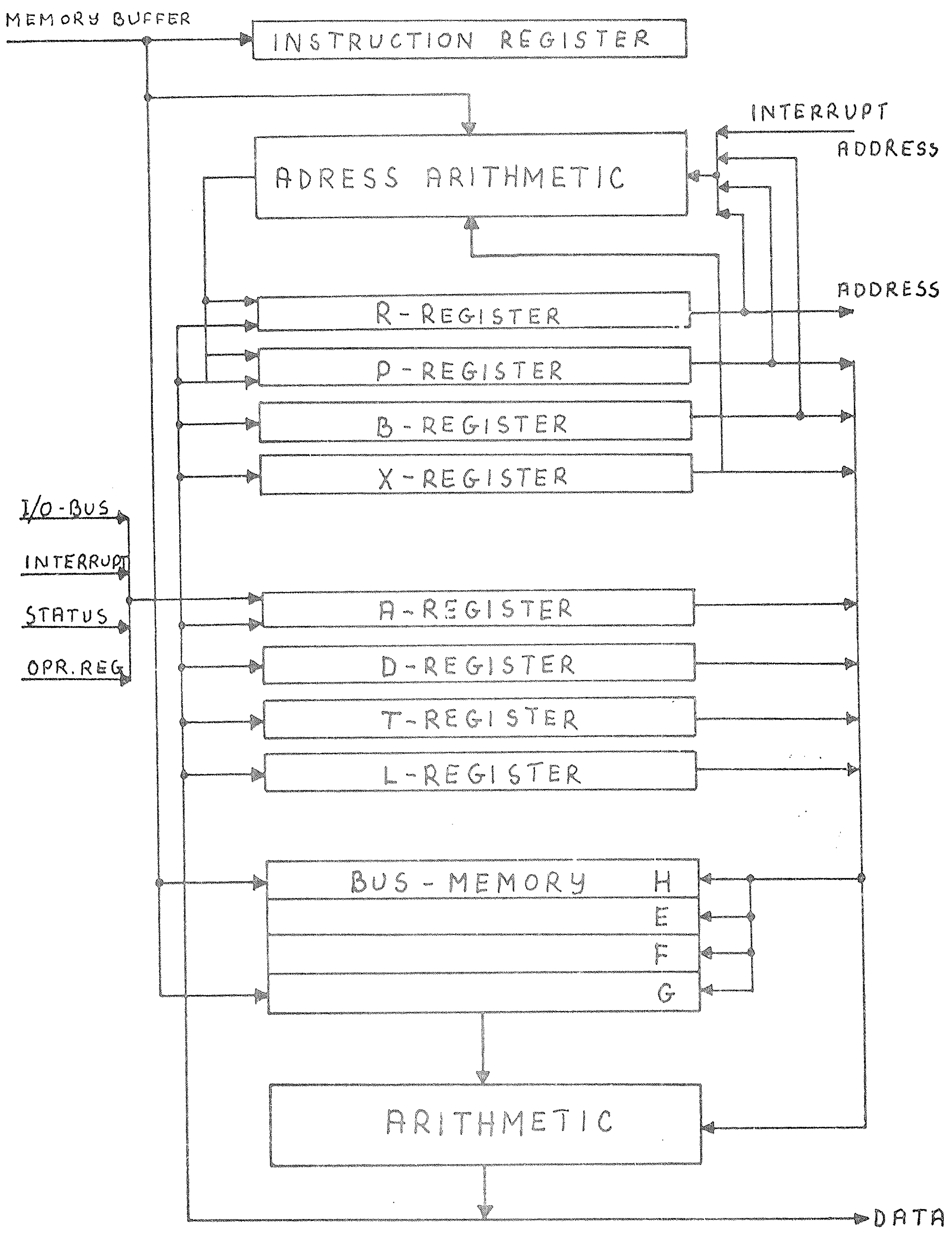
2.3.3 Arithmetic and control units.

Figure 1 shows a block diagram of the central processing unit. The address and index computations are performed in a special address arithmetic unit. All programmed arithmetic and logical operations are performed in a 16 bit high-speed arithmetic unit. Therefore all such operations may be performed on any of the registers.

The control unit contains the necessary logic circuitry to access data and instruction words, to modify instruction addresses, to perform arithmetic and logical operations and to control the interrupt system.



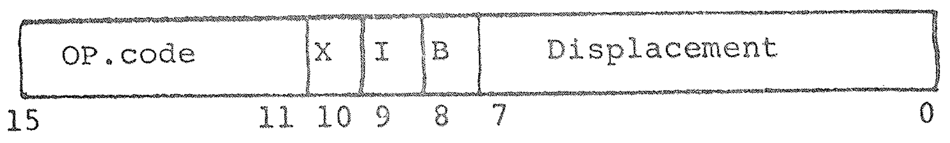
CENTRAL PROCESSING UNIT



GENERAL REGISTERS AND ARITHMETIC

2.4 Instruction and data word formats

2.4.1 Instruction word.



One instruction word always accupies one location, 16 bits, of core memory. The operation code occupies the five most significant bits (11 - 15), and specifies one of 32 instructions.

For memory reference instructions bits 1 - 10 are used to specify the address of the instruction. The instructions which do not have an address, use these bits to further specifications. Bits 8, 9 and 10, called B, I and X, are used to control the address computation.

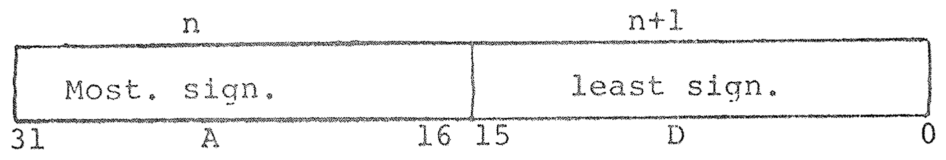
The displacement is an 8 bit signed number ranging from -128 to +127, using two's complement for negative numbers and sign extension.

2.4.2 Data word.

Three different types of data words exist:

a) Single length numbers: a 16 bit number which occupies one memory location. Representation of negative numbers are in 2's complement. Range as integers: $-32768 \leq |x| \leq 32767$.

b) Double length numbers: a 32 bit number which occupies two consecutive locations in memory, and where negative numbers also are in 2's complement.



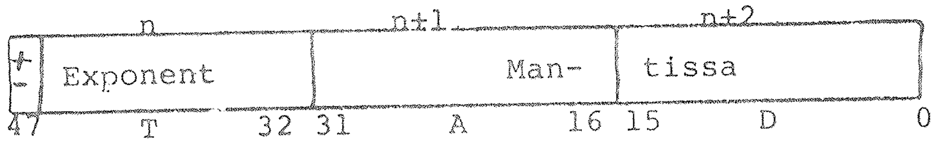
A double word is always referred to by the address of its most significant part. Normally a double word is transferred to the registers so that the most significant part is contained in the A-register and the least significant in the D-register. Range as integers: $-2\ 147\ 483\ 648 \leq |x| \leq 2\ 147\ 483\ 647$.

c) Floating point numbers: The data format of floating point words is 32 bits mantissa magnitude, one bit for the sign of the number and 15 bits for a signed exponent. The mantissa is always normalized, $0{,}5 \leq$ mantissa < 1, for all non zero numbers bit 31 equals one. The exponent base is 2, the exponent is biased with 2^{14}, so that a standardized floating zero contains zero in all 48 bits.

In core store one floating point data word occupies three 16 bit core locations, which are addressed by the address of the exponent part.

n	exponent and sign
n+1	most significant part of mantissa
n+2	least significant part of mantissa

In CPU registers bits 0 - 15 of the mantissa is in the D-register, bits 16 - 31 in the A-register and bits 32 - 47, exponent and sign, in the T-register. These three registers together are defined as the floating accumulator.



The accuracy is 32 bits or approximately 10 decimal digits, any integer up to 2^{32} has an exactly floating point representation.
The range is

$$2^{-16384} \cdot 0,5 \leq |x| < 2^{16383} \cdot 1 \text{ and zero}$$

or

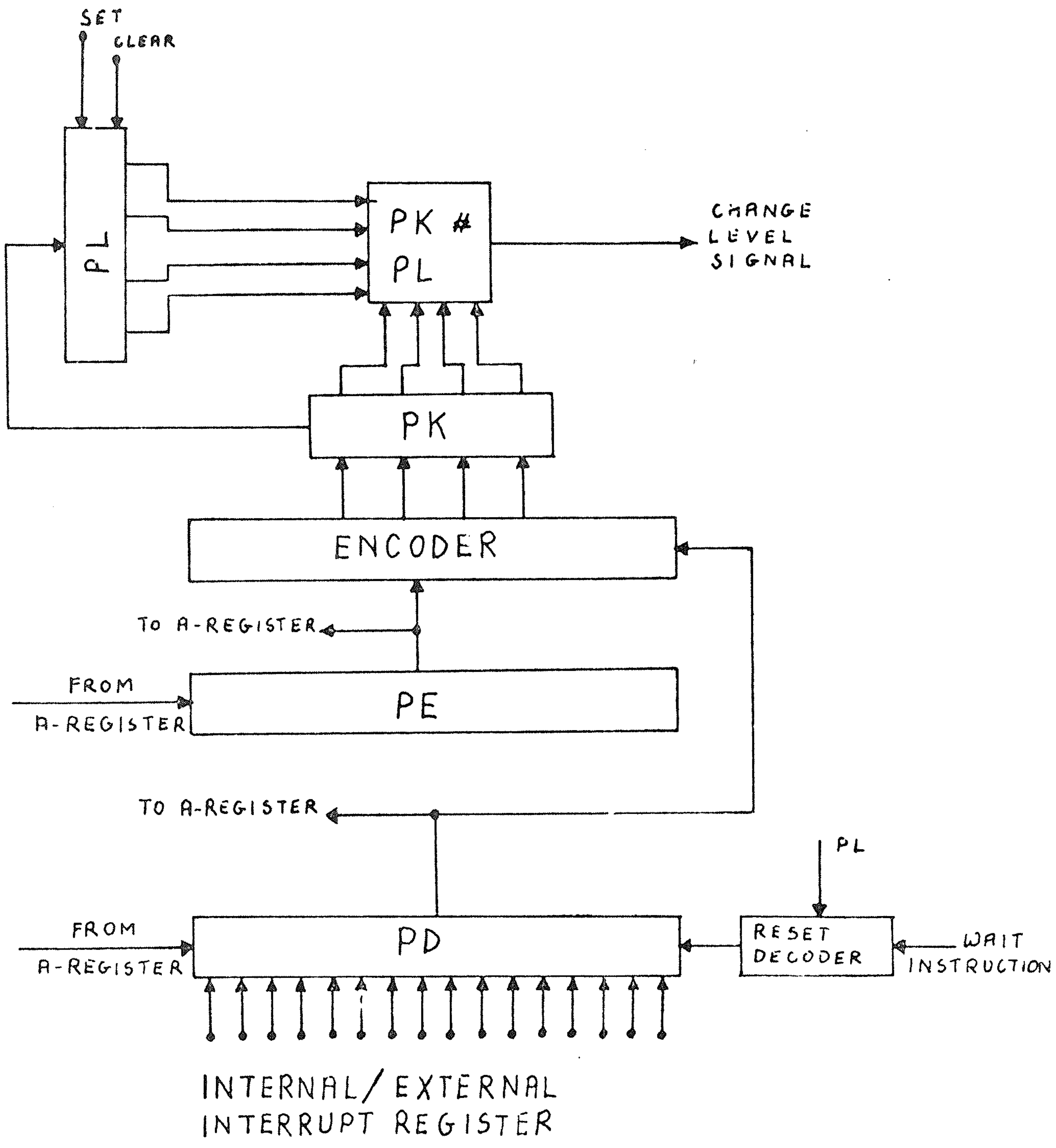
$$10^{-4920} < |x| < 10^{4920}$$

Any other data word format than those three described here may be programmed. These three data word formats have corresponding instructions which make these formats easy and natural to use. It is also rather easy to program data word formats using one bit data word (logical variables) and 8 bits data word (character byte).

2.5 Interrupt system

The NORD-1 computer has a priority interrupt system with 16 different priority levels. The interrupt system has been designed for real-time applications and multi-programming systems. The 16 different priority levels may be triggered either from external signals or from program. Some of the levels are also triggered by control signals from the central processor, for instance if the memory protection system is violated or if a floating point instruction causes overflow. External interrupt request signals may be grouped and connected to the same interrupt level, the priority between interrupt requests on the same level is then determined by program.

When the computer makes a transfer from one level to another the content of all seven central registers and the setting of the status flip-flops are automatically saved in locations in core memory which are associated with the level which was interrupted. Before the new level is entered the seven central registers and the status flip flops are loaded from locations in core memory which again are associated with the level now to be entered. This automatic saving and unsaving of all the programmable registers and flip-flops make multiprogramming extremely easy, and the programs on the different levels may be completely independent of each other.



PRIORITY INTERRUPT SYSTEM

The interrupt system is controlled from two 16 bit registers where each level is controlled from one bit in each of the two registers.

The two registers are:

PID	Priority interrupt detect
PIE	Priority interrupt enable

Both registers are programcontrolled, see section 3.83, the setting of individual bits in the PID register is also for some predetermined levels controlled by wired in interrupt requests.

The PID register is used to detect and hold an interrupt request. Each individual bit may be set either by internal or external interrupt requests or by program. Usually individual bits in PID are automatically reset when the interrupt requests have been processed. A WAIT instruction, "give up priority", causes the bit in the PID register which corresponds to the level now operating to be reset.

The PIE register is used to enable the different levels. Any interrupt level can only have its corresponding program operating if the corresponding bit in PIE is a one. The PIE register is controlled only by program. Because of the automatic saving and unsaving of all register and status information when changing from one level to another, it is possible to disenable an interrupt level for a while, and enable it afterwards regardless which levels have been operating in the meantime.

The interrupt levels are numbered from 0 until 15, where level number 15 has the highest priority. Associated with each level is a corresponding program. At any time the program with the highest priority is running. The highest priority is determined as the highest level which has a one in the corresponding bits both in the PID and the PIE register.

A change from a lower to a higher priority level is usually caused by an interrupt request(internal, external or programmed request). A change from a higher to a lower priority level usually takes place when the higher level program gives up its priority (the WAIT instruction causes the corresponding bit in PID to be reset).

In core memory each level is associated with one location called level-pointer. The level-pointer gives the address of the corresponding level-head. Each level-head consists of 8 consecutive locations which may then be located anywhere in core memory. The level-head is used to hold the content of the seven central registers and the status information when the program on the corresponding level is not running.

Whenever a program is interrupted the register and status are saved in the level-head corresponding to this program, then a new level-pointer is chosen and the registers and status are loaded from the level-head which corresponds to the new level-pointer. The total time involved when changing from one level to another is 32 memory cycles.

2.6 Memory protection system

The NORD-1 protection system provides operation protection for input/output instructions, interrupt control instructions, jump instructions and memory write instructions. Input/output and interrupt control instructions can be executed from protected area only, and memory instructions in unprotected area may write in unprotected area only. Jump from unprotected to protected area is not permitted. Any instructions violating the protection rules will produce interrupt on a specified level. In machines without a priority interrupt system the illegal instruction will be equal to a WAIT instruction.

The standard protection system divides the core memory into two equal parts, one protected area (upper half) and one unprotected area (lower half).

An optional expansion of the protection system which devides the memory into 16 equal parts is also provided. The protection of individual blocks of core memory is controlled by a 16 bit register. With 4096 words in the memory one block is a region of 256 consecutive locations, and with a 16K memory one block is a region of 1024 consecutive locations. A protection bit of 0 designates an unprotected memory block and a protection bit of one designates a protected block.

The protection register can be loaded from the A-register with the instructions MCL, masked clear, or MST, masked set, instructions which of course are privileged (subinstructions of RTR).

Operation of the protection system is under control of the PROTECT switch on the operators panel. If the protection system is operative the following rules apply:

1. The privileged instructions IOT, TRR, MCL, MST, WAIT, ION and IOF can be executed only if they are accessed from protected memory. If a privileged instruction is accessed from unprotected memory, the instruction is not executed; instead, the protection violation interrupt level is triggered.

2. If a jump-instruction or ROP DP or BOP DP is accessed from unprotected memory and the effective new address is in protected memory, the instruction is not executed; instead, the protection violation interrupt level is triggered.

3. The instructions STZ, STT, STA, STX, STD, STF and MIN can be used to alter protected memory only if the instruction is accessed from protected memory. If an attempt is made to alter protected memory with an instruction accessed from unprotected memory, the operation is not performed; instead the protection violation interrupt level is triggered.

3. INSTRUCTION REPERTOIRE

3.1 Memory reference instructions

In the instruction word, 11 bits are used to specify the address, 3 address mode bits, and an 8 bit signed displacement using two's complement for negative numbers and sign extension.

15 11	10	9	8	7 0
OP.Code	X	I	B	Displacement

NORD-1 uses a relative addressing system, which means that the address is specified relative to the content of the Program counter, or relative to the content of the B- or X-register.

Bits 8, 9, 10 called B, I, X, are used to specify the address mode.

If B, I, and X all are zero, the normal relative addressing mode is specified, the effective address equals the content of the Program counter plus the displacement.

The displacement may consist of a number ranging from - 128 to + 127, therefore this addressing mode gives a dynamic range for directly addressing 128 locations backwards and 127 locations forwards.

Otherwise the B, I and X bits are decoded as follows:

B=0 means the address is relative to the Program counter (address of current instruction).

If B=0, X=1 and I=0, this is decoded in a special way giving the address only relative to the X-register.

B=1 means the address is relative to the content of the B-register, also called preindexing. The indexing by B takes place before a possible indirect addressing.

I=1 specifies indirect addressing.

There is only one level of indirect addressing.

X=1 specifies address modification by X, also called post indexing, which takes place after the indirect addressing.

The address computation is summarized in Table 1. The symbols used are defined as follows:

X	Bit 10 of the instruction
I	Bit 9 of the instruction
B	Bit 8 of the instruction

D	Content of bits 0 - 7 of the instruction (displacement)
(X)	Content of the X-register
(B)	Content of the B-register
(P)	Content of the P-register
( )	Means content of the register or word

B I X	Mneumonic	Effective address
0 0 0		(P) $\pm$ D
0 0 1	,X	(X) $\pm$ D
0 1 0	I	((P) $\pm$ D)
0 1 1	I,X	((P) $\pm$ D) + (X)
1 0 0	,B	(B) $\pm$ D
1 0 1	,B ,X	(B) $\pm$ D + (X)
1 1 0	,B I	((B) $\pm$ D)
1 1 1	,B I,X	((B) $\pm$ D) + (X)

Table 1 Addressing modes

The instruction CJP, conditional jump, uses B, I and X to specify the jump condition, see section 3.1.4.

In the following a short description of each memory reference instruction is given. The same mneumonics as used in the assembly language, are specified. For each instruction the registers and indicators that can be affected by the instruction are listed. The execution time of each instruction is specified in memory cycles (mc).

If indirect addressing is specified, an additional memory cycle is required.

The following abbreviations are used in the descriptions:

A	A-register
D	D-register
P	Program counter
X	X-register
T	T-register
L	L-register
B	B-register
EL	Effective location
EW	Effective word, or (EL)
C	Carry indicator
Q	Dynamic overflow indicator
O	Static overflow indicator
Z	Floating point overflow indicator
mc	memory cycle
μs	micro-second

3.1.1 Store instructions

STZ Store zero.

The effective location is cleared.
Affected: (EL) Time 2 mc

STA Store A-register.

The content of the A-register is stored in the effective location.
Affected: (EL) Time 2 mc

STT Store T-register.

The content of the T-register is stored in the effective location.
Affected: (EL) Time 2 mc

STX Store X-register.

The content of the X-register is stored in the effective location. The address of this instruction may be modified by the content of the X-register.
Affected: (EL) Time 2 mc

MIN Increment memory and skip if zero.

Effective word is read and incremented by one and then restored in the effective location. If the result becomes zero, the next instruction is skipped.
Affected: (EL), (P) Time 3 mc

3.1.2 Load instructions

LDA Load A-register.

The effective word is loaded into the A-register.
Affected: (A) Time 2 mc

LDT Load T-register.

The effective word is loaded into the T-register.
Affected: (T) Time 2 mc

LDX Load X-register.

The effective word is loaded into the X-register. The address of this instruction may be modified by the previous content of the X-register.
Affected: (X) Time 2 mc

3.1.3 Arithmetical and logical instructions

ADD Add to A-register.

The effective word is added to the A-register with the result in the A-register. The carry indicator is set to 1 if a carry occurs from the sign bit position of the adder, otherwise the carry indicator is reset to 0. If the signs of the two operands are equal but the sign of the result is different, overflow has occurred, and both the dynamic- and static overflow indicators are set to one. If the condition for overflow does not exist, the dynamic overflow indicator is reset to 0, while the static overflow indicator is left unchanged. The static overflow indicator is automatically reset when sensed by a skip instruction (see BOP).
Affected: (A), C, O, Q Time 2 mc

SUB Subtract from A-register.

The two's complement of the effective word is formed and added to the content of the A-register with the result in the A-register. The same rules as for ADD apply for the setting of the overflow and carry indicators.
Affected: (A), C, O, Q Time 2 mc

AND Logical and.

The logical product of the effective word and the content of the A-register is formed, with the result in the A register. The logical product contains a 1 in each bit position for which there is a corresponding 1 in both the A-register and the effective word, otherwise the bit position contains a zero.
Affected: (A) Time 2 mc

ORA Logical inclusive or.

Logic inclusive or is formed between the effective word and the content of the A-register, with the result in the A-register. Logic inclusive or contains a zero in each bit position for which there is a corresponding zero in both the A-register and the effective word, otherwise the bit position contains a 1.
Affected: (A) Time 2 mc

MPY Multiply integer.

The effective word and the A-register is multiplied and the result is placed in the A-register. Both numbers are regarded as signed integers and the result as a 16 bit signed integer.
Affected: (A) Time 2 mc + 6 μs

3.1.4 Sequencing instructions

JMP Jump

The effective address is loaded into the program counter, and the next instruction is taken from the effective address of the JMP instruction.
Affected: (P) Time 1 mc

JPL Transfer P to L and jump.

The content of the program counter is transferred to the L-register, the effective address is loaded into the program counter, and the next instruction is taken from the effective address of the JPL instruction.
Affected: (P), (L) Time 1 mc

CJP Conditional jump

Bits B, I and X are used to specify one of 8 jump conditions. If specified condition becomes true the displacement is added to the program counter and a jump relative current location takes place. The range is 128 locations backwards and 127 locations forwards. If specified condition is false no jump takes place.
Affected: (P) Time 1 mc

The eight jump conditions are:

JAP	Jump if A-register positive, A bit 15 = 0.
JAN	Jump if A-register negative, A bit 15 = 1.
JAZ	Jump if A-register zero.
JAF	Jump if A-register filled (not zero).
JXN	Jump if X negative, X bit 15 = 1.
JXZ	Jump if X zero.
JPC	Jump if X positive and count. X is incremented by one, and if X bit 15 equals zero after the incrementations, the jump takes place.
JNC	Jump if X negative and count. X is incremented by one, if then X bit 15 equals one, the jump takes place.

A conditional jump instruction must be specified by means of the 8 mneumonics listed above. It is illegal to specify CJP followed by any combination of ,B I and ,X.

3.1.5 Double wordlength instructions

STD Store doubleword.

The content of the A-register is stored into the effective location, and the content of the D-register is stored into

the effective location plus one.
Affected: (EL), (EL+1) Time 3 mc

LDD Load doubleword.

The content of the effective location is loaded into the A-register, and the content of the effective location plus one is loaded into the D-register.
Affected: (A), (D) Time 3 mc

3.1.6 Floating point instructions

A floating point word consists of 48 bits. The floating accumulator consists of the three registers, T, A, D where the exponent is contained in the T-register, the most significant part of the mantissa in the A-register and the least significant part of the mantissa in the D-register.

STF Store floating accumulator.

The content of the floating accumulator is stored in three memory locations, starting with exponent part in effective location.
Affected: (EL), (EL+1), (EL+2) Time 4 mc

LDF Load floating accumulator.

The content of the effective location and the two following locations are loaded into the floating accumulator.
Affected: (T), (A), (D) Time 4 mc

FAD Add to floating accumulator.

The content of the effective location and the two following locations are added to the floating accumulator.
Affected: (T), (A), (D) Time minimum 4mc+ 6μs
Time maximum 4mc+30μs

FSB Subtract from floating accumulator.

The content of the effective location and the two following locations are subtracted from the floating accumulator.
Affected: (T), (A), (D) Time minimum 4mc+ 6μs
Time maximum 4mc+30μs

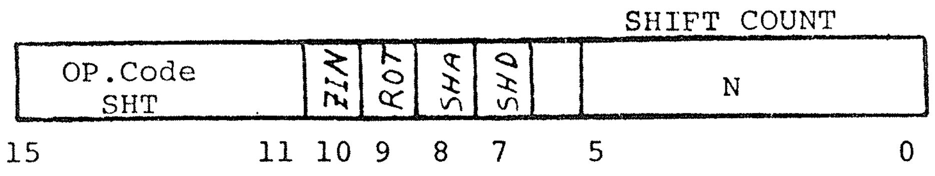
FMU Multiply floating accumulator.

The content of the floating accumulator is multiplied with the number in the effective floating word locations. Result in floating accumulator.
Affected: (T), (A), (D) Time 4 mc + 28 μs

FDV Divide floating accumulator.

The content of the floating accumulator is divided by the number in the effective floating word locations. Result in floating accumulator. If division by zero is tried the floating point overflow indicator is set to one. This indicator is static and remains set until sensed by a skip instruction (see BOP).
Affected: (T), (A), (D), Z Time 4 mc + 28 µs

3.2 Shift instructions



SHT Shift.

Single shifts	Time lmc + 0,4x Nµs
Double shifts	Time lmc + 0,8x Nµs

The shift instruction uses the address bits to specify the type and the number of shifts to be performed.

N is a signed number which specifies shift direction and number of shifts.

N > 0,	bit 5 = 0:	Shift left
N < 0,	bit 5 = 1:	Shift right

Maximum number of shifts is 31 left shifts and 32 right shifts.

Bit 9 and 10 specify the type of shift operation. The decoding is as follows:

Bit 10	Bit 9	Mneumonic	
0	0		Arithmetic shift. During right shifts the signbit (bit 15) is extended during the shifting, in left shifts zeros are fed into vacated bit positions.
0	1	ROT	Rotational shift. In single register shifts bit 0 is connected to bit 15, in double shifts bit 0 of the D-register is connected to bit 15 of the A-register. Only left rotate shift is possible.
1	0	ZIN	Zero end input.
1	1	LIN	Link end input. Every shift instruction causes the last bit which is vacated to be contained in the M, multi shift

link flip-flop, this may then be used as end input for the next shift instruction.

Bit 7 and 8 specify the register(s) to be shifted. The decoding is as follows

Bit 8	Bit 7	Mneumonic	
0	0	SHT	Shift the T-register
0	1	SHD	Shift the D-register
1	0	SHA	Shift the A-register
1	1	SAD	Shift the A- and D-registers connected. Bit 0 of the A-register is connected to bit 15 of the D-register.

Only the A, T and D-registers may be shifted, if any other register is to be shifted, its content must first be placed in the A, T or D-register.

If no shift direction is specified, left shift is assumed.

The number of shifts is interpreted by the assembler as an octal number.

A right shift may be specified either by the correct 6 bits negative shift count or by writing the mneumonic code SHR followed by the positive number of right shifts. A shift instruction to shift the accumulator 3 positions to the right, may be specified by one of the following identical instructions.

```
SHA      75
SHA     100-3
SHA     SHR    3
```

In a right shift nothing should be written between the SHR mneumonic and the number of right shifts (a space to distinguish between SHR and the number is necessary). SHR must be the last mneumonic used in the instruction.

Some examples of correctly specified shift instructions:

```
SAD    10
```

Shift the A- and D-register connected 8 positions (octal 10) left.

```
SHT    ROT    6
```

Rotate the T-register 6 places to the left.

```
SAD    ROT    20
```

Shift the connected A- and D-register 16 positions to the left. Rotate shift is specified, which in this case will cause the content of the A- and D-register to be exchanged. The same effect may be obtained by means of a SWAP SA DD instruction (see ROP instruction).

```
SHD    ZIN    SHR    2
```

Shift the D-register two places to the right, zeros are fed into the right end during the shifting. Bit 15 and 14 in the D-register will become zero.

3.3 Arithmetic register operations

OP.CODE ROP	RAD	C	I	CM1	CLD	S	D
15 11	10	9	8	7	6	5 3	2 0

ROP Register operation. Time 1 mc

The ROP instruction specifies operations between any two general registers.

The instruction decodes bit 0 - 10 as follows:

Bit 0 - 2 specifies one out of 7 registers to be the destination register. The destination register will be loaded with the result of the ROP instruction.

D = 0 is a no operation instruction.

Bit 3 - 5 specifies one out of 8 registers which contains the value to be used as the source register operand.

S = 0 produces a source value equals zero.

CLD = 1: Clear destination register before operation. If the source and the destination register is the same, the register as source is not cleared.

CM1 = 1: Use complement (one's complement) of source register as operand. The source register remains unchanged.

Bit 8 and 9 are decoded in two different ways, dependent on the RAD-bit being zero or a one.

RAD = 1: Add source to destination.

When RAD = 1, bit C and I are decoded as follows:

C = 1, I = 0: Also add old carry to destination, ADC
C = 0, I = 1: Also add 1 to destination, AD1

It is not possible to both add previous carry and to add 1 in the same ROP instruction. (If it is tried, only 1 will be added independent of the status of the carry flip-flop).

RAD = 0: Binary register operations.

The C and I bits are decoded as follows:

C,I=0,0	Register swap, destination and source exchanged,	SWAP
0,1	Logical and ,	RAND
1,0	Logical exclusive or,	REXO
1,1	Logical inclusive or,	RORA

If RAD = 1, the overflow and carry indicators are set after the same rules as applied for ADD, if RAD = 0, the overflow and carry indicators remain unchanged.

The source registers are specified as follows:

SD	D-register	as source
SP	Program counter	as source
SB	B-register	as source
SL	L-register	as source
SA	A-register	as source
ST	T-register	as source
SX	X-register	as source

If no source register is specified, zero will be taken as source register.

The destination registers are specified as follows:

DD	D-register	as destination
DP	Program counter	as destination
DB	B-register	as destination
DL	L-register	as destination
DA	A-register	as destination
DT	T-register	as destination
DX	X-register	as destination

The following groups of ROP mneumonics are mutually exclusive, i.e. only one may be used in a ROP instruction.

(SD, SP, SB, SL, SA, ST, SX)

Only one source register must be specified.

(DD, DP, DB, DL, DA, DT, DX)

Only one destination register must be specified.

(ADC, AD1)

Both 1 and old carry can not be added in the same instruction

(RADD, RSUB, SWAP, RAND, REXO, RORA, COPY).

Only one type of operation must be specified.

(ADC, AD1, SWAP, RAND, REXO, RORA)

Add 1 or add carry may not be used together with the binary register operations.

The recommended way to specify ROP instructions is to use the following mneumonics which will be correctly translated by the assembly language.

RADD,	$D + S \rightarrow D$	Register addition
RSUB,	$D - S \rightarrow D$	Register subtraction
RAND,	$D \cdot S \rightarrow D$	Register logical and
RORA,	$D + S \rightarrow D$	Register logical or
REXO,	$D \oplus S \rightarrow D$	Register logical exclusive or
SWAP,	$D \rightarrow S$ $S \rightarrow D$	Register exchange
COPY,	$S \rightarrow D$	Register transfer

Note that the ROP mneumonic is included in the above mentioned mneumonics.

The assembly language will also permit use of the following combined mneumonics.

CM2 =	CM1	AD1		Two's complement
EXIT =	COPY	SL	DP	Return from subroutine
RCLR =	COPY			Register clear
RINC =	COPY	AD1		Register increment
RDCR =	COPY	CM1		Register decrement

The mneumonics RCLR, RINC and RDCR should be followed only by the destination register specification.

Some examples of use of the ROP instruction.

```
RADD   SA   DX
```

The content of the A-register and X-register is added, with the result in the X-register.

```
COPY   CM2   SA   DA
```

Complement (2' complement) the A-register.

```
RSUB   ST   DB
```

The content of the T-register is subtracted from the content of the B-register, with the result in the B-register.

```
RINC   DX
```

The X-register is incremented by one.

```
RDCR   DL
```

The L-register is decremented by one. (One's complement of zero equals -1 in two's complement).

```
RCLR   DT
```

T-register is cleared.

```
RCLR   AD1   DX
```

X-register is set equal to one.

```
RCLR   CM1   DB
```

B-register is set equal to minus one.

```
          COPY    SX     DT
```

The content of the X-register is copied into the T-register.

```
          SWAP    SA     DA
```

The content of the A-register and the D-register is exchanged.

```
          RAND    SL     DX
```

Logical and is formed between the content of the L-register and the X-register, with the result in the X-register.

Some short programs using ROP instructions.

```
          COPY    CM2    SD      DD
          COPY    CM1    ADC     SA      DA
```

The two's complement of the 32 bit doubleword in A and D is formed.

```
          LDD     PER
          SWAP    SA     DD
          ADD     OLA+1
          SWAP    SA     DD
          COPY    ADC    SA      DA
          ADD     OLA
```

The two double wordlength numbers PER and OLA are added together, with the result in the A- and D-registers.

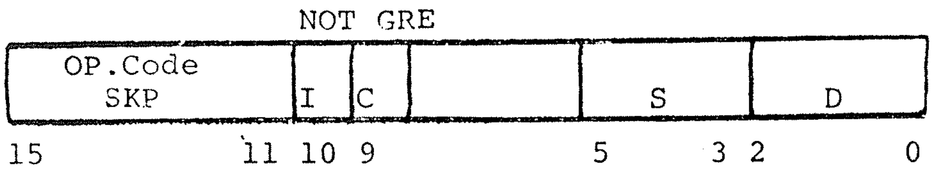
```
                  JPL    SUBR
          ERR,    WAIT
          NORM,
          -------------------------

          SUBR,   LDA    OLA
                  SUB    PER
                  SKP    IF      DA      EQL     0
                 EXIT                          % ERROR EXIT
                 EXIT    AD1                   % NORMAL EXIT
```

Subroutine jump, and return from subroutine to main program.

The JPL instruction will place the address of the WAIT instruction into the L-register. (When JPL is executed the Program Counter points to the address after this instruction.)

The subroutine SUBR has two exits, one to the location immediately following the jump (EXIT), which in this case is an error exit, and one to the location two addresses after the jump.

3.4 Skip instructions



SKP Skip next instruction if specified condition is true. Time 1 mc

The skip instruction makes it possible to test the relationship between any two general registers.

The decoding is as follows:

I = 1:	Invert skip condition	, NOT
C = 0:	Test condition =	, EQL
C = 1:	Test condition $\geq$	, GRE

The S and D field specifies the two registers to be compared and tested.

The arithmetic expression D - S is tested, where D stands for one out of 7 general registers, and S is one out of the 7 general registers or zero.

The D and S registers are specified using the same mneumonics as the ROP instruction, see section 3.3.
If S = 0, the destination register is compared against zero. Only one destination register may be compared against only one source register in the same SKP instruction.

If D = 0, the instruction is a no operation.

If the skip condition is false, the instruction is a no operation.

Because of the great flexibility of the SKP instruction, it may be found difficult to use this flexibility. Therefore, the programmer is advised to use the following format when specifying a SKP instruction.

a) The comparison should be specified as follows:

=	EQL	(Equal,	C = 0,	I = 0)
$\neq$	UEQ	(Not equal,	C = 0,	I = 1)
$\geq$	GRE	(Greater or equal	C = 1,	I = 0)
$<$	LST	(Less,	C = 1,	I = 1)

b) The destination (D) register should be specified before the source (S) register.

c) The mneumonic IF and the number 0, which both have the value zero, may be used freely to obtain easy readability.

SKP	IF	DL	EQL	0	Skip if $L = 0$
SKP	IF	DT	LST	0	Skip if $T < 0$
SKP	IF	DD	GRE	SA	Skip if $D \geq A$
SKP	IF	DB	LST	SX	Skip if $B < X$

3.5 Argument instructions

Time 1 mc

Bits B, I and X are used to specify one of 8 argument instructions. All these instructions use the displacement part of the instruction as a signed number ranging from - 128 until 127. This number is either placed in or added to the specified register.

The eight argument instructions are

SAA	Set argument to A-register
AAA	Add argument to A-register
SAX	Set argument to X-register
AAX	Add argument to X-register
SAT	Set argument to T-register
AAT	Add argument to T-register
SAB	Set argument to B-register
AAB	Add argument to B-register

An argument instruction should be specified by means of one of the eight mneumonics listed above. It is illegal to specify ARG followed by any combination of ,B I and X.

Examples of argument instructions.

```
SAT      13
```

Set the content of the T-register equal to 13 (octal). Bits 8 - 15 will become zero.

```
SAB      - 26
```

Set the content of the B-register equal to - 26 (octal). Bits 8 - 15 will become one, sign extension.

```
AAX      3
```

Add 3 to the content of the X-register. The addition is modulo 2^{15}.

```
AAA      - 6
```

Subtract 6 from the content of the A-register (modulo 2^{15}).

```
SAA      240
```

The content of the A-register will be 177640_8 after the execution of this instruction (sign extension).

```
SAA    ## A
IOT    SKA    ACT    PNT
JMP    * - 1
```

Program to print the letter A.

In an add argument instruction the carry and overflow indicators are set according to the same rules as applied for the ADD-instruction, see section 3.1.3.

3.6 Bit operation instructions

BOP	Sub.instr.	B	D
15 11	10 7	6 3	2 0

BOP Bit operation Time 1 mc

The BOP instruction specifies operations on a single bit in one of the seven general registers, if D ≠ 0.

D = 0 together with the number in B specify operations on one of the program controllable status or control flip-flops (carry and overflow indicators).

For register operations B defines the bit in the register to be manipulated, B = 0 is the rightmost bit and B = 170 (octal) is the leftmost bit. The register is specified by means of the same mneumonics as used in the ROP and SKP instructions, see section 3.3.

The BOP instruction also uses a one bit "accumulator" register, K, to hold temporary results.

16 different subinstructions are available in the BOP instruction.

In the following description B means the bit specified by D (register) and B (bit-number).

3.6.1 Skip instructions.

Four subinstructions are available to test the setting of the specified bit.

BSKP	ZRO	Skip next instruction if B = 0
BSKP	ONE	Skip next instruction if B = 1
BSKP	BCM	Skip next instruction if B = K
BSKP	BAC	Skip next instruction if B^{o} = K

3.6.2 Setting of bit instruction.

Four subinstructions are available to set the specified bit.

BSET	ZRO	0 → B
BSET	ONE	1 → B
BSET	BCM	B → B, complement bit
BSET	BAC	K^{o} → B

3.6.3 Instructions using the one bit accumulator.

Eight subinstructions are available to specify operations between the specified bit and the K, one bit register.

BSTA	$K \rightarrow B\ ,\ 0 \rightarrow K$	Store and clear
BSTC	$K_o \rightarrow B\ ,\ 1 \rightarrow K$	Store complement and set
BLDA	$B \rightarrow K\ ,$	Load
BLDC	$B_o \rightarrow K\ ,$	Load complement
BANC	$B_o \cdot K \rightarrow K$	Logic and complement
BORC	$B_o + K \rightarrow K$	Logic or complement
BAND	$(B \cdot K) \rightarrow K$	Logic and
BORA	$(B+K) \rightarrow K$	Logic or

When the carry and overflow indicators are tested by means of the BSKP subinstruction, the tested indicator is automatically reset.

Some examples of correctly specified bit operation instructions.

```
BSKP      ONE    CRY
```

Skip next instruction if the carry indicator is set, the carry indicator is automatically reset.

```
BSET      ZRO    SO
```

Reset the static overflow indicator.

```
BSET      BCM    170    DT
```

Complement the sign bit in the T-register (complementation of a floating point number).

```
BSET      ZRO    170    DT
```

Set the sign bit in the T-register to zero (absolute value of a floating point number).

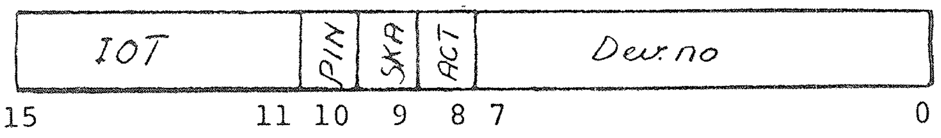
```
BSET      ONE     60    DX
```

Set bit 6 in X-register to one.

```
BLDA      160    DA
BSET      BAC    160    DX
```

Copy A-register bit 16 into X-register bit 16.

3.7 Input-output control



IOT	Operate specified device according to function.	Time 1 mc + 0,4-11 µs

The IOT instruction is used both for starting an output device, in which case the data word is taken from the A-register, or for reading a data word from an input device into the A-register. Other functions again may read or change the status of the device only.

The input-output devices are grouped together and as much as 64 different devices may be gathered in one group. Maximum 4 groups of input-output devices may be connected to one CPU. Each group is connected to the CPU by means of a bus system with three cables, a data-input cable a data-output cable and a control-information cable. These cables connect all devices in the same group. Each group has two interrupt request lines connected to two different interrupt levels. Each device may have its interrupt request signal connected to one of these levels.

The three function bits (8 - 10) usually have the following meaning:

Bit 8:	ACT	Activate the specified device.
Bit 9:	SKA	Skip if start acceptable. If the device accepts this input-output instruction (i.e. the device is ready), the next instruction is skipped.
Bit 10:	PIN	Prepare interrupt. Turn on the interrupt system of the specified device.

The three function bits, ACT, SKA and PIN may in the same IOT instruction be given any possible combination.

If these function bits are all zero, this is interpreted as a different instruction:

SNI	Skip if not interrupt. If the specified device has not transmitted an interrupt request the next instruction is skipped, otherwise the interrupt system of this device is disabled.

Example of use of input-output instructions.

A programmed wait-loop until the device becomes ready will normally look like:

```
IOT   SKA    DVN   % DVN = DEVICE NUMBER
JMP   x-1
```

To print the content of bit 0 - 7 of the A-register on the Teletype paper tape and/or punch:

```
IOT   SKA   ACT   PNT
JMP   *-1
```

To read one character from the on-line Teletype into the A-register bit 0 - 7, bit 8 - 15 will be cleared:

```
IOT   SKA   ACT   RKE
JMP   *-1
```

To program a scanning of several input devices operated in parallel and read the information in the random order it is given (for instance if several Teletypes are connected to the same computer) the following type of program will do:

```
IOT   SKA   DV1
JMP   *2
JMP   RDV1        % JUMP TO ROUTINE FOR
                    READING DEVICE 1
IOT   SKA   DV2
JMP   *2
JMP   RDV2
IOT   SKA   DV3
JMP   *2
JMP   RDV3
----
----
```

A program to recognize an input-output interrupt may look like:

```
IOT   SNI   DV1
JMP   SDV1        % ROUTINE TO SERVICE DEVICE 1
IOT   SNI   DV2
JMP   SDV2
IOT   SNI   DV3
JMP   SDV3
----
----
```

3.8 Miscellaneous instructions

There are some instructions that do not require memory addresses. Some of these instructions are grouped together in the WBT instruction, where bits 0 - 10 give further specifications to this instruction.

3.8.1 Floating point convertion.

Two subinstructions are available. A single precision fixed point number may be converted to a standard form floating point number. A floating point number may be converted to a fixed point single precision number. For both instructions the scaling factor is specified in the displacement part of the instruction. The range of the scaling factor is from -128 until +127 which gives a convertion range from approximately 10^{-39} to 10^{39}.

The two subinstructions are

NLZ Normalize

Convert the number in the A-register to a standard form floating number in floating accumulator, using the displacement of the NLZ instruction as a scaling factor. For integers the scaling factor should be +16, a greater scaling factor will cause a greater floating point number. Because of the single precision fixed point number, the D-register will be cleared.
Affected: (T), (A), (D) Time 1 mc +(0,4-6)μs

DNZ Denormalize

Convert the floating number in the floating accumulator to a single precision fixed point number in the A-register, using the displacement of the DNZ instruction as a scaling factor. When converting to integers the scaling factor should be -16, a greater scaling factor will cause the fixed point number to be greater. The T- and D-registers are not affected by the DNZ instruction.
Affected: (A) Time 1 mc +(0,4-6)μs

If the convertion should be to or from double precision fixed point, special subroutines are available for this purpose.

3.8.2 Transfer to A-register.

The subinstruction TRA, transfer to A-register, is used for reading those registers which cannot be reached by means of the ROP or IOT instructions. The following registers may be read by means of the TRA subinstruction.

OPR	Operator panel register, setting of switches on the operators panel.
STS	Status word, it consists of the six programmable status flip-flops, carry indicator, static overflow indicator, floating point overflow indicator, K one bit accumulator, dynamic overflow indicator, multi shift link flip-flop.
PID	Priority interrupt detect register.
PIE	Priority interrupt enable register.

The TRA subinstruction should be specified by TRA followed by one of the mneumonics listed above.

3.8.3 Transfer from A-register.

Those registers which cannot be reached by the ROP or IOT instructions can be set by three subinstructions in the WBT group.

The transfer from the A-register may be either an ordinary transfer of all 16 bits or a selective set of zeros or ones depending on the content of the A-register.

The three subinstructions are

TRR	Transfer to register.
MCL	Masked clear, for each bit which is a one in the A-register the corresponding bit in the specified register will be reset.
MST	Masked set, for each bit which is a one in the A-register the corresponding bit in the specified register will be set.

The STS, status register, may only be set by means of TRR subinstruction.

The PID and PIE, priority interrupt detect and enable, registers may be set or reset selectively by means of the MCL and MST subinstructions.

3.8.4 Control of interrupt system.

The priority interrupt system may be turned on or off by means of the subinstructions.

ION	Turn on priority interrupt system.
IOF	Turn off priority interrupt system.

3.8.5 Programmed stop of the computer.

The instruction WAIT will cause the computer to stop if the interrupt system is not enabled. The program counter will contain one more than the address of the WAIT instruction (it points to the next instruction after the wait).

In this programmed wait the STOP/CONTINUE button on the operator's panel is lighted red. To start the program in the instruction after the WAIT, push the button STOP/CONTINUE.

If the priority interrupt system is enabled, WAIT will cause an exit from the level now operating (the corresponding bit in PID is reset) and the program with the current highest priority will be entered, which normally then will have a lower priority than the program which contained the WAIT instruction. Therefore the WAIT instruction means "Give up priority". When the program is interrupted in such a WAIT instruction, the P-register points to the instruction after this WAIT, which will be the first instruction the next time this program is entered.

If there are no interrupt requests on any level when the WAIT instruction is executed, the program is exited and the registers saved, but the computer will stop in and IDLE instruction and wait until any interrupt requests occur.

Note that it is legal to specify WAIT followed by an octal number less than 377. This may be useful to detect in which location the program stopped. The WAIT instruction is displayed at the operators panel (IR-register).

A/S NORDATA
NORSK DATA
ELEKTRONIKK

Title

NORD - 1
INSTRUCTION CODE

Drawing no.

			15	14	13	12	11	10 9 8 7 6 5 4 3 2 1 0
0	000.000	STZ	0	0	0	0	0	
	004.000	STA	0	0	0	0	1	
	010.000	STT	0	0	0	1	0	
	014.000	STX	0	0	0	1	1	
1	020.000	STD	0	0	1	0	0	
	024.000	LDD	0	0	1	0	1	
	030.000	STF	0	0	1	1	0	
	034.000	LDF	0	0	1	1	1	
2	040.000	MIN	0	1	0	0	0	X I B — Displacement Δ
	044.000	LDA	0	1	0	0	1	
	050.000	LDT	0	1	0	1	0	
	054.000	LDX	0	1	0	1	1	
3	060.000	ADD	0	1	1	0	0	
	064.000	SUB	0	1	1	0	1	
	070.000	AND	0	1	1	1	0	
	074.000	ORA	0	1	1	1	1	
4	100.000	FAD	1	0	0	0	0	
	104.000	FSB	1	0	0	0	1	
	110.000	FMU	1	0	0	1	0	
	114.000	FDV	1	0	0	1	1	
5	120.000	MPY	1	0	1	0	0	
	124.000	JMP	1	0	1	0	1	
	130.000	CJP	1	0	1	1	0	
	134.000	JPL	1	0	1	1	1	
6	140.000	SKP	1	1	0	0	0	NOT GRT — S — D
	144.000	ROP	1	1	0	0	1	RAD ADI ADC CPM CLD — S — D
	150.000	MIS	1	1	0	1	0	sub in.
	154.000	SHT	1	1	0	1	1	ZIN ROT SHA SAD — Number of shifts
7	160.000	IOT	1	1	1	0	0	PIN SAT ACT — Device number
	164.000		1	1	1	0	1	
	170.000	ARG	1	1	1	1	0	Fncn. — Argument
	174.000	BOP	1	1	1	1	1	Function — Bit no — D

100.000 40.000 20.000 10.000 4.000 2.000 1.000 400 200 100 40 20 10 4 2 1

DRAWN BY PB	Remarks	Replacement for	Date
APPROVED BY			
DATE 21/11 - 67		Replaced by	Date