NORD

COMPUTER SYSTEMS

THE NORD-5 INSTRUCTION SET

September 1971

A/S NORSK DATA-ELEKTRONIKK

Erich Mogensøns vei 38, Oslo 5

THE NORD-5 INSTRUCTION SET

September 1971

TABLE OF CONTENTS

INTRODUCTION

The NORD-5 is designed to be an auxiliary computer in a general computer system called the NORDIC system or the NORD Integrated Computer System

In NORDIC, NORD-5 works as a slave computer being monitored by two or more NORD-1 computers.

One of the main design criteria for NORD-5 was high performance on number crunching, but in addition NORD-5 is an effective general purpose computer. The intention of this manual is to describe the NORD-5 instruction set only, not the connection of NORD-5 to the general system.

The wordlength of NORD-5 is 32 bits for instructions and integers. A floating point number is represented in a 64 bit floating word (two consecutive 32 bit words). Cfr. Section 1.5, Floating Word.

The central processing unit (CPU) contains 64 general registers, each 32 bits long. 16 of the registers may be used as base or index registers, and to each of them is associated one modification register which may contain increments used in certain jump instructions.

64 bit registers for holding floating point numbers are formed by connecting the general registers two by two to obtain floating registers. Cfr. Section 1.9.4, Floating Register.

The instructions are classified in three groups: The memory reference instructions, the inter register instructions, and the argument instructions. The memory reference instructions normally affect one register and one memory word. The memory addressing may be direct or indirect. Up to 16 levels of indirect addressing may be used.

The inter register operations normally use two operand registers called source register A and source register B. The result is stored in a third register, the destination register. These three registers may be any of the general registers. The inter register floating point instructions affect pairs of registers.

The argument instructions are generally two-operand instructions with one of the operands contained in the instruction itself.

The memory cycle time is approximately 1 μs. An inter register multiply of floating point numbers takes about 1 μs; a floating divide takes about 8 μs.

1 GENERAL DEFINITIONS

1.1 Word

The word is the basic storage unit, both in memory and in the central processing unit, CPU. The wordlength of NORD-5 is 32 bits.

Each word in NORD-5 has a unique name which is the name of a register or an address to a word in memory. The content of a word is denoted: (name of word).

1.2 Instruction

Contents of a word interpreted as an instruction to the central processor, CPU.

The instructions in NORD-5 are always one word long. The instructions are divided into three groups as follows:

a) memory reference instructions, Section 2

b) inter-register instructions, Section 3

c) argument instructions, Section 4.

1.3 Integer

Contents of a word interpreted as a binary number.

Negative integers are represented in two's complement. Arithmetic is performed in two's complement.

One's complement is obtained by setting each bit in the word to its opposite value. Two's complement is obtained by adding one to the one's complement of the word.

1.4 Bits and Bit fields

Each bit in a word or floating word has an unique index as follows.

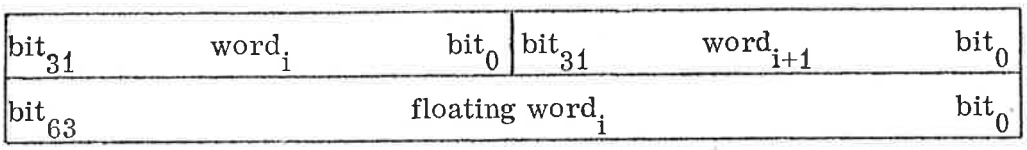
The least significant bit is bit 0. The most significant bit is bit 31. In floating words bit 63 is bit 31 in the first of the two words making the floating words. Bit 0 in the floating word is bit 0 in the second of the two words.

The content of bit i in a word is denoted: (name of word$_i$). A number of contiguous bits in a word is called a bit field. The bit field ranging from bit i to bit j is denoted: name of word$_{i-j}$. The content of a bit field is denoted: (name of word$_{i-j}$).

1.5 Floating Word

A floating word consists of two contiguous 32 bit words in memory, logically connected to give one 64 bit word. The first word of the floating word is called the left portion word, the second word is called the right portion word. Thus, bit field 0-31 is in the right portion word and bit field 32-63 is in the left portion word of the floating word. The address of a floating word is the address of its left portion word. Cfr. Section 1.9.4, Floating Register.

Relation between words and floating words in memory:



In the description of the instructions, floating point $word_i$ will be denoted: $word_i$, $word_{i+1}$.

1.6 Floating Point Number

A floating point number is given by its mantissa, m, and exponent, n, as follows:

$$number = m \cdot 2^n$$

A floating point number is stored in a floating word, cfr. Section 1.5, Floating Word.

The representation of m and n in the floating word is as follows:

bit 0-51	mantissa , m
bit 52-62	exponent , n
bit 63	sign of mantissa, m is negative if bit 63 = 1

The range of the exponent is from -2000_8 to $+1777_8$ making the range of the floating point number approximately from -10^{308} to 10^{308}. The exponent is represented in a 11 bit field but is biased by 2000_8; i.e., 2000_8 is added to the exponent to make it a positive number.

The length of the mantissa is 52 bits, which corresponds to about 15.5 decimal digits.

The floating point numbers have to be normalized before CPU can operate properly on them. A floating point number is said to be on normalized form when m and n have values so that $1 > m \geq 0.1_2$.

1.7 Integer Arithmetic

Integer operands are always one word. Negative numbers are represented in two's complement.

In inter register arithmetic the add and subtract functions may affect or be affected by a carry bit, CB, if the proper subcode is specified.

A multiply instruction will affect an overflow register, OR, which will contain the 32 most significant bits of the product. A divide instruction will affect a remainder register, RR, which will contain a 32 bits remainder after the division.

Arithmetical overflow or a positive product where OR is non-zero (OR is all ones for negative products) may, if specified, cause a monitor call.

1.8 Floating Point Arithmetic

The results of the floating add and subtract functions are rounded. As rounding indicator is used the most significant of the bits which are shifted out to keep a 52 bit mantissa.

In floating multiply, bit 0 in the final result is normally set to one. Bit 0 is not set to one if the eight most significant shifted-out bits are all zero.

In floating divide, bit 0 in the final result is normally set to one. Bit 0 is not set to one if the remainder is zero.

1.9 Register Structure of NORD-5

Registers are storage cells in the central processing unit, CPU. The register used by the programmers in NORD-5 is one word long (32 bits) or one floating word long (64 bits).

1.9.1 P-Register

The program counter, P, contains the address of the instruction being read from meory for execution by the CPU. P is one word long. All instructions affect P. The instructions increment P by one, except the stop, jump and skip instructions.

1.9.2 Instruction Register

The instruction register, IR, contains the instruction being executed by the CPU. IR is one word long.

1.9.3 General Registers

The NORD-5 CPU has 64 general single word registers. These 64 word registers may also be used as 32 general floating registers.

The general registers are normally used by the programs as storage cells for operands and results.

Some of the registers may be used as base registers, index registers and modification registers. Cfr. Section 1.9.4 - Floating Register, Section 1.9.5 - Base Register, Section 1.9.6 - Index Register, and Section 1.9.7 - Modification Register.

The 64 general registers are denoted GR_0, $GR_1, \ldots, GR_{63}$ in the descriptions.

Note: GR_0 always contains zero. This implies that $(FR_0)=0$, $(BR_0)=0$, $(XR_0)=0$ and that MR_0 cannot be used as modification register to GR_0.

1.9.4 Floating Register

A floating register has the storage capacity of one floating word. It is used to store floating point number operands and results.

There are 32 floating registers FR_0 - FR_{31}. The floating registers are organized from the general registers as follows:

$FR_{i\ left} = GR_i$, $FR_{i\ right} = GR_{i+16}$ when $0 \leq i < 16$, $32 \leq i < 48$

We shortly write this as:

$FR_i = GR_i, GR_{i+16}$ when $0 \leq i < 16$, $32 \leq i < 48$

1.9.5 Base Registers

The 16 first general registers may be used as base register, BR.

$BR_i = GR_i \qquad 0 \leq i < 16$

Cfr. Section 2.3 - Memory Addressing - for further description.

1.9.6 Index Registers

The 16 first general registers may be used as index registers, XR.

$XR_i = GR_i \qquad 0 \leq i < 16$

Cfr. Section 2.3 - Memory Addressing - for further description.

If the register has been used as a destination register of a floating point arithmetic operation or double shift, the result cannot be used as a base or index address modification.

1.9.7 Modification Registers

Associated with each of the first 16 general registers there is one modification register, MR. The modification registers are organized from the general registers as follows:

$MR_i = GR_{i+16} \qquad 0 \leq i < 16$

The modification registers are used to hold an increment to base or index registers in certain jump instructions. Cfr. Section 2 - Memory Reference Instruction.

1.9.8 Auxiliary Register

The overflow register, OR, is used to hold the upper 32 bit part of the product of a multiplication.

The remainder register, RR, is used to hold the remainder after a division. OR and RR can be read by an RIO-instruction.

1.10 Syntax of Instruction Descriptions

1.10.1 := Assignment Operator

Set data element to the left of := equal to the data element at the right side of :=.

1.10.2 +, -, *, / Arithmetical Operators

The mode of arithmetic operation is showed by using integer word indicators for integer arithmetic and floating word indicators for floating point arithmetic.

1.10.3 $>\geqslant=\neq<\leqslant$ Relation Operators

The relation operators are used to show arithmetical relations. A relation has the value true or false.

1.10.4 Logical Operators

The logical operations are done by using two operands, each one word long, giving a one word result. However, the logical operations are by its nature single bit operations. They are done on pairs of operand bits, which are made by taking the bits with same bit address from the two operand words. The result is stored in the same bit position in the result word.

Following is a description of the logical operations as single bit operations.

Logical OR , $\vee$

A	0	0	1	1
B	0	1	0	1
A$\vee$B	0	1	1	1

Logical exclusive OR , $\forall$

A	0	0	1	1
B	0	1	0	1
A$\forall$B	0	1	1	0

Logical AND , $\wedge$

A	0	0	1	1
B	0	1	0	1
A$\wedge$B	0	0	0	1

Logical Complement , $\overline{\ }$ (one's complement)

A	0	0	1	1
$\overline{A}$	1	1	0	0

The logical complement is a one operand operation. Each bit is set to its opposite value. The one's complement of a word is not its corresponding negative value. Negative integer numbers are represented in two' complement.

1.10.6 Non-conditional Instructions

A non-conditional instruction is described by one or more assignment statements. If there are more than one assignment statement, they will be connected by the word AND. The assignment statement is executed in the sequence they are shown.

Note: In all instructions P is incremented by one unless otherwise specified.

1.10.7 Conditional Instructions

The conditional instructions are described as follows:

IF relation is true

THEN non-conditional instructions
Instruction finished

ELSE non-conditional instructions
Instruction finished

If there are statements before the conditional phase, the word AND is used to connect the "IF. . . ." to the rest of the description.

1.11 Execution Times

Timing for each instruction is found in the instruction descriptions.

1.11.1 Time Definitions

1.11.1.1 Memory Cycle Time, TMC

Time necessary to read or write one word in memory, TMC = 1.0 μs.

1.11.1.2 Read Instruction Time, TRI

Time necessary to read one instruction from memory to control processor, TRI = TMC

1.11.1.3 Read Single Word Operand Time, TSW

Time necessary to load or store one single word operand in memory, TSW = TMC

1.11.1.4 Read Floating Word Operand Time, TFW

Time necessary to load or store one floating word operand in memory, TFW $\leq$ 2*TMC.

1.11.1.5 Central Processor Time, TCP

Time used by central processor to perform a specified operation when all operands are available in processor registers (Time used to store the result not included). TCP depends on the specified instruction.

1.11.1.6 Execution Time (T)

Total time used to execute one instruction; i.e., time used from the read instruction cycle starts until the result is in the specified register or memory word.

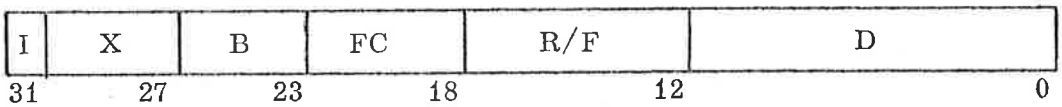
2 MEMORY REFERENCE INSTRUCTION

2.1 Introduction

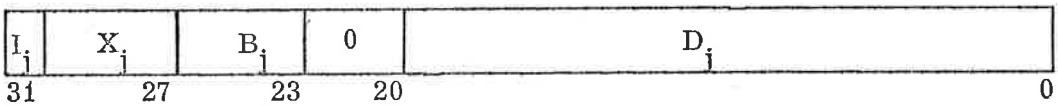
The memory reference instructions have in common that the execution of an instruction involves calculation of a memory address. In some of the memory reference instructions the memory address is used as operand, i.e., jumps and remote execution.

2.2 Instruction Word

Instruction Word, IW



Indirect Address Word of level j, IAW_j



IW	:	Instruction Word
I	:	Indirect addressing flag
X	:	Index register designator
B	:	Base register designator
FC	:	Function Code
R/F	:	General register or floating register designator
D	:	Displacement $0 \leq D < 4096$
IAW_j	:	Indirect Address Word at level j
I_j	:	Indirect addressing flag in IAW_j $I_0 = I$
X_j	:	Index register designator in IAW_j
B_j	:	Base register designator in IAW_j
D_j	:	Displacement in IAW_j, $0 \leq D_j < 1\ 048\ 576$

2.3 Memory Addressing

All memory reference instructions calculate a memory address. The result of this calculation is called the effective address, Ea.

The memory addressing may be direct or indirect. When the addressing is direct, Ea is calculated without memory reference (except for the read of the instruction). When indirect addressing, the CPU has to reference the memory for operands to the calculation of Ea. We say that each memory reference, which is done to calculate Ea in one instruction, requires one level of indirect addressing. The maximum number of levels possible in the NORD-5 is 16 in addition to the read of the instruction.

The algorithm for calculating Ea is as follows: (symbols defined in 2.2)

$$Ea_0 = (R_B) + (R_X) + D$$

if I = 0, then $Ea = Ea_0$

$$Ea_1 = (R_{B_1}) + (R_{X_1}) + D_1 \text{ , } IAW_1 = (Ea_0)$$

$$Ea_j = (R_{B_j}) + (R_{X_j}) + D_j \text{ , } IAW_j = (Ea_{j-1})$$

$$Ea = Ea_j,\ I_j = 0,\ 0 < j \leq 16,\ I_i = 1,\ 0 \leq i < j$$

Note:

Each level of indirect addressing adds one "read instruction time" to the execution time.

A store operation to the location immediately following the storing instruction will have a special effect. The storing will be executed correct, but the next instruction will be the old content of the location. Example: STR REG, * +1.

Indirect addressing is not allowed for the instruction EXC.

2.4 Instruction List

	FC				
2.4.1	0	Refer to inter register and argument instructions.			
2.4.2	1	RTJ	: Return Jump (R): = (P) + 1 AND (P): = Ea	T = 2 μs	
2.4.3	2	EXC	: Remote Execute (Two instructions) R = 0, (IR): = (Ea) R = 1, (IR): = Ea	T = 3 μs T = 2 μs	
2.4.4	3	MIN	: Memory Increment (Ea): = (Ea) + (R) + 1 AND IF (Ea): = 0 THEN (P): = (P) + 2 ELSE (P): = (P) + 1	T = 3 μs	
2.4.5	4	CRG	: Skip if register is greater or equal memory word IF (R) $\geqslant$ (Ea) THEN (P): = (P) + 2 ELSE (P): = (P) + 1	T = 2 μs T = 3 μs	No skip Skip
2.4.6	5	CRL	: Skip if register is less than memory word IF (R) $\prec$ (Ea) THEN (P): = (P) + 2 ELSE (P): = (P) + 1	T = 2 μs T = 3 μs	No skip Skip
2.4.7	6	CRE	: Skip if register and memory word is equal IF (R) = (Ea) THEN (P): = (P) + 2 ELSE (P): = (P) + 1	T = 2 μs T = 3 μs	No skip Skip
2.4.8	7	CRD	: Skip if register not equal memory word IF (R) $\neq$ (Ea) THEN (P):=(P) + 2 ELSE (P): = (P) + 1	T = 2 μs T = 3 μs	No skip Skip

	FC				
2.4.9	8	JRP	: Jump if register is positive IF (R) ≥ 0 THEN (P): = Ea ELSE (P): = (P) + 1	T = 1 μs T = 2 μs	No jump Jump
2.4.10	9	JRN	: Jump if register is negative IF (R) < 0 THEN (P): = Ea ELSE (P): = (P) + 1	T = 1 μs T = 2 μs	No jump Jump
2.4.11	10	JRZ	: Jump if register is zero IF (R) = 0 THEN (P): = Ea ELSE (P): = (P) + 1	T = 1 μs T = 2 μs	No jump Jump
2.4.12	11	JRF	: Jump if register is non-zero IF (R) ≠ 0 THEN (P): = Ea ELSE (P): = (P) + 1	T = 1 μs T = 2 μs	No jump Jump
2.4.13	12	JPM	: Modify register by its modification register. If register is positive, then jump. (R): = (R) + (M_R) AND IF (R) ≥ 0 THEN (P): = Ea ELSE (P): = (P) + 1	T = 1 μs T = 2 μs	No jump Jump
2.4.14	13	JNM	: Modify register by its modification register. If register is negative, then jump. (R): (R) + (M_R) AND IF (R) < 0 THEN (P): = Ea ELSE (P): = (P) + 1	T = 1 μs T = 2 μs	No jump Jump
2.4.15	14	JZM	: Modify register by its modification register. If register is zero, then jump. (R): = (R) + (M_R) AND IF (R) = 0 THEN (P) = Ea ELSE (P) = (P) + 1	T = 1 μs T = 2 μs	No jump Jump

	FC				
2.4.16	15	JFM	: Modify register by its modification register. If contents of register is non-zero, then jump. (R): = (R) +(M_R) IF (R) ≠ 0 THEN (P): = Ea ELSE (P): = (P) + 1	T = 1 μs T = 2 μs	No jump Jump
2.4.17	16	ADD	: Add content of memory word to register (R): = (R) + (Ea)	T = 2 μs	
2.4.18	17	SUB	: Subtract content of memory word from register (R): = (R) - (Ea)	T = 2 μs	
2.4.19	18	AND	: Make "logical AND" between memory word and register. Result in register. (R): = (R) ∧ (Ea)	T = 2 μs	
2.4.20	19	LDR	: Load content of memory word to register. (R): = (Ea)	T = 2 μs	
2.4.21	20	ADM	: Add contents of register to memory word (Ea): = (Ea) + (R)	T = 3 μs	
2.4.22	21		Instruction set aside for future extensions.		
2.4.23	22	XMR	: Exchange contents of registers and memory word (R): = (Ea) AND (Ea): = (R)	T = 3 μs	

	FC			
2.4.24	23	STR	: Store register in memory word (Ea): = (R)	T = 2 μs
2.4.25	24	MPY	: Multiply contents of register and memory word and set result in register (R): = (R) * (Ea) Note: Upper 32 bit part of 64 bit product is in OR	T = 2 μs
2.4.26	25	DIV	: Divide content of register with content of word. Quotient is in register. Remainder is in the remainder register. (R): = (R)/(Ea) Note: Remainder is in RR	T = 10 μs
2.4.27	26	LDF	: Load floating register with content of floating word (F): = (Ea, Ea + 1)	T = 3 μs
2.4.28	27	STF	: Store content of floating register in floating word (Ea, Ea + 1): = (F)	T = 3 μs
2.4.29	28	FAD	: Add content of floating word to content of floating register (F): = (F) + (Ea, Ea + 1)	T = 3 μs
2.4.30	29	FSB	: Subtract content of floating word from floating register (F): = (F) - (Ea, Ea + 1)	T = 3 μs

	FC			
2.4.31	30	FMU	: Multiply content of floating word by floating register. Result in floating register.	
			(F): = (F) * (Ea, Ea + 1)	T = 3 μs
2.4.32	31	FDV	: Divide content of floating register by content of floating word. Result in floating register.	
			(F): = (F)/(Ea, Ea + 1)	T = 10 μs

3 INTER-REGISTER INSTRUCTIONS

3.1 Introduction

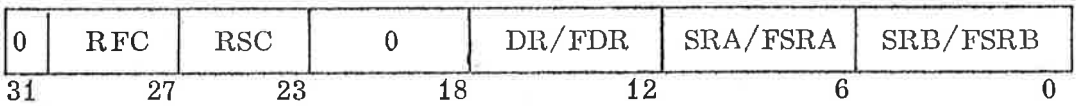
The inter-register instructions normally affect the general registers, or floating registers organized from the general registers. In their general form, the instructions specify two operand registers, called source register A and source B, and one result register, called the destination register.

The usage of source registers and destination registers is completely general. It is possible to specify any registers as source registers and destination registers.

Some of the inter-register instructions have special forms which will be described under the specific instruction.

3.2 Instruction Layout

Instruction Word, IW



IW : Instruction Word
RFC : Inter-Register Function Code
RSC : Inter-Register Sub-function Code
SRA : Source Register A designator
FSRA : Floating Source Register A designator
SRB : Source Register B designator
FSRB : Floating Source Register B designator
DR : Destination Register designator
FDR : Destination Floating Register designator

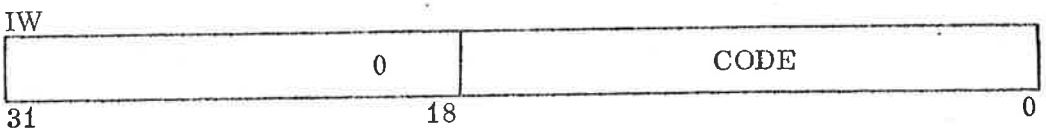
The following abbreviations will also be used:

BN : Bit Number
SC : Shift Count

3.3 Inter-register Instruction Descriptions

3.3.1 STOP

Stop NORD-5 CPU and call monitor program in NORD-1.



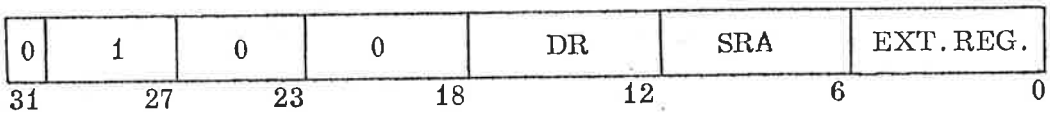
$(IW)_{18-31} = 0$

$(IW)_{0-17}$ = code used to specify monitor function

(P): = (P) T = 1 μs

3.3.2 RIO: Register Input/Output

Transfer content of a specified external register to a specified register.



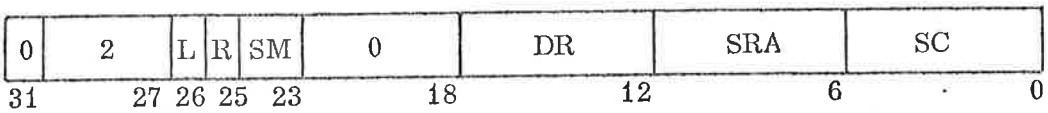
EXT.REG: External register

1 : Overflow, OR-register

3 : Remainder, RR-register

3.3.3 SHR: Shift Register

Copy content of SR to DR and shift DR.



RSC : divided in three fields, L, R, and SM

L = 1 : Shift left

R = 1 : Shift right

SM : Shift mode

SM = 0 : Rotate register

SM = 1 : Rotate register

SM = 2 : Arithmetical shift. For left shift same as logical. For right shift bit 31 is copied to each bit shifted in (sign extension)

SM = 3 : Logical shift. The bits which are shifted out of the word are lost and zeroes are put in the other end.

$0 \leq SC \leq 31$

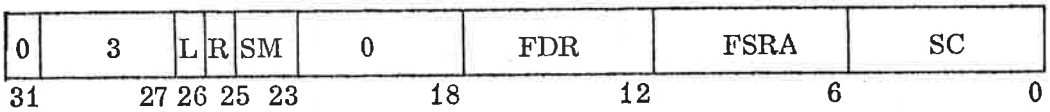
(DR) : = (SR) AND
(DR) : = (DR) * 2^{SC}

If both L = 1 and R = 1, then the right shift will be performed first and then the left shift.

T = 1 μs

3.3.4 SHF: Shift Floating Register

Copy FSRA to FDR and shift FDR.

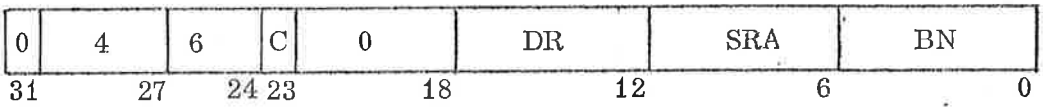


Description as for SHR, Section 3.3.3, except $0 \leq SC \leq 63$.

T = 1 μs

3.3.5 BST: Bit Set

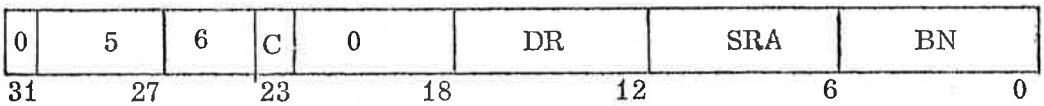
Copy SRA to DR and set specified bit in DR to one.



(DR): = (SRA) AND
IF C = 0 THEN
(DR_{BN}): = 1 ELSE
(DR_{BN}): = (DR_{BN}) T = 1 μs

3.3.6 BCL: Bit Clear

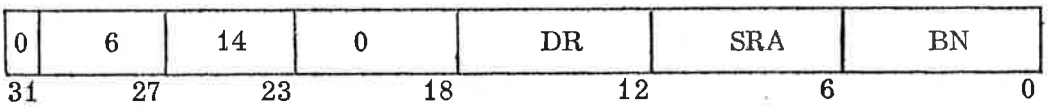
Copy SRA to DR and set specified bit in DR to zero.



(DR): = (SRA) AND
IF C = 0 THEN
(DR_{BN}): = 0 ELSE
(DR_{BN}): = (DR_{BN}) T = 1 μs

3.3.7 BSZ: Bit Skip on Zero

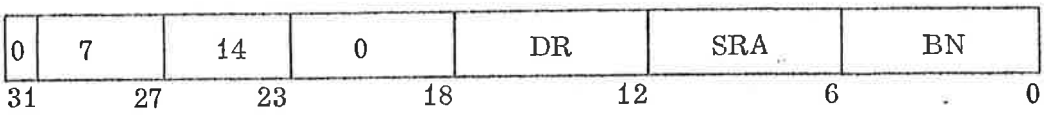
Copy SRA to DR and skip if specified bit in DR is zero.



(DR): = (SRA) AND
IF (DR_{BN}) = 0 THEN (P): = (P) + 2
ELSE (P): = (P) + 1 T = 1.5 μs

3.3.8 BSO: Bit Skip on One

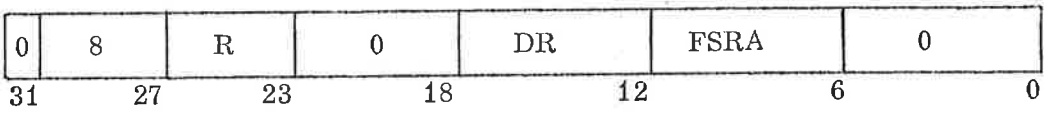
Copy SRA to DR and skip if specified bit in DR is one.



(DR): = (SRA) AND
IF (DR_{BN}) = 1 THEN (P): = (P) + 2
ELSE (P): = (P) + 1 T = 1.5 μs

3.3.9 FIX: Convert Floating to Integer

Convert floating point number in FSRA to integer and place result in DR.

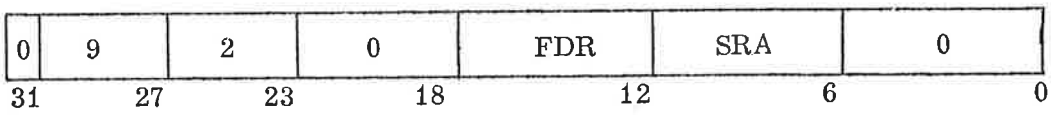


(DR): = integer (RSRA)

The converted number is truncated if R = 0; if R = 10, it is rounded. T = 1 μs

3.3.10 FLO: Convert Integer to Floating

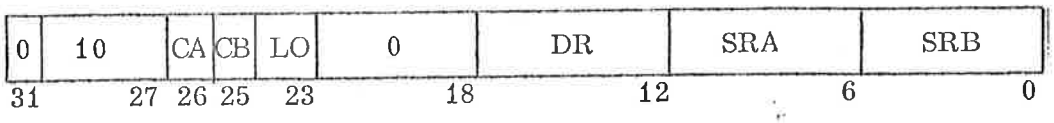
Convert the integer number in SRA to a floating point number in FDR



(FDR): = float (SRA)

T = 1 μs

3.3.11 LRO: Logical Register Operation



The RSC is divided in 3 fields, CA, CB and LO. If CA is set, the operand in SRA is complemented before the logical operation is performed. CB has the same effect on SRB.

The complementation of the operands does not affect the original contents of SRA and SRB.

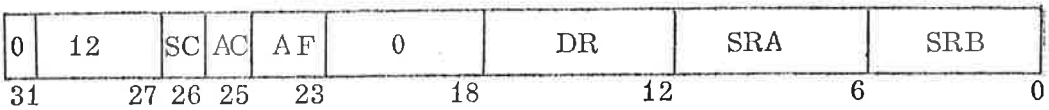
RSC =	CA	CB	LO	
	0	0	0	(DR) : = 0
	0	0	1	(DR) : = (SRA) $\wedge$ (SRB)
	0	0	2	(DR) : = (SRA) $\forall$ (SRB)
	0	0	3	(DR) : = (SRA) $\vee$ (SRB)
	0	1	0	(DR) : = 0
	0	1	1	(DR) : = (SRA) $\wedge$ $\overline{(SRB)}$
	0	1	2	(DR) : = (SRA) $\forall$ $\overline{(SRB)}$
	0	1	3	(DR) : = (SRA) $\vee$ $\overline{(SRB)}$
	1	0	0	(DR) : = 0
	1	0	1	(DR) : = $\overline{(SRA)}$ $\wedge$ (SRB)
	1	0	2	(DR) : = $\overline{(SRA)}$ $\forall$ (SRB)
	1	0	3	(DR) : = $\overline{(SRA)}$ $\vee$ (SRB)
	1	1	0	(DR) : = 0
	1	1	1	(DR) : = $\overline{(SRA)}$ $\wedge$ $\overline{(SRB)}$
	1	1	2	(DR) : = $\overline{(SRA)}$ $\forall$ $\overline{(SRB)}$
	1	1	3	(DR) : = $\overline{(SRA)}$ $\vee$ $\overline{(SRB)}$

T = 1 μs

3.3.12

The inter-register instruction with RFC = 11 is saved for future usage.

3.3.13 IRO: Inter-Register Arithmetic



AF : Arithmetical function

AF = 0	(DR): = (SRA) + (SRB)	T = 1 μs
1	(DR): = (SRA) - (SRB)	T = 1 μs
2	(DR): = (SRA) * (SRB)	T = 1 μs
3	(DR): = (SRA)/ (SRB)	T = 8 μs

The inter-register add and subtract may affect or be affected by a one bit register called the Carry Bit, CB.

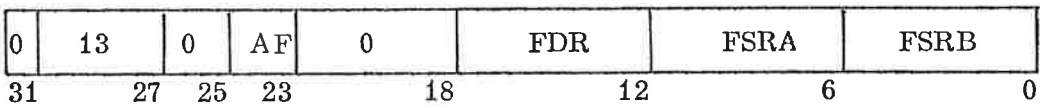
If SC = 1, the content of CB will be set to its proper value after the arithmetical operation. If AC = 1, the content of CB will be added to the result of the arithmetical function.

The carry bit may be used to simulate multiple precision arithemtic. Using SC=1 will have the same effect as extending the operands with one bit containing zero. The result register will be extended by one bit containing one or zero, according to the arithmetical condition.

Integer multiply and divide will affect the auxiliary registers. OR will contain the upper 32 bit part of the 64 bit product. RR will contain a 32 bit remainder after an integer divide.

In all integer arithmetic, overflow conditions may occur. If so specified, an overflow condition may cause a monitor call. Overflow occurs when the result of an arithmetical operation is in size greater than $\pm(2^{23} - 1)$.

3.3.14 FRO: Floating Point Arithmetic

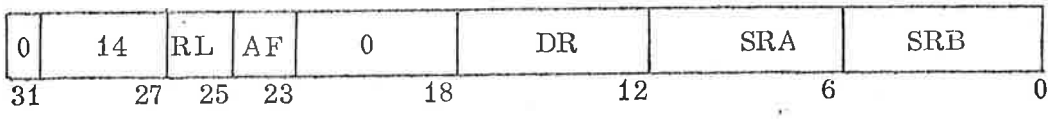


AF : Arithmetical function

AF = 0	(FDR): = (FSRA) + (FSRB)	T = 1 μs
AF 1	(FDR): = (FSRA) - (FSRB)	T = 1 μs
2	(FDR): = (FSRA) * (FSRB)	T = 1 μs
3	(FDR): = (FSRA) / (FSRB)	T = 8 μs

Cfr. Section 1.8, Floating Point Arithmetic.

3.3.15 IRS: Integer Register Skip



The inter-register skip is a general arithmetical instruction followed by a relation between the result and zero.

AF : Arithmetical function

AF = 0 Add

1 Subtract

RL : Relation between result and zero

RL = 0 result $\geqslant 0$

1 result < 0

2 result = 0

3 result $\neq 0$

(DR): = (SRA) arithmetical operation (SRB) AND

IF relation THEN (P): = (P) + 2

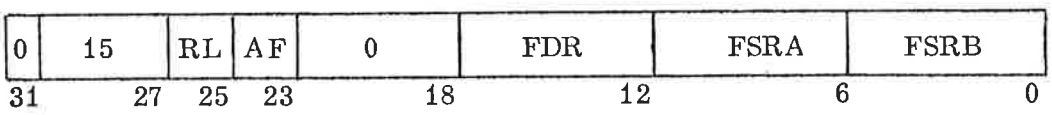
ELSE (P): = (P) + 1

Normally, DR will be affected, but if general register 0 is specified as DR, no destination register is affected. However, the relation on the result will still be effective.

When DR = 0 and AF = 1, the IRS operation is a traditional skip.

Skip	2 μs
No skip	1 μs

3.3.16 FRS: Floating Register Skip



The description of FRS is the same as for IRS, Section 3.3.15, except that the arithmetic is performed on floating registers in floating point mode.

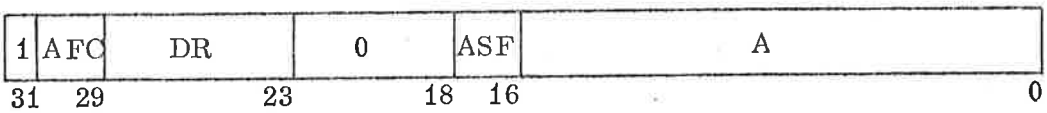
4 ARGUMENT INSTRUCTION

4.1 Introduction

Typical for the argument instruction is that one of the operands is contained in the instruction itself; it is called the argument, A. The other operand is contained in one of the general registers.

The argument is a 16 bit positive number. Before the specified operation takes place, the argument is extended to a 32 bit number with the 16 upper bits all equal to zero.

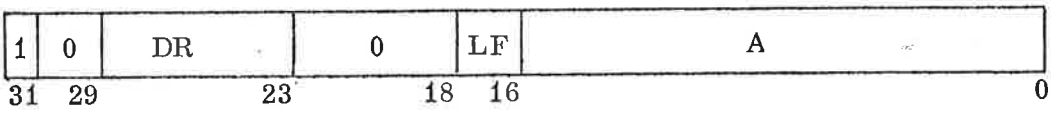
4.2 Argument Instruction Layout



A : Argument, 16 bits
AFC : Argument instruction function code
DR : Destination register
ASF : Argument instruction sub-function code

4.3 Instruction Description

4.3.1 DLR: Direct Logical Operation

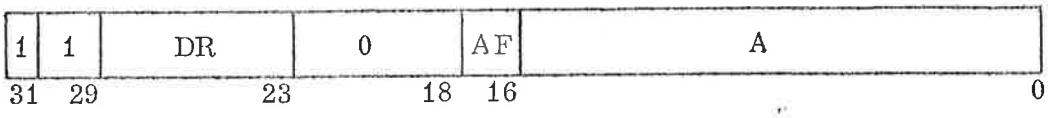


LF : Logical Function

LF = 0 (DR) : = 0
1 (DR) : = (DR) $\forall$ A
2 (DR) : = (DR) $\wedge$ A
3 (DR) : = (DR) $\vee$ A

T = 1 μs

4.3.2 DAR: Direct Arithmetic Function

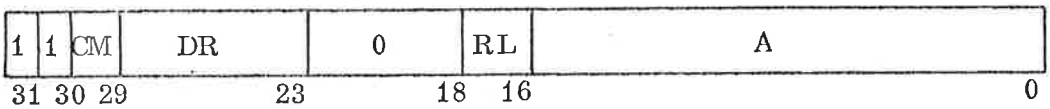


AF : Arithmetical Function

AF = 0 (DR): = A
1 (DR): = -A (2's complement)
2 (DR): = (DR) + A
3 (DR): = (DR) - A

T = 1 μs

4.3.3 DSK: Direct Skip



CM : Complement flag (2's complement)
RL : Relation designator

RL = 0	CM = 0	IF (DR) ≥ A THEN (P): = (P) + 2 ELSE (P): = (P) + 1	DD
1	0	IF (R) < A THEN (P): = (P) + 2 ELSE (P): = (P) + 1	DDN
2	0	IF (DR) = A THEN (P): = (P) + 2 ELSE (P): = (P) + 1	DD2
3	0	IF (DR) ≠ A THEN (P): = (P) + 2 ELSE (P): = (P) + 1	DDF
0	1	IF (DR) ≥ -AR THEN (P): = (P) + 2 ELSE (P): = (P) + 1	DSP
1	1	IF (DR) < -AR THEN (P): = (P) + 2 ELSE (P): = (P) + 1	DSN
2	1	IF (DR) = -AR THEN (P): = (P) + 2 ELSE (P): = (P) + 1	DS2
3	1	IF (DR) ≠ -AR THEN (P): = (P) + 2 ELSE (P): = (P) + 1	DSF

T = 1 μs No skip
T = 2 μs Skip

5 SUMMARY OF INSTRUCTIONS

MEMORY REFERENCE INSTRUCTIONS

+++
+

MNEMONIC	ACTION	SECTION
RTJ	Return jump	2. 4. 2
EXC	Remote execute	2. 4. 3
MIN	Mmeory increment	2. 4. 4
CRG	Skip if (R) $\geq$ (Ea)	2. 4. 5
CRL	Skip if (R) $<$ (Ea)	2. 4. 6
CRE	Skip if (R) = (Ea)	2. 4. 7
CRD	Skip if (R) $\neq$ (Ea)	2. 4. 8
JRP	Jump if (R) ≥ 0	2. 4. 9
JRN	Jump if (R) < 0	2. 4. 10
JRZ	Jump if (R) $= 0$	2. 4. 11
JRF	Jump if (R) $\neq 0$	2. 4. 12
JPM	Modify (R) and jump if (R) ≥ 0	2. 4. 13
JNM	Modify (R) and jump if (R) < 0	2. 4. 14
JZM	Modify (R) and jump if (R) $= 0$	2. 4. 15
JFM	Modify (R) and jump if (R) $\neq 0$	2. 4. 16
ADD	Add (Ea) to (R)	2. 4. 17
SUB	Subtract (Ea) from (R)	2. 4. 18
AND	Logical AND between (Ea)and (R)	2. 4. 19
LDR	Load (R) with (Ea)	2. 4. 20
ADM	Add (R) to (Ea)	2. 4. 21
XMR	Exchange (Ea) and (R)	2. 4. 23
STR	Store (R) in (Ea)	2. 4. 24
MPY	Multiply (R) by (Ea)	2. 4. 25
DIV	Divide (R) by (Ea)	2. 4. 26
LDF	Load (F) with (Ea, Ea + 1)	2. 4. 27
STF	Store (F) in (Ea, Ea + 1)	2. 4. 28
FAD	Add (Ea, Ea + 1) to (F)	2. 4. 29
FSB	Subtract (Ea, Ea + 1) from (F)	2. 4. 30
FMU	Multiply (F) by (Ea, Ea + 1)	2. 4. 31
FDV	Divide (F) by (Ea, Ea + 1)	2. 4. 32

INTER REGISTER OPERATIONS

1 SHIFT INSTRUCTIONS

MNEMONIC	ACTION	SECTION
SLR	Left rotational shift	3. 3. 3
SRR	Right rotational shift	"
SLA	Left arithmetical shift	"
SRA	Right arithmetical shift	"
SLL	Left logical shift	"
SRL	Right logical shift	"
SLRD	Left rotational floating register shift	3. 3. 4
SRRD	Right rotational floating register shift	"
SLAD	Left arithmetical floating register shift	"
SRAD	Right arithmetical floating register shift	"
SLLD	Left logical floating register shift	"
SRLD	Right logical floating register shift	"

2 MISCELLANEOUS OPERATIONS

BST	Bit Set	3. 3. 5
BCL	Bit clear	3. 3. 6
BSZ	Bit skip on zero	3. 3. 7
BSO	Bit ship on one	3. 3. 8
FIX	Convert floating to integer	3. 3. 9
FLO	Convert integer to floating	3. 3. 10

3 ARITHMETIC OPERATIONS

RAD	Register add	3. 3. 13
RSB	Register subtract	"
RMU	Register multiply	"
RDV	Register divide	"
RAF	Floating register add	3. 3. 14
RSF	Floating register subtract	"
RMF	Floating register multiply	"
RDF	Floating register divide	"

4 TEST AND SKIP

MNEMONIC	ACTION	SECTION
SGR	Subtract registers and skip if result ≥ 0	3.3.15
ASG	Add " " " " " " ≥ 0	"
SLE	Subtract " " " " " " < 0	"
ASL	Add " " " " " " < 0	"
SEQ	Subtract " " " " " " $= 0$	"
ASE	Add " " " " " " $= 0$	"
SUE	Subtract " " " " " " $\neq 0$	"
ASU	Add " " " " " " $\neq 0$	"
SGF	Subtract floating registers and skip if result ≥ 0	3.3.16
ASGF	Add " " " " ≥ 0	"
SLF	Subtract " " " " < 0	"
ASLF	Add " " " " < 0	"
SEF	Subtract " " " " $= 0$	"
ASEF	Add " " " " $= 0$	"
SUF	Subtract " " " " $\neq 0$	"
ASUF	Add " " " " $\neq 0$	"

5 LOGICAL OPERATIONS

MNEMONIC	ACTION	SECTION
RND	Register AND	3. 3. 11
RNDA	Register AND, use complement of (SRA)	"
RNDB	Register AND, use complement of (SRB)	"
RXO	Register exclusive OR	"
RXOA	Register exclusive OR, use complement of (SRA)	"
RXOB	Register exclusive OR, use complement of (SRB)	"
ROR	Register OR	"
RORA	Register OR, use complement of (SRA)	"
RORB	Register OR, use complement of (SRB)	"
SZR	Set all zeroes	"

--ooOoo--

6 ARGUMENT INSTRUCTIONS

XORA	Exclusive or
ANDA	And
ORA	Or
SETA	Set register
SECA	Set register to complement
ADDA	Add
ADCA	Add complement

DDP	Skip if (DR) $\geq$ ARG
DDN	" " " $<$ "
DDZ	" " " $=$ "
DDF	" " " $\neq$ "
DSP	Skip if (DR) $\geq$ -ARG
DSN	" " " $<$ "
DSZ	" " " $=$ "
DSF	" " " $\neq$ "

31	30 29 28 27	26 25 24 23	22 21 20 19 18	17 16 15 14 13 12	11 10 9 8 7 6	5 4 3 2 1 0	
0	0 0 0 0	0 0 0 0	0 0 0 0 0	MESSAGE			STOP
0	0 0 0 1	0	0 0 0 0 0	DR	SR	EXT.REG.NO	RIO
0	0 0 1 0	L R SM	0 0 0 0 0	DR	SR	SHIFT COUNT	SHR
0	0 0 1 1	L R SM	0 0 0 0 0	DR	SR	SHIFT COUNT	SHF
0	0 1 0 0	1 1 0 C	0 0 0 0 0	DR	SR	BITNO	BST
0	0 1 0 1	1 1 0 C	0 0 0 0 0	DR	SR	BITNO	BCL
0	0 1 1 0	1 1 0 0	0 0 0 0 0	DR	SR	BITNO	BSZ
0	0 1 1 1	1 1 0 0	0 0 0 0 0	DR	SR	BITNO	BSO
0	1 0 0 0	R 0 0 0	0 0 0 0 0	DR	FSR	0 0 0 0 0 0	FIX
0	1 0 0 1	0 0 1 0	0 0 0 0 0	FDR	SR	0 0 0 0 0 0	FLO
0	1 0 1 0	CA CB LO	0 0 0 0 0	DR	SRA	SRB	LRO
0	1 0 1 1		0 0 0 0 0				Not used
0	1 1 0 0	SC AC AF	0 0 0 0 0	DR	SRA	SRB	IRO
0	1 1 0 1	0 0 AF	0 0 0 0 0	FDR	FSRA	FSRB	FRO
0	1 1 1 0	RL AF	0 0 0 0 0	DR	SRA	SRB	IRS
0	1 1 1 1	RL AF	0 0 0 0 0	FDR	FSRA	FSRB	FRS
I	X	B	0 0 0 0 1	R	D		RTJ
I	X	B	0 0 0 1 0	R	D		EXC
I	X	B	0 0 0 1 1	R	D		MIN
I	X	B	0 0 1 0 0	R	D		CRG
I	X	B	0 0 1 0 1	R	D		CRL
I	X	B	0 0 1 1 0	R	D		CRE
I	X	B	0 0 1 1 1	R	D		CRD
I	X	B	0 1 0 0 0	R	D		JRP
I	X	B	0 1 0 0 1	R	D		JRN
I	X	B	0 1 0 1 0	R	D		JRZ
I	X	B	0 1 0 1 1	R	D		JRF

31	30 29 28 27	26 25 24 23	22 21 20 19 18	17 16 15 14 13	12 11 10 9 8 7 6 5 4 3 2 1 0	
I	X	B	0 1 1 0 0	R	D	JPM
I	X	B	0 1 1 0 1	R	D	JNM
I	X	B	0 1 1 1 0	R	D	JZM
I	X	B	0 1 1 1 1	R	D	JFM
I	X	B	1 0 0 0 0	R	D	ADD
I	X	B	1 0 0 0 1	R	D	SUB
I	X	B	1 0 0 1 0	R	D	AND
I	X	B	1 0 0 1 1	R	D	LDR
I	X	B	1 0 1 0 0	R	D	ADM
I	X	B	1 0 1 0 1	R	D	Not used
I	X	B	1 0 1 1 0	R	D	XMR
I	X	B	1 0 1 1 1	R	D	STR
I	X	B	1 1 0 0 0	R	D	MPY
I	X	B	1 1 0 0 1	R	D	DIV
I	X	B	1 1 0 1 0	FR	D	LDF
I	X	B	1 1 0 1 1	FR	D	STF
I	X	B	1 1 1 0 0	FR	D	FAD
I	X	B	1 1 1 0 1	FR	D	FSB
I	X	B	1 1 1 1 0	FR	D	FMU
I	X	B	1 1 1 1 1	FR	D	FDV
I_j	X_j	B_j	0 0 0	D_j		Ind.addr word
1	0 0 DR		0 0 0 0 0	LF	A	DLR
1	0 1 DR		0 0 0 0 0	AF	A	DAR
1	1 CM DR		0 0 0 0 0	RL	A	DSK