The Algorithmic "Computer"

Zoya Alekseeva

The Algorithmic "Computer"

Zoya Alekseeva

Senior Research Assistant
Moscow Engineering-Physical Institute (MEPI)
zd_alexeeva@inbox.ru

Abstract. This article describes the software creation and development of the computer. Fortran 4 was chosen for scientific and technical problems. They named the language based on Fortran 4 language RTL (Fortran- real time language) treated the allocation of programs in the memory, different kinds of record commands in registers, and instruction structures in memory. It treated the principles of control units and developed an acquisition of some system statistics. Programming in the Fortran-like language increased the coding productivity fivefold in comparison to coding with an ordinary computer language.

Keywords: Algorithmic Computer, Fortran 4, high-level languages, real-time systems, real time languages (RTL)

1 Introduction

Development of computing means goes on two basic directions: development of the equipment and development of methods and means of the software creation. In the 1960s through the 1980s, software producing began at a high-level language (HLL) level. Thus for the creation of links connecting the program written in a HLL and a computer language, it was necessary to create a line of programs – compilers, which made the translation process. Usage of a HLL has certainly simplified the work of a programmer. However, the necessity of application of compilers has essentially increased the size of programs and consequently, has increased the time of performance of a problem; the physical size of the equipment of a computer has increased the time of entering of corrections at debugging.

At any moment of time at designing the specialized systems, the cost for the creation of the software began to exceed the cost of the creation of the equipment. Except for it, one other point appeared; if it is necessary quickly to change the program on the place of usage one mast have the technological equipment there it is not always possible.

The growth of the size of programs using HLL, translation and compilation for reception of a machine code in two to five times demanded an increase in storage size; it was also essential to increase the productivity of the "Computer" at the decision of problems in real time systems. All these circumstances have forced to search for new ways for the rational decision in total of the created problems.

Many language tasks have appeared in computer science because of the different kinds of problems the features of language required.

Therefore, they applied the Cobol language to the decision of problems of economic aspect, LISP for programs of processing of lists, Fortran for scientific and technical problems. The universal languages of priority were PL-1 and the Ada; however, within them appeared no means to address problems of the logical-managing side.

One of search variants of the structure of the "Computer" was a HLL computer language. In the 1960s, high-level source languages witnessed American publications about the "Computer" with hardware interpretation; in particular, Fortran-machines began to appear. The source language for the modified language was Fortran. The Fortran instructions executed in the machine and the device of management had nodes for processing each kind of instructions that transformed them to sequence of microinstructions. This development had not received wide distribution in view of great storage capacity of the equipment.

The idea of creating a "Computer" that worked on HLL has arisen in MEPhI in the beginning of 1980s at the chair of professor Ja. Khetagourov. Having made the analysis widely enough used at that time, HLL and a class of problems, characteristic for specialized real-time systems, came to a conclusion. It is rational to create a language similar to Fortran since for the algorithmic "Computer" it was simple to study, it was logically harmonious, and it was convenient for a wide class of problems and tasks. Having executed preliminary studies, materials had transferred in NPO "Agate" where, making a start from preliminary studies, it created the "Computer" that worked on a Fortran-similar language. Language expanded to include Fortran IV with inclusion of some operators of the PL/1 language. It included special operators and the descriptions allowing it to work real-time systems and with necessary conditions to protect programs and to process of the decision from malfunctioning.

For the aforementioned purposes, the operators used allowed the following:

- Organize protection of programs and files,
- Execute parallel work of programs,
- Carry out blocking of interruptions,
- Set reactions to interruptions.
- Allow language means to fix situations interesting developers,
- Simulate occurrence of signals of interruptions,
- Distribute memory dynamical, statically or under the instructions.

To protect the workings of the machine against consequences of malfunctions, it helps to segment programs into different sites after entering the language; this allows the ability to make repetitions of the decisions in a case malfunctions situations. For example, one can monitor a condition of the "Computer" and the program storage when sensing the occurrence of a malfunction interruption. In addition, the common interrupt error library can organize with protection:

- Task of priorities,
- Introduction of several update wait operators,
- Work with timer, etc.

To use this language, it has created the "Computer", directly realizing problems and the system software written on HLL language.

2 The Language Project

They developed the language based on the language Fortran IV and they named it RTF (Fortran real time). The set of RTF operators appear in Appendix A. Table 1 shows examples of situations of interruptions and reactions to them in a concrete system. (In some systems, other reasons can exist of interruptions and reactions to them.)

Table 1. RTF Examples

The name of a situation	Value
OVERFLOW	Overflow
UNDERFKOW	Loss of the importance
OVERTIME (E)	Time is exceeded (time interval specified in expression E or in a variable has expired)
KEY	Discrepancy of a key and the lock
SUBSCRIPTANCE	Output(Exit) of indexes abroad a file
CHECK (x1, x2, …)	Reference (manipulation) to specified identifiers (e.g. programs, subroutines, labels)

Note that at the occurrence of the interruption signals, if a mask does not forbid it, the system reacts in the standard image if special reaction to interruption is not specified in operator ON.

3 Features of the Algorithmic "Computer"

Introduction of the high-level source language is natural and it demands one of two ways in the creation of software:

1) To process the program, translating it with a computer language (the compiler), and coordinating separate pieces in the concrete program in a single whole
2) To have as a computer language a high-level command language; that is, enter into the management of performance of operators the additional equipment for the analysis and performance of language designs that will distinguish such things as kinds of operands, operations, types of dates, and the order of their performance.

As mentioned above, the experience of developing the Fortran-like machine, where it carried out each operation on the processor, appeared impractical because equipment had a large physical volume. In the 1980s, integrated circuits appeared and they reduced the physical size of the equipment, and the idea has again caused interest since hardware performance of operators increased speed. We already mentioned the program advantages. However, requirements of the "Computer" have essentially grown on a range of numbers and their type such as on its memory size, which at that time was of low speed. In the developed "Computer", they have incorporated two important features: The numbers stored in the memory had a word length of 32 bits, and the memory size consisted of 65K words.

In the “Computer”, words could consist of the following types:

- Short number with the fixed point (16 capacity);
- Number with the fixed point (32 capacity);
- Number with the fixed point of double length (64 capacity);
- Short bit line (16 capacity);
- A bit line (32 capacity);
- Number with a floating point (32 capacity);

Other features included the following. All numbers with the floating- and fixed-point representation appear in a complementary code. They used a bit representation for logic variables and applied logical values TRUE=0 and FALSE=1 in logic management. An attempt to execute operation with operands of different types is perceived as a mistake. If the operands involved in operation are different in type, the system will transform them as the more complex type; that is, the result will be with a floating point at addition of number with the fixed point with number with a floating point.

Expressions get in the arithmetic device in the arithmetic-logic kind, consisting of operations, variable with the indexes, set by the direct or indirect address, with the Polish record of arithmetic and logic operations, i.e. built on a priority. A basis of internal language is one of elementary records consisting of eight kinds. Furthermore, it stipulated four base registers for transformation of expressions. The address of memory is formed by the addition of number of pages (contained in the base register) and number of a half-word, which contains in an address part of kinds 1, 2, 7 records.

4 Kinds of Records

There are seven kinds of records as we now show.

(0) Record of the Zero Kind

000	Number of page
3p	13p

Establish the page number in the second base register

(1) Record of the First Kind

001	NBR	TYPE	Number of a half-word
3p	2p	3p	8p

Simple variable or constant addressing
(NBR = number of the base register, TYPE = type of a variable)

(2) Record of the Second Kind

010	NBR	TYPE	Number of a half-word
3p	2p	3p	8p

Variable addressed by indexing

(3) Record of the Third Kind

011	CI (code instruction)
3p	5p

Operation

(4) Record of the Fourth Kind

100	CI	Code of shift
3p	5p	8p

Shift operation

(5) Record of the Fifth Kind

101	CF
3p	5p

Designates codes of standard functions and managing symbols

(6) Record of the Sixth Kind

110	NBR	TYPE	OPERAND
3p	2p	3p	16-32

Direct operand

Table 2 shows an example of an assignment statement record in the memory of the system for the following Fortran-similar language.

A[I] = (B + C[J, R] * M[J, K]) * SIN(X) – 1.6

= 7.81516232431

Table 2. Records in memory

A	[	I	]
=	(	B	+
C	[	J	,
R	]	*	M
[	J	,	K
]	)	*	SIN
(	X	)	–
1.6			

Note that from the resulting record in memory we can see that the record is dense; we can superimpose blank bytes in cases of transition only at the beginning of

another problem where the following operator is a label on which there can be a jump command.

5 Operations List Carried Out by the Arithmetic Processor

The arithmetic operations that execute on the algorithmic machine correspond to a set of operations of the usual machine; however, they are added with some operations of translation of one type of data to another. Table 3 shows this.

Table 3. Arithmetic Operations of the Algorithmic Machine

N	Designation	Name	Dates types
1	-	Monadic minus	1, 2, 6
2		Shift	1, 2, 3, 6
3	*	Multiplication	1, 2, 6
4	/	Division	1, 2, 6
5	+	Addition	1, 2, 3, 6
6	-	Subtraction	1, 2, 3, 6
7	ABS	Absolute size	1, 2, 6
8	SIGN	=1, if x>0; 0, if x=0;-1, if x <0	1, 2, 6
9	MOD	X=A+By, B-the whole, 0 <A <y	1, 2
10	.EQ	Equally	1, 2, 3, 6
11	.GT	It is more	1, 2, 3, 6
12	.LT	It is less	1, 2, 3, 6
13	.NOT	Logic NOT	4, 5
14	.AND	Conjunct	4, 5
15	.OR.	Disjunction	4, 5
16	.XOR.	Logic excluding OR	4, 5
17		Transformation 1 and 2 types in 3	
18		Transformation 1 and 2 types in 6	
19		Transformation 3 types in 6	
20		Transformation 6 in 1 and 2 types	
21		Transformation 6 types in 3	
22		Transformation 3 types in 1 and 2	

Table 4 shows a list of standard functions and managers of the symbols which are carried out in the arithmetic processor.

6 Instruction Structure

An instruction represents the operator, received after a translation of the command, which appeared in the high-level language. The length of the instruction can be any size, a multiple of half-bytes. Each following instruction begins with the half-byte following the end of the previous instruction. We make exceptions with the first program instruction and the instruction to which programs at performance

management is transferred. Their record always begins with a new cell, irrespective of filling previous cells.

Table 4. Symbol Functions and Managers

N	Function	Name	Special cases
1	LOG (x)	ln x	Trashing
2	EXP (x)	Ex	Trashing
3	SIN (x)		
4	COS (x)		
5	TAN (x)	tg x	Trashing
6	ATG (x)	arctg x	
7	SQRT (x)		x <0
8	ATN (x, y)	arctg (x/y)	y = 0
9	x|y	Xy	x = 0 y <0
10	(		
11	)		
12	,		
13	{		
14	}		
15	=		
16	;		

The first instruction has the following structure:

1	1	Code of the operator	Quantity-in symbols			1st symbol					Sym bols				
				Attribute		Following Attributes						Attributes			
				1 atr.		Following 2 atr.						A trace. 2 atr,			

The first byte of the instruction contains in two senior capacities an attribute at the beginning of the instruction, and in the others - a operation code. The second byte of the instruction includes a code of symbols number in the instruction and the attribute first symbol. Values of the symbols are the following:

00 – Identifier of RAM
01 – Identifier of ROM
10 – Operator or a divider
11 – Constant

The constant as it has been told above, can have length from one up to four bytes. To allocate under a constant standard number of bytes (4) the length of the instruction becomes in a significant amount of cases inefficiently large, with a significant amount on empty bites. To enter attributes of length of a constant, the analysis of the instruction becomes complicated. We have approached this in a different way on conditions of that time. Thus if in first attribute of a symbol was the attribute of a constant instead of second attribute, the attribute of a constant length was stated

followed by the symbol of a constant, instead of following the attribute of the second symbol. If the attribute of a constant appeared in a place of second attribute, the code such as a constant appeared in the second half-byte of attribute in a place of first attribute. In this case, only one further symbol was read. For the symbols, which are distinct from a constant, the order of following became customary. The values of the attributes of a constant length are the following:

00 – 1 byte
01 – 2 bytes
10 – 3 bytes
11 – 4 bytes

The equipment determined the end of the instruction on zeroing account number symbols and on allocating the ends of a code of the instruction.

7 Features of the Managing Device

The performance of the operators in the machine at such internal language demands additional physical size of the equipment. Therefore, in addition to the usual elements of the management devices (for example, start an arithmetic device) in the algorithmic machine appears a set of devices, managing triggers, and counters. All these additional devices represent devices for retrieval and the preliminary processing of instructions. To accelerate the process, two registers act as buffers for consecutive reading the next two instructions words.

Preliminary processing concerns itself with six device entities:

- Arrangement tracking correct passage of the instruction from ROM (calculation number symbols and comparison originally determined)
- Device for forming the address of an operand
- Device for determining type of a symbol
- Device for determining type of a constant
- Microcode, organizing interaction of all devices DM
- Circuit of concurrence (determines concurrence of a symbol to code END and gives out an attribute of the instructions end of the instruction)

8 Some System Statistics

The experimental model of the “Computer” was executed on the 133 series. The arithmetic processor (without the controls circuit) consisted of 412 N-slot, from them on the managements device 138 N-slot were necessary. The time of performance of operation of addition for the first and second type of the data was 0.3 microseconds. For records of the sixth kind, it witnessed speeds from 0.7 up to 9.6 microseconds. Multiplication operations were carried out for records of first and second kind with speeds from 4.9 to 9.7 microseconds and for sixth kind from 5.2 to 10.1microseconds.

Logic operations were carried out for in 0.3 microseconds for first, second, and third kind records, and from 0.6 to 6.8 microseconds for sixth kind records. Logic operations executed in 0.1 microseconds for fourth and fifth kind records for operations such as inversion.

9 Summary Remarks

The results of development, debugging, and testing of work of an experimental sample on the decision computing and solving system problems, we came to the following conclusions.

1. Programming in the Fortran-like language RTL-77 has increased coding productivity in comparison with coding on a computer language by five times.
2. The programs written in the RTL-77 language, in comparison with programs written in a computer language, were reduced in size on the average by 3.3 times.
3. The memory size occupied with the program written on RTL-77 is approximately equal to a memory size occupied with same program written on a computer language, and three to five times smaller than the program written on HLL and compiled code on a computer language.
4. The size occupied by the equipment was 120 without package performance.
5. Speed of the "Computer" was about two million operations per second.

The departmental commission accepting the "Computer" recommended replacing the element base used in the computer by integrated circuits to reduce needful power and volume. However, because of weak development of integrated circuits at that time and the sharp reduction in financing in 1990, work had halted. Currently, this idea is still interesting for developers of computer equipment. The publications appear about development software without programmers with appearance technology "system on a chip" logical depth of operators and machine commands increase. It is desirable to develop method for data in computer in text or vocal forms.

Appendix A

Number of the operator	Operators, the alphabet, the basic concepts (formal record)	Explanations, examples
1 Input (F)	READ (a, n), READ (id, n), READ (a), READ (id), READ (a, w, n),	Ç – devices number-decimal number without a sign, n - number of operator FORMAT, - the identifiers list and cyclic elements (c), c - x = m1, m2, m3, where m1 initial value, m 2 - final value, m3 - a step. If a step = 1 it is possible to omit ид - the identifier of a file id1 id2 – the files identifier, records number. n - operators of input without a format (in the standard form - most frequently meeting in problems) w – the lists name of names of the variables subject to input \ to a conclusion (see operator NAMELIST) is similar to the operator INPUT

Number of the operator	Operators, the alphabet, the basic concepts (formal record)	Explanations, examples
2 Write (F)	WRITE (a, n), WRITE (id, n), WRITE (a), WRITE (id), WRITE (a, w, n)	
	ALLOCATE (id1, id2) SET ALLOCATE (id1)	To make loading a file with name id1 for area of memory with name id2 (Transfer of the information between levels of memory, variables value A=1 when loading is ended.
4 Format	FORMAT (b)	b - formatted expression. The operator is used for the indication of a format of the entered and removed dates and the accompanying text. b (b1 b2 b3 mb4, b5,), where bi - specifies, m – the specifies repeater or formatted group (b1, b2, m (b3, b4, b5,) b6..), m - the whole decimal number without a sign. FORMAT (// A. …) - // - means the passing of 2 lines at a printout. FORMAT (nx, a1, a2...) n blanks before printing the first value or a symbol. b_i specifies can have type and a kind: I - integer, R - real, G –group printing the text, L - a logic variable, F - floating, X – the passing of a position, B - binary. Examples: mIw, mRw. W1, wGzzz---zi where “z” repeat w time, H'zz---z. m-the repeater, w1 - number of digit after a comma, w - the number of symbols (width of a field) FORMAT (E 12,4) - is entered - 824.0123 E-2 means: - 824.0123*10-2
4	The description of variables INTEGER (or begin with letters at absence of the obvious description: I, J, K, L, M, N) REAL DOUBLE PRECISION LOGICAL BINARY	The whole - 640 \0\00\-35 Valid - 04.\-700.\ .015\-.27У2 \6. ô-4 Double accuracy - 0. \ 6D-10\-10.2D4 Logic - .TRUE.\ .FALSE. Binary - 101. IB\-I. 001B\ - I0IE |-10B it is used in a combination to descriptions REAL, INTEDGER, DOUBLE, PRECISION
5	The alphabet: A | B |... |Z letters 0|1|2 |... |9 figures .T. | .F. | =, TRUE. | .FALSE E a designation of logic operations; designations of arithmetic operations: + | - | * | ** | / | +,-, x:; signs on operations of the attitude(relation): .GT. | .GE. | .LT. | .LE. | .EQ. | .NE. or>,>, <, <, =, =; designations of logic operations: .OR. | .AND. | .NOT. | or: "or", "¿", "not"; dividers:, |. | = | (|); indexes of following: GO TO | IF | CONTINUE | PAUSE | RETURN | STOP | WAIT; Descriptors and operators (see corresponding items(points))	
6	The Identifier	Any line of letters and the figures, beginning with the letter, no more than 7 symbols: B | F | A*B, ALFA|TIME (identifier " TIME " concerns to the reserved identifiers)

Number of the operator	Operators, the alphabet, the basic concepts (formal record)	Explanations, examples
7	LINE	- Any sequence of symbols, the prisoner in apostrophes, or with previously wHzz---z = ’ zzz---z ’ ‘ zz---z ’ B - a binary line (from "1" and "0")
8	FILE A (i, j, r...)	A - the identifier of a file, i, j, r-identifiers of variables - indexes or constants
9	The description of file DIMENTION A (k1, k2, … kn)	k1 - the whole decimal number - the top border of a files index. Dimension of a file = to number of indexes n.
10 (F)	The operator appropriate A1... An = E	E - expression (arithmetic, logic) Ai - identifiers of variables by which (or to which) E.Primer's value it is appropriated(given): A1 = TIME - in a cell A1 contents timers value (circuit realization) are remembered
11	The operator of appropriate LABEL ASSIGN n TO m	m - the variable accepts value of a label (the integer without a sign)
12 (F)	GO TO n	The operator of unconditional jump
13	GO TO (n1, n2, … nk)	Jump under the instruction (see operator ASSIGN)
14	GO TO (n, n, … n), m	Calculated jump. m should appear in the left part of the operator of giving m = j, where 1 <j <k
15 (F)	IF (E) n1, n2, n3	The conditional arithmetic operator of jump GO TO n1 - if E> 0 n2 - if E =0 n3 - if E <0
16	IF (L) Q	The conditional logic operator. L - logic expression, Q - the operator - not D0 and not IF
17 (F)	THE OPERATOR OF CYCLE DO n x = m1, m2, m3,	m 1, m2, m3 - initial value, final value and a step accordingly for XX. If m 3=1 it is possible to not write it(him). The cycle goes up to the operator with a label n inclusive. This operator cannot be IF, DO, RETURN, STOP.
18	CONTINUE	The operator of continuation. Carries out jump to the following operator. One of operators replaces at inadmissible connection of adjacent operators
19	PAUSE n	Stop the working program. By pressing button " Start-up " - work proceeds from a place stopping. The label n is given out on the consol
20	STOP n	Stop without an opportunity of start, only button " Start-up "
21 (Rt, pairs)	WAIT (E, k)	E - the variable of type the INTEGER - specifies real time, which is necessary to wait before to continue a course of the program (following for WAIT the operator). To - a constant such as INTEGER, specifying required accuracy of readout of an interval in мкsec
22 (Rt, pairs)	WAIT b1, b2, b … bi	bi - the logic variables, following for WAIT, the operator will be executed, when all bi begin TRUE
23 (Rt, pairs)	WAIT (ид R, A)	IdR - the identifier reserve the register addressed. And - a variable of type binary (BINARY). The subsequent operator will be executed, when (idR) = A Is checked at each interruption
24 (Сис)	Assignment of priority PRIORITY (I=E)	I - the identifier of the program or a branch (TASK), E - a constant or an integer variable.

Number of the operator	Operators, the alphabet, the basic concepts (formal record)	Explanations, examples
25	Descriptor of branch TASK and (A, z, v)	The description of a problem, the identifier, A - a variable of type a priority, z - a variable such as lock-the lock (can be absent). v - a variable such as MASK - a mask.
26	Descriptor of program PROGRAM (I, z, v)	a-the identifier of the program, comes to an end END. z, v that, as in 25, can be absent
27	ATTACH (I, y, r)	A name of a branch (TASK) which can go in parallel with the causing program (if there is a corresponding equipment). I - value of a priority at the moment of input (the number should be appropriated). R- a variable such as EVENT - a condition of inclusion of a branch. Can be absent R and V. y - a variable such as KEY a key.
28 (F)	CALL a name, Y (n1, …, nk)	Start of procedure, procedure of function of the subroutine with a name. n1, nk - values of actual parameters, y - the variable such as KEY, can be absent.
29 (Sis)	ON a b c	The operator of the reactions task interruption. a - a name of one of the situations accepted in system; b - the identifier (SNAP) which can be absent. If it is, the information on a condition of system is given out to the operator at the moment of interruption; C-the operator (for instans GO TO M), specifying reaction of system to interruption - the program with a name "æ" or the subroutine will be started. If “c” is absent, the system reacts in the standard way.
30 (Sys)	REVERT a	The cancellation of reaction to interruption (on ON a …) also restores the reaction specified in the covering block: ON b ON a REVENTA.. Operates b
31 (Pairs)	KEEP a (r1, r2, … rn)	The operator who is carrying out preservation of values of a condition of the program a for variables, whose identifiers are specified in (…), there can be names of registers and time
32	FREE a (Y)	To clear area of the RAM from the information which have been written down in area with a name "a." Y type KEY.
33	FREE (a1, a2, … an)	To clear the RAM from the variables specified in brackets.
34	COMMON | r1 | A, B, C | r2 | R, T, Z …	The operator allocates the common memory with a name r1 for variables A, B, C, (they will be stored(kept) in one area of memory by way of the list), with a name r2 for variables R, T, Z, etc. If the common area is one, it can be not marked. Identifiers in the list can be identifiers of files, but then their characteristics should be set in the operator of description DIMENTION, or in operator COMMON |r1 | A, B (10, 5), C (see item 9)

Number of the operator	Operators, the alphabet, the basic concepts (formal record)	Explanations, examples
35 (Pairs)	EQVIVALENCE (r1, r2)	The variables specified in brackets of operator EQVIVALENCE, will be placed on the same place (in cells) memory. For example: at DIMENTION (r1 (53), r2 (15)) files demand an identical place, but they should not be used simultaneously except for a mark of the same file by different identifiers in different programs for reading.
36	FILE a1 (m1, l1, t1, z1,), a2 …	The description of a file. a1, a2, … identifiers of files or libraries (LIB), mi - quantity of records in a file a i; li - the maximal length of record in a file, ti - the maximal length of the record following on the order after the termination(ending) of reading or record; zi - a variable such as lock.
37	OPEN a1 (Y1), a2 (Y2)	To open files a1, a2, … - on AM (auxiliary memory. Yi - variables such as key
38	CLOSE a1 (Y1), a2 (Y2)	To close files a1, a2, … - on AM Zi Yi can be absent in 36, 37, 38
39	ALLOCATION (a1)	Gives out value '1' B, if the variable a1 is placed in the RAM (ALLOCATE (a1)) and '0' In - otherwise.
40	Descriptors of accommodation of the data in memory AUTOMATIC (a1, a2, …) STATIC (a1, a2, …) CONTROLLED (a1, a2, …)	If there is no description for type of accommodation of a variable type AUTOMATIC is meant. Accommodation of variables in the RAM before performance of the program invariable before its(her) end. Accommodation dynamic occurs before each input(entrance) in a segment (in sense Fortran the common) or in the block (a LIFO principle) It is control operator ALLOCATE
41	NAMELIST |w1 |, A1, B1 …, | w2 |, A2, B2, …, |wi |, Ai, Bi, …	wi - the name of the list of names Ai, Bi, … wi is used in operators WRITE, READ for reduction of record of entered or deduced(removed) variables.
42	DATA a1, a2, (n1, n2, …,)	a1, a2, … - the names list of variables by which numerical values n1 are appropriated, n2, … by way of conformity. The element ai can be an element such as a cycle: (ai (I)), where I=m1, m2, m3 …. For example, If ni repeats some times successively enters the name so: m*ni - number of repetitions on each m-cycle.

Number of the operator	Operators, the alphabet, the basic concepts (formal record)	Explanations, examples
43	The subroutine - function FANCTION B (x1, x2, …)	t - or it is empty, or a descriptor such as function, B - a name of function, (x1, x2, …) - the list of formal parameters. The result will be worn out on identifier B. In (a1, a2, …) can meet in operators, that causes calculation B for values of actual parameters (a1, …) the body in words RETURN and END.
44	The Subroutine (procedure) SUBROUTINE P Y (x1, x2,..)	P - a name of the subroutine Terminates. The call goes on operator CALL P (n1, n2, …). Comes to an end RETURN and END. Y - variable such as mask (in 43 and 44 can be absent).
45	Descriptor EXTERNAL a1, a2, …	This descriptor is used in case arguments of function or the subroutine can be names of functions or subroutines. In this case in the basic program the descriptor (45) in which names of external functions and procedures which should be handed as actual parameters are listed is included.
46	The operator simulating a situation of interruption SIGNAL a	á - a name of one of situations of interruption which can arise in system. The operator simulates occurrence of a situation "á" and transfers management to operator ON working in the given area of the program. If the situation specified in a, is not included, that is operator ON where there is a parameter and interruptions will not be is not executed yet. Management will be handed to the following operator for SIGNAL a. This will take place and after the termination of processing of interruption ON .If a case of performance of operator SIGNAL a situation CONDITION (a) and if operator ON a, b, c has instead of and operator CONDITION (a) follows as it mentioned above, interruption is developed.
47 (Def)	The description of "lock" LOCK Z (n)	z (n) - the file of binary numbers being "lock" which should be open by "key" KEY Y (n) (see 48). "Lock" is established in programs, files and branches.
48 (Def)	The description of "key" KEY Y (N)	Y (n) - a file of the binary numbers representing "key". It(he) is compared to "lock" LOCK z (n)
49	The operator of a supply of file REWIND a	The file with a name "a" in a condition when 1-n record can be read out or be deduced(removed) is established.
50	The operator of return BACK PACE a	Return from the current record of a file to a place of interruption is carried out.
51 (Sys)	The operator of search of record of file FIND (an)	Record of a file "a" with number n is established in position when input-output of this record will demand minimal time
52 (Sys)	To place in library PUT INTO LIB (r1, r2, … rn) (m1, m2, … mn)	To place variables from the program with the specified name in which there was operator PUT INTO LIB, r1, r2... rn, in library in cells m1, m2, … mn,
53 (Sys)	To take from library GETEROM LIB name Y (m1, m2, … mr) (r1, r2, … rr)	To choose from library the program with the identifier "name" and contents of cells m, m, … m to place in cells r, r, … r

Number of the operator	Operators, the alphabet, the basic concepts (formal record)	Explanations, examples
54 (Sys)	Reservation and the name of a place in RAM AREA a name, z (n, b)	At performance of this operator in the RAM the area from n words (n - decimal number) if the OS will find enough place not closed by operations system (or more high priority the program) and if it is not specified "b" - the identifier of a variable which accepts value of the physical address of a cell of the RAM is allocated(removed). If "b" is present at operator AREA OS checks availability of area in the size "n", since a cell "b".
55 (Sys)	MASK V	The description variable V as a binary variable for masking the register of interruptions.
56	CHECK (x1, x2, …,)	Interruption of performance of the program at an output on identifiers or the labels specified in the list. Start-up from the consol.