MEMORANDUM
RM-3804-PR
SEPTEMBER 1963

SOVIET CYBERNETICS TECHNOLOGY: III, PROGRAMMING ELEMENTS OF THE BESM, STRELA, URAL, M-3, AND KIEV COMPUTERS

Translated by A. S. Kozak
Edited by Willis H. Ware and Wade B. Holland

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-3804-PR
SEPTEMBER 1963

SOVIET CYBERNETICS TECHNOLOGY: III, PROGRAMMING ELEMENTS OF THE BESM, STRELA, URAL, M-3, AND KIEV COMPUTERS

Translated by A. S. Kozak
Edited by Willis H. Ware and Wade B. Holland

This research is sponsored by the United States Air Force under Project RAND—contract No. AF 49(638)-700 monitored by the Directorate of Development Planning, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA

PREFACE

This Memorandum contains a translation from the Russian detailing the instruction formats for five of the better known Soviet digital computers. This is probably one of the most poorly documented areas of Soviet computer technology, and with recent announcements of Soviet intentions to offer the Ural computer for sale in Western markets, it is of interest to determine the exact nature of the command structures utilized on Soviet machines.

The present translation was undertaken in order to bring together in one place descriptions relative to several computers as given by a single source. It is therefore felt that the descriptions from machine to machine are quite consistent, and do not exhibit the ambiguities which have characterized research into this subject in the past where different authors have discussed different machines.

This Memorandum is essentially a translation, not a detailed analysis. However, some notes are included in the RAND Editors' Introduction which will help place the machines in perspective. The RAND editors have also prepared charts which give the operation codes for the five machines. These charts are presented with the original Russian statements and the English translations in parallel columns, which should be of interest to translators, researchers, and others caught up in the intricacies of Soviet computer terminology and, especially, abbreviations.

The series Soviet Cybernetics Technology is part of a continuing program of research in computer technology conducted by The RAND Corporation, under U. S. Air Force Project RAND. A list of related publications of The RAND Corporation is found on p. 69. All translations undertaken at RAND are registered with the Office of Technical Services, Department of Commerce, Washington 25, D.C.

SUMMARY

This Memorandum is a translation of the Appendix of the book, Elements of Programming.* The book itself is a general introduction to the subject of electronic computing, with particular emphasis on serving as a guide to programmers. It is intended for use by university students and students in advanced technological institutions.

The interesting feature of the book is the Appendix, which details the command structures and systems of notation for the five Soviet computers: BESM, Strela, Ural, M-3, and Kiev. Additionally, some specifications for each of the computers are included.

The Introduction prepared by the editors of the Soviet Cybernetics Technology series contains summaries of the operations codes for the five machines in chart form. It also contains comments on the problems one encounters in trying to make analyses of the Soviet state of the art, pointing out the vast differences that are found when comparing two different authors' comments on the same machine.

---

*Elementy Programmirovaniya, B. V. Gnedenko, V. S. Korolyuk, E. L. Yushchenko, Fizmatgiz, Moscow, 1961, 346 pp. Appendix translated at The RAND Corporation by A. S. Kozak.

CONTENTS

RAND EDITORS' INTRODUCTION

This translation of the Appendix of Elements of Programming presents the programming fundamentals for five Soviet computers from one viewpoint and in a comparable manner. This is its principal value. It would be presumptive to claim that the notations given here are absolutely correct, since for each of the computers covered (with the exception of the M-3, for which there is only a small amount of literature available) different sources give divergent operation formats. In some cases, only minor inconsistencies are to be noted, while in others, the schemes are radically different. But the present article, dealing with several computers from the same standpoint, employing a consistent approach to the subject matter, is of some value for purposes of comparison, even if it isn't accepted as the final word for the machines with which it deals.

There are several possible explanations for the lack of agreement among different Soviet authors who purport to speak authoritatively on the same machine. First, in many instances we suspect that, although not explicitly stated, these articles sometimes deal with locally modified versions of a machine. Further, in many cases the names of Soviet computers are really generic terms; if an author uses only the broad designation for the machine he is describing, some analysis is necessary to determine whether

he is talking about a particular model (e.g., in the Ural series there is the Ural-1, the Ural-2, and the Ural-4), about a design prototype, or is actually considering the series as a whole. The BESM machine, about which much has been written, is an excellent example of an aggravated case of inconsistencies in the literature. BESM-I never got beyond the prototype stage, and the production model is referred to as BESM-II (or, by some authors, the BESM-2). The modifications which triggered the separate designation of the II-model evolved over a period of several years, so that some of the earlier articles on the BESM-II describe something midway between the two models. Some authors fail to draw any distinction between the two; they simply talk about the BESM that was current at the time they wrote their article.

There has never been the emphasis in the Soviet design centers on software that U.S. manufacturers exhibit. Soviet computers are products of computing centers at universities and in the various Academies of Sciences. The logical design and the construction of the machines are carried out separately from the programming and software development, the latter often being handled by a computing mathematics faculty or comparable group. Also, it often happens that laboratory-constructed versions of a particular machine are shipped out to other universities or computing centers, where they are fitted out with auxiliary components according to the specific intended application. The local center

often develops its own software, which probably accounts for some of the variations noted in the literature. The remaining mystery, though, is why there is such a reluctance to make this factor clear in writing about such a machine.

We touched briefly above on the BESM, one of the machines covered in this translation. This article is an example of an author's failure to distinguish between BESM-I and BESM-II. The machine being specified in the opening paragraphs at first appears to be a cross between the two versions. The date, 1951, is definitely indicative of BESM-I vintage, since by 1959 the II-version was only at the prototype stage. But, the stated speed (8000-10,000 opns/sec) and the core store size (2047 words) are figures associated with BESM-II.* If the 1951 date is accepted as simply the time when work first started on the initial version of the machine, this article can probably be safely identified with BESM-II. In any case, it is doubtful that the authors would have been quite so concerned with BESM-I, since it was never actually produced. (As far as the software is concerned, the question is perhaps academic, since the two versions are understood to utilize the same programming techniques.† The differences

---

*See item 1 in the Bibliography for a comparison of the BESMs.

†Ibid., p. 63.

between the two are mainly in the auxiliary units and in the elegance of the construction techniques.)

The Ural under consideration is the Ural-1. The Ural-2 and Ural-4 have since been developed, but the Ural-1 does not seem to have been obsoleted, although it is quite primitive by present standards. New literature has appeared on Ural-1 as recently as late-1962,* in which mention is made of the more recent 2- and 4-models. (This points out another interesting facet of the Soviet technology: Even though a machine might be succeeded by improved versions, it is not thereby considered obsolete, nor does its "refinement" cease. It is likely, although there is no direct evidence one way or the other, that Ural-1 machines are still being manufactured.) But, even though we can with a high degree of assurance identify the Ural in this work as the initial model in the series, the op codes here detailed are substantially at odds with another, more recent, survey.†

It should be pointed out that in preparing the accompanying charts (pp. xiv-xxi, below) of the operations of the five computers, only information derived from Elements of Programming has been used. Not as much information is given in some cases as in others, reflecting the differing levels

---

*Gavrilenko, E. T., et al., Programmirovanie dlya elektronnoj vychislitel'noj mashiny "Ural-1" [Programming for the "Ural-1" Computer], Mashgiz, Moscow, 1962.

†Ibid., pp. 261-63.

of detail among the five machines covered. Although by drawing on other sources, it would have been perhaps possible to maintain the same depth of detail throughout, this was felt unwise, again, because of the probability of introducing inaccuracies in trying to reformulate information from other sources to make it consistent with the approach adopted by the editors of Elements of Programming. It will be readily noted that there are columns on the BESM and Kiev charts (labeled "Designation") which are missing on the charts for the other three machines. In this column is given the mnemonic representation of the operation. In the cases where letter designations (as opposed to mathematical symbols) are used, these two charts have the additional, and very useful, function of defining standard Russian abbreviations--many of which are frequently encountered in the literature, but are rarely defined.*

Finally, a bibliography of other RAND Corporation publications in the area of Soviet cybernetics and computer technology is found on p. 69. Further information on some of the machines covered here is to be found in items 1 and 4.

---

*In a related project, work is underway at RAND on the formulation of glossaries of Russian-English computer terminology and of Soviet computer abbreviations.

Table I--Command Structure for the BESM Computer

БЭСМ				BESM	
Назначение	Обозначен.	№п No.	№ кода опер. Op. code	Designat'n	Description
I. ОПЕРАЦИИ ПЕРЕДАЧИ УПРАВЛЕНИЯ С МЕСТНОГО НА ЦЕНТРАЛЬНОЕ И С ЦЕНТРАЛЬНОГО НА МЕСТНОЕ				I. OPERATIONS OF CONTROL TRANSFER FROM LOCAL TO CENTRAL AND FROM CENTRAL TO LOCAL	
Передача на местное управление.	ПМУК	1	$'K_0$=011000	PMUK	Transfer to local control.
Переход на центральное управление команд.	ПЦУК	2	$'K_2$=011001	PTsUK	Branch to central command control.
Изменение счетчика местного управления.	ИМУК	3	$'K_0$=011010	IMUK	Change local [command] control counter.
Изменение счетчика центрального управления командами.	ИЦУК	4	$'K_0$=011011	ITsUK	Change central command control counter.
II. ОПЕРАЦИИ ПРЕОБРАЗОВАНИЯ И ПЕРЕСЫЛКИ КОДОВ				II. CODE CONVERSION AND TRANSFER OPERATIONS	
Сложение.	+	1	$'K_0$=000001	+	Addition.
Сложение с блокировкой нормализации.	, +	2	$'K_0$=100001	, +	Addition without normalization.
Вычитание.	-	3	$'K_0$=000010	-	Subtraction.
Вычитание с блокировкой нормализации.	, -	4	$'K_0$=100010	, -	Subtraction without normalization.
Умножение.	x	5	$'K_0$=000011	x	Multiplication.
Умножение с блокировкой нормализации.	, x	6	$'K_0$=100011	, x	Multiplication without normalization.
Деление.	:	7	$'K_0$=000100	:	Division.
Сложение порядков.	+ П	8	$'K_0$=000101	+ P	Addition of exponents.
Сложение порядков с блокировкой нормализации.	, + П	9	$'K_0$=100101	, + P	Addition of exponents without normalization.
Вычитание порядков.	- П	10	$'K_0$=000110	- P	Subtraction of exponents.
Вычитание порядков с блокировкой нормализации.	, - П	11	$'K_0$=100110	, - P	Subtraction of exponents without normalization.
Изменение порядка по адресу.	ИПА	12	$'K_0$=000111	IPA	Change in the exponent according to the address.
Изменение порядка по адресу с блокировкой нормализации.	,ИПА	13	$'K_0$=100111	,IPA	Change in the exponent according to the address without normalization.
Умножение с выводом удвоенного количества разрядов.	x а	14	$'K_0$=001000	x a	Multiplication with output of twice the number of bits.
	x б	15	$'K_0$=001001	x b	
То же, что и 14 и 15, но с блокировкой нормализации.	, x а	16	$'K_0$=101000	, x a	Same as 14 and 15, but without normalization.
	, x б	17	$'K_0$=101001	, x b	
Деление с выводом остатка.	: а	18	$'K_0$=001010	: a	Division with output of the remainder
	: б	19	$'K_0$=001011	: b	
Передача числа нормальная.	ПЧ	20	$'K_0$=001100	PCh	Normal transfer.
Передача числа нормальная с блокировкой нормализации.	,ПЧ	21	$'K_0$=101100	,PCh	Normal transfer without normalization.
Передача числа на печать.	,ПЧТ	22	$'K_0$=101100*	,PChT	Transfer for printout.
Передача числа с регистров пульта управления.	ПЧР	23	$'K_0$=001100*	PChR	Transfer from console control registers.

Передача числа с регистров пульта управления с блокировкой нормализации.	,ПЧР	24	$'K_0$=101100*	,PChR	Transfer from console control registers without normalization.
Передача числа с изменением знака.	ПЧ-	25	$'K_0$=001101	PCh-	Transfer with a change in the sign.
Передача числа с изменением знака и с блокировкой нормализации.	,ПЧ-	26	$'K_0$=101101	,PCh-	Transfer with a change in the sign and without normalization.
Передача модуля числа.	|ПЧ|	27	$'K_0$=001110	|PCh|	Transfer of the modulus.
Передача числа по модулю с блокировкой нормализации.	,|ПЧ|	28	$'K_0$=101110	,|PCh|	Transfer according to the modulus without normalization.
Передача числа с изменением знака в зависимости от знака другого числа.	ПЧ±	29	$'K_0$=001111	PCh±	Transfer with a sign change depending on the sign of another number.
Передача числа с изменением знака в зависимости от знака другого числа с блокировкой нормализации.	,ПЧ±	30	$'K_0$=101111	,PCh±	Transfer with a sign change depending on the sign of another number, but without normalization.
Передача порядка числа.	↓	31	$'K_0$=010000	↓	Transfer of the exponent.
Передача порядка числа с блокировкой нормализации.	,↓	32	$'K_0$=110000	,↓	Transfer of the exponent without normalization.
Сдвиг с блокировкой порядков.	←	33	$'K_0$=010001	←	Shift without exponents.
Сдвиг по всем разрядам.	,←	34	$'K_0$=110001	,←	Shift over all bits.
Логическое умножение.	Л	35	$'K_0$=011101	L	Logical multiplication.
Сложение команд.	СК	36	$'K_0$=010010	SK	Addition of commands.
Циклическое сложение.	,СК	37	$'K_0$=110010	,SK	Cyclical addition.
Выделение целой части.	ЦЧ	38	$'K_0$=010011	TsCh	Division of an integral.
III. ОПЕРАЦИИ ПЕРЕДАЧИ УПРАВЛЕНИЯ				III. CONTROL TRANSFER OPERATIONS	
Сравнение с учетом знаков.	<	1	$'K_0$=010100	<	A comparison taking the signs into account.
Сравнение на равенство кодов.	,<	2	$'K_0$=110100	,<	A comparison of equality.
Сравнение по модулю.	|<	3	$'K_0$=010101	|<	Comparison according to the modulus
IV. ОПЕРАЦИИ ОБРАЩЕНИЯ К ВНЕШНИМ УСТРОЙСТВАМ				IV. OPERATIONS FOR USING EXTERNAL UNITS	
Команды обращения к внешним запоминающим устройствам (магнитная запись).	МЗа МЗб	1 2	$'K_0$=010110 $'K_0$=010111	MZa MZb	Commands for return to external memory units (magnetic recording).
Останов условный (по тумблеру).	ост.усл.	3	K_0=011100	ost.usl.	Conditional stop (by a tumbler).
Останов.	ост.	4	K_0=011111	ost.	Stop.

*See text, p. 9.

Table II--Command Structure for the Strela Computer

СТРЕЛА			STRELA
Назначение	№п No.	№ кода опер. Op. code	Description
I. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ			I. ARITHMETIC OPERATIONS
Сложение.	1	$'k_0 = 01$	Addition.
Вычитание.	2	$'k_0 = 03$	Subtraction.
Вычитание модулей.	3	$'k_0 = 04$	Modulus subtraction.
Умножение.	4	$'k_0 = 05$	Multiplication.
Сложение порядков.	5	$'k_0 = 06$	Addition of exponents.
Вычитание порядков.	6	$'k_0 = 07$	Subtraction of exponents.
Перенос числа с присвоенным знаком другого числа.	7	$'k_0 = 10$	Carry-over of a number with the sign taken from another number.
Сложение чисел без округления.	8	$'k_0 = 12$	Addition without rounding.
Контрольное суммирование кодов.	9	$'k_0 = 17$	Control summing of codes.
Специальное сложение (для команд).	10	$'k_0 = 02$	Special addition (for commands).
Специальное вычитание (для команд).	11	$'k_0 = 15$	Special subtraction (for commands).
II. ЛОГИЧЕСКИЕ ОПЕРАЦИИ			II. LOGICAL OPERATIONS
Поразрядное логическое умножение.	1	$'k_0 = 11$	Digital logical multiplication.
Поразрядное логическое сложение.	2	$'k_0 = 13$	Digital logical addition.
Сдвиг.	3	$'k_0 = 14$	Shift.
Поразрядная операция "неравнозначно."	4	$'k_0 = 16$	The digital operation "non-equivalent."
Сравнение и останов при несовпадении.	5	$'k_0 = 26$	Comparison and stop for noncoincidence.
III. КОМАНДЫ УСЛОВНОГО ПЕРЕХОДА И ПРЕДВАРИТЕЛЬНЫЕ КОМАНДЫ ГРУППОВЫХ ОПЕРАЦИЙ			III. CONDITIONAL BRANCHING COMMANDS AND PRELIMINARY COMMANDS FOR GROUP OPERATIONS
Условный переход первого типа.	1	$'k_0 = 20$	Conditional branch of the first type.
Условный переход второго типа.	2	$'k_0 = 27$	Conditional branch of the second type.
Предварительные команды групповых операций.	3	$'k_0 = 30\text{-}37$	Preliminary commands for group operations.
IV. КОМАНДЫ ОБРАЩЕНИЯ К ВНЕШНИМ УСТРОЙСТВАМ			IV. COMMANDS FOR THE USE OF EXTERNAL UNITS
Подвод под считывающую головку зоны $'k_1$ магнитной ленты.	1	$'k_0 = 25$	Placement of zone $'k_1$ of the magnetic tape under reading head.
Перенос кодов с магнитной ленты в ячейки памяти.	2	$'k_0 = 43$	Transfer from magnetic tape to core memory.
Перенос кодов внутреннего ЗУ на магнитную ленту.	3	$'k_0 = 46$	Transfer from core storage to magnetic tape.
Перенос кодов с перфокарт во внутреннюю память.	4	$'k_0 = 41$	Transfer from punched cards to core storage.
Перенос кодов из ячеек ЗУ на перфокарты.	5	$'k_0 = 44$	Transfer from core storage cells to punched cards.
Перенос кодов из одних ячеек ЗУ в другие.	6	$'k_0 = 45$	Transfer from certain core storage cells to others.
Групповая передача с контролем.	7	$'k_0 = 60$	Controlled group transfer.
Операция останова.	8	$'k_0 = 40$	Halt.

V. СТАНДАРТНЫЕ ПРОГРАММЫ ПОСТОЯННОЙ ПАМЯТИ			V. STANDARD ROUTINES
Вычисление обратной величины.	1	$'k_0 = 62$	Computation of a reciprocal.
Вычисление квадратного корня.	2	$'k_0 = 63$	Computation of square root.
Вычисление показательной функции.	3	$'k_0 = 64$	Computation of an exponential function.
Вычисление логарифма.	4	$'k_0 = 66$	Computation of a logarithm.
Вычисление синуса.	5	$'k_0 = 67$	Computation of a sine.
Вычисление функции arctg.	6	$'k_0 = 73$	Computation of an arctangent.
Вычисление функции arcsin.	7	$'k_0 = 74$	Computation of an arcsine.
Перевод кодов из двоичной системы счисления в двоично-десятичную.	8	$'k_0 = 70$	Conversion from binary to binary-decimal.
Обратная операция.	9	$'k_0 = 72$	The reverse operation.

Table III--Command Structure for the Ural-1 Computer

УРАЛ			U R A L
Назначение	№п No.	№ кода опер. Op. code	Description
I. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ			I. ARITHMETIC OPERATIONS
Сложение.	1	$'k_0 = 01$	Addition.
Засылка на сумматор.	2	$'k_0 = 02$	Transfer to the accumulator.
Вычитание.	3	$'k_0 = 03$	Subtraction.
Вычитание модулей.	4	$'k_0 = 04$	Modulus subtraction.
Умножение с накоплением в сумматоре.	5	$'k_0 = 05$	Multiplication with storage in accumulator.
Умножение.	6	$'k_0 = 06$	Multiplication.
Деление.	7	$'k_0 = 07$	Division.
Циклическое сложение.	8	$'k_0 = 26$	Cyclical addition.
II. ЛОГИЧЕСКИЕ ОПЕРАЦИИ			II. LOGICAL OPERATIONS
Присвоение знака числа содержимому сумматора.	9	$'k_0 = 10$	Assumption of the sign of a number by the contents of the accumulator.
Сдвиг содержимого регистра r сумматора.	10	$'k_0 = 11$	Shift of the contents of register r of the accumulator.
Поразрядное логическое умножение.	11	$'k_0 = 12$	Digital logical multiplication.
Поразрядное логическое сложение.	12	$'k_0 = 13$	Digital logical addition.
Поразрядная логическая операция "неравнозначно ≅".	13	$'k_0 = 14$	The digital logical operation "non-equivalent ≅".
Операция нормализации.	14	$'k_0 = 15$	Normalization.
Посылка из сумматора по адресу.	15	$'k_0 = 16$	Transfer from the accumulator according to the address.
Посылка на регистр сумматора.	16	$'k_0 = 17$	Transfer to the accumulator register.
Посылка на сумматор.	17	$'k_0 = 20$	Transfer to the accumulator.
III. ОПЕРАЦИИ ПЕРЕДАЧИ УПРАВЛЕНИЯ			III. CONTROL TRANSFER OPERATIONS
Условная передача управления.	18	$'k_0 = 21$	Conditional control transfer.
Безусловная передача управления.	19	$'k_0 = 22$	Unconditional control transfer.
Передача управления по ключу.	20	$'k_0 = 23$	Control transfer by key.
Начало групповой операции.	21	$'k_0 = 25$	Beginning of a group operation.
Конец групповой операции.	22	$'k_0 = 24$	End of a group operation.
Изменение команд.	23	$'k_0 = 30$	Command change.
Останов и засылка на сумматор.	24	$'k_0 = 37$	Stop and transfer to the accumulator.
IV. ОПЕРАЦИИ ОБРАЩЕНИЯ К ВНЕШНИМ УСТРОЙСТВАМ			IV. OPERATIONS FOR USE OF EXTERNAL UNITS
Подготовительная команда для передачи кодов.	25	$'k_0 = 31$	Preparatory command for transfer operation.
Печать содержимого сумматора.	26	$'k_0 = 32$	Printout of accumulator contents.
Пропуск одной строки на бумажной ленте. .	27	$'k_0 = 34$	Skip one line on the paper tape.

Table IV--Command Structure for the M-3 Computer

М - 3			М - 3
Назначение	№п No.	№ кода опер. Op. code	Description
I. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ			I. ARITHMETIC OPERATIONS
	1	$'k_{01} = 0$	
	2	$'k_{01} = 1$	
	3	$'k_{01} = 2$	
	4	$'k_{01} = 3$	
	5	$'k_{01} = 4$	
	6	$'k_{01} = 5$	
	7	$'k_{01} = 6$	
	8	$'k_{01} = 7$	
II. КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ			II. CONTROL TRANSFER COMMANDS
Безусловный переход по первому адресу.	1	$'k_0 = 24$	Unconditional transfer according to the first address.
Безусловный переход по первому адресу с печатью содержимого регистра r.	2	$'k_0 = 64$	Unconditional transfer according to the first address with a printout of the contents of register r.
Безусловный переход по второму адресу.	3	$'k_0 = 74$	Unconditional transfer according to the second address.
Условный переход.	4	$'k_0 = 34$	Conditional transfer.
Останов.	5	$'k_0 = 37$	Stop.
III. ОПЕРАЦИИ ОБРАЩЕНИЯ К ВВОДУ (ВЫВОДУ) И ПЕРЕНОСА			III. INPUT (OUTPUT) AND TRANSFER OPERATIONS
Ввод с перфоленты.	1	$'k_0 = 07, 27$	Input from perforated tape.
Перенос.	2	$'k_0 = 05, 15$	Transfer.
Перенос с печатью.	3	$'k_0 = 45, 55$	Transfer with printout.

Table V--Command Structure for the Kiev Computer

КИЕВ				KIEV	
Назначение	Обозначен.	№п No.	№ кода опер. Op. code	Designat'n	Description
I. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ				I. ARITHMETIC OPERATIONS	
Сложение.	+	1	$'k_0 = 01$	+	Addition.
Вычитание.	−	2	$'k_0 = 02$	-	Subtraction.
Сложение команд.	$+_1$	3	$'k_0 = 03$	$+_1$	Addition of commands.
Вычитание модулей.	|-|	4	$'k_0 = 06$	|-|	Modulus subtraction.
Циклическое сложение.	Ц+	5	$'k_0 = 07$	Ts+	Cyclical addition.
Умножение без округления.	x	6	$'k_0 = 10$	x	Multiplication without rounding.
Умножение с округлением.	⊗	7	$'k_0 = 11$	⊗	Multiplication with rounding.
Деление.	:	8	$'k_0 = 12$	:	Division.
II. ЛОГИЧЕСКИЕ ОПЕРАЦИИ				II. LOGICAL OPERATIONS	
Нормализация.		9	$'k_0 = 35$		Normalization.
Сдвиг логический.		10	$'k_0 = 13$		Logical shift.
Поразрядное логическое сложение.	∨	11	$'k_0 = 14$	∨	Digital logical addition.
Поразрядное логическое умножение.	∧	12	$'k_0 = 15$	∧	Digital logical multiplication.
Поразрядная логическая операция "неравнозначно."	≃	13	$'k_0 = 17$	≃	The digital logical operation "non-equivalence."
III. ОПЕРАЦИИ ПЕРЕДАЧИ УПРАВЛЕНИЯ				III. CONTROL TRANSFER OPERATIONS	
Условная передача управления по равенству модулей.	=	14	$'k_0 = 16$	=	Conditional control transfer according to equality of moduli.
Условная передача управления по соотношению "меньше или равно."	≤	15	$'k_0 = 04$	≤	Conditional control transfer in relation to "less or equal."
Условная передача управления по соотношению "меньше или равно" без учета знаков.	|≤|	16	$'k_0 = 05$	|≤|	Conditional control transfer in relation to "less or equal" without considering the signs.
Условная передача управления по знаку числа.	УПЧ	17	$'k_0 = 31$	UPCh	Conditional control transfer according to the sign of a number.
Условный переход на подпрограмму.	УПП	18	$'k_0 = 32$	UPP	Conditional transfer to a subroutine.
Переход по регистру возврата.		19	$'k_0 = 30$		Transfer according to the return register.
IV. ОПЕРАЦИИ ОБРАЩЕНИЯ К ВНЕШНИМ УСТРОЙСТВАМ				IV. OPERATIONS FOR USE OF EXTERNAL UNITS	
Ввод чисел с перфоленты (ПЛ).	ВЧ	20	$'k_0 = 20$	VCh	Data input from perforated tape.
Ввод команд с перфоленты.	ВК	21	$'k_0 = 21$	VK	Input of commands from perforated tape.
Вывод кодов на ПЛ.	ВПЛ	22	$'k_0 = 22$	VPL	Output of data onto perforated tape.
Обмен кодами ОЗУ и внешним ЗУ (МБ) в режиме "запись."	· МБЗ	23	$'k_0 = 23$	MBZ	Exchange of codes between core and external units (magnetic drums) in the "write" mode.

Обмен кодами ОЗУ и МБ в режиме "считка."	МБЧ	24	$'k_0 = 24$	MBCh	Exchange of data between core storage and the magnetic drums in the "read" mode.
Подготовительная операция для операций "МБЗ" и "МБЧ", обеспечивающая надлежащую подводку магнитного барабана.		25	$'k_0 = 25$		Preparatory operation for the MBZ and MBCh operations, ensuring proper feed from/to the magnetic drum.
Останов.		26	$'k_0 = 33$		Stop.
V. ОПЕРАЦИИ МОДИФИКАЦИИ АДРЕСОВ				V. ADDRESS MODIFICATION OPERATIONS	
Начало групповой операции.	НГО	27	$'k_0 = 26$	NGO	Beginning of a group operation.
Конец групповой операции.	КГО	28	$'k_0 = 27$	KGO	End of a group operation.
Заполнение регистра А по фиксатору.	Ф	29	$'k_0 = 34$	F	Load register A according to locator.

ELEMENTS OF PROGRAMMING

B. V. Gnedenko, V. S. Korolyuk, E. L. Yushchenk[redacted]
[A. S. Kozak, Translator]

APPENDIX

BESM

The BESM high-speed electronic computer, built in 1951 under the direction of Academician S. A. Lebedev at the Academy of Sciences, USSR, uses 39-bit words and has floating point. Six bits are used for the exponent of a number (the first bit is the sign of the exponent), and 33 bits are for the mantissa (the first bit is the sign of the mantissa), as shown in Fig. 21.

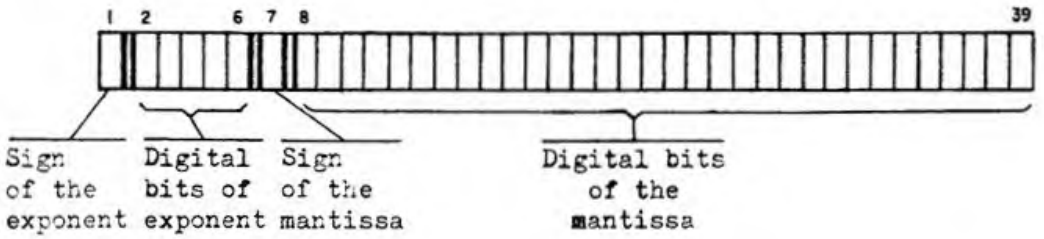

Fig. 21.

A three-address command system is used, where six bits are for coding the type of operation; each address uses 11 bits.

The speed of the computer is about 8000-10,000 operations per second. Core storage has a capacity of

2047 cells, and auxiliary storage is on two magnetic drums and four magnetic tapes. The capacity of a drum is 5120 words; the capacity of a tape is 30,000 words (the tapes are replaceable). Input is from punched cards and perforated tape, and output is onto punched cards or paper tape.

Unlike a number of other Soviet computers (Strela, Kiev, Ural), the BESM has two controls--central and local, providing convenience in the use of subroutines.

In addition to command register K, there are two command counters in the BESM:*

C_1 is the command counter for central control;

C_2 is the command counter for local control.

In addition there is a control trigger T. A 1 in trigger T corresponds to the central control mode and a 0 to local control.

Separate machine cycles are used for execution of command 'K--that is, the command whose code is contained in command register K--and for sending the code of the next command to this register.

For a description of the operations of the BESM, we shall separate and designate the parts of command register K. (See Table 10.)

---

*Here we designate only those registers and switches needed to describe the basic operations of the computer.

Table 10

Bits of the BESM command register (from left to right)	Designation and name of bit groups (registers)
1 - 6	K_0 -- operation code register
7 - 17	K_1 -- I address register
18 - 28	K_2 -- II address register
29 - 39	K_3 -- III address register

In addition we shall designate the $i^{\underline{th}}$ bit of register K by ε_i when necessary.

Figure 22 shows the composition of the command register (and the cells in which the command is written).

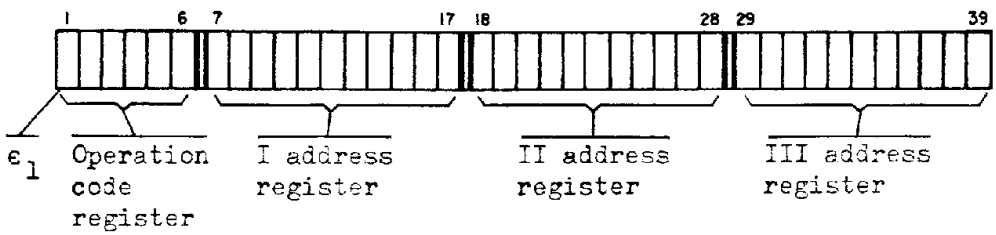

Fig. 22.

For recording $'K_0$ we shall use binary notation. The permissible set of values of $'K_0$ will be 000000, 000001, ...,111111. We shall designate the conditional halt switch in the computer by T_1.

Ordinary execution of a command (arithmetic, logical, and auxiliary commands for computational purposes) consists of:

1) properly converting the codes and sending them according to the address; i.e., execution of an operation of the form

$$f(^2K_1, {}^2K_2) = {}'K_3;$$

2) adding a 1 to counter C_1 if $'T = 1$, or to counter C_2 if $'T = 0$; i.e.,

$$'C_1 + T \Rightarrow C_1;$$
$$'C_2 + (1 - 'T) \Rightarrow C_2;$$

3) branching to the next command whose number is in counter C_1 if $'T = 1$, or in counter C_2 if $'T = 0$:

$$'('C_1'T + 'C_2\ (1 - 'T)) \Rightarrow K.$$

Thus the special operations of the BESM are those which change the state of control trigger T.

I. OPERATIONS OF CONTROL TRANSFER FROM LOCAL TO CENTRAL AND FROM CENTRAL TO LOCAL

1. $'K_0 = 011000$ (PMUK)--transfer to local control. By this command:

a) $0 \Rightarrow T$;

b) $^2C_2 \Rightarrow K$.

Thus, control is transferred to local; commands are executed beginning with the number in local control counter C_2.

2. $'K_2$ = 011001 (PTsUK)--branch to central command control. By this command:

a) $1 \Rightarrow T$;

b) $^2C_1 \Rightarrow K$.

Thus, control is transferred to central; commands are executed beginning with the address in central control counter C_1.

Here the contents of registers K_1, K_2, K_3 are insignificant.

3. $'K_0$ = 011010 (IMUK)--change local [command] control counter:

a) $0 \Rightarrow T$;

b) $'K_3 \Rightarrow C_2$;

c) $^2C_2 \Rightarrow K$.

Thus, if an operation is carried out in central control, it then transfers to local and commands are executed beginning with address $'K_3$. If an operation is carried out in local control, it then continues, but beginning with address $'K_3$. The contents of registers K_1 and K_2 do not influence the result of the operation.

4. $'K_0$ = 011011 (ITsUK)--change central command control counter:

a) $'C_1{}'T + 'C_2(1 - 'T) + 1 \Rightarrow ('K_2)_3$; $011011 \Rightarrow ('K_2)_0$;

b) $1 \Rightarrow T$;

c) $'K_3 \Rightarrow C_1$;

d) $^2C_1 \Rightarrow K$.

An operation is executed in the same manner as above; however, the command with operational code ITsUK* is sent according to address $'K_2$; in the third address is the address of the following command, increased by 1 (point a). The contents of register K_1 do not affect the result of the operation.

The operation IMUK is convenient for executing a branch to subroutines. For this purpose the basic program is written in central control, and the subroutines in local. Branching from central to local control for a subroutine is coded by the command IMUK. Here the address of the first command of the subroutine is entered into IIIA [third address] of the command IMUK.

For return to the basic program from a subroutine, there is the operation PTsUK, which permits return to the interrupted place in the basic program (to the command following the command IMUK, from which the branch to the subroutine was effected). Here it is obvious that in the process of executing a subroutine, the execution of commands which transfer control from local back to central must be taken into special account. Thus, if during execution of a certain subroutine it becomes necessary to transfer to the program, the command ITsUK can be used.

---

*Remember that $(\ )_3$ designates bits which correspond to the third address of a cell; the address is indicated in the parentheses ( ).

By means of this command, a counter indicator of the central control command (the place of exit from the basic program to the subroutine) can be fixed in the III address of the cell whose address is indicated in K_2.

Let us now proceed to a list of the remaining BESM operations.

II. CODE CONVERSION AND TRANSFER OPERATIONS

1. $'K_0$ = 000001 (+)--addition:

a) $^2K_1 + {}^2K_2 \Rightarrow {}'K_3$;

b) $'C_1 + 'T \Rightarrow C_1$; $'C_2 + (1 - 'T) \Rightarrow C_2$;

c) $'('C_1\ 'T + 'C_2\ (1 - 'T)) \Rightarrow K$.

Prior to the addition of numbers, their exponents are equalized. The result is normalized and rounded off.

2. $'K_0$ = 100001 (, +) is the same as $'K_0$ = 000001, but without normalization.

The following commands are completed in the same manner:

3. $'K_0$ = 000010 (-)--subtraction.

4. $'K_0$ = 100010 (, -)--subtraction without normalization.

5. $'K_0$ = 000011 (x)--multiplication.

6. $'K_0$ = 100011 (, x)--multiplication without normalization.

7. $'K_0$ = 000100 (:)--division.

8. $'K_0$ = 000101 (+ P)--addition of exponents. The exponent of the number $^2K_2$ is added to the number $^2K_1$, and

the normalized result is sent according to address $'K_3$.

9. $'K_0$ = 100101 (, + P)--the same as the preceding operation, but without normalization.

10 and 11. $'K_0$ = 000110 and $'K_0$ = 100110 (- P) and (, - P)--subtraction of exponents. The same as for $'K_0$ = 000101 and $'K_0$ = 100101, only the operation being executed is subtraction.

12. $'K_0$ = 000111 (IPA)--a change in the exponent according to the address.

The number $'K_2$ is added to the exponent of number $^2K_1$, and the normalized result is sent according to address $'K_3$. The sixth bit of K_2 is taken as the sign bit.

13. $'K_0$ = 100111 (,IPA)--the same as for $'K_0$ = 000111, but without normalization.

14 and 15. $'K_0$ = 001000 (x a) and $'K_0$ = 001001 (x b)--multiplication with an output of twice the number of bits. The operation is executed with two commands: by the command $'K_0$ = 001000, which is multiplication of the numbers $^2K_1$ and $^2K_2$ without rounding, with normalization, and with output of the first 32 bits of the result with their exponent according to address $'K_3$; and by the command (following it) $'K_0$ = 001001, which is normalization of 32 low-order bits of the accumulator (with the remaining exponent from the preceding command), and output according to address $'K_3$ ($'K_1$ and $'K_2$ in the latter command are not used).

16 and 17. $'K_0$ = 101000 and $'K_0$ = 101001--the same as 14 and 15, but without normalization.

18 and 19. $'K_0$ = 001010 (: a) and $'K_0$ = 001011 (: b)--division carried out with output of the remainder, similarly as in the preceding operation.

20. $'K_0$ = 00110 (PCh)--normal transfer of a number. The number $^2K_1$ is normalized and transferred according to address $'K_3$.

a) $(^2K_1)^n \Rightarrow 'K_3$,

b) and c) are the same as for addition; $'K_2$ is not used. The normalized code $^2K_1$ is designated by $(^2K_1)^n$.

21. $'K_0$ = 101100 (,PCh)--normal transfer of a number without normalization.

22. $'K_0$ = 101100 and $'\varepsilon_{19} = 1$ (,PChT)--transfer of a number for printing. $^2K_1$ is printed.

23. $'K_0$ = 001100 and $'\varepsilon_{26} = 1$ or $'\varepsilon_{27} = 1$, or $'\varepsilon_{28} = 1$ (PChR)--transfer of a number from the console control registers.

If $'\varepsilon_{26} = 1$, the code from the trigger control register is transferred; if $'\varepsilon_{27} = 1$, transfer is from the second diode register; if $'\varepsilon_{28} = 1$, transfer is from the first diode register.

24. $'K_0$ = 101100 and $'\varepsilon_{26} = 1$, or $'\varepsilon_{27} = 1$, or $'\varepsilon_{28} = 1$ (,PChR)--transfer of a number from the console control registers without normalization.

25. $'K_0$ = 001101 (PCh-)--transfer of a number with a sign change. A normalized number $^2K_1$ with the opposite sign is transferred according to address $'K_3$. $'K_2$ is not used:

$$(-\ ^2K_1)_n \Rightarrow\ 'K_3.$$

26. $'K_0$ = 101101 (,PCh-)--transfer of a number with a sign change and without normalization.

27. $'K_0$ = 001110 (|PCh|)--transfer of the modulus of a number. The normalized number $^2K_1$ is transferred according to the modulus to address indicated by $'K_3$

$$|^2K_1|^n \Rightarrow\ 'K_3.$$

28. $'K_0$ = 101110 (,|PCh|)--transfer of a number according to the modulus without normalization.

29. $'K_0$ = 001111 (PCh$\pm$)--transfer of a number with a sign change depending on the sign of another number. The transferred number $^2K_1$ is normalized; the sign of this number is reversed, if the number $^2K_2$ is negative, and retained in the opposite case; the result is sent according to address $'K_3$.

30. $'K_0$ = 101111 (,PCh$\pm$)--transfer of a number with a sign change depending on the sign of another number, without normalization.

31. $'K_0$ = 01000 ($\downarrow$)--transfer of the exponent of a number. The exponent of the number $^2K_1$ is in the form of a normalized number with its own exponent, and is trans-

ferred according to address 'K_3. 'K_2 is not used.

32. 'K_0 = 110000 (,↓). Transfer of the exponent of a number without normalization.

33. 'K_0 = 010001 (←)--shift without exponents. The shift of the number $^2K_1$ by the number of bits equal to the number 'K_2, is to the left if sign 'K_2 = 0, and to the right if sign 'K_2 = 1. Here 'ε_{22} is taken as the sign bit (the 7th bit of register K_2) (the code of the exponent is neither shifted nor transferred); the result is sent according to address 'K_3.

34. 'K_0 = 110001 (,←)--shift over all bits. The same as for 'K_0 = 010001, but the shift occurs over all bits.

35. 'K_0 = 011101 (L)--logical multiplication. Digital logical multiplication of codes $^2K_1$ and $^2K_2$ with placement of the result according to address 'K_3.

36. 'K_0 = 010010 (SK)--addition of commands. Addition of the digital parts of the numbers $^2K_1$ and $^2K_2$ is carried out (including the signs of the numbers); the exponent of the second number is not taken into account. The result (without normalization) is sent according to address 'K_3.

37. 'K_0 = 110010 (,SK)--cyclical addition. Codes $^2K_1$ and $^2K_2$, considered as one integer, are summed up cyclically (with transfer from the sign bit of the number to the 1st bit of the exponent and from the exponent sign

bit to the low-order bit of the digital part of the number); the result is placed according to address $'K_3$.

38. $'K_0$ = 010011 or 110011 (TsCh)--division of an integral. The integral part of the number $^2K_1$ with its sign is sent according to address $'K_3$ in the form of a number with a fixed point after the low-order bit. Its fractional part, reduced to a zero exponent, is sent according to address $'K_2$.

III. CONTROL TRANSFER OPERATIONS

In addition to the control transfer operations cited in the beginning, the following operations, which maintain the operating functions at either local or central control, are found in BESM.

1. $'K_0$ = 010100 (<)--a comparison taking the signs into account:

$$\begin{array}{lll} & & 'K_3\,(1 - 'T) + 'C_2'T \Rightarrow C_2; \\ \text{a)} & R\,\{^2K_1 < {}^2K_2\} & 'K_3'T + 'C_1\,(1 - 'T) \Rightarrow C_1; \\ & & 'C_1 + 'T \Rightarrow C_1;\ 'C_2 + (1 - 'T) \Rightarrow C_2; \end{array}$$

b) $'('C_1'T + 'C_2\,(1 - 'T)) \Rightarrow K.$

Thus, if $^2K_1 < {}^2K_2$, control is transferred according to address $'K_3$; if $^2K_1 \geq {}^2K_2$, control is transferred to the next command in sequence. The following two operations are completed analogously.

2. $'K_0$ = 110100 (,<)--a comparison of equality:

$$'K_3(1-'T) + 'C_2'T \Rightarrow C_2;$$

a) $R\ \{{}^2K_1 \neq {}^2K_2\}\ 'K_3'T + 'C_1(1-'T) \Rightarrow C_1;$

$$'C_1 + 'T \Rightarrow C_1;\ 'C_2 + (1-'T) \Rightarrow C_2;$$

b) $'('C_1'T + 'C_2(1-'T)) \Rightarrow K.$

3. $'K_0$ = 010101 (|<|)--a comparison according to the modulus:

$$'K_3(1-'T) + 'C_2'T \Rightarrow C_2;$$

a) $R\ \{|{}^2K_1| < |{}^2K_2|\}\ 'K_3'T + 'C_1\ (1-'T) \Rightarrow C_1;$

$$'C_1 + 'T \Rightarrow C_1;\ 'C_2 + (1-'T) \Rightarrow C_2;$$

b) $'('C_1'T + 'C_2(1-'T)) \Rightarrow K.$

IV. OPERATIONS FOR USING EXTERNAL UNITS

1 and 2. $'K_0$ = 010110 (MZa) and $'K_0$ = 010111 (MZb)--commands for return to external memory units (magnetic recording). Operations are completed in two commands. The first and last numbers of the external memory unit code to which the operation refers, and the first address of the internal memory unit cell with which the operation is concerned, are indicated in the addresses of the commands. The nature of the operation is additionally determined by the first address of the MZa command (recording, reading, or rewinding; drum or tape; the number of the drum or tape). Thus, provision is made for return to perforated tape, magnetic drum, and magnetic tape.

3. K_0 = 011100 (ost. usl.)--conditional stop (by a tumbler). A stop occurs if special tumbler T_1 is switched on ('T_1 = 1), and in the opposite case, the next command is completed; i.e.,

a) R {'T_1 = 1} ost. [stop] 'C_1 + 'T = C_1; 'C_2 + (1-'T) = C_2;

b) '('C_1'T + 'C_2(1-'T) = K.

4. K_0 = 011111 (ost.)--stop.

A stop occurs with the corresponding signaling.

STRELA

The Strela, a general-purpose automatic digital computer, has 43-bit words. Data is displayed in the system in floating point with 36 bits designated for the binary mantissa (the first bit is the sign of the mantissa), and 7 bits for the binary exponent (the first bit is the sign of the exponent) (Fig. 23).

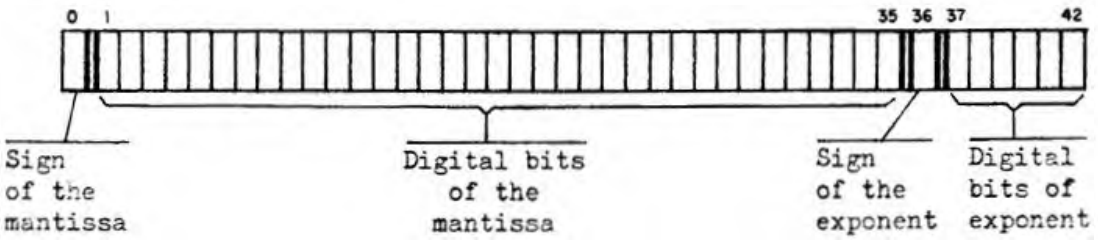

Fig. 23.

We shall call 0 through 35 the mantissa register, and 36 through 42 the exponent register, and shall designate the cells with an index of either m [mantissa] or p [exponent].

For storing data in a binary-decimal word, 37 bits (one is the sign) are used for the mantissa, and 6 bits (one is the sign) for the exponent. The 36 digital bits of the mantissa are divided into 9 tetrads, one for each decimal place.

The command system is three-address with a normal order of execution. For writing a command in a memory cell, each address occupies 12 bits, the operational code uses 6 bits, and one bit is for the control sign. The distribution of bits over a cell in which a command is written is shown in Fig. 24.

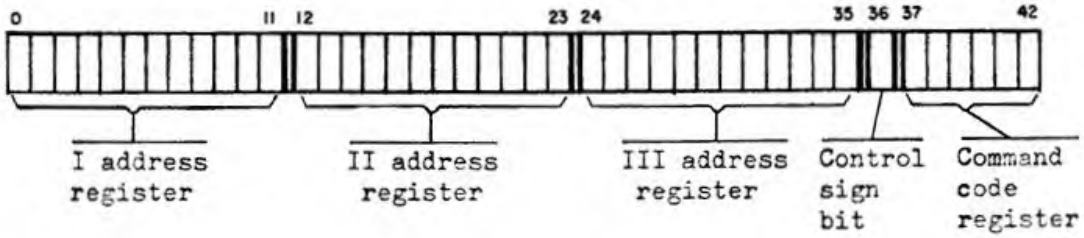

Fig. 24.

The Strela high-speed computer completes 2000-3000 operations per second. Internal storage, with a capacity of 2048 cells, consists of cathode-ray tubes. In addition to internal storage, there is a permanent storage. Here is located the so-called constants-yielding unit (UVK), in which certain frequently encountered constants are permanently stored; numbers from 7400 to 7777 are allocated to these cells. Also in the permanent storage is a standard subroutines store (NSP), in which subroutines for computation of elementary functions ($\frac{1}{x}$, e^x, sin x, ln x and others) are located.

The auxiliary storage is located on two reels of ferromagnetic tape. The magnetic tapes are divided into zones. One tape can have up to 511 zones; on each zone up to 2048 digits can be recorded. Accordingly, magnetic tape zones are numbered from 4001 to 4777, and from 5001 to 5777. Reading codes from each zone can be done in any groups, beginning with the first code.

Input is from punched cards, and output is onto punched cards. There is a unit for transfer of information from punched cards to printout. Each word (data or command) is punched in a card in the form of one line; in all, 12 words are punched into a card.

For punching commands, each octal digit is punched in the form of a triad of binary digits; the control sign is punched in the form of one binary digit. For punching decimal numbers each decimal digit is punched in the form of a tetrad, and the sign is in the form of one binary number. Conversion of numbers into binary is automatic.

For a description of operations we need the following designations:

1) s--the command counter,
2) K--the command register,
3) G--the command register for a group operation,
4) A--the address modification register,

5) ω--the trigger for indication of conditional transfer,

6) T--the operating condition toggle switch.

We shall divide command register K, corresponding to the given placement of commands (see Fig. 24), into bit groups (registers) (see Table 11).

Table 11

Bit groups of the command register	Designation and name of bit groups (registers)
0 - 11	k_1 - register of I address
12 - 23	k_2 - register of II address
24 - 35	k_3 - register of III address
36	α - control sign bit
27 - 42	k_0 - register of the command code

The command register for a group operation G (and the cells) is broken down into such registers as needed. They are designated by the letter "G" (or by the number of the cell) with a corresponding index (0, 1, 2, 3).

The indicated registers can contain octal codes:

$$'k_1, 'k_2, 'k_3 = 0000;\ 0001;\ \ldots;\ 7777;$$

$$'\alpha = 0;\ 1;$$

$$'k_0 = 00;\ 01;\ \ldots;\ 77.$$

However, $'k_1$, $'k_2$, $'k_3$, in all except certain specific cases, are less than 4000 (2048 in a decimal system), which corresponds to the ZU [storage] capacity of the computer.

Provision has been made for a so-called controlled stops system, transfer to which is provided by tumbler T('T = 1). In this event, after the execution of each command in which $'\alpha = 1$, the computer stops. In other words, after the completion of each command, the following action is executed:

$$R\{'\alpha \times 'T = 1\}\ \text{ost.}\ [\text{stop}]$$

Depending on the code of $'k_0$, subsequent operations are carried out.

I. ARITHMETIC OPERATIONS

1. $'k_0 = 01$--addition. The computer completes the following actions:

a) $^2k_1 + {}^2k_2 \Rightarrow 'k_3$;

b) $R\{{}^2k_3 \leq -0\}\ \begin{matrix}1 \Rightarrow \omega;\\ 0 \Rightarrow \omega;\end{matrix}$

c) $'s + 1 \Rightarrow s$;

d) $^2s \Rightarrow K$.

Action a) denotes addition of codes $^2k_1$, $^2k_2$ with normalization of the result and its transfer according to address $'k_3$; b) is the processing of the signal ω (used in conditional control transfer commands; see below);

c) and d) denote transfer to the next command, and are the same for all arithmetic operations. In addition to this, a stop occurs upon overflow--i.e., if the exponent of the result is greater than 77; if, however, it is smaller than the minimum permissible (77), than the result is considered to be zero. Therefore action a) here, as in other arithmetic operations, must be written in greater detail in the form:

$$\text{a)}\quad R\,\{(^2k_1 + {}^2k_2)_p > 77\}\ \text{ost. [stop]};$$

$$R\,\{(^2k_1 + {}^2k_2)_p > -77\}\ {}^2k_1 + {}^2k_2 \Rightarrow {}'k_3;$$
$$0 \Rightarrow k_3.$$

2. $'k_0 = 03$--subtraction.
3. $'k_0 = 04$--modulus subtraction.

These operations are carried out the same as in addition, but with a corresponding change in the sign of the operation as in a).

4. $'k_0 = 05$--multiplication:
   a) $^2k_1 \times {}^2k_2 \Rightarrow {}'k_2$;
   b) $R\,\{|^2k_3| \geq 1\}\ \begin{matrix}1 \Rightarrow \omega;\\ 0 \Rightarrow \omega;\end{matrix}$
   c) and d).
5. $'k_0 = 06$--addition of exponents and
6. $'k_0 = 07$--subtraction of exponents:
   a) $'('k_1)_m \Rightarrow ('k_3)_m$;
   $'('k_1)_p \pm {}'('k_2)_p \Rightarrow ('k_3)_p$;

b) $R\ \{('k_3)_p \geq 1\}$ $\begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) and d).

7. $'k_0$ = 10--carry-over of a number with the sign taken from another number:

a) $|{}^2k_1| \vee \text{sign}\ {}^2k_2 \Rightarrow 'k_3$;

b) $R\ \{'('k_3)_p \geq 1\}$ $\begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) and d).

8. $'k_0$ = 12--addition of numbers without rounding:

a) ${}^2k_1 + {}^2k_2 \Rightarrow 'k_3$;

b) $R\ \{{}^2k_3 = 0\}$ $\begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) and d).

9. $'k_0$ = 17--the control summing of codes ${}^2k_1$ and ${}^2k_2$, understood as indivisible binary codes, with carry-over from the high-order bit to the low-order bit:

a) ${}^2k_1(\text{Ts} +){}^2k_2 \Rightarrow 'k_3$;

b) $0 \Rightarrow \omega$;

c) and d).

10. $'k_0$ = 02--special addition (for commands) and

11. $'k_0$ = 15--special subtraction (for commands):

a) $'('k_1)_i \pm '('k_2)_i \Rightarrow ('k_3)_i$, where i = 1,2,3;

$'('k_1)_0 \Rightarrow ('k_3)_0$;

b) $0 \Rightarrow \omega$;

c) and d).

II. LOGICAL OPERATIONS

1. $'k_0 = 11$--digital logical multiplication:

   a) $(^2k_1) \wedge (^2k_2) \Rightarrow 'k_3$ (step-by-step)

   b) $R\{^2k_3 = 0\}$ $\begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

   c) and d).

2. $'k_0 = 13$--digital logical addition:

   a) $(^2k_1) \vee (^2k_2) \Rightarrow 'k_3$ (step-by-step)

   b) $R\{^2k_3 = 0\}$ $\begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

   c) and d).

3. $'k_0 = 14$--shift. Two modifications of this operation are possible. If $'k_2 < 4000$--i.e., if it is the address of the operative cell--then the code $^2k_1$ is shifted by the number of bits equal to the exponent of the number $^2k_2$; if the sign of the exponent is +, the shift is to the left, and if it is -, the shift is to the right. If $'k_2 = 4000 + \ell$ or $'k_2 = 5000 + \ell$, then the shift is ℓ bits to the left. If however, $'k_2 = 4100 + \ell$ or $'k_2 = 5100 + \ell$, then the shift is ℓ bits to the right ($\ell < 77$).

Designating the shift of the code β by α bits by $\beta \Leftrightarrow \alpha$ (left, if $\alpha > 0$, or right if $\alpha < 0$), we shall write the operation in address form:

a) 1) R {'k_2 < 4000} $^2k_1 \Rightarrow$ '('k_2) $\Rightarrow$ 'k_3;

2) R {'k_2 < 4100} $^2k_1 \Rightarrow$ ('k_2 - 4000) $\Rightarrow$ 'k_3;

3) R {'k_2 < 5000} $^2k_1 \Rightarrow$ (4100 - 'k_2) $\Rightarrow$ 'k_3;

4) R {'k_2 < 5100} $^2k_1 \Rightarrow$ ('k_2 - 5000) $\Rightarrow$ 'k_3;
$^2k_1 \Rightarrow$ (5100 → 'k_2) $\Rightarrow$ 'k_3;

b) R {$^2k_3 = 0$} 1 $\Rightarrow \omega$;
0 $\Rightarrow \omega$;

c) and d).

4. 'k_0 = 16--a digital operation "non-equivalent" (see Chapter II):

a) ($^2k_1 \tilde{=} (^2k_2) \Rightarrow$ 'k_3 (step-by-step);

b) R {$^2k_3 \neq 0$} 1 $\Rightarrow \omega$;
0 $\Rightarrow \omega$;

c) and d).

5. 'k_0 = 26--comparison and stop for noncoincidence:

a), b), d) are the same as in 4;

c) R {$\omega = 1$} ost. [stop]
's + 1 $\Rightarrow$ c.

III. CONDITIONAL BRANCHING COMMANDS AND PRELIMINARY COMMANDS FOR GROUP OPERATIONS

1. 'k_0 = 20--conditional branch of the first type:

a) R {'ω = 0} '$k_1 \Rightarrow$ s;
'$k_2 \Rightarrow$ s;

b) $0 \Rightarrow {}'k_3$;

c) ${}^2s \Rightarrow K$.

2. ${}'k_0 = 27$--conditional branch of the second type:

a) $R\{{}'\omega = 0\}$ ${}'k_1 \Rightarrow s$;
${}'k_2 \Rightarrow s$;

b) ${}'s + 1 \Rightarrow ({}'k_3)_1$;
${}'s + 1 \Rightarrow ({}'k_3)_2$;
${}'k_3 \Rightarrow ({}'k_3)_3$;
$20 \Rightarrow ({}'k_3)_0$.

Point a) is conditional transfer according to ω. Point b) sends the unconditional branch command according to address $({}'k_3)$.

b) ${}^2s \Rightarrow K$.

3. ${}'k_0 = 30, 31, 32, 33, 34, 35, 36, 37$--preliminary commands for group operations.

Group operations consist of two commands: a preliminary command and an execution command (the code of the latter is an arbitrary command of an arithmetic or logical operation).

A number, which is 1 less than the number of completions of the execution command, is shown in the II address in the preliminary command.

The operation code of the preliminary command contains information concerning the rule for modifying addresses of an execution command. In addition to this,

the address indicating where a 0 is to be sent, is shown in the III address of the preliminary command.

Let us designate by ε_1, ε_2, ε_3, respectively, the 40th, 41st, and 42nd bits of register K (that is, the three low-order bits of the register for the operation code k_0). Group operations are described by the following program:

a) $0 \Rightarrow A;\quad 0 \Rightarrow {}'k_3;$

b) $'('s + 1) \Rightarrow G;$

c) $'('G_1 + {}'\varepsilon_1 \cdot {}'A)\theta\ {}'('G_2 + {}'\varepsilon_2{}'A) \Rightarrow {}'G_3 + {}'\varepsilon_3{}'A;$

d) $'A + 1 \Rightarrow A;$

e) $R\{'A \leq {}'k_2\}\quad b);$

f) $'s + 2 \Rightarrow s;$

g) ${}^2s \Rightarrow K.$

Here we designate any arithmetic or logical operation by θ.

Point a) is clearing of modification register A and clearing of a certain cell $'k_3$ (if the latter clearing is not necessary, a zero is placed in the III address).

Point b) sends the execution command to group operations register G.

Point c) is execution of the command contained in register G with address modification according to the three last bits of the operation code of the preliminary command; address G_i, for which $\varepsilon_i = 1$ $(i = 1, 2, 3)$, is increased by the contents of register A (since upon the first completion of this point register A contains zero, the

command is executed the first time in the same form as it was encoded).

Point d) is an increment of the contents of register A by 1.

Point e) is a verification of the end of group operations; as a result action c) will be repeated ('k_2 + 1) times.

Points f) and g) are transfer to the execution of the command, followed by its execution.

IV. COMMANDS FOR THE USE OF EXTERNAL UNITS

1. 'k_0 = 25--placement of zone 'k_1 of the magnetic tape under the reading head.

2. 'k_0 = 43--transfer from magnetic tape [ML] to core memory. The word ('k_2 + 1) from zone 'k_1 of the magnetic tape is transferred to cells 'k_3, 'k_3 + 1, ..., 'k_3 + 'k_2. We write this conventionally in the form

$$'[('k_1)ML] \Rightarrow \begin{pmatrix} 'k_3 \\ 'k_3 + 'k_2 \end{pmatrix} .$$

3. 'k_0 = 46--transfer from core storage to magnetic tape:

('k_2 + 1) code

$$'k_1, 'k_1 + 1, \ldots, 'k_1 + 'k_2$$

is transferred from core to zone $'k_3$ of the magnetic tape.

Conventionally, we shall designate this:

$$'\begin{pmatrix} 'k_1 \\ 'k_1 + 'k_2 \end{pmatrix} \Rightarrow ('k_3)\ \mathrm{ML}.$$

Here the zone of the magnetic tape number $'k_3$ is designated by $('k_3)$ML.

4. $'k_0 = 41$--transfer from punched cards [PK] to core storage. $('k_2 + 1)$ code is transferred from punched cards to cells

$$'k_3,\ 'k_3 + 1, \ldots,\ 'k_3 + 'k_2;$$

$$'\mathrm{PK} \Rightarrow \begin{pmatrix} 'k_3 \\ 'k_3 + 'k_2 \end{pmatrix} .$$

5. $'k_0 = 44$--transfer from core cells to punched cards. $('k_2 + 1)$ code from cells $'k_1$, $'k_1 + 1$, ..., $'k_1 + 'k_2$ is transferred to punched cards:

$$'\begin{pmatrix} 'k_1 \\ 'k_1 + 'k_2 \end{pmatrix} \Rightarrow \mathrm{PK}\ .$$

6. $'k_0 = 45$--transfer from certain cells in core to others. $('k_2 + 1)$ code from cells $'k_1$, $'k_1 + 1$, ..., $'k_1 + 'k_2$ is transferred to cells $'k_3$,

$'k_3 + 1, \ldots, 'k_3 + 'k_2$:

$$'\begin{pmatrix} 'k_1 \\ 'k_1 + 'k_2 \end{pmatrix} \Rightarrow \begin{pmatrix} 'k_3 \\ 'k_3 + 'k_2 \end{pmatrix} .$$

7. $'k_0 = 60$--group transfer with control. Various modifications of this operation are possible. If

$$'k_1 = 0000, \ 0001 \leq 'k_3 \leq 3777,$$

then transfer from punched cards to core is carried out. If

$$0001 \leq 'k_1 \leq 3777, \ 'k_3 = 0000,$$

transfer is from core to punched cards. If

$$0001 \leq 'k_3 \leq 3777 ,$$

$$4001 \leq 'k_1 \leq 4777 \text{ or } 5001 \leq 'k_3 \leq 5777,$$

transfer is from magnetic tape to core. If

$$0001 \leq 'k_1 \leq 3777,$$

$$4001 \leq 'k_3 \leq 4777 \text{ or } 5001 \leq 'k_3 \leq 5777,$$

transfer is from core to magnetic tape. If

$$0001 \leq 'k_1 \leq 3777, \ 0001 \leq 'k_3 \leq 3777,$$

transfer is from certain cells of core to others. A cyclic summation of ($'k_0 = 17$) codes is carried out before and after transfer, simultaneously with execution of the

transfer. The resultant sums are compared and should they not coincide, the computer halts.

8. Operation halt. $'k_0 = 40$. The numbers $^2k_1$ and $^2k_2$ are output to the console.

V. STANDARD ROUTINES

The commands described below are group operations using standard subroutines in a permanent memory. After completion of these subroutines, control is transferred to the next command of the main program. The next command must not be a conditional transfer (codes 20 and 27), since w is not treated in these operations.

1. $'k_0 = 62$--computation of a reciprocal:

$$\frac{1}{'('k_1 + i)} \Rightarrow 'k_3 + i, \text{ where } i = 0, 1, \ldots, 'k_2.$$

2. $'k_0 = 63$--computation of square root:

$$\sqrt{'('k_1 + i)} \Rightarrow 'k_3 + i, \; i = 0,1,\ldots,'k_2.$$

Analogously the following operations are completed:

3. $'k_0 = 64$--computation of an exponential function.
4. $'k_0 = 66$--computation of a logarithm.
5. $'k_0 = 67$--computation of a sine.
6. $'k_0 = 73$--computation of an arctangent.
7. $'k_0 = 74$--computation of an arcsine.

In all of these operations a stop occurs in case of overflow in cell $'k_3 + i$, and in case of inapplicability

of the function to code $'k_1 + i$ in the field of real numbers.

8. $'k_0 = 70$--conversion of codes from binary to binary-decimal notation:

$$'('k_1 + i)_{dec.} \Rightarrow 'k_3 + i, \text{ where } i = 0, 1, \ldots, 'k_2.$$

9. $'k_0 = 72$--the reverse operation:

$$'('k_1 + i)_{bin.} \Rightarrow 'k_3 + i, \; i = 0,1,\ldots,'k_2 .$$

The following actions are also carried out:

$$'s + 1 \Rightarrow s; \; "s \Rightarrow K.$$

URAL

The Ural general-purpose automatic digital computer uses 36-bit full, or 18-bit partially full words. Commands are in a single-address system, and are coded in partially filled cells; five bits are for the operational code, 12 are for the address, and 1 is for changing (modifiability) of the command.

A fixed point system is used; the first bit is the sign.

When data is stored in short cells, the bits are distributed as shown in Fig. 25.

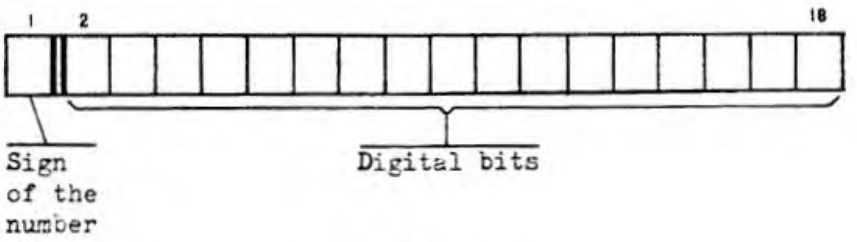

Fig. 25.

The distribution of bits in the storing of data in the so-called long (i.e., full) cells is shown in Fig. 26 for binary codes, and in Fig. 27 for binary-decimal.

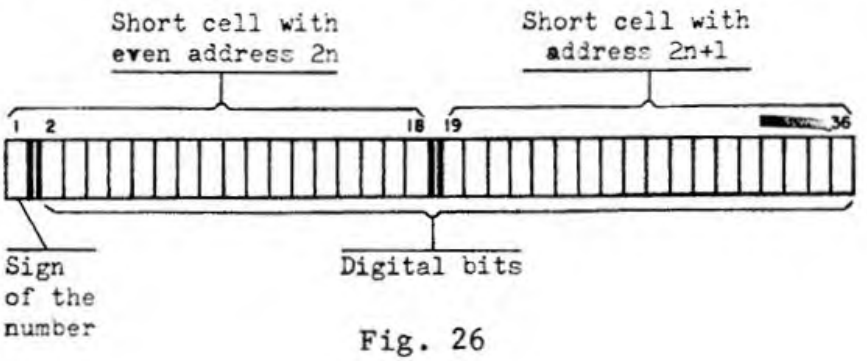

Fig. 26

The speed of the computer is 100 operations per second. The internal memory capacity (on a magnetic drum) is 2048 short cells or 1024 full cells. Full cells have

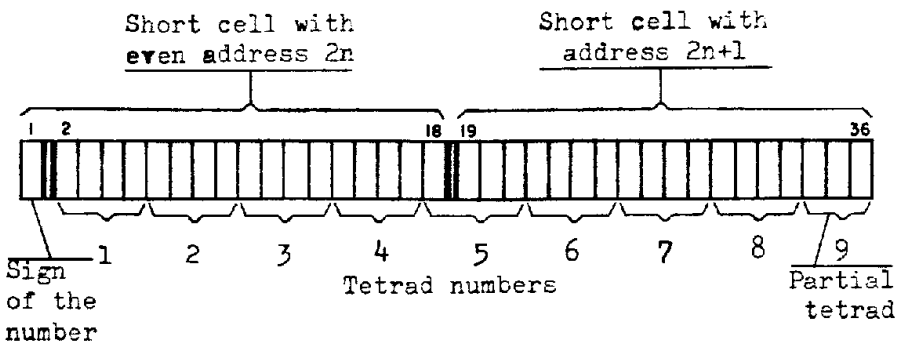

Fig. 27.

only even addresses. Auxiliary storage is on magnetic tapes. Input is from perforated tapes, and output can be onto perforated tape or paper tape.

Special registers and keys of the Ural:

1) S--the accumulator;
2) r--the arithmetic unit register;
3) C--the command counter;
4) K--the command register;
5) ω--the trigger indicator for conditional control transfer;
6) Ts--the cycle register;
7) $K_i (i = 1,2,\ldots,7)$--the keys;
8) T_1--toggle switch 1;
9) T_2--toggle switch 2.

Having absorbed these letter designators, let us proceed to a description of the fundamental operations of the Ural.

A single operation cycle of the Ural consists of:

1) execution of command K--that is, commands whose code is stored in the command register at that time;

2) sending the code of the next command to the command register.

For a description of the set of basic operations of the Ural, we shall present a breakdown of the bits (registers) in the command register (see Table 12).

Table 12

Bits of the Ural command register (left to right)	Designation and use of groups of bits (registers)
1	ε_0--the bit for changing (modifiability) of command
2 - 6	k_0--the operation code register
7	ε--the bit for indicating a full cell
8 - 18	k--the command address register

Fig. 28 shows a breakdown of the command register (and of a cell when a command is stored in it) into groups of bits.

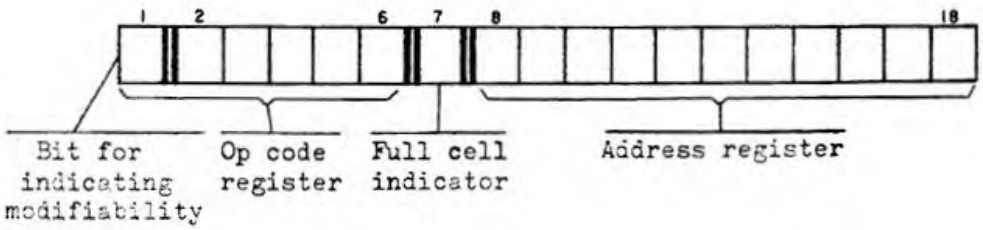

Fig. 28.

The sets of permissible codes for storing in the indicated registers (in octal notation) are the following:

$$'\varepsilon_0,\ '\varepsilon = 0;\ 1;$$
$$'k_0 = 00,\ 01,\ldots,37;$$
$$'k = 0000,\ 0001,\ldots,3777.$$

I. ARITHMETIC OPERATIONS

1. $'k_0 = 01$--addition. The following actions are completed by this command:

a) $'S + '('k - '\varepsilon_0 'Ts) \Rightarrow S;$

$\text{sign } 'S \vee |'C'k - '\varepsilon_0 'Ts)| \Rightarrow r;$

b) $R\ \{\text{sign } 'S = 1\}\ \begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) $'C + 1 \Rightarrow C;$

d) $^2C \Rightarrow K.$

If $'\varepsilon_0 = 0$--i.e., there is no indication of varying the command--the operation $'S + {}^2k \Rightarrow S$ is executed.

If, however, $'\epsilon_0 = 1$, the address is modified; i.e., it is decreased by the contents of the cycle register.

Here, if $'\epsilon = 0$, the contents of the partially full cell with the address 'k is selected; if $'\epsilon = 1$, the contents of the full cell with 'k is selected. In the latter case, 'k must be an even number.

b) designates sending a 1 or a 0 to the trigger indicator for conditional transfer of control ω, depending upon the sign of the number in the accumulator.

c) and d) consist of preparation for the execution of the next command in numerical sequence.

Actions b), c), and d) are completed also upon execution of all other commands of a given group. The same thing can be said for the ε bit. In further operational descriptions nothing more will be said about this.

2. $'k_0 = 02$--sending to the accumulator:

a) $'('k - '\epsilon_0'Ts) \Rightarrow S$; $'('k - '\epsilon_0'Ts) \Rightarrow r$;

b), c) and d), the same as for $'k_0 = 01$.

3. $'k_0 = 03$--subtraction:

a) $'S - '('k - '\epsilon_0'Ts) \Rightarrow S$:

$\text{sign } 'S \vee |'('k - '\epsilon_0'Ts)| \Rightarrow r$.

4. $'k_0 = 04$ (the same as for $'k_0 = 03$; only the operation executed is modulus subtraction):

a) $|'S| - |'('k - '\epsilon_0'Ts)| \Rightarrow S$;

$\text{sign } 'S \vee |'('k - '\epsilon_0'Ts)| \Rightarrow r$.

5. $'k_0$ = 05--multiplication with storage in the accumulator:

a) $'r \times '('k - '\varepsilon_0'Ts) + 'S \Rightarrow S; \quad 0 \Rightarrow r;$

the contents of a cell is multiplied by the contents of the accumulator register and the result is added to the contents of the accumulator.

6. $'k_0$ = 06--multiplication:

a) $'S \times '('k - '\varepsilon_0'Ts) \Rightarrow S;$

$0 \Rightarrow r.$

7. $'k_0$ = 07--division:

a) $'S:'('k - '\varepsilon_0'Ts) \Rightarrow S, \ 0 \Rightarrow r.$

Besides the registers indicated in the list, there is an overflow register φ for various conditions, and switches φ_1 and φ_2 connected with it.

If, upon execution of arithmetic operations 01, 03, 04, 05, 06, 07,

$$|'S| < 1 ,$$

a 1 is sent to the φ trigger:

$$1 \Rightarrow \varphi;$$

if ('S) < 1, a 0 is sent to φ:

$$0 \Rightarrow \varphi.$$

Here, if $'\varpi = 1$ and

a) $'\varphi_1 = 1,$

then the action

$$'C + 1 \Rightarrow C$$

is completed (transfer to the next command of the program); if

b) $'\varphi_1 = 0$, $'\varphi_2 = 1$,

then the computer stops; if

c) $'\varphi_1 = 0$, $'\varphi_2 = 0$,

then $'C + 2 \Rightarrow C$ is executed; that is, one command is skipped.

8. $'k_0 = 26$. Cyclical addition of the contents of the accumulator with the contents of cell $'k - '\varepsilon_0'Ts$ (with transfer from the high-order to the low-order bit). The role of bits ε_0 and ε is the same as for the preceding operations. Also, points b), c), and d) are executed.

II. LOGICAL OPERATIONS

9. $'k_0 = 10$. Assumption of the sign of the number by the contents of the accumulator:

a) $|'S| \vee \text{sign } '('k - '\varepsilon_0'Ts) \Rightarrow S \ (\Rightarrow r)$;

b) $R\ \{\text{sign } 'S = 1\}$ $\begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) $'C + 1 \Rightarrow C$;

d) $^2C \Rightarrow K$.

10. $'k_0 = 11$. Shift of the contents of register r of the accumulator (including the sign bit) by the number

of bits which corresponds to the modulus of the number distributed in bits 13 through 18 of the accumulator S:

a) to the left if sign 'S = 0, and to the right if sign 'S = 1. The result of the shift is placed in the accumulator; $0 \Rightarrow r$;

b) $R\ \{'S = 0\}\ \begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) $'C + 1 \Rightarrow C$;

d) $^2C \Rightarrow K$.

The operation does not depend on $'\varepsilon_0$, $'\varepsilon_1$, 'k.

11. $'k_0 = 12$. Digital logical multiplication:

a) $'S \wedge '('k - '\varepsilon_0 'Ts) \Rightarrow S\ (\Rightarrow r)$,

where 'k is an even number;

b) $R\ \{'S = 0\}\ \begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) $'C + 1 \Rightarrow C$;

d) $^2C \Rightarrow K$.

12. $'k_0 = 13$. The same as for $'k_0 = 12$; only the executed operation is digital logical addition.

13. $'k_0 = 14$--digital logical operation "non-equivalent $\tilde{=}$"; the bits of the result are clarified in Table 13.

Table 13

a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

a) $['S \tilde{=} '('k - '\varepsilon_0'Ts)] \Rightarrow S \ (\Rightarrow r);$

b) $R\ \{'S = 0\}\ \begin{matrix} 0 \Rightarrow \omega; \\ 1 \Rightarrow \omega; \end{matrix}$

c) $'C + 1 \Rightarrow C;$

d) $^2C \Rightarrow K.$

14. $'k_0 = 15$--normalization:

a) the contents of the accumulator is shifted left by the number of bits equal to the number of zeros between the decimal point and the first significant digit if $|'S| < \frac{1}{2}$, and is shifted to the right one bit if $|'S| > 1$. After the shift, the number corresponding to the number of shifts, in the first case negative, and in the second case positive, is recorded according to the address $'k - \varepsilon_0'Ts$, and the exponent of the number is fixed in the accumulator. The sign of the exponent is fixed in the accumulator in the sign bit; the low-order bit of the exponent is the 19th bit of the accumulator.

b) $R\ \{'('k - '\varepsilon_0'Ts) = 0\}\ \begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) $'C + 1 \Rightarrow C;$

d) $^{2}C \Rightarrow K$.

15. $'k = 16$. Sending from the accumulator according to address:

a) $'S \Rightarrow 'k - '\varepsilon_{o}'Ts$;

b) R {sign $'S = 1$} $\begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) $'C + 1 \Rightarrow C$;

d) $^{2}C \Rightarrow K$.

16. $'k_{0} = 17$. Sending to the accumulator register:

a) $'('k - '\varepsilon_{0}'Ts) \Rightarrow r$;

b) R {$'r = 0$} $\begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) $'C + 1 \Rightarrow C$;

d) $^{2}C \Rightarrow K$.

17. $'k_{0} = 20$. Sending to the accumulator:

a) $'k - '\varepsilon_{0}'Ts \Rightarrow S \ (\Rightarrow r)$.

The number $'k - '\varepsilon_{0}'Ts$ is placed in the accumulator (from the 7th through the 18th bits), and not the contents of this address as in operation 02. The contents ε is considered the sign of the number $'k - '\varepsilon_{0}'Ts$, and $'\varepsilon$ is sent to the sign bit of the accumulator;

b) R {sign $'S = 1$} $\begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) $'C + 1 \Rightarrow C$;

d) $^{2}C \Rightarrow K$.

III. CONTROL TRANSFER OPERATIONS

Control transfer operations do not change the contents of the addresses in the computer, and concern only the contents of certain special registers.

18. $'k_0 = 21$. Conditional control transfer:

a) $R\{'\omega = 1\}$ $\begin{matrix} 'k - '\varepsilon_0 'Ts \Rightarrow C; \\ 'C + 1 \Rightarrow C; \end{matrix}$

b) $^2C \Rightarrow K$.

If 1 is in trigger ω, transfer is made to the execution of the command preserved according to address $'k - '\varepsilon_0 'Ts$, and in the opposite case, transfer is to the next command.

19. $'k_0 = 22$--unconditional transfer of control:

a) $'k - '\varepsilon_0 'Ts \Rightarrow C$;

b) $^2C \Rightarrow K$.

20. $'k_0 = 23$--control transfer by key. If key $'k$ is switched on, one command is omitted; if it is not on, transfer is made to the next command; that is:

a) $R\{'K_{'k} = 1\}$ $\begin{matrix} 'C + 2 \Rightarrow C; \\ 'C + 1 \Rightarrow C; \end{matrix}$

b) $^2C \Rightarrow K$.

21. $'k_0 = 25$--beginning of a group operation. The contents of cycle register Ts is prepared by this command:

a) $'k \Rightarrow Ts$;

b) $'C + 1 \Rightarrow C$;

c) $^{2}C \Rightarrow K$.

22. $'k_0 = 24$--the end of a group operation:

a) $R\{'Ts \neq 0\}$ $\begin{matrix} 'k - '\varepsilon_0'Ts \Rightarrow C;\ 'Ts - '\varepsilon - 1 \Rightarrow Ts; \\ 'C + 1 \Rightarrow C; \end{matrix}$

b) $^{2}C \Rightarrow K$.

a) signifies transfer of control to the cycle (to the command stored according to address $'k - '\varepsilon_0'Ts$) if $'Ts \neq 0$, and a decrease in the contents of cycle register Ts by 1 if $'\varepsilon = 0$ (for a group operation on short cells), and a decrease by two if $'\varepsilon = 1$ (for a group operation on full cells); in the opposite case, exit is made from the cycle.

23. $'k_0 = 30$. Command change. By this command:

a) $'k - '\varepsilon_0'Ts \Rightarrow K$;

b) $'C + 1 \Rightarrow C$;

c) $'K + {}^{2}C \Rightarrow K$.

Thus, the "next" command enters the command register, modified by the quantity of the contents of address $'k - '\varepsilon_0'Ts$.

24. $'k_0 = 37$. The computer stops on this command, and in addition, sends the contents of cell $'k - '\varepsilon_0'Ts$ to the accumulator:

$$'('k - '\varepsilon_0'Ts) \Rightarrow S.$$

IV. OPERATIONS FOR USE OF EXTERNAL UNITS

25. $'k_0$ = 31--a preparatory command for a [redacted] transfer operation; 'k is the address of the first cell of the internal memory (a magnetic drum), with which the operation begins. Operations for using external units are coded by a sequence of three commands:

31 α ;

β_1 β ;

00 γ .

Here 31, β_1, 00 are codes corresponding to register k_0; α, β, and γ are codes corresponding to register k. Here the following values are allowed for β_1:

$$\beta_1 = 01;\ 02;\ 03;$$

which also determine the type of operation; β is the zone number of magnetic and perforated tape.

1) β_1 = 01--executes a transfer from the input unit (perforated tape) to the internal memory (magnetic drum).

2) β_1 = 02--executes a transfer from the external (magnetic tape) to the internal memory unit.

3) β_1 = 03--executes a transfer from the internal to the external memory units.

In each of these cases the code γ determines the address of the next cell of the internal memory, to which the operation is being transferred.

After transfer, the next command is completed.

26. $'k_0 = 32$. Printout of the contents of the accumulator. Here, if $'T_1 = 1$, the printout proceeds on paper tape; if $'T_1 = 0$, printout is on perforated tape; if $'T_2 = 1$, printout is in octal codes (output of the program); if $'T_2 = 0$, printout is in decimal (output of the numerical material).

27. $'k_0 = 34$--omission of a line on paper tape.

M-3

The M-3 automatic digital computer uses 31-bit words, and has fixed point. One bit is the sign bit and 30 are digital (Fig. 29).

For recording numbers in a binary-decimal system, the last two bits are not used, and seven decimal digits are in the remaining 28 bits.

The command system is two-address with sequential execution. The operational code occupies six bits; the addresses, 12 bits each. The scheme for the distribution of bits over a cell for storing a command is shown in Fig. 30.

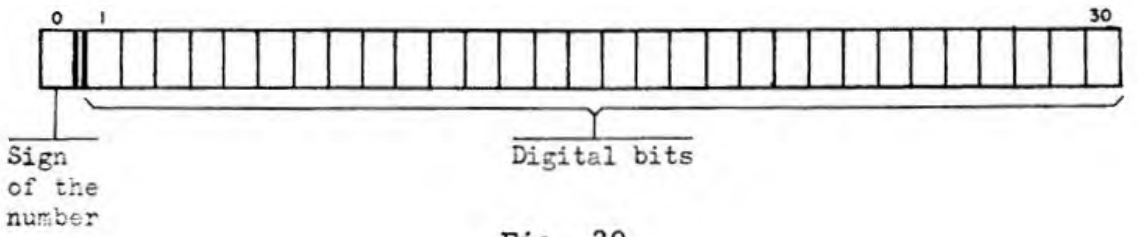

Fig. 29

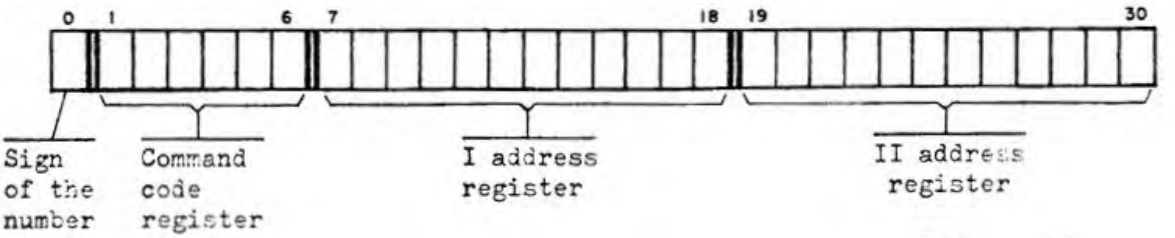

Fig. 30

Table 14

Bit groups of the command register	Designation and name of bit groups (registers)	
1 - 3	k_{01}--the first register of the command code	k_0--the code register
4 - 6	k_{02}--the second register of the command code	
7 - 18	k_1--the I address register	
19 - 30	k_2--the II address register	

The internal memory of the computer consists of 2048 cells (numbered from 0000 to 3777 in octal) and is located on a magnetic drum. The speed of the computer is 30 operations per second. The arithmetic unit and control unit are high-speed. Therefore replacing the magnetic drum with a memory unit of ferrite cores increases the speed to 1500 operations per second. Input is from perforated paper tape, and output is printed. Cell 0000, unlike the Strela, BESM, and Kiev computers, is a normal cell (not containing the code + 0).

For a description of operations we introduce the designations:

1) C--the command counter,

2) K--the command register,

3) r--the arithmetic unit register,

4) ω--the trigger for indicating conditional transfer.

We shall divide the command register K into bit groups (registers) (see Table 14). The value $'k_{02}$ determines the type of operation (see Table 15).

Table 15

Type of Operation	$'k_{02}$
Addition	0
Subtraction	1
Multiplication	3
Division	2
Digital logical multiplication	6

The value $'k_{01}$ determines the operation to be carried out. Let θ designate any of the operations shown in the table. The computer executes subsequent operations depending on the nature of $'k_{01}$.

I. ARITHMETIC OPERATIONS

1. $'k_{01} = 0$.

a) $^2k_1 \; \theta \; ^2k_2 \Rightarrow r; \; 'r \Rightarrow 'k_2;$

b) $R \; \{'r \leq -0\} \; \begin{matrix} 1 \Rightarrow \omega; \\ 0 \Rightarrow \omega; \end{matrix}$

c) $'C + 1 \Rightarrow C;$

d) $^2C \Rightarrow K.$

In other words, in this case the addresses of the arguments of an operation are coded in the addresses of the command; the result of the operation is contained according to the second address $'k_2$ and in r--the arithmetic unit register. Point b) is treatment of the sign ω, used in conditional control transfer commands.

Points b), c), and d), transfer to execution of the next command, are the same for all commands of this group.

2. $'k_{01} = 1$.

a) $^2k_1 \theta\, ^2k_2 \Rightarrow r$.

(The contents of address $'k_2$, unlike the case $'k_{01} = 0$, is not changed by the operation.)

3. $'k_{01} = 2$

a) $'r \theta\, ^2k_1 \Rightarrow r$;

$'r \Rightarrow 'k_2$.

(The operation θ is executed upon the contents of r--the arithmetic unit register and cell $'k_1$; the result is placed in register r according to address $'k_2$.)

4. $'k_{01} = 3$.

a) $'r \theta\, ^2k_1 \Rightarrow r$.

5. $'k_{01} = 4$.

In this case, the following operations are executed:

a) $^2k_1\theta^2k_2 \Rightarrow r;$

$'r \Rightarrow 'k_2;$

print 'r.

('r is printed on paper tape.)

6. $'k_{01} = 5.$

a) $|^2k_1|\theta|^2k_2 \Rightarrow r.$

7. $'k_{01} = 6.$

$'r\theta^2k_1 \Rightarrow r;$

$'r \Rightarrow 'k_2;$

print 'r.

8. $'k_{01} = 7.$

$|'r|\theta|^2k_1| \Rightarrow r.$

II. CONTROL TRANSFER COMMANDS

1. $'k_0 = 24.$ Unconditional transfer according to the first address:

a) $'r \Rightarrow 'k_2;$

b) $'k_1 \Rightarrow C;$

c) $^2C \Rightarrow K.$

(The contents of r, the arithmetic unit register, is sent simultaneously according to address $'k_2$.)

2. $'k_0 = 64.$ Unconditional transfer according to the first address with printing of the contents of register r:

a) $'r \Rightarrow {}'k_2$;

b) print $'r$;

c) $'k_1 \Rightarrow C$;

d) ${}^2C \Rightarrow K$.

3. $'k_0 = 74$. Unconditional transfer according to the second address ($'k_1 = 0$):

a) $|'r| \Rightarrow r$;

b) $'k_2 \Rightarrow C$;

c) ${}^2C \Rightarrow K$.

4. $'k_0 = 34$. Conditional transfer:

a) $R\ \{'\omega = 1\}$ $\begin{matrix} 'k_1 \Rightarrow C; \\ 'k_2 \Rightarrow C; \end{matrix}$

b) ${}^2C \Rightarrow K$.

5. $'k_0 = 37$. Stop:

a) ${}^2k_1$ is put out on the console;

b) stop.

III. INPUT (OUTPUT) AND TRANSFER OPERATIONS

1. $'k_0 = 07, 27$. Input from perforated tape [PL] ($'k_1 = 0$):

a) $'PL \Rightarrow {}'k_2$;

b) $'C + 1 \Rightarrow C$;

c) ${}^2C \Rightarrow K$.

One code from the perforated tape is transferred to memory cell $'k_2$.

2. $'k_0$ = 05, 15. Transfer:

a) $^2k_1 \Rightarrow 'k_2$;

b) $'C + 1 \Rightarrow C$;

c) $^2C \Rightarrow K$.

3. $'k_0$ = 45, 55. Transfer with printing:

a) $^2k_1 \Rightarrow 'k_2$;

b) print $^2k_1$;

c) $'C + 1 \Rightarrow C$;

d) $^2C \Rightarrow K$.

KIEV

The Kiev general-purpose automatic digital computer uses 41-bit words. For commands in a three-address system five bits are provided for the operation code and 12 bits for each address. The first bit of each address is used as an address modification (changeability) indicator. It has fixed point and the first bit is the sign bit.

The distribution of bits over a cell when a binary number is recorded is shown in Fig. 31.

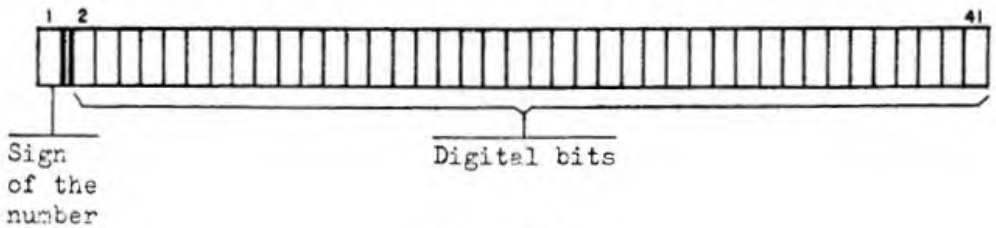

Fig. 31.

The speed of the computer is 15-20 thousand addition type operations per second and 3-4 thousand multiplication and division type operations per second, or an average of 9 thousand operations per second. The capacity of the internal memory unit (VZU) is 2048 cells; the external memory is on magnetic drums and tapes. The internal memory contains both operative and passive cells. Passive

cells are of three types:

a) Soldered--permanent,

b) Soldered--removable,

c) Plugged in--removable.

The permanently soldered memory is designed for storing the more frequently encountered constants and subroutines. The removably soldered memory is in the form of soldered blocks designed for storing various standard subroutines, and can be connected for operation according to need. The eight cells of the plug-board memory represent sets of tumbler selectors, by means of which various codes can be introduced by hand. Conversion of data from decimal to binary is combined with input, and does not require additional time. Input is from perforated tape, and output is onto perforated tape or by printer; facility for punched card use has been provided.

In addition to the cells of the VZU with addresses 0000 + 3777 in octal, the Kiev has the following special registers:

1) C--the command counter (11-bit)

2) K--the command register (41-bit)

3) R--the return register (11-bit)

4) Ts--the cycle register (10-bit)

5) A--the address modification register (10-bit)

6) T--the register-trigger for emergency stop (1-bit).

For a description of operations including the operation of these registers, we shall limit ourselves to alphabetic designations--addresses, which we shall assume to be different from the address codes of the VZU.

According to the principle of program control (see Chapter II), a separate operational cycle is used to execute the command 'K--i.e., a command whose code is stored at a given moment in the command register--and to send to register K the code of the next command.

Execution of command K depends in turn on its contents (the code).

For storing a number (address) of the "next" command in the Kiev, command counter C is used, as in many other computers.

For convenience in describing the set of elementary operations carried out by the Kiev's commands, we shall divide the bit groups (registers) in command register K, as is shown in Table 16. Such a division is carried out when necessary also in the cells of the VZU.

The command register (and the memory cell in which a command is stored) is broken down as shown in Fig. 32.

Table 16

Bits of the Kiev command register (left to right)	Designation and name of bit groups (registers)
1 - 5	k_0--the operational code register
6	ϵ_1--the bit which indicates modifiability of the I address
7 - 17	k_1--the I address register
18	ϵ_2--the bit which indicates modifiability of the II address
19 - 29	k_2--the II address register
30	ϵ_3--the bit which indicates modifiability of the III address
31 - 41	k_3--the III address register

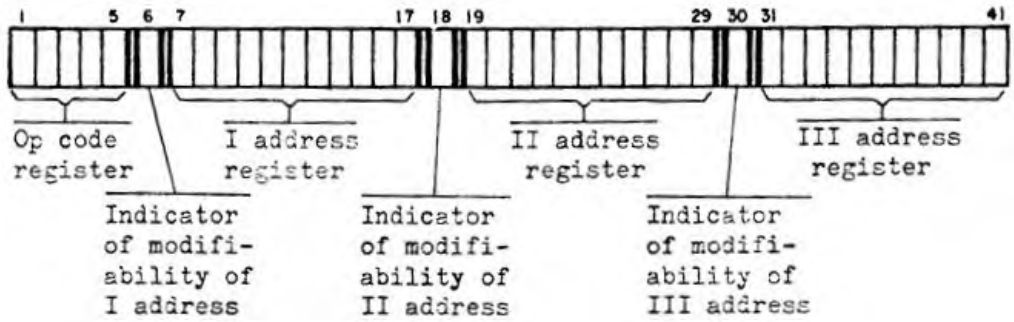

Fig. 32.

The following octal codes can be stored in the indicated registers:

$'k_0 = 00, 01, 02, \ldots, 37;$

$'\varepsilon_1, '\varepsilon_2, '\varepsilon_3 = 0, 1;$

$'k_1, 'k_2, 'k_3 = 0000, 0001, \ldots, 3777.$

Depending on $'k_0$, the operation code, the computer executes subsequent operations.

I. ARITHMETIC OPERATIONS

1. $'k_0 = 01$. Addition "+". For this operation the computer completes the following actions:

a) addition of numbers which are the contents of addresses $'k_1 + '\varepsilon_1'A$ and $'k_2 + '\varepsilon_2'A$, and sending of the result of the addition according to address $'k_3 + '\varepsilon_3'A$:

$$'('k_1 + '\varepsilon_1'A) + '('k_2 + '\varepsilon_2'A) \Rightarrow 'k_3 + '\varepsilon_3'A;$$

b) incrementing the contents of command counter C by 1:

$$'C + 1 \Rightarrow C;$$

c) sending the next command to command register K:

$$^2C \Rightarrow K.$$

Point a) designates that for $'\varepsilon_i = 1$ $(i = 1,2,3)$, the corresponding address is modified; specifically, it is increased by the value of the contents of address modification register A. For $'\varepsilon_i = 0$, the operation is executed directly according to the address. Points b) and

c) designate preparation for the execution of the next command in numerical order.

These features will refer to all operations of a given group.

2. $'k_0 = 02$. Subtraction "-". The same as for "+" only the executed operation is subtraction; i.e.,

a) $'('k_1 + '\varepsilon_1'A) - '('k_2 + '\varepsilon_2'A) \Rightarrow 'k_3 + '\varepsilon_3'A;$

b) $'C + 1 \Rightarrow C;$

c) $^2C \Rightarrow K.$

3. $'k_0 = 03$. Addition of commands "$+_1$":

a) $|'('k_1 + '\varepsilon_1'A) + |'('k_2 + '\varepsilon_2'A)||\vee$ sign

$'('k_2 + '\varepsilon_2'A) \Rightarrow 'k_3 + '\varepsilon_3'A;$

b) $'C + 1 \Rightarrow C,$

c) $^2C \Rightarrow K.$

The operation "$+_1$" differs from addition "+" in that here the contents of cell $'('k_1 + '\varepsilon_1'A)$--the re-addressing constant--is added to the contents of cell $'('k_2 + '\varepsilon_2'A)$--i.e., to its modulus--which is considered as command code, and the sign of command $'('k_2 + '\varepsilon_2A)$ is appropriated to the result.

4. $'k_0 = 06$. Modulus subtraction "$|-|$". The same as "-", but the diminuend and subtrahend are moduli of the corresponding codes.

5. 'k_0 = 07. Cyclical addition, "Ts +", is addition of codes with transfer from the high order-bit to the low-order bit. As in the preceding operations, address modification is possible. The following operations are completed analogously:

6. 'k_0 = 10. Multiplication without rounding, "x".

7. 'k_0 = 11. Multiplication with rounding, " ⓧ ".

8. 'k_0 = 12. Division, ":".

In addition to the above, for completion of "+", "-", ":", 1 is sent to register T, if the result of the operation turns out to be greater than 1, according to the modulus; and 0 in the opposite case.

In a corresponding arrangement for the emergency stop toggle switch, sending

$$1 \Rightarrow T$$

designates stop.

II. LOGICAL OPERATIONS

9. 'k_0 = 35. Normalization.

a) The number '('k_1 + 'ε_1'A) is normalized, the exponent of the number is placed in the six low-order bits of cell ('k_2 + 'ε_2'A), and the mantissa is according to address 'k_3 + 'ε_3'A:

b) 'C + 1 ⇒ C;

c) ^{2}C ⇒ K.

10. $'k_0 = 13$. Logical shift.

a) The code $'('k_1 + '\varepsilon_1'A)$ (including the sign bit) is shifted by the number of bits indicated in the III address of cell $'k_2 + '\varepsilon_2'A$, right or left, depending on the sign of this number; the result is placed according to address $'k_3 + '\varepsilon_3'A$;

b) $'C + 1 \Rightarrow C$;

c) $^2C \Rightarrow K$.

11. $'k_0 = 14$. Digital logical addition "$\vee$" (see Chapter II):

a) $'('k_1 + '\varepsilon_1'A) \vee '('k_2 + '\varepsilon_2'A) \Rightarrow ('k_3 + '\varepsilon_3'A)$;

b) $'C + 1 \Rightarrow C$;

c) $^2C \Rightarrow K$.

The following two operations are completed analogously.

12. $'k_0 = 15$. Digital logical "multiplication" "$\wedge$".

13. $'k_0 = 17$. Digital logical operation "non-equivalence $\tilde{=}$".

III. CONTROL TRANSFER OPERATIONS

All control transfer operations do not change the contents of the cells in the VZU [internal memory unit]; the result of their action affects only certain special registers.

14. $'k_0 = 16$. Conditional control transfer according to equality of moduli "=".

The operation "=" obtains an address program:

a) $R\ \{'('k_1 + '\varepsilon_1'A) = '('k_2 + '\varepsilon_2'A)\}\ \begin{matrix} 'k_3 + '\varepsilon_3'A \Rightarrow C; \\ 'C + 1 \Rightarrow C \end{matrix}$

and

b) $^2C \Rightarrow K.$

The next three operations are completed analogously.

15. $'k_0 = 04$. Conditional control transfer in relation to "less or equal $\leq$":

a) $R\ \{'('k_1 + '\varepsilon_1'A) \leq '('k_2 + '\varepsilon_2'A)\}\ \begin{matrix} 'k_3 + '\varepsilon_3'A \Rightarrow C; \\ 'C + 1 \Rightarrow C; \end{matrix}$

b) $^2C \Rightarrow K.$

16. $'k_0 = 05$. Conditional control transfer in relation to "less or equal" without taking into account the signs "$|\leq|$":

a) $R\ \{|'('k_1 + '\varepsilon_1'A)| \leq |'('k_2 + '\varepsilon_2'A)|\}\ \begin{matrix} 'k_3 + '\varepsilon_3'A) \Rightarrow C; \\ 'C + 1 \Rightarrow C; \end{matrix}$

b) $^2C \Rightarrow K.$

17. $'k_0 = 31$. Conditional control transfer according to the sign of the number, "UPCh":

a) $R\ \{'('k_1 + '\varepsilon_1'A) \leq - 0\}\ \begin{matrix} 'k_3 + '\varepsilon_3'A \Rightarrow C; \\ 'k_2 + '\varepsilon_2'A \Rightarrow C; \end{matrix}$

b) $^2C \Rightarrow K.$

18. $'k_0 = 32$. Conditional transfer to a subroutine, "UPP":

a) $R\ \{'('k_1 + '\varepsilon_1'A) \leq - 0\}\ \begin{matrix} 'k_2 + '\varepsilon_2'A \Rightarrow R;\ 'k_3 + '\varepsilon_3'A \Rightarrow C; \\ 'C + 1 \Rightarrow C; \end{matrix}$

b) $^2C \Rightarrow K$.

Here $'k_3 + '\varepsilon_3'A$ is the address of the first command of the subroutine, and $'k_2 + '\varepsilon_2'A$ is the number of the command to which control must be transferred after execution of a subroutine. When the condition is not met transfer is made to the next command in numerical order.

The corresponding subroutine is ended by a special command.

19. $'k_0 = 30$. Transfer according to the return register. By this command these actions are completed:

a) $'R \Rightarrow C$;

b) $^2C \Rightarrow K$.

Here the contents of registers k_1, k_2, k_3 do not affect the result of the operation.

C;

IV. OPERATIONS FOR USE OF EXTERNAL UNITS

All operations of this group are group operations; that is, they refer to code sequences.

20. $'k_0 = 20$. Data input, VCh, from perforated tape (PL). By this command, codes which beforehand have been converted from decimal-binary to binary are put into operative core storage (OZU) cells from perforated tape, with addresses

;;

$$'k_1, 'k_1 + 1, \ldots, 'k_2.$$

For convenience we shall write this group operation in the form

a) $'(\mathrm{PL})_{\text{converted to binary}} \Rightarrow \begin{pmatrix} '\mathrm{k}_1 \\ '\mathrm{k}_2 \end{pmatrix}$.

In addition, the following are executed by this command:

b) $'\mathrm{C} + 1 \Rightarrow \mathrm{C}$;

c) ${}^{2}\mathrm{C} \Rightarrow \mathrm{K}$.

21. $'\mathrm{k}_0 = 21$. Input of commands, "VK," from perforated tape. The same as "VCh," only transfer of codes is carried out without conversion:

a) $'(\mathrm{PL}) \Rightarrow \begin{pmatrix} '\mathrm{k}_1 \\ '\mathrm{k}_2 \end{pmatrix}$;

b) $'\mathrm{C} + 1 \Rightarrow \mathrm{C}$

c) ${}^{2}\mathrm{C} \Rightarrow \mathrm{K}$.

22. $'\mathrm{k}_0 = 22$. Output of codes onto perforated tape, "VPL."

Return operation: the contents of cells

$$'\mathrm{k}_1,\ '\mathrm{k}_1 + 1,\ \ldots,\ '\mathrm{k}_2$$

are output onto PL

a) $'\begin{pmatrix} \mathrm{k}_1 \\ '\mathrm{k}_2 \end{pmatrix} \Rightarrow \mathrm{PL}$;

b) $'\mathrm{k}_3 \Rightarrow \mathrm{C}$;

c) ${}^{2}\mathrm{C} \Rightarrow \mathrm{K}$.

23. $'k_0 = 23$. Exchange of codes between OZU [core storage] and external ZU [storage units] (MB [magnetic drums]) in the "write" mode, "MBZ". Codes contained in a sequence of cells of the OZU

$$'k_1, 'k_1 + 1, \ldots, 'k_2,$$

are transferred to MB:

a) $' \begin{pmatrix} 'k_1 \\ 'k_2 \end{pmatrix} \Rightarrow MB;$

in addition,

b) $'k_3 \Rightarrow C;$

c) $^2C \Rightarrow K.$

24. $'k_0 = 24$. Exchange of codes between OZU and MB in the "read" mode "MBCh". Codes from the MB are transferred to a sequence of cells in the OZU, $'k_1$, $'k_1 + 1$, ...; that is,

a) $'(MB) \Rightarrow \begin{pmatrix} 'k_1 \\ 'k_2 \end{pmatrix}$;

in addition,

b) $'k_3 \Rightarrow C;$

c) $^2C \Rightarrow K.$

25. $'k_0 = 25$. A preparatory operation for the "MBZ" and "MBCh" operations, ensuring proper feed from/to the magnetic drum. Here

$$'k_1 = \begin{cases} 0 \text{ for operation 23} \\ 1 \text{ for operation 24,} \end{cases}$$

$'k_2$--the track number of the MB, from which code exchange is carried out;

$'k_3$--the number of the data item on the MB track from which it is necessary to start the appropriate operation.

In this case the actions

$$'C + 1 \Rightarrow C; \quad {}^2C \Rightarrow K$$

are also executed.

26. $'k_0 = 33$. Stop.

V. ADDRESS MODIFICATION OPERATIONS

Address modification operations are group operations in the sense that the contents of a modification register formed by them can be used as a group of commands.

27. $'k_0 = 26$. "Beginning of group operation," NGO. By the operation NGO:

a) a number is sent to cycle register Ts, characterizing the number of cycles in the cyclical process,

$$'k_1 \Rightarrow Ts;$$

b) the readdressing constant is sent to address modification register A

$$'k_2 \Rightarrow A;$$

c) the predicated formula is obtained

$$R\ \{'Ts = 'A\}\ \begin{matrix} '\mathrm{k}_3 \Rightarrow C; \\ 'C + 1 \Rightarrow C; \end{matrix}$$

d) $^2C \Rightarrow K$.

Thus the command NGO prepares the contents of the readdressing register, and by the same token provides the appropriate modification of the changed addresses.

If the number of cycles N is known earlier, and r is a readdressing step, then we assume:

$$'\mathrm{k}_1 = '\mathrm{k}_2 + Nr.$$

For each cycle repetition in this case the contents of register A is increased by the quantity r. As long as $'Ts \neq 'A$, computations continue according to the cycle; when $'Ts = 'A$ exit is made from the cycle to the command with the number $'\mathrm{k}_3$.

An increase of the contents of register A by readdressing step r can be obtained by taking into account the readdressing of the corresponding command NGO (increase of its second address by r), and the repeated execution of this command, or by means of the next command, KGO.

28. $'\mathrm{k}_0 = 27$. End of group operation, "KGO." By the command KGO:

a) the contents of register A is increased by readdressing step $'\mathrm{k}_1 + 'A \Rightarrow A$ ($'\mathrm{k}_1$ = r--the re-addressing step);

b) the predicated formula is obtained

$$R\ \{'Ts = 'A\}\ \begin{matrix} '\mathrm{k}_3 \Rightarrow C; \\ '\mathrm{k}_2 \Rightarrow C; \end{matrix}$$

c) ${}^2C \Rightarrow K.$

Here $'\mathrm{k}_3$ is the command number to which control is transferred at the end of the cycle; $'\mathrm{k}_2$ is the command number to which control is transferred during the course of computations over the cycle. Here we note that $'\mathrm{k}_2$ is not the number NGO, since the latter is not now repeated upon transfer from cycle to cycle, since its repetition would lead to a restoration of the initial loading of register A.

29. $'\mathrm{k}_0 = 34$. Loading of register A according to a locator (call-up according to locator), "F".

Besides the "NGO" operation, which provides loading of the address modification register, the "F" operation is provided, which also executes this function. However, while the quantity in the command NGO is being sent to register A in explicit form (as $'\mathrm{k}_2$), in the "F" command this quantity is assigned only by its address. Under the "F" command:

a) $'('\mathrm{k}_1 + '\varepsilon_1'A)_2 \Rightarrow A;$

b) $'('('\mathrm{k}_1 + '\varepsilon_1'A)_2) \Rightarrow '\mathrm{k}_3;$

c) $'C + 1 = C;$

d) ${}^2C \Rightarrow K,$

$'k_2$ is not used in the execution of the command.

Let us designate by $'\alpha_2$ the contents of the II address of cell α. Then, if

$$'k_1 + \alpha; \quad '\alpha_2 = \beta,$$

we have, according to operation F,

a) $\beta \Rightarrow A$;

b) $'\beta \Rightarrow 'k_3$;

c) $'C + 1 \Rightarrow C$;

d) $^2C \Rightarrow K$.

The convenience of using operation F can be made clear from Chapter V. It is possible to show that the presence of operation F makes it possible to compose programs for arbitrary cyclical parametric processes, which do not vary during their operation (without re-addressing commands).

BIBLIOGRAPHY OF RAND CORPORATION PUBLICATIONS IN SOVIET CYBERNETICS AND COMPUTER TECHNOLOGY

1. Ware, W. H., (ed.), Soviet Computer Technology--1959, RM-2541, March 1, 1960. Reprinted in IRE Transactions on Electronic Computers, Vol. EC-9, No. 1, March 1960.

An account of a trip taken by two RAND computer specialists to the Soviet Union as part of an eight-man delegation representing the U.S. National Joint Computer Committee and its member societies. The genesis of the delegation and its itinerary in the Soviet Union are traced. The state of the art in Soviet computer technology as observed by the delegates is examined, showing the development, constructions, applications, routines, and components of the major Soviet computing machines. Impressions are included on Soviet education, the role of the Academy of Sciences, and Chinese developments in computer technology. Many photographs of Soviet machines, components, people, and places are included. First-hand information is also given on the BESM-I, BESM-II, Strela, Ural, and Kiev computers, plus several other machines. Machine specifications are presented in chart form, facilitating comparisons; op codes are given for the Ural-1 and Ural-2. 205 pp. Illus.

2. Feigenbaum, E. A., Soviet Cybernetics and Computer Sciences, 1960, RM-2799-PR, October 1961. Reprinted in IRE Transactions on Electronic Computers, Vol. EC-10, No. 4, December 1961.

A description of the author's experiences as a delegate to the International Congress on Automatic Control, held in Moscow, June 27-July 7, 1960. The Memorandum discusses (1) certain aspects of the conference; (2) some Soviet research projects in artificial intelligence and biocybernetics; and (3) general Soviet attitudes, techniques, and directions in the cybernetic and computer-related sciences. It is concluded that Soviet research in the computer sciences lags behind Western developments, but that the gap is neither large nor based on a lack of understanding of fundamental principles. The Soviets will progress rapidly if and when priority, in terms of accessibility to computing machines, is given to their research. 77 pp. Illus.

3. Ware, Willis H., and Wade B. Holland, (eds.), Soviet Cybernetics Technology: I. Soviet Cybernetics, 1959-1962, RM-3675-PR, June 1963.

Seven sets of translations in the area of Soviet cybernetics, together with commentary and analyses on the status of cybernetics in the Soviet Union, and the directions of Soviet cybernetics research. This volume is concerned with general computer technology and cybernetics applications, rather than with specific machines. Particular emphasis was placed in selection of items for translation which surveyed the activities of organizations and conferences, and current literature. 104 pp. Illus.

4. Ware, Willis H., and Wade B. Holland, (eds.), Soviet Cybernetics Technology: II. General Characteristics of Several Soviet Computers, RM-3797-PR, August 1963.

Several sets of translations detailing specifications for the Ural-2, Ural-4, BESM-II, Razdan-2, MN-10 and MN-14, Luch, and EPOS computers. The level of detail varies widely among the several articles, which were taken from such diverse sources as specification brochures, items in the popular press, technical journals, etc. Included is a set of instructions for the BESM-II which is quite dissimilar to that presented in Elements of Programming. 67 pp. Illus.