ABOUT CRYPTOGRAPHIC EXTENSION OF RUSSIAN PROGRAMMING LANGUAGE AND ITS SOFT AND HARD IMPLEMENTATIONS

G. P. Agibalov, V. B. Lipsky, I. A. Pankratova

Tomsk state university
Tomsk, Russia
E-mail: agibalov@isc.tsu.ru, *lipsky@.mail.tsu.ru, pank@isc.tsu.ru*

A cryptographic extension of the Russian programming language LYaPAS called LYaPAS-T is presented. The extension concerns the size of operands and the set of elementary operations over them. It is motivated by the need of trustworthy and effective soft and hard implementations of contemporary cryptographic algorithms and secure computer systems of logical control. A LYaPAS-T compiler generating a load module for operating system Linux, and the project of a processor implementing LYaPAS-T in hardware are presented too.

Keywords: Russian programming language, cryptographic extension, LYaPAS-T, compiler, implementation in hardware.

Introduction

Here by the Russian programming language is meant the algorithmic language LYaPAS elaborated at the beginning of the 1960th years at Tomsk State University (Russia) by the leadership of A.D. Zakrevskij and designed for the representation of logical combinatorial algorithms solving the problems of applied discrete mathematics appearing in the synthesis of discrete automata [1, 2]. The name of Russian programming language was given to it by American scientists [3]. Up to 1990th years, LYaPAS was applied in USSR [2], USA [4], Germany, Poland, Czechoslovakia and other countries. This time, LYaPAS is successfully reanimated at Tomsk State University by the Information Security and Cryptography Department especially for elaborating the trustworthy system and applied software destined for the computer-aided design of secure logical control computer systems and for the secure and effective implementation of cryptographic algorithms [5]. Among many programming languages known today, LYaPAS seems to be the most appropriate one for these purposes.

At the same time, there is an essential and perhaps single drawback of LYaPAS – the lack of many elementary operations that are widely used in contemporary cryptographic algorithms: for the long integer arithmetic, for calculating in many-dimensional spaces over finite fields and rings, for solving combinatorial problems on large sets, etc. By the way, this drawback is usual for all other programming languages including ones being much younger than LYaPAS. In some of them, the drawback is got over by writing classes of long integers, large discrete functions and others. As for LYaPAS, this its lack is more effectively got over by extending the language itself, i.e., by spreading elementary operations in LYaPAS for logical complexes and by adding some new operations to it defined for variables and logical complexes. The last version of LYaPAS (known as LYaPAS-M [6, 7]) slightly revised and then exTended in such a way is called LYaPAS-T.

The revision of LYaPAS-M concerns the alphabet of the language and the arithmetic operations of multiplication and division for integers. The result of the revision is called vLYaPAS (from reVised LYaPAS). In it, small Latin letters are used instead of capital Russian ones, symbols of some operations are replaced by other more proper ones, and multiplication and division operations are defined saving the values of overflow and remainder respectively.

Revised LYaPAS

A program in vLYaPAS is a series of sentences each starting with a pair §*s* where *s* is a non-negative integer called the number of the sentence. Every sentence is a sequence of operations applied to operands.

Operands in vLYaPAS are constants, variables, complexes and complex elements. They are used for representing non-negative integers, Boolean vectors, Unicode symbols and sequences of them. Components in a Boolean vector are numbered beginning with 0 in the direction from the right to the left end. In vLYaPAS, non-negative integers are bounded by $2^{32} - 1$, and the length of Boolean vectors – by 32. Boolean vector of the length 32 is called a word. It is also considered as a non-negative integer. Boolean vector of any length $n \geq 1$ with only one component 1 is called a unit or identical vector.

There are natural, unit and symbol constants in vLYaPAS. Natural constants are written as decimal, hexadecimal, octal or binary numbers. A unit constant is a unit vector. Symbol constant is a sequence of Unicode symbols.

Variables in vLYaPAS take values of Boolean vectors of length 32. The number of all variables equals 27. They are denoted by straight (not italic) letters a, b, ..., z, Z, where Z is used for keeping overflow and remainder under multiplication and division operations. Besides, there is a virtual variable called the own variable of LYaPAS. Unlike the other variables, it is not written in LYaPAS programs, but it appears in them in implicit way as a result of any elementary operation and can be used as operand by any next operation in the program. For convenience of exposition, this variable is accepted to name τ.

Complex is a linearly ordered set of elements being symbols – in a symbol complex or Boolean vectors of length 32 – in a logical complex. Every complex has its unique number – a non-negative integer. The real number and the greatest number of elements in a complex are the parameters of the complex and are called its cardinality and capacity respectively.

There are elementary, complex, input-output and macro operations in vLYaPAS. Elementary operations are of four classes: value transfer (assigning), logical, arithmetic and transition operations. Among complex operations, there is the operation of forming (creating) a complex of a given cardinality. A complex which is created with a constant or variable capacity is called respectively static or dynamic one. Input-output operations allow to output symbol complexes and constants for console and to input symbol constants from keyboard to a symbol complex. Macro operations are calls for subprograms with the given external (input/output) parameters.

The more detail definitions and notations of all operations and operands in vLYaPAS are given in [8].

LYaPAS extension

Natural constants in LYaPAS-T are the integers 0, 1, ... , $2^n - 1$ where n is a multiple of 32 and depends on actual implementation of LYaPAS-T. Nowadays the value $n = 2^{14}$ seems to be quite sufficient for contemporary cryptographic applications. By letting δ be 2^{32} a natural constant c may be expressed by the following series:

$$c = c_0 + c_1\delta + c_2\delta^2 + ... + c_{r-1}\delta^{r-1} \quad (1)$$

for some $r > 0$ and $c_i \in \Omega = \{0, 1, ... , 2^{32} - 1\}$, $i = 0, 1, ... , r - 1$. In their standard binary representation, the elements of the set Ω are Boolean vectors of length 32. Therefore in LYaPAS-T, the sequence $c_0, c_1, \dots , c_{r-1}$ is represented by a logical complex L of length r with c_i being the ith element of L.

All operations defined in vLYaPAS for variables can be used in LYaPAS-T for logical complexes. In case of arithmetic operation the sequence of complex elements is considered as a natural constant c expressed by the series in (1). Different operands for an arithmetic operation may be of different lengths and types (one of them – a variable, another – a complex). In case of logical operation the complex value is considered as a Boolean vector being the concatenation of the complex elements. Logical complexes of cardinality $n/32$ with values being unit vectors are unit constants in LYaPAS-T.

So, unlike LYaPAS, there are two types of operands for elementary operations in LYaPAS-T: variables of the length of one word and logical complexes of different lengths – from 1 to $n/32$ words. Accordingly, in LYaPAS-T, there are two types of the own variable – prime and complex. The first one is the traditional τ in LYaPAS. It has the length of one word and may take the values of any variable of the language. In any implementation of LYaPAS-T, soft or hard, it is kept in a processor register. The own variables of the 2nd type take values of logical complexes and have their lengths. In hard implementation of LYaPAS-T, each of them is kept in one and the same register of the maximal possible length – n. In soft implementation of LYaPAS-T, to exclude time-spending operations for complex transfer between a register and the data memory, it is expediently, for the time of executing a series of operations beginning with the address to a logical complex, the role of the complex own variable to pass directly to this complex and to keep it in the data memory at the address of the complex itself.

In addition to operations in vLYaPAS, the extension LYaPAS-T contains some new logical operations used in cryptographic algorithms: permutation of components in a Boolean vector, projection – taking a sub-vector in a Boolean vector, insertion of a Boolean vector into another one and inverse operation, concatenation of Boolean vectors, left and right cyclic shifts of a Boolean vector, finding the largest and the least elements in a complex and others. The definitions and notations of these operations are given in [8].

LYaPAS-T compiler

It is a program in C++ which converts a program in LYaPAS-T to a load module in an assembly language for the OS Linux. The compiler is elaborated in the traditional way and is written using the library of regular expressions making it simple and transparent.

The operation of creating a complex is accompanied with the control of the free section of a necessary size in the memory. It is done by comparing the values of the complex capacity, address of the free area, and the memory bound granted by OS for complexes. In the case of enough place, the address of the complex beginning takes the value of the free section address, and then the free section address is increased by the value of the complex ca-

pacity. If the place is not enough, then an appeal to OS is made for increasing the accessible memory bound.

Under this organization, the memory is protected against attacks through the stack or heap overflow because, first, buffers (complexes) are taken away from the stack, and there is no possibility to rewrite the return address, and, second, there are no operations for setting memory free by means of OS.

LYaPAS-T processor

For being executed by LYaPAS-T processor, a LYaPAS-T program should be preliminary represented by a sequence of instructions in the executive code (called LE-code) for the processor. Each instruction in it has fields containing information about operation code, operand type (constant or not, complex type – logical, symbol, static or dynamic if the operand is a complex or its element) and complex and variable addresses in the data memory. A complex address is the address of the first element of the complex. For the static complex, it is explicitly written into the complex address field. For the dynamic complex, this field contains not the address of a complex itself but the address where the address of this complex is kept. The variable address field is used to keep the operand, if it is a constant, or the address of the operand, if it is a variable or a complex element. In the last case the variable address field contains the address of this element, if the complex is a static one and the number of the element in it is a constant, or the number of this element in a complex given immediately or as a value of a variable, if the complex is a dynamic one.

LE-code of a LYaPAS-T program is generated for the processor by the L-preprocessor – a special compiler translating the program to LE-code. A functioning algorithm of the L-preprocessor is described in [9].

LYaPAS-T processor consists of the following units: Memory, Arithmetic Logical Unit (ALU), Control Device (CD), Instruction Counter (IC), Instruction Register (IR) and two decoders – Address Decoder (ADec) and Operation Decoder (ODec).

The Memory is divided into two segments – IM (Instruction Memory) and DM (Data Memory) used to store, respectively, an LE-code of a LYaPAS-T program P and data for it – unit constants and complexes and variables for every subprogram in the hierarchical structure of P. Accordingly, for P with k subprograms, DM is conditionaly divided into four sections: I – for keeping n unit constants; C and G divided into k subsections C_j and G_j – for keeping respectively static and dynamic complexes in jth subprogram; and W also divided into k subsections W_j – for keeping local variables a, b, ..., z, Z belonging to jth subprogram, parameters and addresses of all complexes in jth subprogram.

The data allocation in sections I, C, W is made by the L-preprocessor (before the execution of the program P), in section G – by the processor itself (during the execution of P). Instructions in IM and data in DM are disposed compactly, with no gaps and in the order of section enumeration: I, C, W, G. The quantity of occupied elements in the subsection G_j is kept as a value of an element in W_j. The address of this element is denoted a_j. Its value is the least address of free element in G_j.

ALU contains three registers τ, Z and O of the maximal possible length n called the Registers of Common Use (CURs) and destined for keeping, respectively, variables τ, Z and operand being read from DM, Operational Devices (OD) used for executing arithmetic and logical operations in LYaPAS-T over operands and with the operation results represented in registers τ, Z and O, and a circuit CEA (Complex Element Address) for

computing the address of a complex element in DM: if v is the address of a complex κ, then the address of qth element in κ is $t = v + 4q$ for logical κ or $t = v + q$ for symbol κ.

CD organizes the work of the processor according to the following algorithm.

1. CD selects from IM an instruction at the address pointed in IC and writes it to IR.

2. ODec decodes the contents of the operation code field, ADec decodes the contents of type and operand (complex and/or variable) address fields in IR.

3. CD takes the information from ODec and Adec and generates the signals either for selecting from DM (possibly by means of the circuit CEA) an operand (constant, variable, logical complex or complex element) at the corresponding address and writing it into one of the CURs, or, if the operand is a constant given explicitly in the instruction, for writing it to the register O or to IC.

4. If the operation in the instruction is related to the functional type being a logical or arithmetic one, CD generates a signal initiating the corresponding OD.

5. Initiated OD fulfils the operation pointed in the instruction in IR, and CD writes results into CURs according to the operation.

6. If the operation code is of the transition type, the value of the variable address field in the instruction in IR is written to IC; otherwise the state of IC is increased for selecting the next instruction from IM.

7. If the instruction in IR implies the creation of a dynamic complex with a variable capacity ξ in a jth subprogram, then the cardinality 0 and capacity ξ of this complex are written at the addresses of its parameters in W_j, and the value at address a_j is stored in W_j as the address of this complex in G_j and then it is increased by ξ.

For the subset of vLYaPAS containing no calls for subprograms and operations over complexes, the student S.Ye. Soldatov has implemented the processor in a programmable logical integrated circuit designed with the help of the computer-aided design system ISE WebPACK 9.2i by Xilinx. The maximal operating frequency of the circuit equals 50 MHz. The circuit occupies the third part of the Nexys2 FPGA debugging board.

Bibliography

1. LYaPAS, a Programming Language for Logic and Coding Algorithms. New York, London: Academic Press, 1969.
2. *Торопов Н. Р.* Язык программирования ЛЯПАС // Прикладная дискретная математика. 2009. № 2(4). С. 9–25.
3. *Nadler N.* User Group for Russian Programming Language // IEEE, Newsletter for Computer-Aided Design. 1971. Issue № 3. May/June.
4. *Charles J., Albright Jr.* An Interpreter for the Language LyaPAS. University of North Carolina at Chapel Hill: Department of Computer Science, 1974.
5. *Агибалов Г. П.* К возрождению русского языка программирования // Прикладная дискретная математика. 2012. № 3(17). С. 77–84.
6. *Закревский А. Д., Торопов Н. Р.* Система программирования ЛЯПАС-М. Минск : Наука и техника, 1978.
7. *Торопов, Н. Р.* Диалоговая система программирования ЛЕС. Минск: Наука и техника, 1985.
8. *Agibalov G. P., Lipsky V. B., Pankratova I. A.* Cryptographic extension of Russian programming language // Прикладная дискретная математика. Приложение. 2013. № 6. С. 93–98.
9. *Agibalov G. P., Lipsky V. B., Pankratova I. A.* Project of hardware implementation of Russian programming language // Прикладная дискретная математика. Приложение. 2013. № 6. С. 98–102.