no possibility to rewrite the return address, and, second, there are no operations for setting memory free by means of OS.

BIBLIOGRAPHY

1. LYaPAS, a Programming Language for Logic and Coding Algorithms / eds. M. A. Gavrilov and A. D. Zakrevskii. New York, London: Academic Press, 1969. 475 p.
2. *Toropov N. R.* Programming language LYaPAS // Applied Discrete Mathematics. 2009. No. 2(4). P. 9–25. (in Russian).
3. *Nadler N.* User group for Russian programming language // IEEE, Newsletter for Computer-Aided Design. 1971. Iss. 3.
4. *Charles J. and Albright Jr.* An Interpreter for the Language LYaPAS. University of North Carolina at Chapel Hill: Department of Computer Science, 1974. 125 p.
5. *Agibalov G. P.* To reanimation of Russian programming language // Applied Discrete Mathematics. 2012. No. 3(17). P. 77–84. (in Russian).
6. *Zakrevskij A. D. and Toropov N. R.* Programming system LYaPAS-M. Minsk: Nauka i Technika, 1978. 240 p.(in Russian).
7. *Toropov N. R.* Dialogue programming system LES. Minsk: Nauka i Technika, 1985. 263 p. (in Russian).
8. *Agibalov G. P., Lipsky V. B., and Pankratova I. A.* Project of hardware implementation of Russian programming language // Applied Discrete Mathematics. Application. 2013. No. 6. P. 98–102.
9. *Broslavskiy O. V.* AES in LYaPAS // Applied Discrete Mathematics. Application. 2013. No. 6. P. 102–104.

УДК 004.43, 004.056

PROJECT OF HARDWARE IMPLEMENTATION OF RUSSIAN PROGRAMMING LANGUAGE

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

The projects of a LYaPAS-T processor implementing the programming language LYaPAS-T in hardware and of a preprocessor translating LYaPAS-T programs into the executive code of the processor are presented. It is also told that for a LYaPAS-T subset containing neither subprograms, nor operations over complexes and long operands, the architecture of the processor was described in VHDL, tested by means of a computer simulation, and implemented in a programmable logical integrated circuit obtained with the help of a computer-aided design.

Keywords: *Russian programming language, LYaPAS-T, hardware implementation, LYaPAS-T preprocessor.*

In [1] a cryptographic extension LYaPAS-T of Russian programming language and a compiler for its implementation in software were presented. The objective of this paper is to present an information about the project of a processor implementing LYaPAS-T programs in hardware, and a preprocessor translating LYaPAS-T programs to executive code for the processor.

1. Parameters

In LYaPAS-T implemented in hardware, the operand length, the largest number of a complex and the maximal quantity of subprograms in the hierarchical structure of a program are assumed to be bounded by natural n, m and k respectively. Hence, the quantities of different complexes (logical and symbol) and variables in a LYaPAS-T program executed by a processor do not exceed, respectively, mk and $27k$. Nowadays the values like $m = 64$, $k = 128$ are quite sufficient for the majority of practical algorithms.

2. Executive code

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. The field for complex address is used if the operation deals with the data of the form $\varkappa v$ or L. In the first case the address of static complex $\varkappa$ being, by definition, the address of the first element in $\varkappa$ is explicitly written in this field, and the address of the variable v is written in the field for variable address. In the second case the field of the complex address contains the address of the complex L, and the field of the variable address remains empty (equals 0). In all other cases the field of the complex address is empty. The same is true for dynamic complex with the following exception: the field of its address contains not the complex address itself but the address where it is kept.

3. Architecture

The architecture of the processor implementing LYaPAS-T in hardware consists of three units: Memory, Arithmetic Logical Unit (ALU) and Control Device (CD).

3.1. Memory

The Memory is divided into two segments — IM (Instruction Memory) and DM (Data Memory) used to store, respectively, a sequence of instructions in LE-code representing a LYaPAS-T program P and data for it — unit constants Ii, complexes and variables for every subprogram in the hierarchical structure of the program P. Accordingly, for P with k subprograms, DM is conditionally divided into four sections: section I — for keeping vectors I$i, i = 0, 1, \ldots, n-1$; sections C and G divided into k subsections C_j and G_j — for keeping respectively static and dynamic complexes in jth subprogram; and section W also divided into k subsections W_j — for keeping local variables a, b, …, z, Z belonging to jth subprogram, and parameters and addresses of all complexes in jth subprogram.

Sections I, C, W form so called static memory and section G — the dynamic memory of the processor. The data allocation in the first is made by the L-preprocessor (before the execution of the program P), in the second — 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 engaged elements in the subsection G_j is fixed 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. Particularly, before the beginning of the processor work addresses a_j for all j equal the number of elements in the static memory.

Static complexes being created in jth subprogram are placed in C_j by L-preprocessor pointing for them parameters and addresses in W_j. The cardinality and the address in W_j of a dynamic complex are remained by L-preprocessor undetermined. A dynamic complex being created in jth subprogram with the cardinality equaled the value of a variable ξ is placed in G_j as an array of a certain size with the address of the initial (first) element written in the element of W_j whose address is a_j. It is done in the same way by the processor at the moment when it executes the given operation: parameters and the address of the complex are stored in W_j, the value of the element having address a_j is increased by ξ.

3.2. A L U

The unit ALU consists of 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, and also of Operational Devices (OD) and a circuit CEA (Complex Element Address) for determining the address in DM of a complex element.

Operational devices are used for executing arithmetic and logical operations in LYaPAS-T over operands and with the operation results represented in registers τ, Z and O.

For a complex element $\varkappa\upsilon$, its address in DM is computed by the circuit CEA as $\theta = \varphi + 4\upsilon$ for logical $\varkappa$ or as $\theta = \varphi + \upsilon$ for symbol $\varkappa$ where φ is the address in DM of the first element in $\varkappa$ (also called the address of the complex $\varkappa$ itself).

3.3. C o n t r o l d e v i c e

The main functions of CD are the following: to receive and analyse information signals of other units, to select the next instruction from IM, to identify the operation code in it, to determine the operand address in DM and the next instruction address in IM, and to form and send control signals to other executive units. For carrying out these functions, CD contains Instruction Counter (IC), Instruction Register (IR) and two decoders — Address Decoder (ADec) and Operation Decoder (ODec).

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

In this algorithm, the work of the control device CD is formally described by the finite automata model.

5. L-preprocessor

For translating LYaPAS-T program into LE-code, a special compiler (called L-preprocessor) is used. It can be written in any programming language (for example in LYaPAS-T) and run on any computer provided with a proper compiler. The following is the sequence of main actions which L-preprocessor should make over a LYaPAS program P:

1) for every subprogram in the hierarchical structure of the program P, to prescribe to it a unique number from the series 1, 2, ..., k;

2) for every $j = 1, 2, \ldots, k$, to give a unique address in W_j to every local variable and to every static local complex met in the jth subprogram of P;

3) for all $i, j \in \{1, 2, \ldots, k\}$ where $i \neq j$, for every call for jth subprogram from ith subprogram in P, the real parameters, pointed in the call, to substitute for the corresponding abstract (external) parameters in the text of jth subprogram; for every dynamic complex in jth subprogram, the address in W_j, where the address of this complex in G_j is written, to substitute for the name of this complex in the text of jth subprogram; and the sequence of instructions a_i, $\Rightarrow a_j$ and the text of jth subprogram to substitute for the call itself in P;

4) every other operation in P to replace with the equivalent sequence of instructions in LE-code — either directly or through the intermediate replacement by the series of unary operations each having not more than one operand being different from τ.

6. Alternative variant

In the alternative variant of the processor, the segment IM saves the executive codes both of a head program and of all subprograms in its hierarchical structure. In this case, if the program P contains the call for a subprogram S, the L-preprocessor substitutes for this call in P an instruction A with the operation code of the transition to the address of the subprogram S executive code in IM, and, at the end of this code, it writes an instruction with the operation code of return back, that is, of transition to the address of an instruction in IM next to A. Besides, when executing the program P, the processor, before the execution of the instruction A, transfers the values of the input operands of the subprogram S, pointed in its call, at the corresponding addresses in DM, written in its executive code, and, before returning back to the instruction next to A, transfers the results of the subprogram code work at the addresses of the corresponding output operands, pointed in its call.

The transferring the values of the program operands decreases the rate of the processor work in comparison with its basic variant, but essentially reduces the necessary size of the segment IM. At the same time, the absence of transferring between operands in the basic variant does not guarantee the value integrity for the input operands of a subprogram if it is not provided by the programmer.

7. Applications

There are, at least, two possible applications of LYaPAS-T processor: as a cryptoprocessor and as a control processor. In the first case the segment IM is filled in LE-code of a program in LYaPAS-T representing a cryptographic algorithm, and data for it (a plaintext, a ciphertext, a key and others) are written in the segment DM. In the second case IM and DM are used for storing, respectively, LE-code of a LYaPAS-T program destined for the secure control of a critically important object (cosmic, energetic, transport, etc.) and data for this program.

8. vLYaPAS subset processor

Let L_1 be the vLYaPAS subset that includes all operations at the first level of vLYaPAS and does not contain (calls for) subprograms and operations over complexes. For the first time, a processor architecture for L_1 (also called L_1-processor) was elaborated and implemented in VHDL in 2012 by S. E. Soldatov, a student of the Information Security and Cryptography Department of Tomsk State University. For preliminary verification, all individual units in L_1-processor and its architecture on the whole were simulated by means of the program product ModelSim PE Student Edition 10.1d. Besides, the programmable logical integrated circuit of L_1-processor was synthesized 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 which is equivalent to the circuit delay of 20 ns. The size of the circuit is the third of the size of Nexys2 FPGA debugging board by Digilent Inc.

This result shows that the implementation in hardware of the processor for LYaPAS-T is the quite real affair promising trustworthy means for effective performance of cryptographic and other combinatorial algorithms.

BIBLIOGRAPHY

1. *Agibalov G. P., Lipsky V. B., and Pankratova I. A.* Cryptographic extension of Russian programming language // Applied Discrete Mathematics. Application. 2013. No. 6. P. 93–98.

УДК 004.43, 004.056

AES IN LYAPAS

O. V. Broslavskiy

Programs in vLYaPAS representing the encryption and key expansion algorithms for symmetric block cipher AES are presented.

Keywords: *AES, LYaPAS.*

The objective of the paper is to present the description of the AES encryption and key expansion algorithms [1, 2] in the revised Russian programming language vLYaPAS [3]. The presented programs show the compactness, transparency and effectiveness of cryptographic algorithm representations in the language which was originally aimed at the representation of logical synthesis algorithms. It is assumed that the number of the cipher rounds is 10, and the lengths of the cipher block and key equal 128 bits. A ciphertext block is considered as a 2-measured array of 4×4 bytes. It is called a *state* and is represented by a logical complex of cardinality 4 whose elements are the rows of the state.

Further, the texts of the head programs and their subprograms are given. The external parameters in them are the following: L1 — the state (with the initial value equaled a plaintext block); L2 — the array of eleven 128-bit round keys (the complex of cardinality 44); L3 — ciphertext block; L4 — substitution table (S-box) for the operation of byte substitution; L5 — private key.

Encryption of a block

```
Encrypt(L1,L2,L4/L3)
   *AddRoundKey(L1,L2,0/L3) Oi
§1 ∆i⊕10↪2
   *SubBytes(L3,L4/L3)
   *ShiftRows(L3/L3)
   *MixColumns(L3/L3)
   *AddRoundKey(L3,L2,i/L3) → 1
§2 *SubBytes(L3,L4/L3)
   *ShiftRows(L3/L3)
   *AddRoundKey(L3,L2,10/L3) **
```