МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 004.43, 004.056

О КРИПТОГРАФИЧЕСКОМ РАСШИРЕНИИ И ЕГО РЕАЛИЗАЦИИ ДЛЯ РУССКОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ

Г. П. Агибалов, В. Б. Липский, И. А. Панкратова

Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: agibalov@isc.tsu.ru, lipsky@mail.tsu.ru, pank@isc.tsu.ru

Представлено расширение русского языка программирования ЛЯПАС, получившее название ЛЯПАС-Т и заключающееся в увеличении длины операндов и расширении множества элементарных операций над ними. Необходимость в нём продиктована, в первую очередь, потребностями страны в доверенных и эффективных программной и аппаратной реализациях современных криптографических алгоритмов в безопасных компьютерных системах логического управления критически важными объектами, такими, как космические системы, энергетические установки, ядерное оружие, подводные лодки, беспилотники и т. п. Представлены также компилятор ЛЯПАСа-Т, генерирующий его загрузочный модуль для операционной системы Linux, и проекты процессора, реализующего ЛЯПАС-Т аппаратно, и препроцессора, конвертирующего программы на ЛЯПАСе-Т в исполняемый код процессора. Сообщается о процессоре для подмножества ЛЯПАСа-Т без подпрограмм, операций над комплексами и длинных операндов, описанном на VHDL, протестированном средствами компьютерного моделирования и реализованном на ПЛИС с помощью системы автоматизированного проектирования.

Ключевые слова: *Русский язык программирования, ЛЯПАС-Т, компилятор, препроцессор, процессор, аппаратная реализация.*

Введение

Здесь под Русским языком программирования подразумевается алгоритмический язык ЛЯПАС, разработанный в начале 1960-х годов в Томском государственном университете под руководством А. Д. Закревского и предназначенный для представления логико-комбинаторных алгоритмов решения задач прикладной дискретной математики, встречающихся в синтезе дискретных автоматов [1, 2]. Имя Русского языка программирования ему дали американские учёные [3]. До 1990-х годов ЛЯПАС интенсивно использовался в Советском Союзе [2], США [4], Германии, Польше, Чехословакии и многих других странах. В настоящее время ЛЯПАС успешно возрождается силами кафедры защиты информации и криптографии Томского государственного университета, главным образом, с целью разработки доверенного системного и прикладного программного обеспечения для автоматизированного проектирования безопасных компьютерных систем логического управления и для безопасной и эффективной реализации криптографических алгоритмов [5]. Среди многочисленных языков програм-

мирования, известных сегодня, ЛЯПАС предстаёт как наиболее подходящий для этих целей.

Вместе с тем есть один существенный и, возможно, единственный недостаток ЛЯПАСа — отсутствие в нём ряда элементарных операций, которые широко используются в современных криптографических алгоритмах: для арифметики длинных чисел, вычислений в многомерных пространствах над конечными полями и кольцами, решения комбинаторных задач над большими множествами и др. Кстати сказать, этот недостаток присущ всем современным языкам программирования, включая и те, что моложе ЛЯПАСа. В некоторых из них он преодолевается путём написания классов длинных чисел, больших дискретных функций и т. п. Что касается ЛЯПАСа, этот его недостаток более эффективно преодолевается расширением самого языка путём распространения элементарных операций в ЛЯПАСе на логические комплексы, которые, как известно, допускают арифметическую интерпретацию, и добавлением к нему некоторых новых элементарных операций, определённых над переменными и логическими комплексами. Последняя версия ЛЯПАСа — язык ЛЯПАС-М [6, 7], подвергнутый предварительно некоторой ревизии и затем расширенный (exTended) указанным образом, — называется ЛЯПАС-Т.

Ревизия ЛЯПАСа-М касается символики языка и арифметических операций умножения и деления целых чисел. Её результат назван vЛЯПАСом (от vLYaPAS, или reVised LYaPAS). В нём вместо больших русских букв используются большие латинские буквы, символы некоторых операций заменены другими, более подходящими знаками, и операции умножения и деления определены с сохранением переполнения и остатка соответственно. Для хранения последних в язык введена специальная переменная Z.

Цель данной работы — представить информацию о некоторых из первых результатов, полученных в процессе возрождения ЛЯПАСа, а именно: о расширении ЛЯПАС-Т, обусловленном, главным образом, требованиями криптографических алгоритмов; о компиляторе ЛЯПАСа-Т, генерирующем код в формате исполняемого файла операционной системы (ОС) Linux; о процессоре, реализующем программы на ЛЯПАСе-Т аппаратно; о препроцессоре, транслирующем эти программы в исполняемый код процессора; о воплощении процессора в ПЛИС для подмножества vЛЯПАСа.

Начнём изложение с краткого обзора языка vЛЯПАС.

1. vЛЯПАС

Программа на vЛЯПАСе представляет собой последовательность предложений, каждое из которых (кроме, возможно, первого) начинается с §s, где s — целое неотрицательное число, и которые, в свою очередь, являются последовательностями операций над операндами языка.

1.1. Операнды

Операндами в vЛЯПАСе являются константы, переменные, комплексы и элементы комплексов. Они используются для представления неотрицательных целых чисел, булевых векторов, символов Unicode и последовательностей из них. Компоненты в булевом векторе нумеруются, начиная с 0 в направлении справа налево. В vЛЯПАСе неотрицательные целые ограничиваются числом $2^{32} - 1$, а длина булева вектора — числом 32. Булев вектор длины 32 называется словом и рассматривается также как неотрицательное целое, представленное в двоичном позиционном коде. Булев вектор любой длины $n \geqslant 1$ с одной единичной компонентой называется далее единичным вектором.

В vЛЯПАСе имеются натуральные, единичные и символьные константы. Натуральные константы записываются как десятичные, шестнадцатеричные, восьмеричные и двоичные числа. Единичная константа — это единичный вектор. Она обозначается Ii, если номер компоненты 1 в ней равен i. Символьная константа — это последовательность символов Unicode. В ней символы $'$ и \ записываются как пары \$'$ и \\ соответственно.

Переменные в vЛЯПАСе принимают значения булевых векторов длины 32. Их количество равно 27. Они обозначаются буквами a, b, ..., z, Z и, как сказано выше, Z используется в специальных целях. Кроме того, есть ещё виртуальная переменная, называемая собственной или внутренней переменной языка. В отличие от остальных, реальных, переменных, она не участвует в записи программ на ЛЯПАСе, но появляется в них неявно как результат любой элементарной операции и может быть использована любой последующей операцией в программе. Для удобства изложения эту переменную принято называть τ.

Комплекс есть конечное линейно упорядоченное множество элементов, которые в символьном комплексе суть коды символов (булевы векторы длины 8, или байты), а в логическом — булевы векторы длины 32, или слова. Каждый комплекс имеет уникальный номер из ряда 0, 1, 2, ... Логический или символьный комплекс, имеющий номер i, обозначается Li или Fi соответственно. Действительное и максимально возможное количества элементов в комплексе являются параметрами комплекса и называются его мощностью и ёмкостью соответственно.

Далее все эти понятия вводятся формально:

— переменная υ ::=a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z;
— особая переменная Z;
— десятичная цифра ϱ ::= 0|1|2|3|4|5|6|7|8|9, десятичая константа ∂ ::= $\varrho\{\varrho\}$;
— шестнадцатеричная цифра η ::= ϱ|A|B|C|D|E|F, шестнадцатеричная константа $\hbar$ ::= $\eta\{\eta\}$h;
— восьмеричная цифра ω ::= 0|1|2|3|4|5|6|7, восьмеричная константа ϖ ::= $\omega\{\omega\}$o;
— двоичная цифра ε ::= 0|1, двоичная константа β ::= $\varepsilon\{\varepsilon\}$b ;
— натуральная константа ι ::= $\partial|\hbar|\varpi|\beta$;
— символ σ ::= любой символ Unicode, символьная константа $\Sigma ::='\ \sigma\{\sigma\}'$;
— единичная константа $\mathcal{I}$ ::= I∂|Iυ; константа ::= $\iota|\Sigma|\mathcal{I}$;
— символьный комплекс $\mathcal{F}$::= F∂, логический комплекс $\mathcal{L}$ ::= L∂, комплекс $\varkappa$ ::= $\mathcal{F}|\mathcal{L}$;
— индекс $\mathcal{J}$ ::= $.\partial|\upsilon|(\upsilon+\partial)|(\upsilon-\partial)$; элемент комплекса ::= $\varkappa\mathcal{J}$;
— Q∂ ::= мощность комплекса с номером ∂, S∂ ::= ёмкость комплекса с номером ∂;
— значение натуральной константы ::= неотрицательное целое|булев вектор;
— значение символьной константы ::= строка символов;
— значение переменной ::= неотрицательное целое|булев вектор;
— значение элемента логического комплекса ::= неотрицательное целое|булев вектор;
— значение элемента символьного комплекса ::= неотрицательное целое|булев вектор|символ.

В отличие от других языков программирования, типы значений в vЛЯПАСе не фиксированы. Тип значения (целое или вектор) для константы, переменной и элемента комплекса определяется типом операции (арифметической или логической соответственно), которая применяется к этому операнду.

1.2. Элементарные операции

Передача значения

Пусть, как обычно, τ есть собственная (внутренняя) переменная ЛЯПАСа, α — произвольные переменная или элемент комплекса, γ — переменная, элемент комплекса или константа, χ — значение на выходе генератора псевдослучайных чисел в компьютере (PRNG), ζ — начальное состояние PRNG, θ — значение на выходе таймера в компьютере. Тогда:

$\mathrm{O}\alpha$ — присвоение наименьшего значения (нулей): $\alpha := 00...0$;
$^{-}\alpha$ — присвоение наибольшего значения (единиц): $\alpha := 11...1$;
$\Rightarrow \alpha$ — присвоение τ: $\alpha := \tau$;
γ — присвоение γ: $\tau := \gamma$;
$\Leftrightarrow (\upsilon_1\upsilon_2)$ — обмен значениями переменных υ_1 и υ_2;
$\Leftrightarrow (\varkappa\upsilon_1\upsilon_2)$, или $\Leftrightarrow (\varkappa\upsilon\partial)$, $\Leftrightarrow (\varkappa.\partial\upsilon)$, или $\Leftrightarrow (\varkappa.\partial_1.\partial_2)$ — обмен значениями элементов комплекса $\varkappa\upsilon_1$ и $\varkappa\upsilon_2$, или $\varkappa\upsilon$ и $\varkappa.\partial$, или $\varkappa.\partial_1$ и $\varkappa.\partial_2$ соответственно;
X означает $\tau := \chi$; $\Rightarrow$X означает $\zeta := \tau$; T означает $\tau := \theta$.

Логические и арифметические операции

! — вычисление номера правой единицы: $\tau := !\tau$;
$\neg$ — отрицание $\tau := \neg\tau$;
% — вычисление веса булева вектора: $\tau := \%\tau$;
$\vee\gamma$ — дизъюнкция: $\tau := \tau \vee \gamma$;
$\&\gamma$ — конъюнкция: $\tau := \tau\&\gamma$;
$\oplus\gamma$ — сложение по модулю 2: $\tau := \tau \oplus \gamma$;
$< \gamma$ — левый сдвиг: $\tau := \tau < \gamma$;
$> \gamma$ — правый сдвиг $\tau := \tau > \gamma$;
$+\gamma$ — сложение по модулю 2^{32}: $\tau := \tau + \gamma$;
$-\gamma$ — вычитание по модулю 2^{32}: $\tau := \tau - \gamma$;
$*\gamma$ — умножение по модулю 2^{32}: $\tau := \tau * \gamma$, Z := переполнение;
$:\gamma$ — деление двойного числа: $\tau :=$ частное от деления Zτ на γ, Z := остаток;
$/\gamma$ — частное целочисленного деления: $\tau :=$ частное от деления τ на γ, Z := остаток;
$;\gamma$ — остаток целочисленного деления: $\tau :=$ остаток от деления τ на γ, Z := частное;
$\triangle\alpha$ — увеличение на 1: $\tau := \alpha := \alpha + 1$;
$\bigtriangledown\alpha$ — уменьшение на 1: $\tau := \alpha := \alpha - 1$.

1.3. Операции перехода

$\to \partial$ — безусловный переход к параграфу ∂;
$\hookrightarrow \partial$ — переход к параграфу ∂ по условию $\tau = 0$;
$\mapsto \partial$ — переход к параграфу ∂ по условию $\tau \neq 0$;
$\uparrow (\gamma_1 \diamond \gamma_2)\partial$ — переход к параграфу ∂ по отношению $\diamond \in \{=, \neq, <, >, \leqslant, \geqslant\}$;
$\rightarrowtail \partial$ — уход к параграфу ∂ с возвратом;
$\leftarrowtail$ — возврат к точке, следующей за точкой ухода $\rightarrowtail \partial$;
$\uparrow (\gamma)\partial$ — переход к параграфу ∂ по времени γ;
$\uparrow$X$\partial\alpha_1\alpha_2$ — перечисление единиц: если $\alpha_1 = 0$, то $\alpha_2 := 0$ и $\to \partial$, в противном случае правая 1 в α_1 заменяется на 0, её номер присваивается α_2 и выполняется следующая операция;
{assembly program} — ассемблерная вставка: выполняется программа на языке Ассемблера, указанная между { и }.

1.4. Операции над комплексами

Пусть ξ есть ∂ или υ; тогда:

@ + $\varkappa(\xi)$ — образование (создание) комплекса $\varkappa$ ёмкости ξ и мощности 0;

@ − $\varkappa$ — уничтожение комплекса $\varkappa$;

@%$\varkappa$ — сокращение ёмкости комплекса до его мощности;

O$\varkappa$ – очистка комплекса (в пределах его мощности): $\varkappa ::= 00\ldots0$;

@′$\sigma_1\sigma_2...\sigma'_m > \mathcal{F}$ — символы $\sigma_1, \sigma_2, ..., \sigma_m$ добавляются к символьному комплексу $\mathcal{F}$;

@ > $\varkappa\xi$ — вставка элемента: значение τ вставляется в комплекс $\varkappa$ перед ξ-м элементом (в отсутствие ξ — после последнего элемента);

@ < $\varkappa\xi$ — удаление элемента: ξ-й элемент (в отсутствие ξ — последний элемент) комплекса $\varkappa$ перемещается в τ, мощность комплекса уменьшается на 1;

@#$\varkappa_1\varkappa_2(\xi_1, \xi_2, \xi_3)$ — ξ_1 элементов комплекса $\varkappa_1$, начиная с ξ_2-го элемента (в отсутствие ξ_2 — первые ξ_1 элементов $\varkappa_1$), копируются в $\varkappa_2$, начиная с ξ_3-го элемента (в отсутствие ξ_3 — в конец $\varkappa_2$), мощности $\varkappa_1, \varkappa_2$ не изменяются.

Впредь комплекс, образованный с фиксированной ёмкостью (∂), называется статическим, а комплекс с переменной ёмкостью (υ) — динамическим.

Заметим также, что операция объявления части комплекса самостоятельным комплексом, существующая в ЛЯПАСе, не включена в vЛЯПАС по причине её потенциальной опасности.

1.5. Операции ввода-вывода

/$\mathcal{F}$ >C — вывод символьного комплекса $\mathcal{F}$ на консоль;

/′ς' >C — вывод символьной константы ς на консоль;

/$\mathcal{F}$ <C — ввод символьной константы с клавиатуры: вводимые символы добавляются к символьному комплексу $\mathcal{F}$, мощность комплекса увеличивается.

2. Расширение vЛЯПАСа

2.1. Длинная арифметика

Натуральные константы в расширении ЛЯПАС-Т являются целыми из множества $\{0, 1, \ldots, 2^n - 1\}$, где n кратно 32 и зависит от фактической реализации ЛЯПАСа-Т. В настоящее время значение $n = 2^{14}$ вполне приемлемо для криптографических приложений.

Обозначая число 2^{32} символом δ, любую натуральную константу c можно выразить в виде

$$c = c_0 + c_1\delta + c_2\delta^2 + \ldots + c_{r-1}\delta^{r-1} \tag{1}$$

для некоторых $r > 0$ и $c_i \in \Omega = \{0, 1, \ldots, 2^{32} - 1\}$, $i = 0, 1, \ldots, r - 1$. В стандартном двоичном представлении элементы множества Ω являются словами — булевыми векторами длины 32. Поэтому в ЛЯПАСе-Т последовательность $c_0, c_1, \ldots, c_{r-1}$ представляется логическим комплексом мощности r с c_i в качестве i-го элемента.

Все операции, определённые в vЛЯПАСе над переменными, в ЛЯПАСе-Т могут применяться к логическим комплексам. В случае арифметической операции последовательность элементов комплекса рассматривается как натуральная константа c, заданная формулой (1). Различные операнды для одной и той же арифметической операции могут иметь различные длины и типы (один из них — переменная, другой — комплекс). В случае логической операции значение комплекса рассматривается как булев вектор, являющийся конкатенацией элементов комплекса. Логические комплексы мощности $n/32$ со значениями единичных векторов являются единичными константами в ЛЯПАСе-Т.

2.2. Множественность собственной переменной

Таким образом, в отличие от ЛЯПАСа, в ЛЯПАСе-Т есть два типа операндов для элементарных операций: переменные длины в одно слово и логические комплексы различных длин — от одного до $n/32$ слов. Соответственно этому, в ЛЯПАСе-Т имеются и два типа собственной переменной — простая и комплексная. Первая — из ЛЯПАСа, имеет длину одного слова. Она может принимать значение любой переменной языка. В любой реализации ЛЯПАСа-Т, программной или аппаратной, она представляется в регистре процессора. Собственные переменные 2-го типа имеют длины логических комплексов и могут принимать значения последних. В аппаратной реализации ЛЯПАСа-Т все они могут представляться в одном и том же регистре максимальной возможной длины — n разрядов. В программной реализации ЛЯПАСа-Т, из-за отсутствия такого регистра, роль комплексной собственной переменной на время выполнения цепочки операций, начинающейся с обращения к логическому комплексу, возлагается непосредственно на этот комплекс, и хранится она по месту его расположения в памяти компьютера.

В дальнейшем изложении, там, где это не вызывает двусмысленности, собственная переменная любого типа, будь то простая или комплексная, обозначается (как в ЛЯПАСе) буквой τ и называется соответственно простой или комплексной τ.

2.3. Дополнительные операции

В дополнение к операциям в vЛЯПАСе расширение ЛЯПАС-Т содержит некоторые новые логические операции, используемые в записи криптографических алгоритмов, включая следующие, где $\lambda ::= \upsilon|\mathcal{L}|\varkappa\upsilon|\varkappa.\partial|\partial$:

1) $\Updownarrow \mathcal{L}$ — перестановка: компоненты булева вектора τ переставляются в соответствии с их порядковыми номерами, указанными в элементах логического комплекса $\mathcal{L}$;

2) $\_(\xi_1, \xi_2)$ — проекция: выбирается часть булева вектора τ с компонентами, имеющими номера в интервале (ξ_1, ξ_2);

3) $\Uparrow \xi_1\lambda(\xi_2, \xi_3)$ — вставка: часть булева вектора λ с компонентами, имеющими номера в интервале (ξ_2, ξ_3), вставляется в τ перед ξ_1-й компонентой (в отсутствие ξ_3 вставляется правая часть булева вектора λ длины ξ_2, в отсутствие (ξ_2, ξ_3) — весь булев вектор λ);

4) $\Downarrow (\xi_1, \xi_2)$ — редукция: часть булева вектора τ с компонентами, имеющими номера в интервале (ξ_1, ξ_2), вычёркивается из вектора;

5) $|\,\lambda$ — конкатенация: булев вектор λ присоединяется к τ;

6) $\ll \xi_1(\xi_2, \xi_3)$ или $\gg \xi_1(\xi_2, \xi_3)$ — левый или правый циклические сдвиги: часть булева вектора τ с компонентами, имеющими номера в интервале (ξ_2, ξ_3), циклически сдвигается на ξ_1 бит влево или вправо соответственно (в отсутствие ξ_3 сдвигается правая часть булева вектора τ длины ξ_2, в отсутствие (ξ_2, ξ_3) — весь булев вектор τ);

7) $\geqslant \varkappa$ — максимальный элемент комплекса $\varkappa$ помещается в τ, его номер — в Z;

8) $\leqslant \varkappa$ — минимальный элемент комплекса $\varkappa$ помещается в τ, его номер — в Z.

Что касается арифметических операций по модулю N (для некоторого натурального N), таких, как сложение и вычитание (mod N), умножение (mod N), возведение в степень (mod N) и другие, широко используемые в криптографии, фактически нет никакой реальной возможности для включения их в список элементарных операций ЛЯПАСа-Т из-за существования великого множества алгоритмов их выполнения, имеющих разные эффективности в различных случаях. Вместо этого решено эти алгоритмы реализовывать по мере надобности и возможности на ЛЯПАСе-Т и (или) на

языке Ассемблера и включать их в библиотеку для использования в программах на ЛЯПАСе-Т в качестве подпрограмм.

3. Компилятор ЛЯПАСа-Т

3.1. Что это такое?

Далее для краткости любая программа на ЛЯПАСе-Т и её подпрограммы называются L-программой и L-подпрограммами соответственно.

Для выполнения L-программы компьютером она должна быть подана в качестве параметра компилятору, который преобразует её в загрузочный модуль (исполняемый код) для ОС Linux. Компилятор запускается по команде последней
`>$ lc <prog>.l`,
где `<prog>.l` — это имя файла с L-программой, являющейся списком L-подпрограмм. (Рекомендуется файлу с L-программой давать имя с расширением l, но это не обязательно.) Первая в списке L-подпрограмма является головной. ОС Linux передаёт ей управление после загрузки файла в оперативную память компьютера. Порядок следования других L-подпрограмм в списке несущественный. Файл может содержать не все необходимые L-подпрограммы. В этом случае компилятор находит недостающие в файле `lib10.l`, являющемся библиотекой пользователя. В ней рекомендуется хранить наиболе часто используемые L-подпрограммы.

Результатом работы компилятора является загрузочный модуль, который хранится под именем `<prog>` (без расширения) в той же папке, где находится и L-программа. Запуск скомпилированной программы на выполнение осуществляется по команде `>$ ./<prog>`.

Компилятор написан на языке C++ с использованием библиотеки регулярных выражений, делающим его максимально простым и прозрачным.

3.2. Структура загрузочного модуля

Загрузочный модуль состоит из двух сегментов — сегмента программы (`.text`) и сегмента данных (`.data`). В свою очередь, первый сегмент состоит из подпрограмм, генерируемых компилятором для L-подпрограмм, вызываемых в процессе исполнения L-программы. Сегмент данных содержит: текущий адрес в памяти для размещения новых комплексов и границу памяти, отводимой ОС под комплексы; текущее состояние PRNG; единичные константы; веса всех булевых векторов в $\{0,1\}^8$ (нужны для реализации операции взвешивания %); все символьные константы, встречающиеся в L-программе.

3.3. Организация памяти

Для каждой L-подпрограммы все её локальные переменные, начала, мощности и ёмкости её локальных комплексов размещаются в стеке, образующем фрейм в 1420 байтов. Доступ к локальным данным осуществляется по фиксированным смещениям от начала фрейма (регистр `ebp`). Значение регистра `ebp` фрейма родительской подпрограммы также сохраняется во фрейме, образуя список фреймов вызванных L-подпрограмм.

Локальные комплексы L-подпрограммы размещаются в «куче». Адрес свободного участка кучи на момент вызова подпрограммы также сохраняется во фрейме.

Создание локального комплекса сопровождается проверкой свободного места путём сравнения величин ёмкости создаваемого комплекса, адреса свободного участка и границы памяти, отводимой под комплексы. Если места достаточно, то адрес начала комплекса получает значение адреса свободного участка, а адрес свободного участка

увеличивается на значение ёмкости комплекса. Если места недостаточно, то выполняется обращение к ОС для увеличения границы доступной памяти.

Такая организация памяти защищена от атак переполнения стека и «кучи», поскольку, во-первых, буферы (комплексы) убраны из стека вместе с возможностью переписать адрес возврата и, во-вторых, нет операций для освобожения «кучи» посредством ОС.

4. Процессор ЛЯПАСа-Т

4.1. Параметры

В ЛЯПАСе-Т, реализованном аппаратно, длина операнда, наибольший номер комплекса и наибольшее количество подпрограмм в иерархической структуре программы обозначаются натуральными n, m и k соответственно. Следовательно, количества различных комплексов (логических и символьных) и переменных в программе на ЛЯПАСе-Т, исполняемой процессором, равны соответственно mk и $27k$. В настоящее время максимальные значения $m = 64$, $k = 128$ вполне достаточны для большинства практических алгоритмов.

4.2. Исполняемый код

Для исполнения процессором программа на ЛЯПАСе-Т должна быть предварительно представлена последовательностью инструкций в исполняемом коде (называемом LE-код) для процессора. Каждая инструкция в нём содержит в себе поля под код операции, тип операнда (константа или нет, тип комплекса — логический, символьный, статический или динамический, если операнд — комплекс или его элемент) и под адреса комплекса и переменной в памяти данных. Поле адреса комплекса используется, если операция имеет дело с данными вида $\varkappa\upsilon$ или $\mathcal{L}$. В первом случае адрес статического комплекса $\varkappa$, являющийся, по определению, адресом его первого элемента, записывается в этом поле явно, и адрес переменной υ записывается в поле адреса переменной. Во втором случае поле адреса комплекса содержит адрес комплекса $\mathcal{L}$ и поле адреса переменной остаётся пустым (равно 0). Во всех других случаях поле адреса комплекса в инструкции пустое. То же самое справедливо и в отношении динамического комплекса с одним исключением: в поле адреса динамического комплекса указывается не сам адрес комплекса, а адрес, по которому он хранится.

4.3. Архитектура

Архитектура процессора, реализующего ЛЯПАС-Т аппаратно, содержит блоки: Память, Арифметико-логическое устройство (АЛУ), Устройство управления (CD — Control Device), Счётчик инструкций (IC — Instruction Counter), Регистр инструкций (IR — Instruction Register) и два Дешифратора — адреса (ADec) и операции (ODec).

Память

Память процессора подразделяется на два сегмента — IM (Instruction Memory) и DM (Data Memory), используемых для запоминания соответственно последовательности инструкций в LE-коде, представляющей некоторую программу P на ЛЯПАСе-Т, и данных для неё — единичных констант Ii, переменных и комплексов для каждой подпрограммы в иерархической структуре программы P, а также параметров (мощностей и ёмкостей) и адресов этих комплексов в DM. Соответственно, для P с k подпрограммами, DM подразделяется условно на четыре секции: секция I — для хранения векторов I$i, i = 0, 1, \ldots, n-1$; секции C и G, разбитые на k подсекций C_j и G_j — для хранения соответственно статических и динамических комплексов в j-й подпрограмме; и секция W, также разбитая на k подсекций W_j — для хранения принадлежащих

j-й подпрограмме локальных переменных a, b, ..., z, параметров и адресов всех комплексов j-й подпрограммы.

Секции I, C и W образуют так называемую статическую память, а секция G — динамическую память процессора. Размещение данных в первой выполняет препроцессор (до исполнения программы P), во второй — сам процессор (во время исполнения P).

Инструкции в IM и данные в DM располагаются плотно, без пропусков элементов памяти, в порядке следования секций I, C, W и G. Количество занятых элементов подсекции G_j в DM отображается значением одного из элементов в W_j. Адрес этого элемента обозначается далее a_j. Его значение есть наименьший из адресов свободных элементов в G_j. В частности, до запуска процессора все a_j совпадают с числом элементов в статической памяти DM.

Статические комплексы, создаваемые в j-й подпрограмме, размещаются в C_j препроцессором путём указания их параметров и адресов в W_j. Ёмкость и адрес в W_j динамического комплекса препроцессор оставляет неопределёнными. Динамический комплекс, создаваемый в j-й подпрограмме с ёмкостью, задаваемой значением некоторой переменной ξ, располагается в G_j в массиве требуемого размера с начальным элементом по адресу, записанному по адресу a_j. Это делается аналогичным образом процессором в момент исполнения им данной операции: параметры и адрес комплекса запоминаются в W_j, значение по адресу a_j увеличивается на ξ.

АЛУ

АЛУ состоит из трёх регистров τ, Z и O максимальной возможной длины n, называемых регистрами общего назначения (CURs – Common Use Registers) и предназначенных для представления соответственно переменных τ, Z и операнда, считываемого из DM, а также операционных устройств (OD — Operational Devices) и схемы CEA (Complex Element Address) определения адреса элемента комплекса в DM.

Операционные устройства используются для выполнения арифметических и логических операций в ЛЯПАСе-Т с участием операндов и результатов операций, представленных в регистрах τ, Z и O.

Для элемента комплекса $\varkappa\upsilon$ его адрес в DM вычисляется схемой CEA как $\theta = \varphi + 4\upsilon$, если $\varkappa = \mathcal{L}$, или как $\theta = \varphi + \upsilon$, если $\varkappa = \mathcal{F}$, где φ есть адрес в DM первого элемента в $\varkappa$ (называемый также адресом самого комплекса $\varkappa$).

Устройство управления

Основные функции блока CD следующие: получать и анализировать информационные сигналы от других блоков, выбирать следующую инструкцию из IM, идентифицировать код операции в ней, определять адрес операнда в DM и адрес следующей инструкции в IM, формировать и посылать управляющие сигналы к другим исполнительным блокам.

4.4. Алгоритм функционирования

1. Устройство управления CD выбирает из IM инструкцию по адресу, указанному в IC, и записывает её в IR.

2. ODec дешифрует содержимое поля кода операции, а ADec — содержимое полей типа и адреса операнда (комплекса и/или переменной) из инструкции в IR.

3. CD по информации от ODec и ADec генерирует сигналы либо для выбора из DM (возможно, посредством схемы CEA) операнда (константы, переменной, комплекса или элемента комплекса) по соответствующему адресу и для его записи в один из CURs, либо, если операнд является константой, заданной в инструкции явно, для записи его в регистр O или в IC.

4. Если операция в инструкции относится к функциональному типу, т. е. является арифметической или логической, то CD генерирует сигнал, инициализирующий соответствующее OD.

5. Это OD выполняет данную операцию, и результат записывается в CURs в соответствии с кодом операции.

6. Если код операции указывает на переход, то содержимое поля адреса переменной в инструкции в IR записывается в IC; в противном случае содержимое в IC увеличивается для выбора следующей инструкции из IM.

7. Если инструкция в IR указывает на создание некоторого динамического комплекса с переменной ёмкостью ξ в j-й подпрограмме, то мощность 0 и ёмкость ξ этого комплекса записываются по адресам его параметров в W_j, а значение по адресу a_j запоминается в W_j как адрес этого комплекса в G_j и затем увеличивается на величину ξ.

В этом алгоритме функционирования процессора поведение устройства управления CD описывается формально на языке конечных автоматов.

4.5. L - п р е п р о ц е с с о р

Для трансляции программы на ЛЯПАСе-Т в LE-код используется специальный компилятор, называемый L-препроцессор. Он может быть написан на любом языке программирования (на ЛЯПАСе-Т, например) для исполнения на любом компьютере, снабжённом соответствующим компилятором. L-препроцессор должен выполнять следующую последовательность действий над заданной L-программой P:

1) каждой подпрограмме в иерархической структуре программы P назначить уникальный номер из ряда 1, 2, ..., k;

2) для каждого $j = 1, 2, \ldots, k$ каждой локальной переменной и каждому статическому локальному комплексу j-й подпрограммы в структуре программы P поставить в соответствие уникальный адрес в W_j;

3) для каждого вызова j-й подпрограммы из i-й подпрограммы в P, $i, j \in \{1, 2, \ldots, k\}$, подставить реальные параметры, указанные в вызове, вместо соответствующих абстрактных (внешних) параметров в теле j-й подпрограммы и вместо самого вызова — инструкции a_i, $\Rightarrow a_j$ и текст j-й подпрограммы, в котором вместо имени всякого динамического комплекса указан адрес в W_j, где лежит адрес этого комплекса в G_j;

4) всякую другую операцию в программе заменить эквивалентной ей цепочкой инструкций в LE-коде — непосредственно либо с промежуточной заменой на цепочку унарных операций, т. е. таких, которые используют самое большее один операнд, отличный от τ.

4.6. А л ь т е р н а т и в н ы й в а р и а н т

В альтернативном варианте процессора в сегменте IM хранятся исполняемые коды как головной программы, так и всех подпрограмм в её иерархической структуре. В этом случае L-препроцессор вместо вызова подпрограммы подставляет в содержащую его (вызов) программу инструкцию A с кодом операции перехода по адресу исполняемого кода данной подпрограммы в IM и в конец последнего записывает инструкцию с кодом операции возврата, т. е. перехода по адресу инструкции, следующей в IM за A. Кроме того, в процессе исполнения программы процессор перед исполнением инструкции A пересылает значения входных операндов подпрограммы, указанных в её вызове, по соответствующим адресам DM в её исполняемом коде, а перед возвратом к инструкции, следующей за A, пересылает результаты исполнения кода подпрограммы по адресам соответствующих её выходных операндов, указанных в её вызове.

Пересылка значений операндов подпрограмм существенно снижает скорость работы процессора по сравнению с его базовым вариантом, но значительно сокращает объём требуемой памяти сегмента IM. Вместе с тем отсутствие пересылок между операндами в базовом варианте не гарантирует сохранения значений входных операндов подпрограммы, если это не предусмотрено программистом.

4.7. Применения

Есть, по меньшей мере, два возможных применения процессора ЛЯПАСа-Т: в качестве криптопроцессора и как управляющего процессора. В первом случае сегмент памяти IM заполняется LE-кодом некоторой программы на ЛЯПАСе-Т, реализующей некоторый криптографический алгоритм, а данные для него (открытый или шифрованный тексты, ключи и др.) записываются в сегмент DM. Во втором случае IM и DM используются для запоминания соответственно LE-кода ЛЯПАС-Т-программы, реализующей некоторый алгоритм, предназначенный для безопасного управления критически важным объектом (космическим, энергетическим, транспортным, технологическим и т. п.), и данных для этой программы.

5. Процессор для подмножества vЛЯПАСа

Пусть L_1 является подмножеством vЛЯПАСа, которое включает все операции первого уровня vЛЯПАСа и не содержит (вызовов) подпрограмм и операций над комплексами. Архитектура процессора для L_1 (называемого также L_1-процессором) была впервые разработана и описана на VHDL в 2012 г. С. Е. Солдатовым, студентом кафедры защиты информации и криптографии Томского государственного университета. С целью предварительной верификации все отдельные блоки в L_1-процессоре и его архитектура в целом промоделированы с помощью программного продукта ModelSim PE Student Edition 10.1d. Кроме того, с использованием автоматизированной системы проектирования ISE WebPACK 9.2i фирмы «Xilinx» синтезирована программируемая логическая интегральная схема L_1-процессора. Максимальная рабочая частота схемы равна 50 МГц, что эквивалентно схемной задержке в 20 нс. Схема занимает 1/3 часть отладочной платы Nexys2 FPGA, Digilent Inc.

Этот результат показывает, что аппаратная реализация процессора для ЛЯПАСа-Т является совершенно реальным проектом, обещающим создание доверенных средств для эффективного исполнения криптографических и безопасных управляющих алгоритмов.

Результаты работы доложены на 12 Всероссийской конференции «Сибирская научная школа-семинар с международным участием „Компьютерная безопасность и криптография“ — SIBECRYPT'13» [8, 9]. Там же был продемонстрирован алгоритм шифрования AES, представленный на языке vЛЯПАС [10].

ЛИТЕРАТУРА

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. *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. 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. *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. 240 с.
7. *Торопов Н. Р.* Диалоговая система программирования ЛЕС. Минск: Наука и техника, 1985. 263 с.
8. *Agibalov G. P., Lipsky V. B., and 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., and Pankratova I. A.* Project of hardware implementation of Russian programming language // Прикладная дискретная математика. Приложение. 2013. № 6. P. 98–102.
10. *Broslavskiy O. V.* AES in LYaPAS // Прикладная дискретная математика. Приложение. 2013. № 6. С. 102–104.