стов на ЛЯПАСе, пока не разработана ОС, поддерживающая ЛЯПАС изначально. Для достижения этой цели поставлены и решены следующие задачи:

— **Описание грамматики ЛЯПАСа-Т**. Из источников [3 – 5], а также из личной переписки с сотрудниками ТГУ собраны и обобщены сведения о синтаксисе и семантике ЛЯПАСа-Т.
— **Создание программы для построения синтаксического дерева**. Написаны программы для flex и Bison, связка которых генерирует LALR-парсер, строящий синтаксическое дерево программы на ЛЯПАСе-Т [6].
— **Создание трансляторов для ОС Windows и Linux**. Реализованы соответствующие программы, генерирующие код на языке ассемблера по синтаксическому дереву.

Реализованы некоторые алгоритмы дискретной математики на ЛЯПАСе-Т, демонстрирующие возможности языка:

— Алгоритм быстрого перемножения многочленов и длинных чисел на основе алгоритма Кули — Тьюки быстрого преобразования Фурье. Примеры комбинации его с алгоритмом Винограда для поиска свёртки длины маленьких простых чисел [7].
— Алгоритмы факторизации: ρ-метод Полларда, метод квадратичного решета [7].
— Алгоритм Миллера — Рабина проверки простоты чисел [7].

ЛИТЕРАТУРА

1. *Агибалов Г. П.* К возрождению русского языка программирования // Прикладная дискретная математика. 2012. № 3. С. 77–84.
2. *Торопов Н. Р.* Язык программирования ЛЯПАС // Прикладная дискретная математика. 2009. № 2. С. 9–25.
3. *Агибалов Г. П., Липский В. Б., Панкратова И. А.* О криптографическом расширении и его реализации для русского языка программирования // Прикладная дискретная математика. 2013. № 3. С. 93–104.
4. *Broslavskiy O. V.* AES in LYaPAS // Прикладная дискретная математика. Приложение. 2013. № 6. С. 102–104.
5. *Гречнев С. Ю., Стефанцов Д. А.* Модификация ЛЯПАСа для разработки ОС // Прикладная дискретная математика. Приложение. 2015. № 8. С. 129–131.
6. *Ахо А. В., Лам М. С., Сети Р., Ульман Д. Д.* Компиляторы: принципы, технологии и инструментарий. М.: Вильямс, 2008.
7. *Крэндалл Р., Померанс К.* Простые числа. Криптографические и вычислительные аспекты. М.: УРСС, 2011.

УДК 004.43+004.4’42+004.451 DOI 10.17223/2226308X/9/49

МОДУЛЬНЫЙ ТРАНСЛЯТОР С ЯЗЫКА ЛЯПАС

Д. А. Стефанцов, В. О. Сафонов, В. В. Першин, С. Ю. Гречнев, П. А. Томских

Сообщается о разработке модульного транслятора с языка программирования ЛЯПАС. Цель разработки — упрощение создания транслятора с ЛЯПАСа, написанного на ЛЯПАСе. Процесс трансляции разделяется на этапы, за каждый из которых ответствен один из модулей транслятора. Модули выполнены в виде исполняемых программ и обмениваются данными с помощью файлов. Для представления промежуточных результатов работы модулей используются вспомогательные языки программирования.

Ключевые слова: *ЛЯПАС, язык программирования, операционная система.*

Язык программирования ЛЯПАС [1] возрождается в Томском государственном университете силами кафедры защиты информации и криптографии с целью иметь высокопроизводительную доверенную вычислительную систему для создания доверенного ПО, разработки и исследования криптографических алгоритмов, управления сетевым оборудованием. Главные компоненты создаваемой системы — транслятор с ЛЯПАСа [2] и ОС ЛЯПАС [3].

В [3] описаны этапы создания ОС ЛЯПАС, после завершения которых появляется возможность разработки и компиляции ОС ЛЯПАС и прикладных программ для неё в самой ОС ЛЯПАС. Для выполнения последних двух этапов требуется транслятор с ЛЯПАСа, написанный на ЛЯПАСе. Модульный транслятор с языка ЛЯПАС — вспомогательное средство для создания требуемого.

Создание транслятора с ЛЯПАСа на ЛЯПАСе — трудоёмкая задача, требующая решения множества отдельных небольших проблем. Предлагается реализация транслятора, разделённого на модули, каждый из которых может быть написан на любом языке программирования независимо от других модулей. Такой транслятор может быть создан изначально, например, на C++ и переписан по частям на ЛЯПАСе. В качестве способа связи модулей выбрана передача данных через файлы в формате, не зависящем от языка реализации модулей.

1. Архитектура транслятора

Транслятор $t : X_1 \to X_{n+1}$ состоит из модулей $m_i : X_i \to X_{i+1}$, таких, что $t = m_1 \circ \circ m_2 \circ \ldots \circ m_n$; X_i — входной язык модуля m_i и выходной язык модуля m_{i-1}. Входной язык X_1 транслятора t — ЛЯПАС, выходной язык X_{n+1} — язык ассемблера; языки X_i, $i \in \{2, \ldots, n\}$, используются для представления промежуточных результатов трансляции; $X_1, \ldots, X_{n+1}$ называются *языками транслятора* t; $X_2, \ldots, X_n$ называются *внутренними языками транслятора* t. Возможно $X_i = X_j$ при $i \neq j$, например, входной и выходной языки оптимизирующего модуля могут совпадать.

Модули реализованы в виде исполняемых программ, каждая из которых может быть написана на любом языке программирования, независимо от других модулей. Обмен данными между модулями осуществляется с помощью временных файлов, содержащих программы на внутренних языках транслятора. Поскольку программы на внутренних языках транслятора анализируются, в основном, модулями транслятора и их разработчиками, эти программы можно представить в виде, лёгком для разбора машиной, и только во вторую очередь — человеком. Дополнительное требование к такому способу представления программ — наличие библиотек для разбора и генерации данных этим способом для большинства существующих языков программирования. В качестве такого способа представления выбран JSON [4]; в качестве библиотек для работы с JSON на языках C и C++ выбраны [5] и [6] соответственно.

На этапе, когда модули транслятора переписываются на ЛЯПАСе, необходима библиотека, конвертирующая программы, представленные с помощью JSON, в программы, представленные с помощью типов данных ЛЯПАСа — логических и символьных комплексов и чисел. Эта библиотека для представления внутренних языков транслятора (ПВЯТ) реализуется как обёртка над библиотекой для формата JSON на C. Такая технология добавления библиотек C в ЛЯПАС уже опробована на работе с файлами и сокетами в ОС GNU/Linux. После того, как все модули будут переписаны на ЛЯПАСе, необходимость в формате JSON отпадёт и модули смогут обмениваться программами, представленными с помощью комплексов и чисел. Промежуточные результаты трансляции можно будет записывать во временные файлы в бинарном виде с помощью

вспомогательной библиотеки, подпрограммы которой имеют те же заголовки, что и подпрограммы библиотеки ПВЯТ.

Примером такой архитектуры модульного транслятора с ЛЯПАСа частично послужил транслятор CompCert [7] с C, обладающий формальным (проверяемым машиной) доказательством корректности своей работы. В будущем планируется разработка формально верифицированного транслятора с ЛЯПАСа по образцу CompCert. Одной из задач, необходимых для достижения этой цели, является создание модульного транслятора, так как формальное доказательство упрощается разбиением его на части, соответствующие модулям. Другая важная задача для достижения указанной цели — создание операционной семантики ЛЯПАСа [8].

2. Языки транслятора

В настоящий момент в модульном трансляторе предусмотрены одиннадцать языков. Первый (входной язык транслятора) — ЛЯПАС. Одиннадцатый (выходной язык транслятора) — язык ассемблера. Внутренние языки модульного транслятора не имеют собственных имён, они пронумерованы числами от 2 до 10.

Язык 2. Программы на языке 2 представляют собой программы на ЛЯПАСе с добавленным к ним именем файла, в котором хранится данная программа. Это необходимо для вывода сообщений об ошибках на следующих этапах трансляции.

Язык 3. Программы на языке 3 представляют собой результат лексического анализа программы на ЛЯПАСе — последовательность *токенов*. Помимо типа и значения лексемы, в токенах хранится информация о нахождении лексемы в исходной программе на ЛЯПАСе: имя файла, номер строки и символа в строке. Транслятор из языка 2 в язык 3 реализуется на языке C с использованием генератора flex [9].

Язык 4. Программы на языке 4 представляют собой результат синтаксического анализа программы — дерево разбора. Узлы дерева разбора, помимо символов грамматики, содержат информацию об их нахождении в исходной программе на ЛЯПАСе. Транслятор из языка 3 в язык 4 реализуется на языке C с использованием генератора bison [10].

Язык 5. Язык 5 представляет собой язык абстрактной одноадресной машины, каждый регистр общего назначения которой соответствует переменной ЛЯПАСа. Каждый вызов подпрограммы на ЛЯПАСе создаёт в машине новый набор регистров. Копирование входных и выходных параметров выполняется абстрактной машиной автоматически. Поскольку ЛЯПАС изначально создавался для компиляции в язык одноадресной машины Урал-1 [1], язык 5 является наиболее простым выходным языком для трансляции с ЛЯПАСа из всех следующих языков модульного транслятора. Транслятор с языка 4 в язык 5 реализуется на языке C++.

Язык 6. Язык 6 отличается от языка 5 тем, что представляет собой язык абстрактной двухадресной машины. Переход от одноадресной к двухадресной машине облегчает явную работу со стеком на следующих этапах трансляции и помогает более эффективно производить оптимизацию получаемой программы. Транслятор с языка 5 в язык 6 реализуется на языке C++.

Язык 7. Язык 7 отличается от языка 6 явной работой со стеком при вызове подпрограмм. Механизм композиции функций, описанный в [2], реализуется транслятором с языка 6 в язык 7.

Язык 8. Язык 8 отличается от языка 7 наличием ограниченного количества регистров общего назначения и неограниченным количеством пронумерованных областей памяти, с которыми регистры могут обмениваться значениями. Набор регистров соот-

ветствующей абстрактной машины — один для всех вызовов всех подпрограмм. Транслятор с языка 7 в язык 8 производит отображение переменных ЛЯПАСа, каждая из которых до сих пор хранилась в собственном регистре абстрактной машины, в ограниченный набор регистров новой абстрактной машины. В необходимых местах значения регистров сохраняются в памяти для последующего восстановления. Количество регистров абстрактной машины для языка 8 — параметр. Это необходимо для обеспечения независимости транслятора в язык 8 от конкретной аппаратной архитектуры.

Язык 9. Язык 9 отличается от языка 8 наличием операций захвата и освобождения памяти. Адресная арифметика в данном языке запрещена. Операции с комплексами производятся в явном виде, в отличие от предыдущих языков, в которых абстрактная машина имела элементарные операции для работы с комплексами.

Язык 10. Язык 10 отличается от языка 9 наличием адресной арифметики при работе с памятью. Из языка 10 происходит трансляция в язык ассемблера. Производится замена некоторых специальных операций абстрактной машины для языка 10 вызовами заранее подготовленных ассемблерных процедур.

3. Этапы разработки

Разработка модульного транслятора с ЛЯПАСа разделяется на три этапа.

1) На первом этапе языки реализации модулей — C и C++, формат представления внутренних языков программирования — JSON.
2) На втором этапе производится перенос модулей на язык ЛЯПАС. Языки реализации модулей — C, C++ и ЛЯПАС, формат представления промежуточных данных — JSON. На этом этапе требуется библиотека ПВЯТ для модулей на ЛЯПАСе.
3) На третьем этапе все модули переписаны на ЛЯПАСе. Формат JSON становится ненужным. Библиотека ПВЯТ модифицируется так, что она записывает в файлы данные не в формате JSON, а напрямую — комплексы и целые числа. На этом этапе производится оптимизация транслятора и его перекомпиляция с помощью предыдущей версии этого же транслятора.

Заключение

В настоящий момент разработка модульного транслятора находится на первом этапе. Создана цепочка модулей, транслирующая исходную программу на ЛЯПАСе в язык 5. Ведётся разработка модуля, переводящего программу с языка 5 в язык 6.

На втором этапе планируется использование существующего монолитного транслятора с ЛЯПАСа, разработанного в ТГУ, который используется в настоящий момент [3].

ЛИТЕРАТУРА

1. *Торопов Н. Р.* Язык программирования ЛЯПАС // Прикладная дискретная математика. 2009. № 2. С. 9–25.
2. *Гречнев С. Ю., Стефанцов Д. А.* Модификация ЛЯПАСа для разработки ОС // Прикладная дискретная математика. Приложение. 2015. № 8. С. 129–131.
3. *Стефанцов Д. А., Томских П. А.* Разработка операционной системы на языке ЛЯПАС // Прикладная дискретная математика. Приложение. 2015. № 8. С. 134–135.
4. JSON. ECMA-404 The JSON Data Interchange Standard. `http://www.json.org/`. 2016.
5. Jansson: C library for encoding, deconding and manipulating JSON data. `https://github.com/akheron/jansson`. 2016.