УДК 004.4

Технологии трансляции языков программирования высокого уровня

А.В. Дроздов[1*], Т.П. Мансурова[2], А.В. Блинников[1]

[1] Сибирский федеральный университет, пр. Свободный, 79, г. Красноярск, 660041, Россия
[2] ККДНиТ, ул. Урицкого, 61, г. Красноярск, 660049, Россия

* E-mail: mansurovatp@mail.ru

Аннотация. В статье приводится краткая информация по истории развития языков программирования и рассматриваются различные подходы к реализации трансляторов для языков программирования высокого уровня. Отмечена роль ассемблера, появление которого ознаменовало переход от языков первого поколения к языкам второго поколения, которые уже использовали методы трансляции. Выделены виды трансляции, различающиеся по типу реализации: компиляция, интерпретация и динамическая компиляция.

Ключевые слова: программирование, трансляторы, ассемблер

Technologies for the translation of high-level programming languages

A.V. Drozdov[1*], T.P. Mansurova[2], A.V. Blinnikov[1]

[1] Siberian Federal University, Svobodny pr., 79, Krasnoyarsk, 660041, Russia
[2] Krasnoyarsk Science and Technology City Hall, Uritskogo Street, 61, Krasnoyarsk, 660049, Russia

* E-mail: mansurovatp@mail.ru

Abstract. The article provides brief information on the history of the development of programming languages and discusses various approaches to the implementation of translators for high-level programming languages. The role of the assembler is noted, the appearance of which marked the transition from languages of the first generation to languages of the second generation, which already used translation methods. Highlighted the types of translation, differing in the type of implementation: compilation, interpretation and dynamic compilation.

Keywords: programming, translators, assembler

1. Введение

На сегодняшний день существует огромное множество языков программирования, используемых специалистами для разработки компьютерных программ. Наиболее полная тематическая интернет-энциклопедия HOPL, по состоянию на 2020 год, насчитывает 8945 уникальных языков программирования. Хотя реальное число языков подсчитать невозможно, по некоторым оценкам сегодня оно может достигать 25,000 [1].

2. Проблема исследования

В оценке истории развития языков программирования, отмечают тенденцию к их общему усложнению, чему способствует увеличение числа абстракций – смысловых конструкций, кратко описывающих такие структуры данных и операции над ними, которые используются в решении прикладных задач наиболее часто.

Достижения в области компьютерной лингвистики, конкуренция на рынке программного и аппаратного обеспечения, ужесточение требований к процессам разработки, влекут за собой эволюцию подходов к реализации современных языков программирования [2].

С появлением первых ЭВМ, в роли языков программирования выступали машинные коды – наборы кодов операций конкретной вычислительной машины, которые интерпретируются непосредственно ее процессором [2]. До сих пор машинный код является самым низким уровнем представления компьютерных программ – он имеет компактное двоичное представление и практически непригоден для чтения человеком.

Для упрощения разработки с применением машинного кода, в 1947 году был разработан язык ассемблера, названый в честь своего транслятора (сборщика). Язык представляет собой систему условных обозначений – буквенных мнемоник, используемых вместо двоичных кодов вызова процессорных команд. Такой код проще и понятнее для чтения и написания со стороны программиста, но он не распознается процессором. Задача транслятора состояла в том, чтобы преобразовать исходные тексты программ на языке понятном программисту – в машинный код [3].

Разработка с применением машинных кодов осложнена тем, что каждая модель процессора имеет собственный набор команд, лишь частично совместимый с кодами других процессоров. Это заставляет программиста на машинном языке писать программы сразу в нескольких версиях, под разные платформы, с особым вниманием к

ISBN 978-5-6045165-2-2

различиям между платформами. Создание трансляторов способствует устранению этой проблемы [4].

Создание языка ассемблера ознаменовало переход от языков первого поколения к языкам второго поколения, использующим методы трансляции – преобразование программы, представленной на одном языке программирования, в программу на другом языке. Впоследствии, эти методы также использовались при разработке трансляторов языков следующих поколений, языком реализации которых выступали транслируемые языки предыдущих поколений [5].

3. Результаты и обсуждение

По типу реализации выделяют следующие виды трансляции:

- компиляция;
- интерпретация;
- динамическая компиляция.

Компиляция – вид трансляции, в которой в роли целевого языка выступают языки по уровню близкие к машинному коду.

Процесс включает в себя несколько этапов: лексический анализ, синтаксический анализ, семантический анализ, создание объектного кода. На стадии лексического анализа исходный текст программы разбивается на распознанные группы символов – лексемы. На выходе лексического анализатора – поток лексем (токенов), классифицированных по типам («оператор», «идентификатор», «литеральное значение» и т.п.). Сформированный поток токенов отправляется на вход синтаксического анализатора.

Целью синтаксического анализа является преобразование последовательности примитивных токенов в структуру данных, использование которой было бы удобно при дальнейшей обработке. Как правило, для этих целей используются древовидные структуры данных, именуемые Абстрактными Синтаксическими Деревьями. Преобразование выполняется в соответствии с правилами, определенными формальной грамматикой данного языка.

Семантический анализатор работает напрямую с продуктом синтаксического анализа – синтаксическим деревом. На этапе семантического анализа устанавливаются смысловые отношения между различными узлами графа, производится сопоставление типов оперируемых объектов, определение областей видимости переменных и прочих идентификаторов, корректность использования операторов и т.п. В случае отсутствия

ISBN 978-5-6045165-2-2

значимых ошибок семантического анализа, по результатам работы выполняется генерация объектного кода на целевом языке программирования.

По выбору целевого языка отдельно выделяют компиляторы транспилирующего типа. Особенность транспайлеров состоит в том, что в роли целевого языка программирования, вместо языков близких к машинным кодам, используются другие высокоуровневые языки. Примерами транспайлеров служат реализации таких языков как TypeScript, Dart и Closure, использующие JavaScript в качестве целевого языка.

Интерпретация – процесс чтения и выполнения исходного кода программой-интерпретатором. Выполнение может происходить сразу (чистая интерпретация) или после создания в памяти промежуточного представления кода (смешанная реализация).

Простые интерпретаторы можно использовать в интерактивном режиме, отсутствует необходимость повторной компиляции исходного кода при переходе на другую платформу или при внесении изменений.

Исходные коды на интерпретируемых языках программирования, как правило, более компактны по своему объему по сравнению с кодами на языках, компилируемых в машинный код. Интерпретатор, как часть виртуальной машины исполнителя, берет на себя ту часть ответственностей, которые при использовании компилируемых в машинный код языков несет сам программист. Так, в некоторых виртуальных машинах используются механизмы автоматического управления памятью, что убирает необходимость в использовании специальных инструкций для выделения / освобождения участков памяти.

Интерпретация исходного кода – процесс более медленный, чем простое выполнение аналогичной программы, скомпилированной в машинный код. Для повышения производительности интерпретаторов, в ряде реализаций используют технологии динамической компиляции.

Суть динамической компиляции заключается в компиляции наиболее часто исполняемых участков интерпретируемого исходного (промежуточного) кода в машинный код «на лету», прямо в момент исполнения. Каждый из участков кода компилируется, получившийся машинный код сохраняется в кеше процесса и при необходимости повторного использования, повторная компиляция не выполняется. Такой подход используется в ряде виртуальных машин (JRE, CLR, PyPy), он позволяет существенно увеличить скорость выполнения программ, однако требователен к ресурсам.

ISBN 978-5-6045165-2-2

3. Заключение

Краткий обзор по истории развития языков программирования позволил рассмотреть различные подходы к реализации трансляторов для языков программирования высокого уровня. Уделено внимание различным видам трансляции. Выделим компиляторы транспилирующего типа, особенностью которых является то, что в роли целевого языка программирования вместо языков, близких к машинным кодам, используются другие высокоуровневые языки.

Список литературы

1. Online Historical Encyclopaedia of Programming Languages [Электронный ресурс]. – URL: https://hopl.info/ (дата обращения 09.06.2020).
2. Языки и методы программирования. Лабораторный практикум. Раздел: Методы трансляции языков программирования: учебно-методическое пособие / Ю.В. Цыганова. – Ульяновск: УлГУ, 2011. – 39 с.
3. Касьянов, В.Н. Методы построения трансляторов / В.Н. Касьянов, И.В. Поттосин. – Москва: "Наука", 1986. – 344 с.
4. Толковый словарь по вычислительным системам = Dictionary of Computing / Под ред. В. Иллингуорта и др.: Пер. с англ. А.К. Белоцкого и др.; под ред. Е. К. Масловского. – М.: Машиностроение, 1990. – 560 с. – 70 000 (доп.) экз. – ISBN 5-217-00617-X (СССР), ISBN 0-19-853913-4 (Великобритания).
5. Вычислительная техника. Терминология: Справочное пособие. Выпуск 1. ГОСТ 19781-83. / Рецензент канд. техн. наук Ю.П. Селиванов. – М.: Издательство стандартов, 1989. – 168 с. – 55 000 экз. – ISBN 5-7050-0155-X.