КОМПИЛЯТОР

Авторы: А. Ю. Дроздов, А. В. Ильин

КОМПИЛЯТОР (от лат. compilo – собирать, составлять, компилировать) в информатике, компьютерная программа, которая преобразует исходный код программы на *языке программирования* высокого уровня (ЯПВУ) в функционально эквивалентный набор инструкций на языке низкого уровня (т. н. объектный код). К. является одним из видов *транслятора* и, как правило, входит в состав системного программного обеспечения компьютера. Каждый К. соответствует определённому ЯПВУ (паскалю, си, фортрану и др.) и одной или нескольким вычислит. платформам. Вычислит. платформа определяется архитектурой семейства центр. процессоров; напр., x86 – Intel 8086 Family Architecture (архитектура семейства Intel 8086), *операционной системой* и, в ряде случаев, дополнит. программным обеспечением (напр., виртуальной машиной), необходимым для работы исполняемого кода на процессорах данной архитектуры.

К. выполняет лексич., синтаксич., семантич. анализ исходного кода программы и генерацию объектного кода. На этапе лексич. анализа исходный код преобразуется в последовательность лексич. единиц – лексем (ключевые слова языка программирования, идентификаторы переменных, константы и др.). Во время синтаксич. анализа осуществляется проверка последовательности лексем на наличие синтаксич. ошибок (в соответствии с синтаксич. правилами языка программирования) и затем – преобразование этой последовательности в т. н. дерево разбора. Семантич. анализ предназначен для выявления логич. ошибок в исходной программе и определения значения языковых конструкций дерева разбора. После этого К. либо переходит к генерации объектного кода, либо завершает работу выводом сообщения об ошибках. Исходный код программы, как правило, содержится в нескольких файлах. К. преобразует каждый из них в отд. объектный модуль (файл, содержащий объектный код), а затем спец. программа-компоновщик собирает исполняемый код

программы из объектных модулей, стандартных библиотечных модулей и др. До создания компоновщиков сборка производилась компилятором.

Для эффективного использования особенностей архитектуры процессоров и улучшения характеристик исполняемого кода (производительности, компактности и др.) в К. встроены компоненты, оптимизирующие код. Различают оптимизацию машинно независимую (т. е. не зависящую от архитектуры процессора) и машинно зависимую. Машинно независимая оптимизация часто выполняется на этапе генерации объектного кода. Машинно зависимые преобразования направлены на эффективное использование особенностей архитектур процессоров, оптимальное распараллеливание программ (для многоядерных процессоров) и др.

Альтернативой К. и компоновщикам служат *интерпретаторы*. Для ряда ЯПВУ существуют как К., так и интерпретаторы. Для некоторых ЯПВУ (напр., Java) применяется двухэтапная компиляция или сочетание компиляции и интерпретации. На первом этапе на спец. промежуточном языке генерируется т. н. байт-код, который не привязан к конкретной операционной системе и архитектуре семейства процессоров. Байт-код предназначен для последующей интерпретации или т. н. JIT-компиляции (Just-In-Time compilation – компиляция «на лету») во время выполнения программы. На втором этапе преобразование байт- кода в исполняемый код (для конкретных операционной системы и процессора) осуществляется спец. программным обеспечением (напр., Java Virtual Machine – виртуальной машиной Java). Такой подход позволяет существенно уменьшить трудозатраты программистов, поскольку один и тот же байт-код может быть использован на разл. вычислит. платформах, включающих необходимую виртуальную машину. С кон. 20 в. большое внимание уделяется созданию т. н. двоичных К., которые позволяют переводить скомпонованный исполняемый код одной платформы в код альтернативной архитектурной платформы, обеспечивая перенос готового программного обеспечения.

К. классифицируют по разл. признакам: по времени запуска процесса компиляции, по классу целевых архитектур, по наличию оптимизирующих преобразований и др. По времени запуска различают К. статические и динамические. Статич. К. запускается программистом из *инструментальной системы* программирования или

командной строки. Динамич. К. запускается во время работы программы, т. е. параллельно с исполнением программы идёт перекомпиляция некоторых её частей. На основе классов целевых архитектур процессоров различают К. для архитектур с явно выраженным параллелизмом (EPIC – Explicitly Parallel Instruction Computing), для встроенных (embedded) архитектур, автоматич. распараллеливатели для многоядерных (multicore) архитектур и др.

Один из первых К. разработан в 1952 Г. Хоппер (США) для созданного ею языка программирования A-0 ЭВМ UNIVAC I. До появления ЯПВУ и их К. программы писали на языках низкого уровня (сначала в кодах машинных команд, позднее – на языках *ассемблера*). Значит. достижением автоматизации программирования стала разработка в 1957 К. для языка фортран ЭВМ IBM 704, выполненная под рук. Дж. Бэкуса (США). В 1950–60-х гг. исходные коды К. писались только на языках ассемблера. Одной из первых рос. работ в области оптимизирующей компиляции стал Альфа-транслятор с языка алгол-60 для ЭВМ М-20, созданный под рук. А. П. *Ершова* (1959–64). В 1970-х гг. реализованы ЯПВУ паскаль, си и др., которые стали применяться как для создания прикладных программ, так и для разработки К., что привело к значит. сокращению трудоёмкости создания К. Рост числа ЯПВУ и различных вычислит. платформ обусловливает появление новых К. и совершенствование методов их конструирования.

Литература

Лит.: Compilers: principles, techniques and tools. 2nd ed. Boston, 2007.