Трансля́тор — программа или техническое средство, выполняющее *трансляцию программы*

Трансля́ция програ́ммы — преобразование программы, представленной на одном из языков программирования, в программу на другом языке. Транслятор обычно выполняет также диагностику ошибок, формирует словари идентификаторов, выдаёт для печати текст программы и т. д.[1]

Язык, на котором представлена входная программа, называется *исходным языком*, а сама программа — *исходным кодом*. Выходной язык называется *целевым языком*.

В общем случае понятие трансляции относится не только к языкам программирования, но и к другим языкам — как формальным компьютерным (вроде языков разметки типа HTML), так и естественным (русскому, английскому и т. п.)[3][4].

Виды транслятор

Существует несколько видов трансляторов[2].

- *Диалоговый* транслятор — транслятор, обеспечивающий использование языка программирования в режиме разделения времени.
- *Синтаксически-ориентированный* (*синтаксически-управляемый)* транслятор — транслятор, получающий на вход описание синтаксиса и семантики языка, текст на описанном языке и выполняющий трансляцию в соответствии с заданным описанием.
- *Однопроходной* транслятор — транслятор, преобразующий исходный код при его однократном последовательном чтении (за один проход).
- *Многопроходной* транслятор — транслятор, преобразующий исходный код после его нескольких чтений (за несколько проходов).
- *Оптимизирующий* транслятор — транслятор, выполняющий оптимизацию создаваемого кода. См. оптимизирующий компилятор.
- *Тестовый* транслятор — транслятор, получающий на вход исходный код и выдающий на выходе изменённый исходный код. Запускается перед основным транслятором для добавления в исходный код отладочных процедур. Например, транслятор

с языка ассемблера может выполнять замену макрокоманд на код.
- *Обратный* транслятор — транслятор, выполняющий преобразование машинного кода в текст на каком-либо языке программирования. См. дизассемблер, декомпилятор.

Реализации

Цель трансляции — преобразование текста с одного языка на язык, понятный адресату. При трансляции компьютерной программы адресатом может быть:

- устройство — процессор (трансляция называется компиляцией);
- программа — интерпретатор (трансляция называется интерпретацией).

Виды трансляции:

- компиляция;
  - в исполняемый код
    - в машинный код
    - в байт-код
  - транспиляция;
- интерпретация;
- динамическая компиляция.

Компиляция

Язык процессора (устройства, машины) называется машинным языком, машинным кодом. Код на машинном языке исполняется процессором. Обычно, машинный язык — язык низкого уровня, но существуют процессоры, использующие языки высокого уровня (например, iAPX-432[5]). Однако, такие процессоры не получили распространения в силу своей сложности и дороговизны.

Компилятор — это вид транслятора, преобразующий исходный код с какого-либо языка программирования на машинный язык[6].

Процесс компиляции, как правило, состоит из нескольких этапов:

- лексический анализ;
- синтаксический анализ;
- семантический анализ;

- создание на основе результатов анализов промежуточного кода;
- оптимизация промежуточного кода;
- создание объектного кода, в данном случае машинного.

Программа может использовать сервисы, предоставляемые операционной системой, и сторонние библиотеки (например, библиотеки для работы с файлами и библиотеки для создания графического интерфейса). Для добавления в объектный файл машинного кода из других объектных файлов (кода статических библиотек) и информации о динамических библиотеках выполняется *связывание* (англ. *link*) или *компоновка*. Связывание или компоновка выполняется редактором связей или компоновщиком. Компоновщик может быть отдельной программой или частью компилятора. Компоновщик создаёт исполняемый файл. Исполняемый файл (программа) запускается следующим образом:

- по запросу пользователя в ядре операционной системы создаётся объект «процесс»;
- загрузчик программ операционной системы выполняет следующие действия:
- читает исполняемый файл;
- загружает его в память;
- загружает в память динамические библиотеки;
- выполняет связывание машинного кода программы с динамическими библиотеками (динамическое связывание);
- передаёт управление программе.

Достоинства компиляции:

- компиляция программы выполняется один раз;
- наличие компилятора на устройстве, для которого компилируется программа, не требуется.

Недостатки компиляции:

- компиляция — медленный процесс;
- при внесении изменений в исходный код, требуется повторная компиляция.

Ассемблер — компилятор, преобразующий текст с языка ассемблера на машинный язык. Язык ассемблера — язык, близкий к машинному языку, язык низкого уровня.

Интерпретация

Интерпретация — процесс чтения и выполнения исходного кода. Реализуется программой — интерпретатором.

Интерпретатор может работать двумя способами:

1. читать код и исполнять его сразу (*чистая интерпретация*[6]);
2. читать код, создавать в памяти промежуточное представление кода (байт-код или p-код), выполнять промежуточное представление кода (*смешанная реализация*[6]).

В первом случае трансляция не используется, а во втором — используется трансляция исходного кода в промежуточный код.

Этапы работы интерпретатора:

- лексический анализ;
- синтаксический анализ;
- семантический анализ;
- создание промежуточного представления кода (при чистой интерпретации не выполняется);
- исполнение.

Интерпретатор моделирует машину (виртуальную машину), реализует цикл выборки-исполнения команд машины. Команды машины записываются не на машинном языке, а на языке высокого уровня. Интерпретатор можно назвать исполнителем языка виртуальной машины.

Чистая интерпретация применяется, обычно, для языков с простой структурой, например, языков сценариев, языков APL и Лисп.

Примеры интерпретаторов, создающих байт-код: Perl, PHP, Python, Erlang.

Достоинства интерпретаторов по сравнению с компиляторами:

- возможность работы в интерактивном режиме;
- отсутствие необходимости перекомпиляции исходного кода после внесения изменений и при переносе кода на другую платформу.

Недостатки интерпретаторов по сравнению с компиляторами:

- низкая производительность (машинный код исполняется процессором, а интерпретируемый код — интерпретатором;

машинный код самого интерпретатора исполняется процессором);
- необходимость наличия интерпретатора на устройстве, на котором планируется интерпретация программы;
- обнаружение ошибок синтаксиса на этапе выполнения (актуально для чистых интерпретаторов).

Сравнение чистого интерпретатора и интерпретатора, создающего байт-код:

- чистый интерпретатор проще в реализации, так как для него не нужно писать код транслятора;
- интерпретатор, создающий байт-код, может выполнять его оптимизацию и добиваться большей производительности, чем чистый интерпретатор;
- интерпретатор, создающий байт-код, потребляет больше ресурсов системы (трансляция в байт-код занимает процессорное время; байт-код занимает место в памяти).

Динамическая компиляция

Динамическая или JIT компиляция — трансляция, при которой исходный или промежуточный код преобразуется (компилируется) в машинный код непосредственно во время исполнения, «на лету» (англ. *just in time*, *JIT*). Компиляция каждого участка кода выполняется только один раз; скомпилированный код сохраняется в кеше и при необходимости используется повторно.

Достоинства динамической компиляции по сравнению с компиляцией:

- скорость работы динамически компилируемых программ близка к скорости работы компилируемых программ;
- отсутствие необходимости перекомпиляции программы при переносе на другую платформу.

Недостатки динамической компиляции по сравнению с компиляцией и чистой интерпретацией:

- бóльшая сложность реализации;
- бóльшие требования к ресурсам.

Динамическая компиляция хорошо подходит для веб-приложений.

Динамическая компиляция появилась и поддерживается в той или иной мере в реализациях Java, .NET Framework, Perl, Python.

Смешение понятий трансляции и интерпретации

Понятия «трансляция» и «интерпретация» различаются. Во время трансляции выполняется преобразование кода программы с одного языка на другой. Во время интерпретации программа исполняется.

Так как целью трансляции является, обычно, подготовка к интерпретации, эти процессы рассматриваются вместе. Например, языки программирования часто характеризуются как «компилируемые» или «интерпретируемые» в зависимости от того, что преобладает при использовании языка: компиляция или интерпретация. Причём, практически все языки низкого уровня и третьего поколения, вроде ассемблера, Си или Модулы-2, являются компилируемыми, а более высокоуровневые языки, вроде Python или SQL — интерпретируемыми.

С другой стороны, существует взаимопроникновение процессов трансляции и интерпретации: интерпретаторы могут быть компилирующими (в том числе с динамической компиляцией), а в трансляторах может требоваться интерпретация для реализации метапрограммирования (например, для макросов в языке ассемблера, условной компиляции в Си или шаблонов в C++).

Более того, один и тот же язык программирования может и транслироваться, и интерпретироваться, и в обоих случаях должны присутствовать общие этапы анализа и распознавания конструкций и директив исходного языка. Это относится и к программным реализациям, и к аппаратным — так, процессоры семейства x86 перед исполнением инструкций машинного языка выполняют их декодирование, выделяя в опкодах поля операндов (указание регистров, адресов в памяти, констант), разрядности и т. п., а в процессорах Pentium с архитектурой NetBurst тот же самый машинный код перед сохранением во внутреннем кэше дополнительно транслируется в последовательность микроопераций.