

Введение в методы трансляции



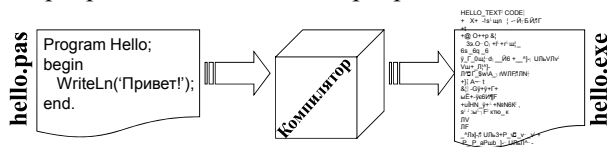
Если в вашей программе нет ошибок – сообщите разработчику транслятора, чтобы он исправил ошибку в трансляторе.
Из программистского фольклора

Содержание

- Понятие языкового процессора
- Компиляторы как класс программного обеспечения
- Упрощенная модель компилятора

Понятие языкового процессора

- *Компьютер* выполняет программы на языке машинных команд. *Программист* создает программы на языке программирования.
- *Языковой процессор* обеспечивает выполнение на компьютере директив и предложений программы, написанной программистом.



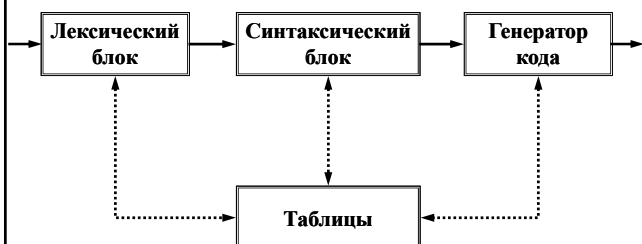
Типы языковых процессоров

- *Интерпретатор* – выполняет программу на исходном языке.
- *Транслятор* – переводит программу на исходном языке в язык машинных команд.
 - *Ассемблер* – транслятор языка низкого уровня.
 - *Компилятор* – транслятор языка высокого уровня.

Компиляторы как класс программного обеспечения

- Многочисленные реализации известных языков высокого уровня.
- Разработка новых языков высокого уровня требует разработки компиляторов.
- Разработка новых аппаратных архитектур требует разработки новых компиляторов для известных языков высокого уровня.

Упрощенная модель компилятора



Таблицы

- *Таблицы* хранят долговременную и/или глобальную информацию о программе.
- Одна из таблиц – *таблица имен (таблица идентификаторов, таблица символов)*, в которой накапливается информация об идентификаторах программы.

Лексический блок

- *Лексический блок* преобразует последовательность символов (*цепочку*) на исходном языке в последовательность *лексем*.
- *Лексема* – наименьшая цепочка исходного языка, имеющая заранее определенный смысл. Лексема состоит из двух частей:
 - *класс* – указывает на характер информации в лексеме
 - *значение* – собственно значение лексемы.



Пример: работа лексического блока

- Цепочка: `if B>=2 then A:=0`

Лексема	Класс	Значение
<code>if</code>	if	–
<code>B</code>	Переменные	Индекс B в таблице имен
<code>>=</code>	Операции отношения	Больше либо равно
<code>2</code>	Константы	2
<code>then</code>	then	–
<code>A</code>	Переменные	Индекс A в таблице имен
<code>:=</code>	Оператор присваивания	–
<code>0</code>	Константы	0

Синтаксический блок

- *Синтаксический блок* преобразует цепочку лексем в цепочку атомов.
- *Атом* – элементарная операция; последовательность атомов отражает порядок выполнения операций. Атом состоит из двух частей:
 - *класс* – вид операции
 - *значение* – набор указателей на операнды операции.



Пример: работа синтаксического блока

- Цепочка: $A+B*C$

Лексема	Класс	Значение
A	Переменные	Индекс A в таблице имен
+	Аддитивные операции	Сложить
B	Переменные	Индекс B в таблице имен
*	Мультипликативные операции	Умножить
C	Переменные	Индекс C в таблице имен

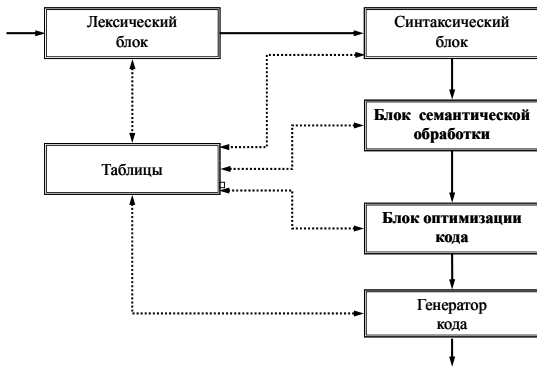
Атом	Класс	Значение
УМНОЖ(B,C,R1)	Умножить	Индексы B, C, R1 (системная переменная) в таблице имен
СЛОЖ(R1,A,R2)	Сложить	Индексы A, R1 и R2 (системные переменные) в таблице имен

Генератор кода

- *Генератор кода* преобразует цепочку атомов в цепочку команд машинного языка.
- Генератор кода является *машинно-зависимым* блоком компилятора.



Дополнительные блоки компилятора



Блок семантической обработки

- *Блок семантической обработки* выполняет часть работы компилятора, связанную со смыслом лексем или атомов.
- Пример:
Порождение команд с фиксированной или плавающей точкой в зависимости от операндов атома **УМНОЖ(В,С,Р1)**.

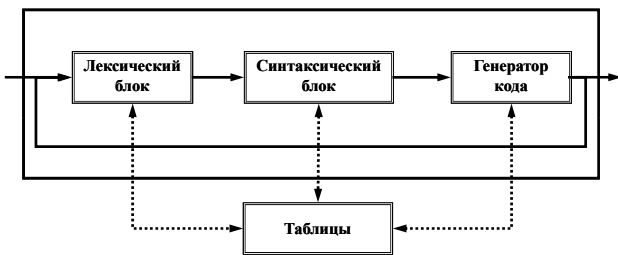
Блок оптимизации

- *Блок оптимизации* повышает эффективность машинного кода.
- Пример:
до оптимизации – атомы кода
for i:=1 to 100000 do
 К:=K+i*(A+B*C);
после оптимизации – атомы кода
 tmp:=A+B*C;
 for i:=1 to 100000 do
 К:=K+i*tmp;

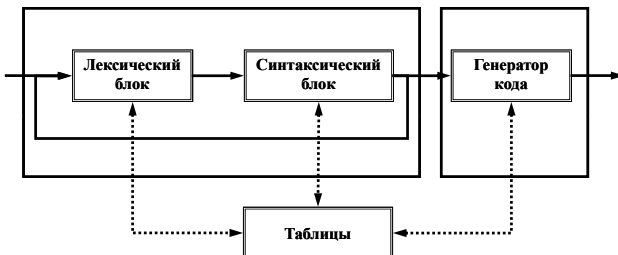
Проходы компилятора

- *Проход компилятора* – это работа блока (или блоков) по выдаче *всей* цепочки объектов (лексем, атомов или машинных команд) до передачи управления другому блоку.
- Классификация компиляторов по числу проходов:
 - 1-проходные
 - 2-проходные
 - 3-проходные

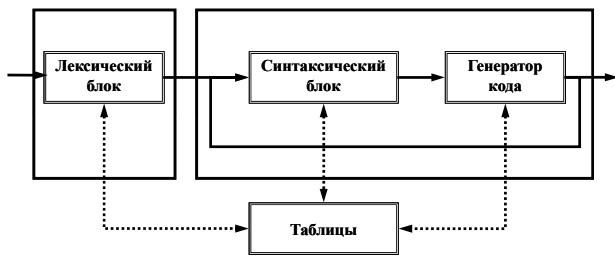
Однопроходный компилятор



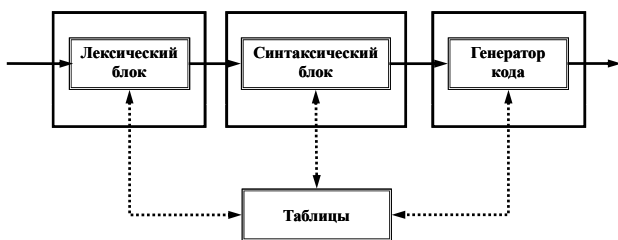
Двухпроходный компилятор (а)



Двухпроходный компилятор (b)



Трехпроходный компилятор



Мотивация разбиения на проходы

- *Логика языка* – компилятору может потребоваться информация из еще не просмотренной части программы.
- *Оптимизация кода* – машинный код можно сформировать более эффективным, если генератору кода доступна информация обо всей программе.

Заключение

- **Языковой процессор** обеспечивает выполнение на компьютере программы, написанной программистом.
- Типы языковых процессоров – **интерпретатор** и **транслятор**. Трансляторы разделяются на **ассемблеры** и **компиляторы**.
- Компиляция осуществляется тремя последовательно соединенными блоками: **лексический блок**, **синтаксический блок** и **генератор кода**, которые имеют доступ к общему набору **таблиц**.
- В зависимости от способа передачи управления от одного блока к другому компиляторы делятся на **однопроходные**, **двухпроходные** и **трехпроходные**.
