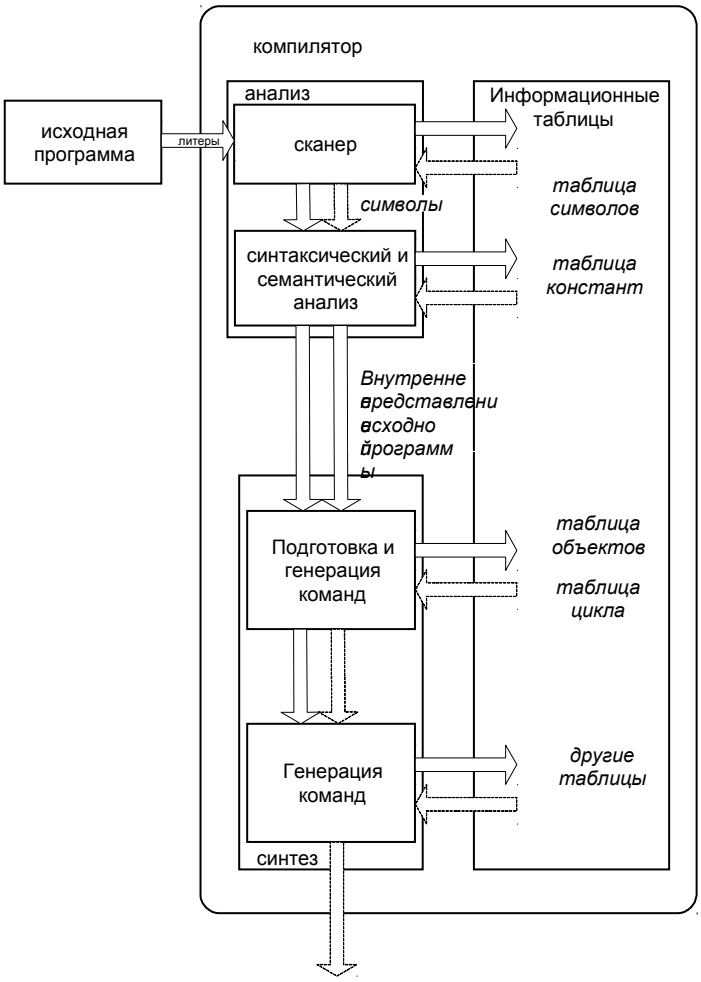
55. Принципы организации и построения компиляторов.

- *Транслятор* – программа, которая переводит исходный код в целевой равносильный.
  - Адресный — преобразование вирт. Адресов в реаьные
  - Одно\Многопроходный
  - Обратный — целевой в исходный \ дизассемблер
  - Оптимизирующий\Не~
- *Компилятор* — транслятор, исх. код - высокоуровневый язык, целевой — машинный или ассемблер.
  - Гибкий — написан на языке высок. уровня. Составлен из модулей или на основе компилятора компиляторов
  - Компилятор компиляторов — транслятор, создающий компилатор по формальному описанию языка
  - Отладочный — устраняет некоторые синт. Ошибки.
  - Самокомпилируемый — напиан на том же языке, что и компилирует
- ВИДЫ КОМПИЛЯЦИИ
  - Пакетная — вместе с модулями
  - Интерпретирующая - построяная
  - Условная - (ifdef)
- *Ассемблер* – программа, которая переводит исходную программу, написанную в автокоде
  (ассемблере), на язык вычислительной машины.
- *Интерпретатор* – принимает программу на исходном языке, как исходную информацию и выполняет её (предварительно анализирует и транслирует в некоторое внутреннее представление, далее программа выполняется).

  Таблица символов — соответствия имени пер-ой с ее атрибутами (тип, область видимости)
  Таблица функций — имя, тип, аргументы и методы их передачи



Процесс компиляции разделяется на ***этапы***, которые, в свою очередь, разделяются на ***фазы***.

Этап анализа состоит из фаз:

1. Лексический анализ — читает входной поток символов и группирует в лексемы. Для лексем строятся токены (имя, адрес в памяти\таблице)
2. Синтаксический анализ — токены > дерево;
3. Семантический анализ — проверка дерева на семантич. Согласованность с языком. Проверка типов;
4. Генерация промежуточного кода.

Этап синтеза состоит из следующих фаз:

Машинно-независимая оптимизация.

- Генерация целевого кода (напр ASM)
- Машинно зависимая оптимизация.

Оптимизация

процесс получения более эффективного кода на основе исходного

корректность

повышение производительности

время оптимизации ограничено

Проблемный код представляется графом.

Узлы — наборы команд. Дуги — потоки данных

Трехадресный код - команда, содержащая только один оператор.

Базовый блок — максимальная последовательность трехадресных кодов.

Оптимизация делится на локальную и глобальную.

Локальная

на уровне базовых блоков

- Устранение неиспользуемого кода
- Применение алгебраических тождеств (2x = x + x, x/2=x*0.5)
- Представление обращений к массивам (создание узла с оператором [] и двумя дочерними узлами a[0] и индекс i)

Глобальная

учитывает потоки между базовыми блоками

общие глобальные выражения (

```
t6 = 4*1;          t6 = 4*1;
x = a[t6];         x = a[t6];
t2 = 4*1;          t9 = a[t6];
t9 = a[t2];        t10 = a[t6];
t8 = 4*1;
t10 = a[t8];
)
```

распространение копий (если встретилось u := v; то дальше вместо u везде v)

перемещение кода (вынесение кода за цикл)

Инструментальные средства

При создании компиляторов используются :

- 1. Генераторы лексических анализаторов — создает лексический анализатор на основе токенов языка с использованием рег. выражений.
- 2. Генераторы синтаксических анализаторов — автомат. Создает синт. Анализатор на основе грамматических правил языка.
- 3. Генератор генераторов кода — генераторы кода на основе набора правил трансляции.