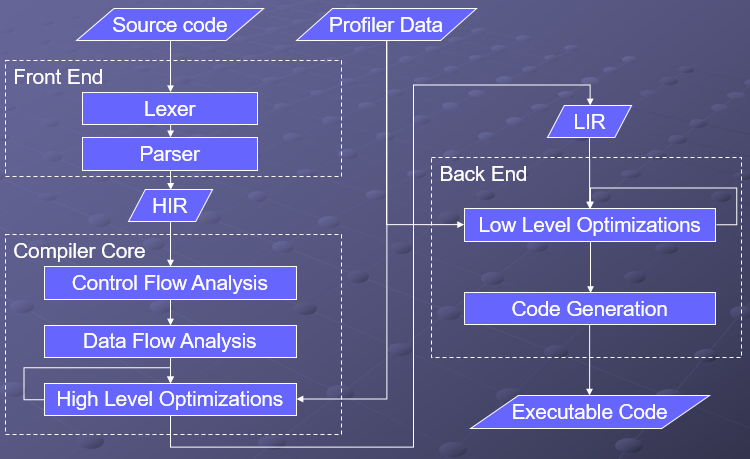
***27. Структура оптимизирующего компилятора.***



Лексер — делает аналитический разбор входной последовательности символов с целью получения на выходе последовательности символов, называемых токенами.

Пример с википедии:

net\_worth\_future = (assets - liabilities);

Легким движением руки превращается

ИМЯ "net\_worth\_future"

ПРИСВАИВАНИЕ

ОТКРЫВАЮЩАЯ\_СКОБКА

ИМЯ "assets"

МИНУС

ИМЯ "liabilities"

ЗАКРЫВАЮЩАЯ\_СКОБКА

ТОЧКА\_С\_ЗАПЯТОЙ

Парсер — получает на вход данные и строит структуру данных, часто что-то вроде синтаксического дерева или другую иерархическую структуру — давая структурное представление ввода, проверяя корректность синтаксиса в процессе.

* Front End (Транслятор) - на выходе промежуточное представление, одно для нескольких языков
  + Зависит от языка программирования высокого уровня
  + Возможно множество трансляторов для различных языков
* Compiler Core (Ядро, оптимизатор) - анализируются граф управления, граф потока данных. Если метод горячий, то он оптимизируется
  + Универсален для широкого класса языков и аппаратных архитектур
  + Наиболее сложен для реализации
  + Составляет основную ценность в структуре компилятора
  + Требует высокой гибкости взаимодействия с трансляторами и генераторами кода
* Back End (Генератор Кода)
  + Зависит от аппаратной платформы
  + Возможно множество генераторов кода для различных платформ
  + Требует тонкой подстройки под конкретную платформу
* [оптимизирующий компилятор](https://ru.wikipedia.org/wiki/Оптимизирующий_компилятор) – википедия
* [читать до 24](http://sp.cmc.msu.ru/proseminar/2011/belevantsev.2011.04.01.pdf)

Преобразование графа управления в граф потока данных излишне

<http://docs.oracle.com/cd/E15289_01/doc.40/e15058/underst_jit.htm>