***26. Оптимизирующий JIT Компилятор. Особенности MRE***

Шаблонный JIT-компилятор каждую инструкцию по шаблону развертывает в кусок кода, который не оптимизируется.

Оптимизирующий компилятор — компилятор, в котором используются различные методы получения более оптимального программного кода при сохранении его функциональных возможностей.

Виды:

* Peephole-оптимизации рассматривают несколько соседних (в терминах одного из графов представления программы) инструкций, чтобы увидеть, можно ли с ними произвести какую-либо трансформацию с точки зрения цели оптимизации. В частности, они могут быть заменены одной инструкцией или более короткой последовательностью инструкций.
* Локальная оптимизация- рассматривается только информация одного базового блока за один шаг. Так как в базовых блоках нет переходов потока управления, эти оптимизации требуют незначительного анализа (экономя время и снижая требования к памяти), но это также означает, что не сохраняется информация для следующего шага.
* Внутрипроцедурные оптимизации — глобальные оптимизации, выполняемые целиком в рамках единицы трансляции (например, функции или процедуры) При наличии глобальных переменных может быть затруднена.
* Оптимизации циклов
* Межпроцедурные оптимизации

По сравнению с интерпретатором и шаблонным JIT компилятором

* Достоинства
  + Ускорение выполнения кода в 4 и более раз
  + Производительность кода сопоставимая с компилируемыми языками (C, C++, Fortran, Ada)
* Недостатки
  + Затраты ресурсов на компиляцию
  + Увеличение времени старта
  + Затруднение отладки пользовательского кода
  + Высокая вероятность ошибки и нарушения семантики пользовательского кода

Особенности систем управляемого исполнения:

Объект — уникальная сущность, обладающая состоянием и поведением и сохраняющая правильное(наблюдаемое) состояние в течение всей жизни.

[основные принципы ООП](http://vschol.ru/TurboPascal/gl10/gl10_1.html)

* Инкапсуляция — реализация требований(правильное наблюдаемое состояние в течение всей жизни) определения объекта
  + Вызов методов для доступа к данным
* Наследование
  + Цепочки вызовов
* Полиморфизм — возможность по одному символу имени в зависимости от контекста вызывать различную реализацию
  + Косвенные вызовы
* Неопределенное время жизни объектов
  + Выделение всех объектов в куче
* Промежуточный код (Byte Code)
  + Сокрытие семантики исходного кода
* Ленивая загрузка классов
  + Неполный граф вызовов во время компиляции
* JIT Компилятор является частью среды
  + Возможность перекомпиляции при загрузке новых классов и изменении характера нагрузки
  + Не требуется спекулятивная оптимизация
* Компиляция происходит параллельно с выполнением
  + Требуется учитывать время компиляции
  + За раз компилируется небольшой блок кода

Температура кода:

* Мертвый код
  + Не исполняется никогда
* Холодный код
  + Исполняется редко, не оказывает влияния на скорость работы системы
* Теплый код
  + Исполняется регулярно, влияет на скорость работы системы
* Горячий код
  + Оказывает доминирующее влияние на скорость работы системы