**2. Основные компоненты MRE и их назначение**

Загрузчик IR

Верификатор

Планировщик

Менеджер памяти

GC

Ядро

Исполнитель

Интерпретатор

JIT компилятор

Профилятор

загрузка/cохранение

Данных и кода

Модель данных

Внешний интерфейс

Инициализация

Инкапсуляция — принцип модульности, система разделяется на модули, «внутренности» которых скрыты от остальных.

Среды управляемого исполнения — эксзоскелет вокруг пользовательского кода.

Цена модульности — накладные расходы при переходе от одного модуля к другому.

Модели данных: модели потоков, нитей, рефлексий, итд

Профилятор — постоянно профилирует(oh, really?). С помощью профилятора достигается локальный оптимальный баланс между скоростью работы и временем компиляции.

Менеджер памяти — выделяет память, обнаруживает живые объекты, собирает мусор(освобождает память).

Для каждого семейства сборщиков мусора — своя стратегия выделения памяти.

Планировщик — следит за компонентами системы и синхронизацией(основной вид синхронизации — исполнение большого числа программ на малом числе процессоров)

* Создание и разрушение нитей
* Распределение нагрузки между процессорами
* Приостановка и возобновление нитей
  + Для синхронизации управляемого кода
  + Для сборки мусора
  + Для синхронизации неуправляемого кода и сборки мусора

Интерпретатор — медленный, работает через внешний интерфейс

JIT — компилятор

Есть возможность сохранить промежуточное представление(Intermediate Representation)(часть работы JIT — компилятора) это ускорит последующие компиляции, однако создает уязвимости.

Один из методов борьбы с такими уязвимостями Trusted computing — все промежуточные данные подписываются цифровой подписью

Верификация

* Проверка формата класса
* Проверка данных класса
  + final классы и методы не перекрыты
  + Определен суперкласс
  + Формат пула констант
* Проверка кода класса
  + Размер стека, тип данных на стеке
  + Правильный тип локальной переменной при доступе
  + Число аргументов при вызове метода
  + Необходимое количество и тип аргументов инструкций
* Проверка ссылок
  + Наличие классов, методов и полей
  + Права доступа

Опр: Указатель — число, адрес ячейки памяти.

Опр: Ссылка — уникальный идентификатор какой- либо сущности.

Внешний интерфейс — позволяет написать методы, для тех случаев, когда невозможно целиком написать приложение на языке Java.

Например, стандартная библиотека класса Java не поддерживает платфоременные особенности или программную библиотеку.

Данный фреймворк позволяет нативному методу использовать Java объекты также, как Java код. Нативный метод может создавать Java объекты, а затем просматривать и использовать для выполнения своих задач. Нативный метод также может просматривать и использовать объекты, созданные кодом приложения Java.

* Вызов внешних неуправляемых функций
* Обратный вызов управляемого кода (callback) — передача внешней библиотеке указателя на функцию, которая будет обрабатывать события.
  + Два сопособа итераций:
    - внешняя итерация

for(i=0; i<=n; i++)  
a.next[i]

* + - внутренняя  
      a.interate (это и есть callback)
* Передача данных между управляемым и неуправляемым кодом (marshaling)
  + Копирование — эффективно для малых данных и неэффективно для больших
  + Обертывание и обратный вызов — решает безопасным способом абсолютно все задачи, но есть проблемы с эффективностью
  + Закрепление объектов (pinning) — запрет объектам перемещаться из памяти
  + Указатель на неуправляемую память

Типы команд

* Простые
  + Стековая арифметика
  + Доступ к локальным данным
  + Ветвления
* Объектные
  + Создание объектов
  + Доступ к полям
  + Вызов методов
  + Исключения