**1. Виртуальная машина и MRE. Достоинства и недостатки MRE по сравнению с естественным кодом платформы**

Изначальное определение «виртуальная машина — изолированный дубликат или копия реальной машины» Попек и Голдберг

*Опр*: ***Виртуальная машина(VM)*** — общее программная реализация определенной машины (например, отдельный компьютер или ), исполняющее программы как данная машина.

Программы могут быть большими как ОС, или маленькими, как процесс.

Код программы может быть интерпретирован, скомпилирован (JIT — выполнение компиляции по запросу, AOT - компиляция выполнена заранее) или просто запущен под контролем VM

*Примеры*: Xen, VMWare, JVM, LLVM, Parrot, ScrummVM, DosBox, Sega Emulator

Изначально JVM задумывалась аппаратной

*Опр*: ***Runtime Environment***(Окружение времени исполнения): Библиотека, реализующая поведение программы конкретного языка

* Вход/выход из программы
* Доступ к сервисам ОС — ввод, вывод итд
* Проверка типов во время исполнения
* Обработка исключений
* Генерация и исполнение кода

Главная функциональность, которую предоставляет ОС — многозадачность (запуск и отслеживание работы программы)

Если программа одна и без ошибок — ОС не нужна (Системы без ОС называют baremetal, «на голом железе»).

Программы на C и ASM могут запускаться без runtime, но это непрактично.

*Опр*: ***Managed Runtime Environment(MRE)*** — среда исполнения, обеспечивающая восстановление от ошибок в ходе исполнения кода

* Ошибочный код не может свободно исполняться
* Исключения должны быть брошены немедленно в контексте утверждения Язык высокого уровня
* Безопасность типов, Безопасность указателей, Безопасность нитей, Сборка мусора, Унификация интерфейсов, Основные управляемые библиотеки, Изоляция «железа», Верификация кода, Кросс-платформенность кода

Альернативой MRE является *Бинарная компиляция* — трансляция кода в бинарные инструкции для целевой архитектуры:

* Статическая компиляция — не надо запускать компилятор, но больший размер бинарника
* Бинарники для конкретных платформ:
  + Отимизация под конкретную платформу
  + Нет кросс-платформенных бинарников
* Уязвимости в безопасности:
  + Код семантически не проверяется на правильность
  + Уязвимость по отношению к исполняемым данным

|  |  |
| --- | --- |
| Плюсы MRE | Минусы MRE |
| * Код меньше * Кросс-платформенные бинарники * Профилирование (сбор профиля по статистике программы) * Код семантически проверяется на правильность при загрузке * Безопаснее * Лучше дебажится * Потенциальная замена ОС | * Большая среда исполнения (300 Кб — минимум для Java) * Больше время исполнения * Более медленный запуск при JIT — компиляции (примерно в 2 раза, AOT(компиляция перед исполнением) — примерно одинаково) * Ограничена оптимизация по платформам * Ограничено использование системного программирования |

*Примеры* MRE/VM

* Специализированные

RIP for PostScript, SQL Server

* Общие

PHP, Python, Ruby, CLISP, TCL, Lua, Java RE, CLI(.NET), Google V8 (Java Script), Mozilla Spider Monkey (Java Script), Android Dalvik (Java)