



Технологии аппаратной виртуализации

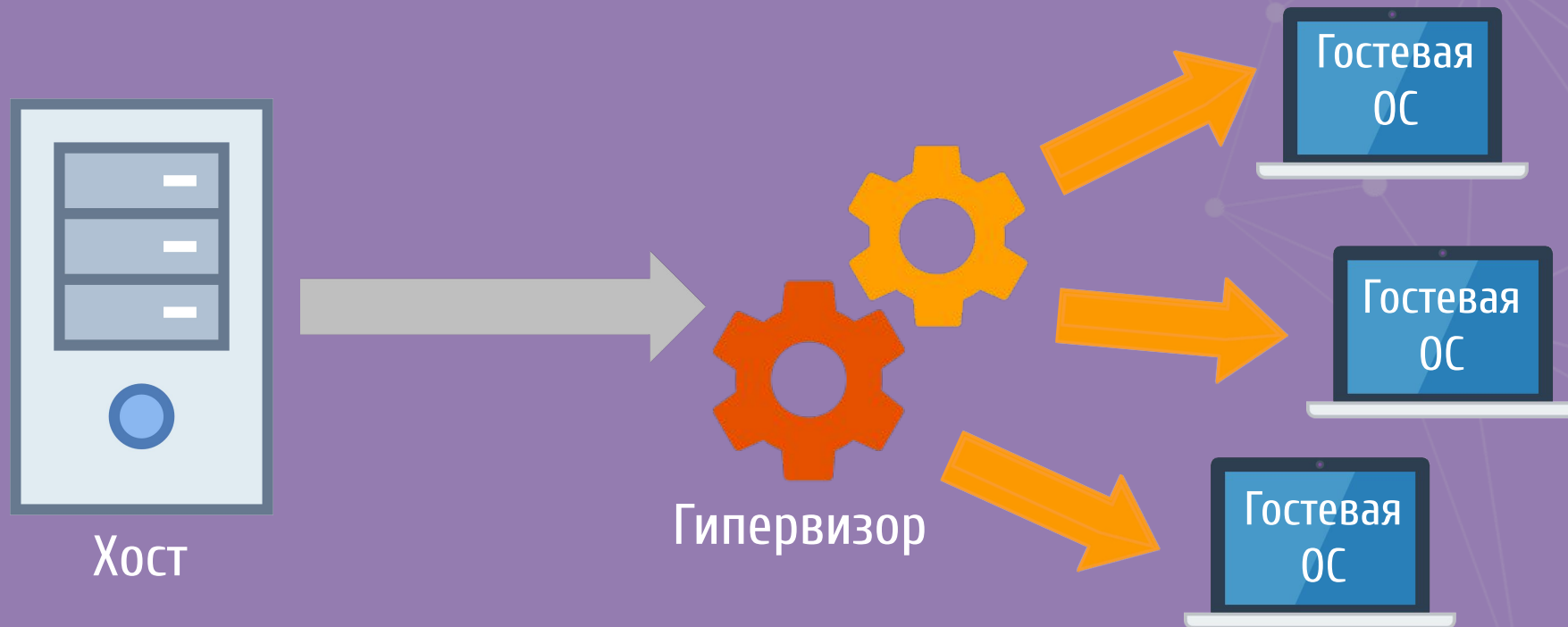
Доклад подготовил:

студент группы ЭВМ6-14-1

Русецкий Никита

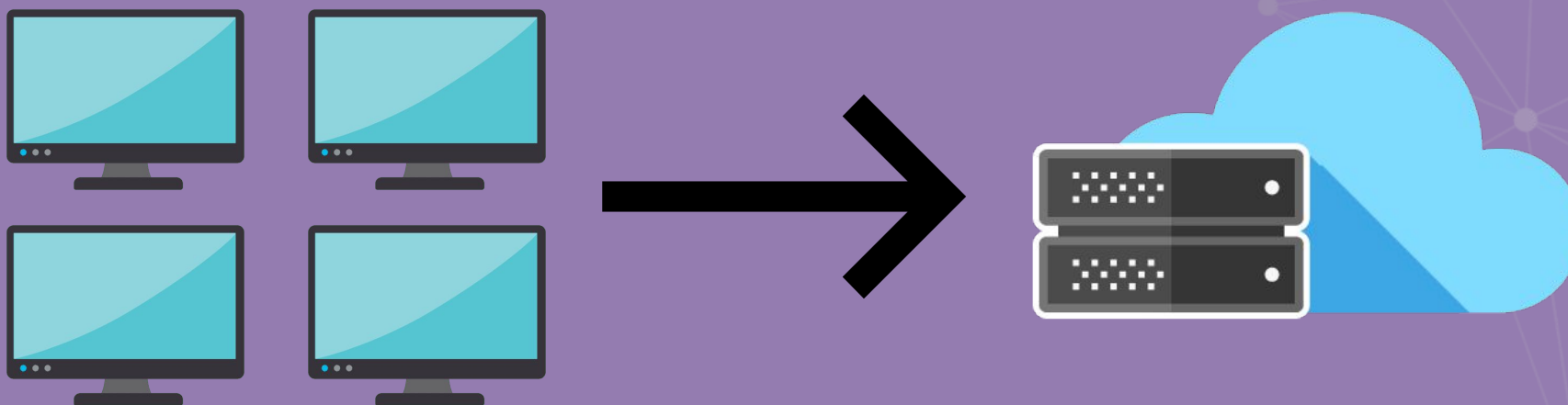
Что это?

- Виртуализация – технология, позволяющая запускать на одном физическом компьютере, называемом «хостом», несколько виртуальных операционных систем, называемых «гостевыми ОС»



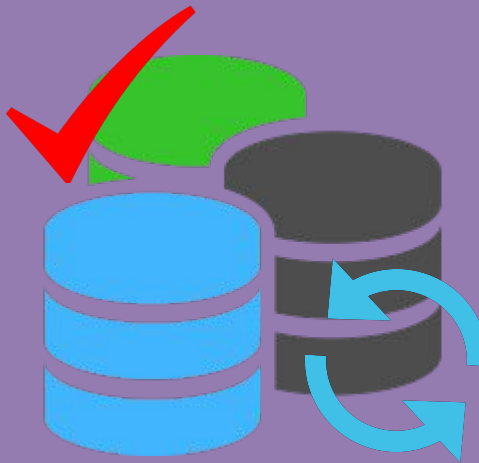
Зачем?

- Для обеспечения независимости гостевых ОС от аппаратной платформы;
- Для сосредоточения нескольких виртуальных машин на одной физической.



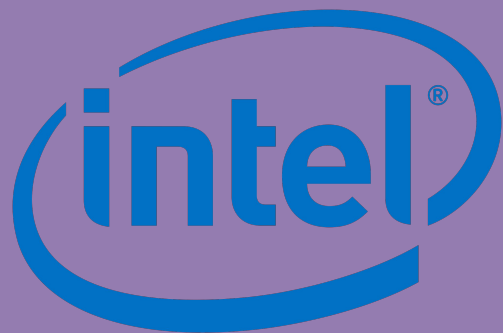
Какие преимущества?

- ✓ Обеспечивается существенная экономия на аппаратном обеспечении и обслуживании;
- ✓ Упрощается процедура резервного копирования и восстановления после сбоев.

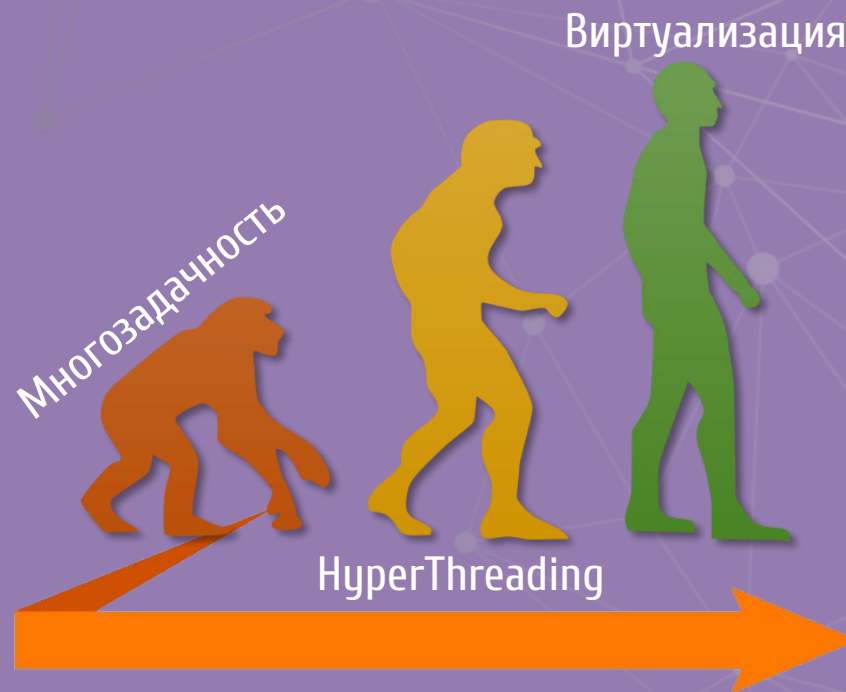


Немного истории

- 1985 год – *аппаратная* виртуализации впервые была воплощена в 386-процессорах (V86 mode)
- 1998 год – компания VMware запатентовала *программные* техники виртуализации
- Вслед за Intel AMD выпускает процессоры с поддержкой аппаратной виртуализации

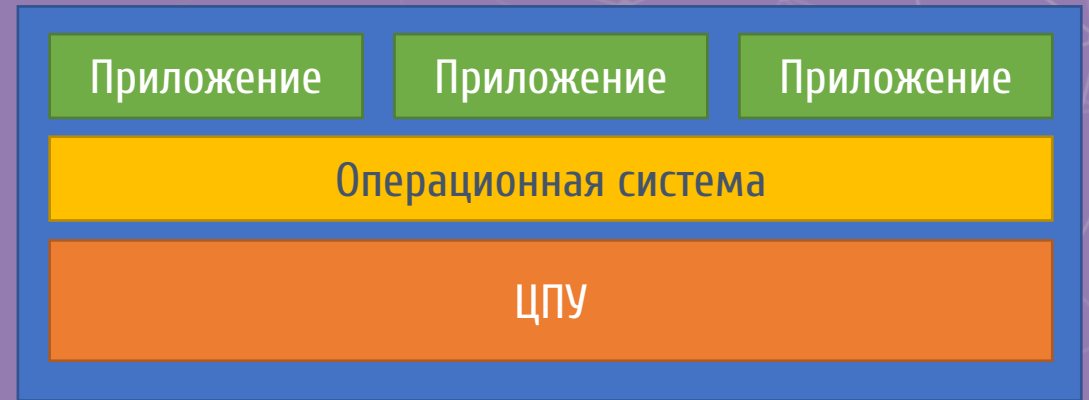


Эволюция уровней абстрагирования программных платформ



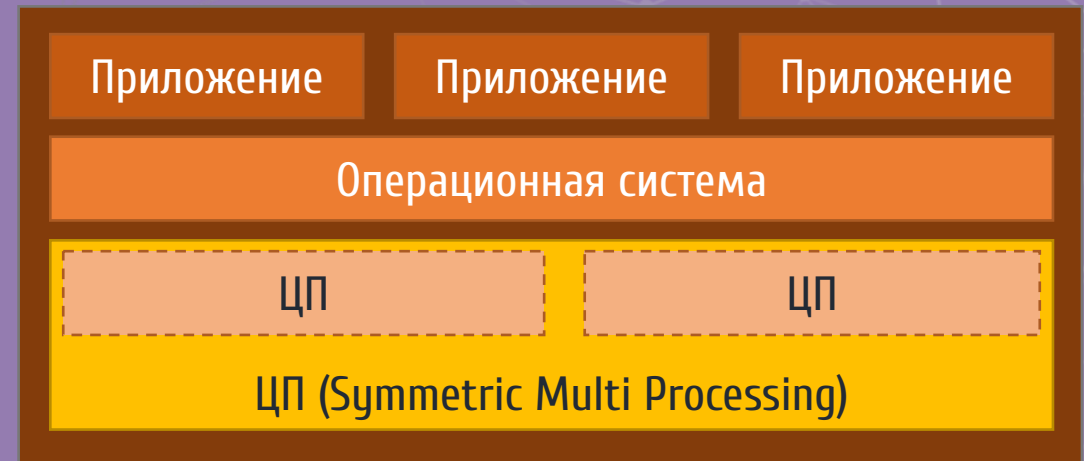
Многозадачность

- Многозадачность – первый уровень абстракции приложений. Каждое приложение разделяет ресурсы физического процессора в режиме деления исполнения кода по времени



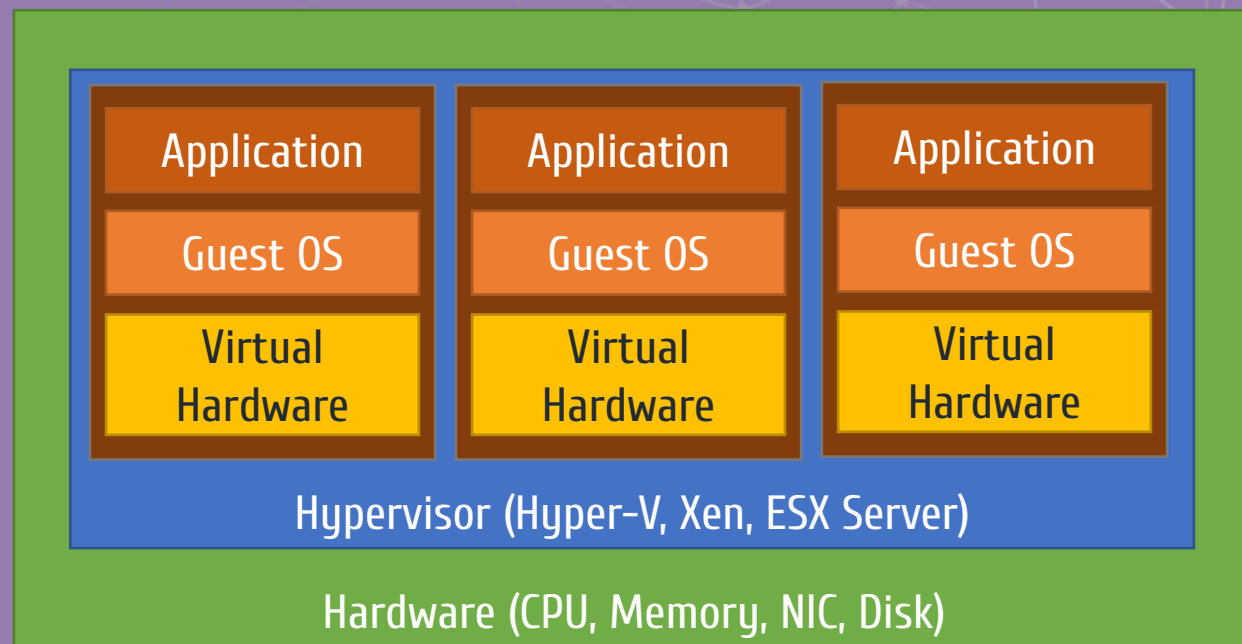
HyperThreading

- Технология HyperThreading в широком смысле представляет собой аппаратную технологию виртуализации.
- В рамках одного физического процессора происходит симуляция двух виртуальных процессоров с помощью техники Symmetric Multi Processing (SMP).



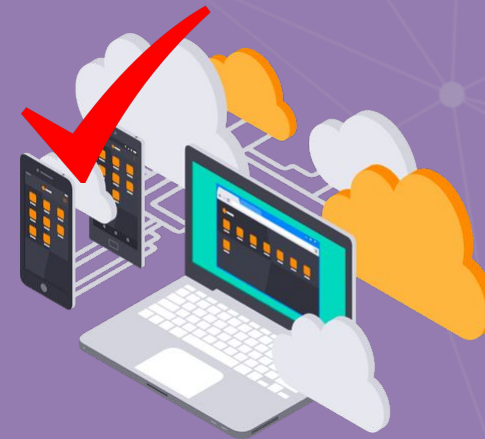
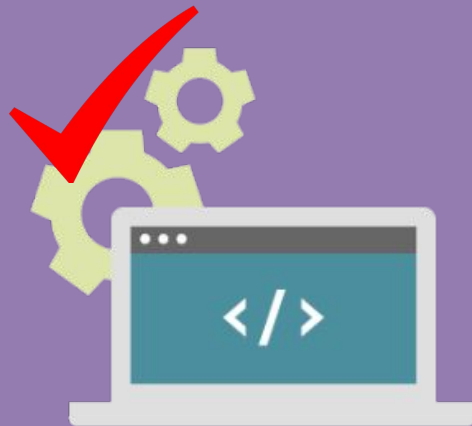
Виртуализация

- Виртуализация представляет собой эмуляцию нескольких виртуальных процессоров для каждой из гостевых ОС.
- Технология SMP позволяет представлять несколько виртуальных процессоров в гостевой ОС при наличии технологии HyperThreading или нескольких ядер в физическом процессоре.



Преимущества аппаратной виртуализации

- + Упрощение разработки платформ виртуализации
- + Возможность увеличения быстродействия платформ виртуализации
- + Возможность независимого запуска нескольких виртуальных платформ
- + Отвязка гостевой системы и архитектуры хостовой платформы и реализации платформы виртуализации



Аппаратная виртуализация

Принципы работы



Архитектура VM

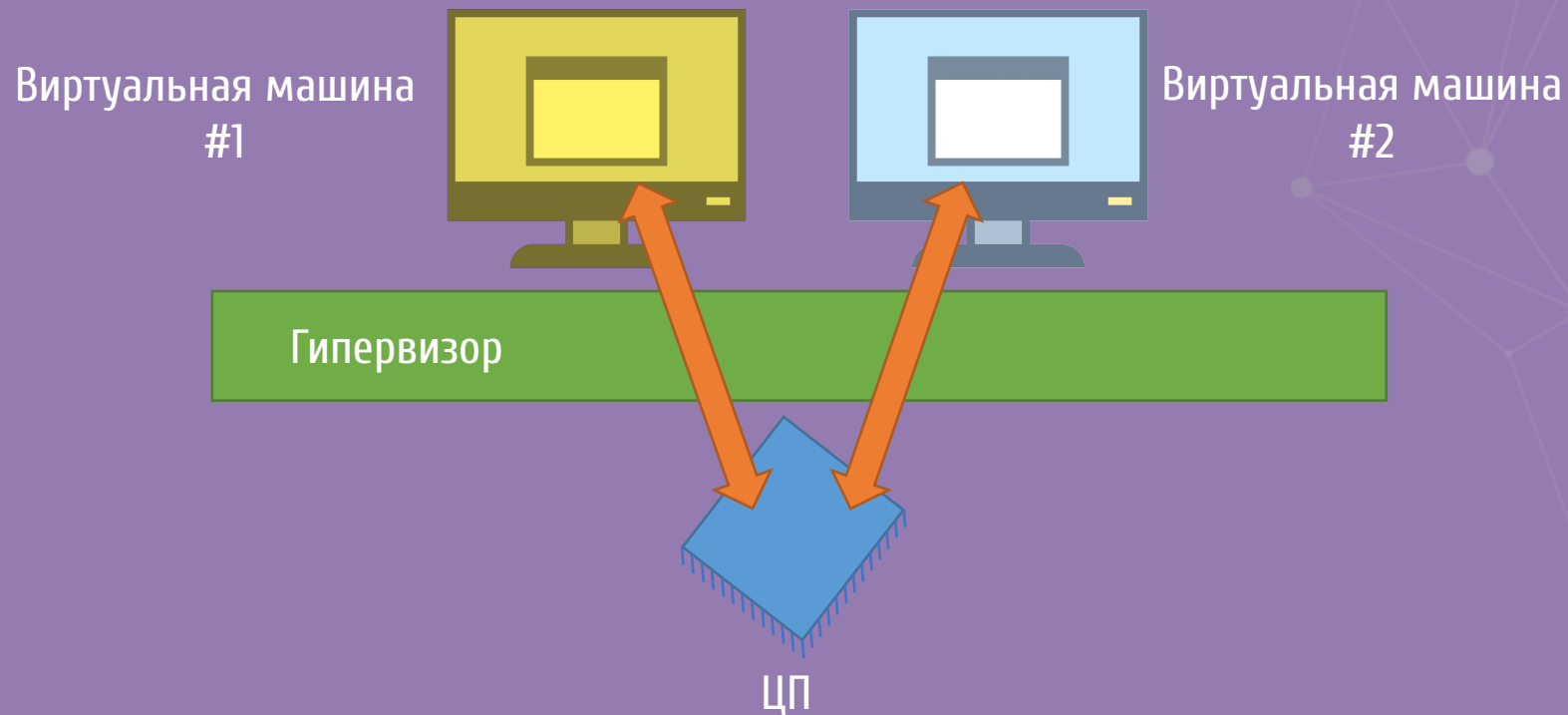
- Монитор виртуальных машин (*Virtual Machine Monitor*) или Гипервизор (*Hypervisor*) обеспечивает или позволяет одновременное, параллельное выполнение нескольких операционных систем на одном и том же хост-компьютере.
- Гипервизор также обеспечивает изоляцию операционных систем друг от друга, защиту и безопасность, разделение ресурсов между различными запущенными ОС и управление ресурсами.

Требования к гипервизору

- Гипервизор должен быть способен к:
 - Самозащите от ПО гостевой машины
 - Изоляции одной гостевой ОС от другой
 - Предоставлению интерфейса гостевому ПО
- Чтобы достичь этого, гипервизор должен иметь доступ к:
 - ЦП, памяти и устройствам ввода/вывода
- Способы разделения ресурсов между виртуальными машинами:
 - Временное мультиплексирование
 - Разделение ресурсов
 - Посреднические аппаратные интерфейсы

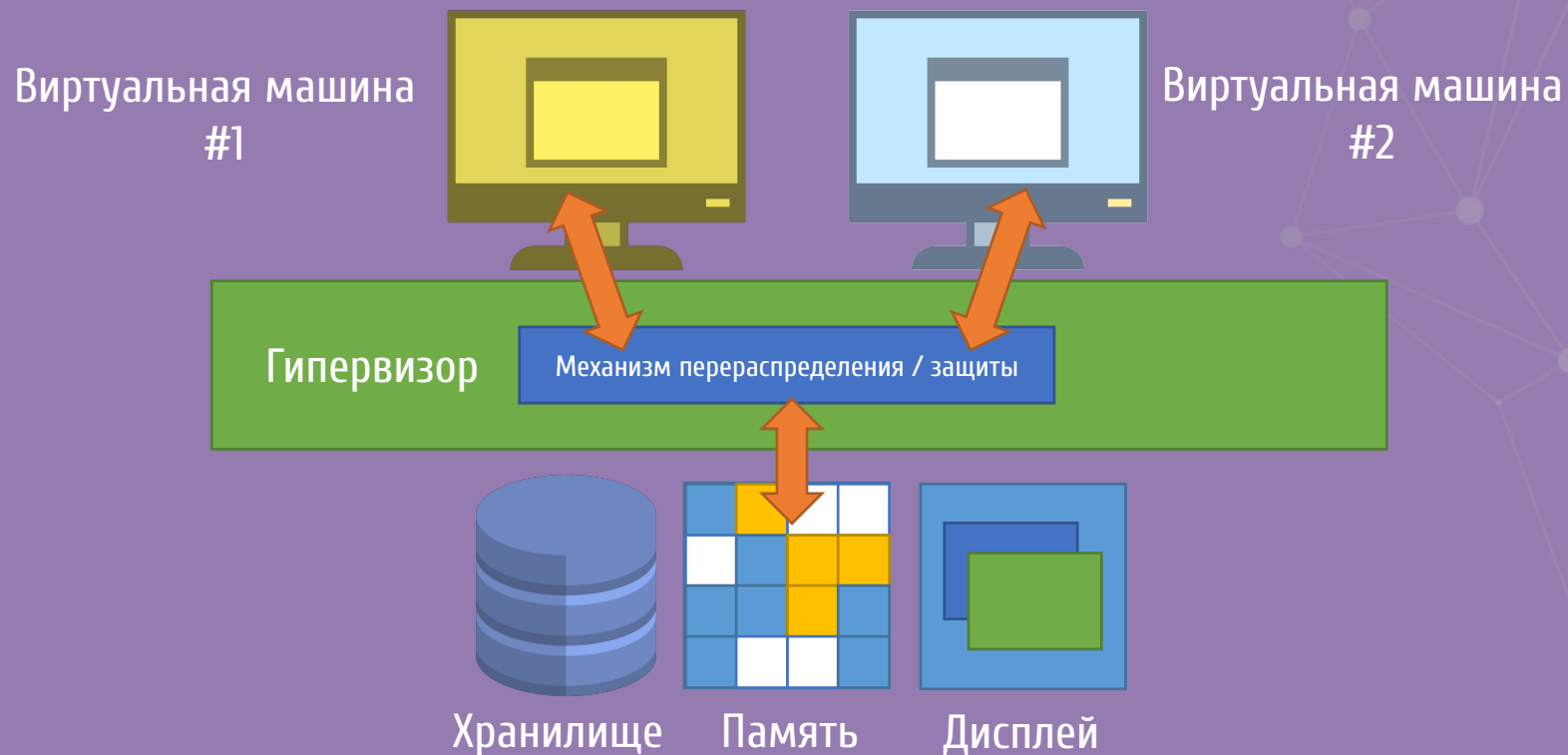
Временное мультиплексирование

- Виртуальной машине разрешен доступ к ресурсам на определенный период времени перед тем, как произойдет переключение к другой виртуальной машине



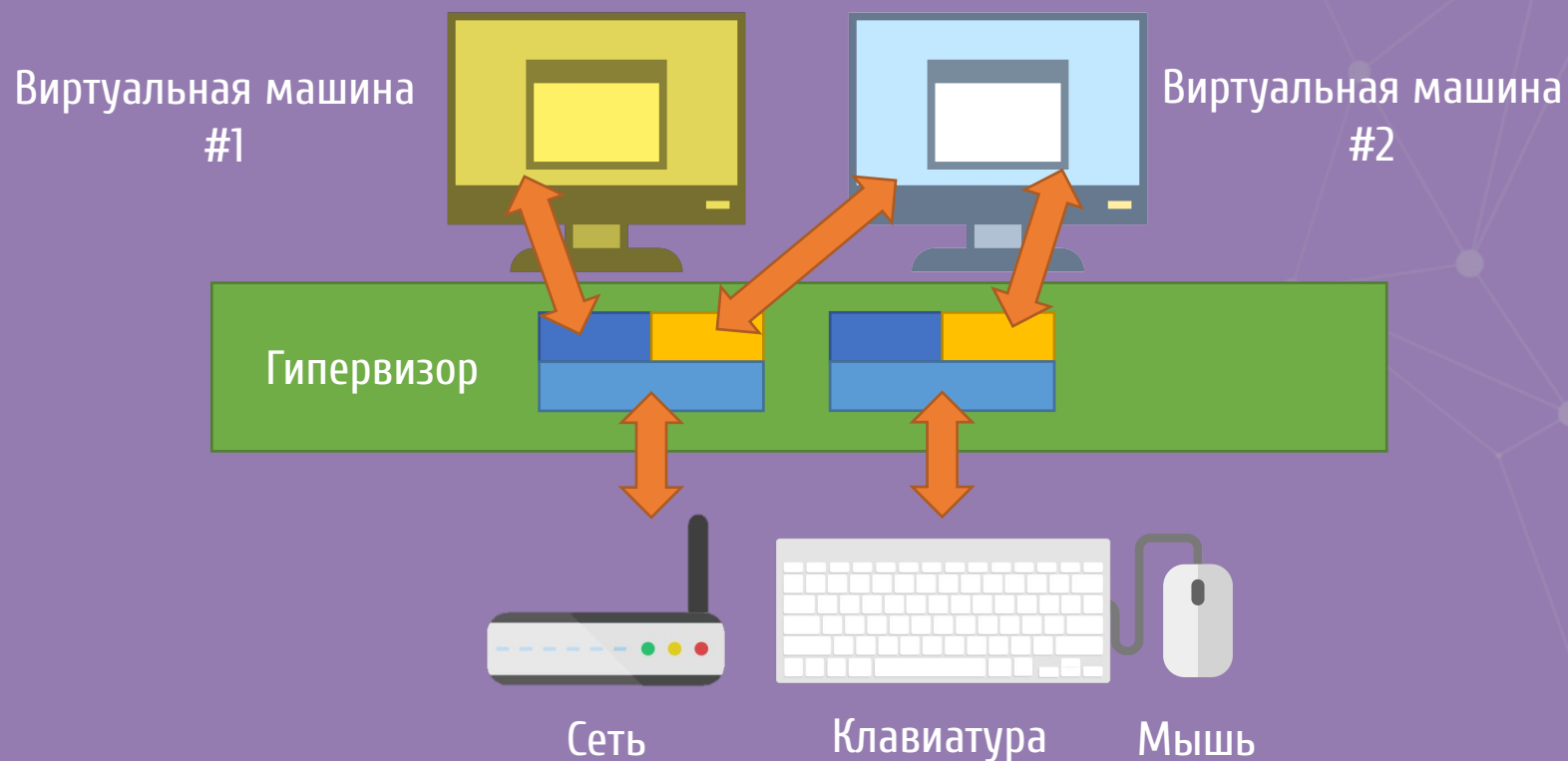
Разделение ресурсов

- Гипервизор распределяет «объем владения» физическими ресурсами между виртуальными машинами



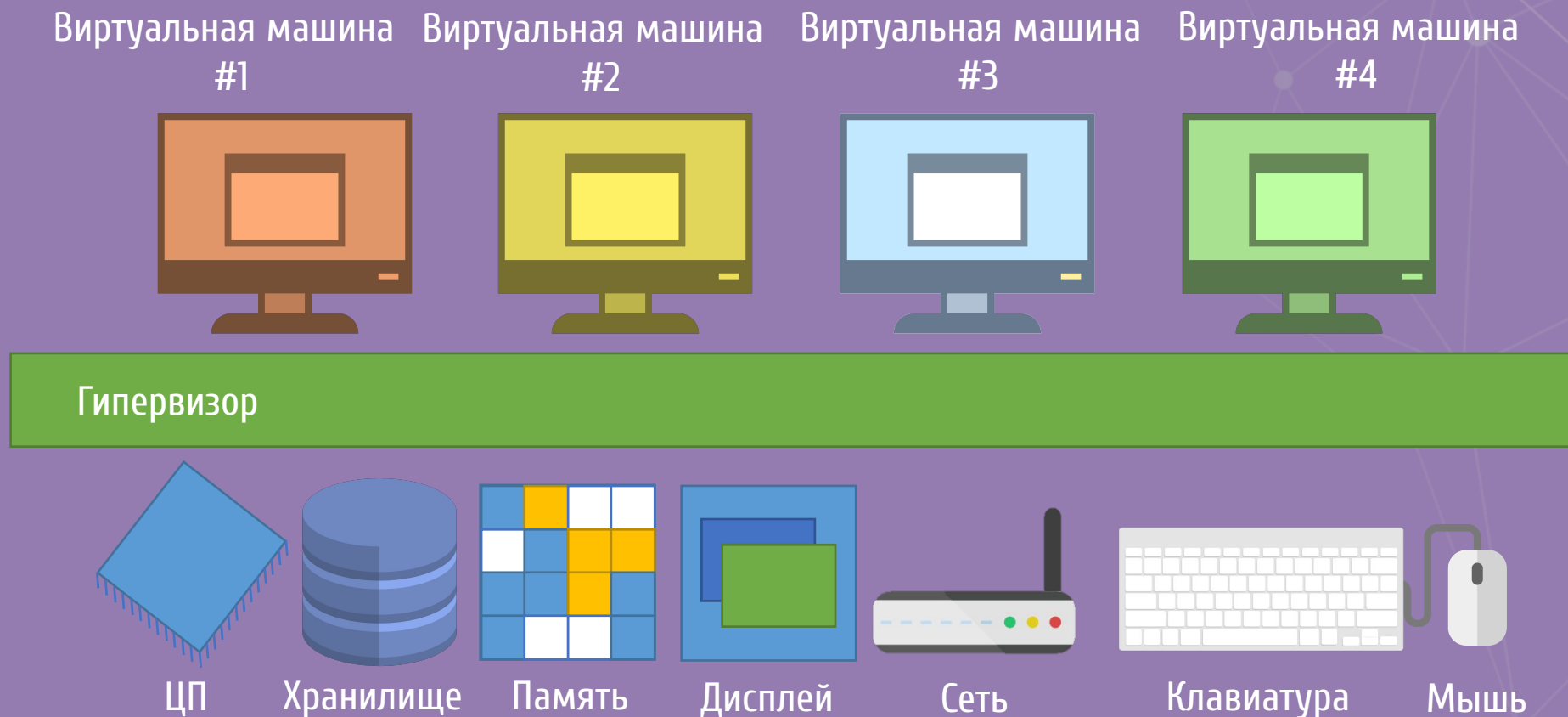
Посреднический доступ к физ. ресурсам

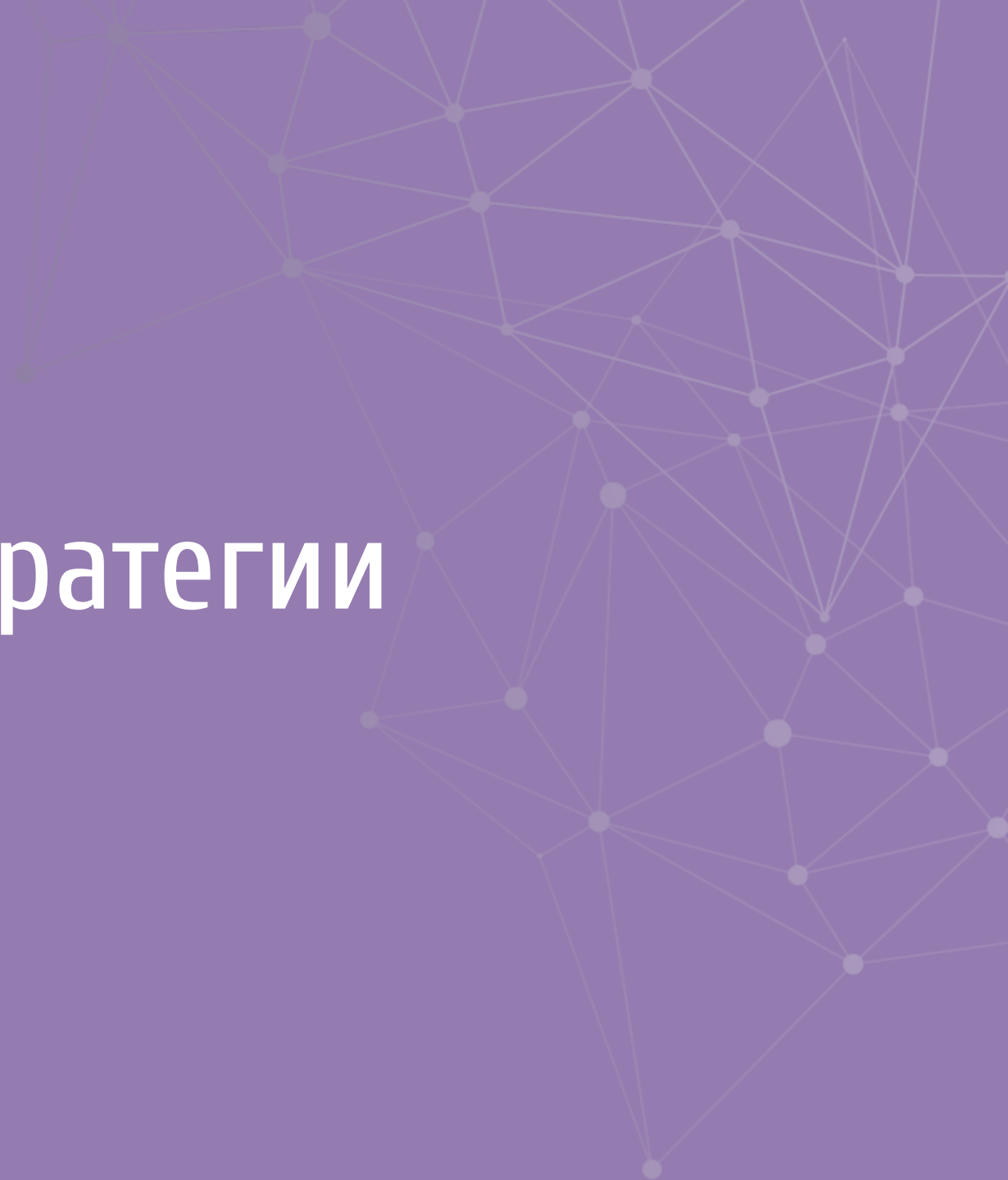
- Гипервизор сохраняет владение физическими ресурсами



Все вместе

- Гипервизор применяет все три метода для создания иллюзии, что гостевая ОС запускается в естественной среде





Виртуализация: стратегии реализации

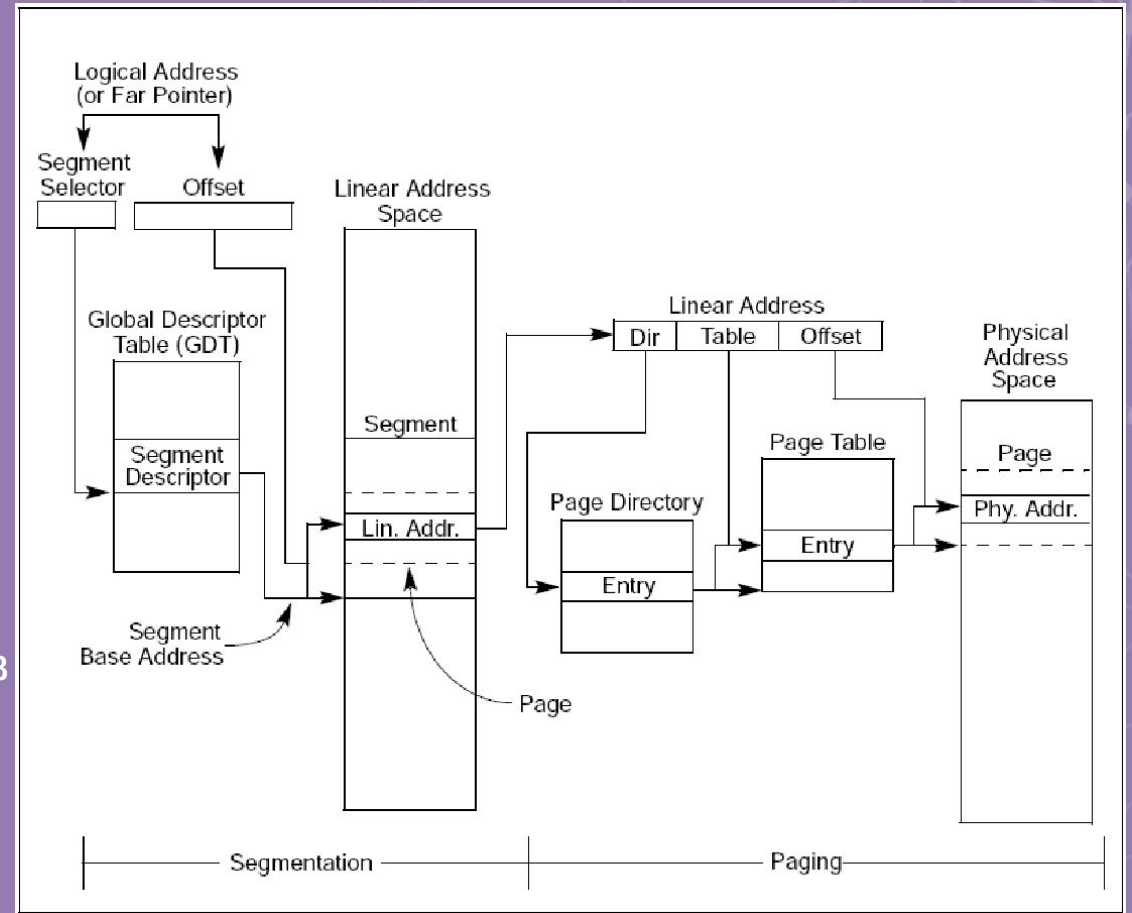
IA-32



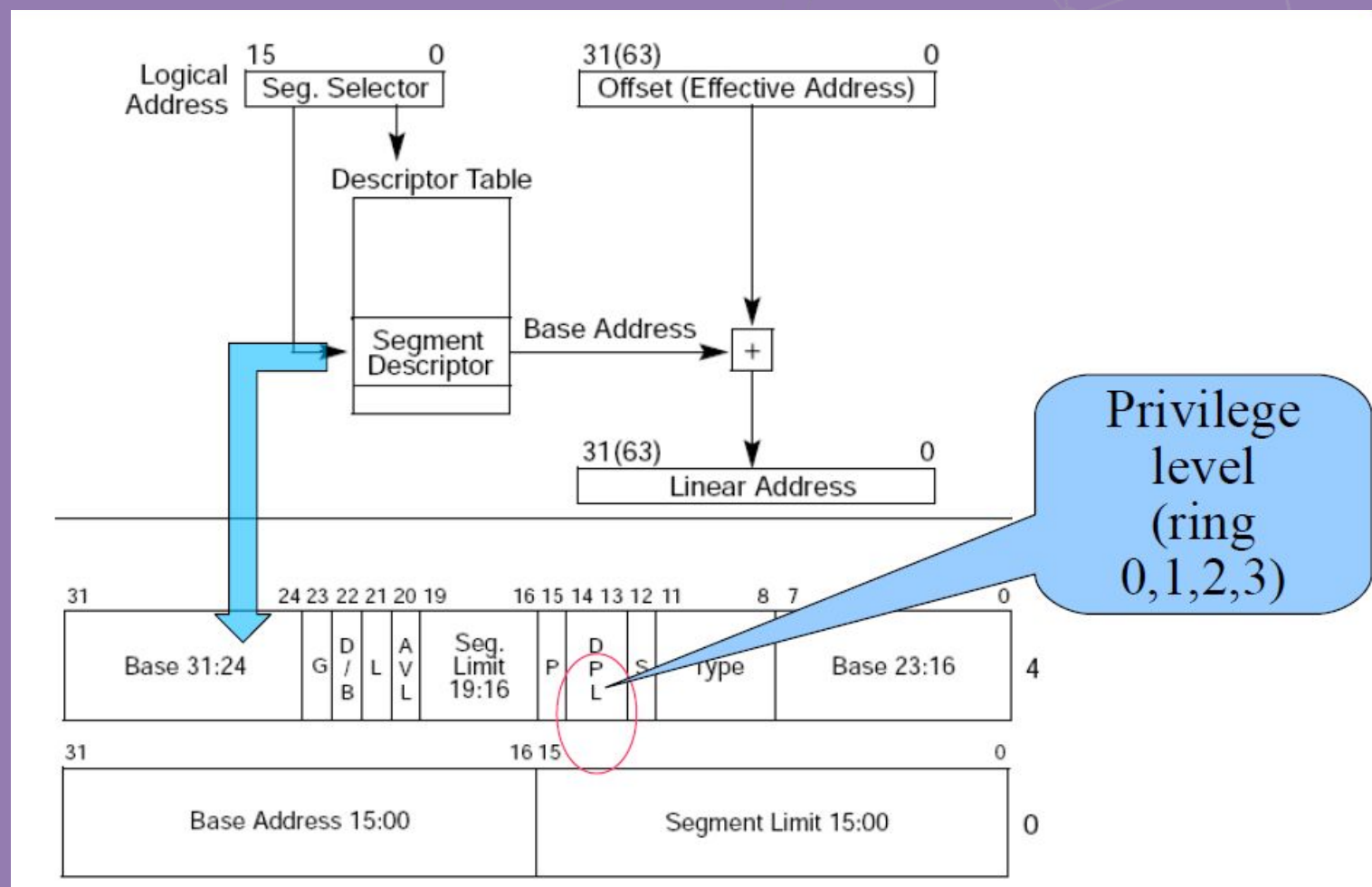
- IA-32 обеспечивает 4 уровня привилегий (кольца защиты)
- Защита, основанная на сегментах
 - Разделенная между 4 кольцами
- Защита, основанная на страницах
 - Разделяет только режимы пользователя и супервизора
 - Режим пользователя: код выполняется в 3 кольце
 - Режим супервизора: код выполняется в 0-2 кольцах

Управление памятью в IA-32

- Микропроцессор в защищенном режиме преобразует логические адреса в физические в два этапа: сначала блок управления сегментами выполняет трансляцию сегмента, преобразуя логический адрес, состоящий из селектора сегмента и относительного адреса (смещения), в 32-х разрядный линейный адрес, а затем блок страничной трансляции выполняет разбиение на страницы, преобразуя линейный адрес в физический. Микропроцессор не предусматривает механизмов запрещения сегментации, с другой стороны, страничная трансляция – опциональный механизм и может использоваться либо не использоваться в зависимости от особенностей операционной системы.

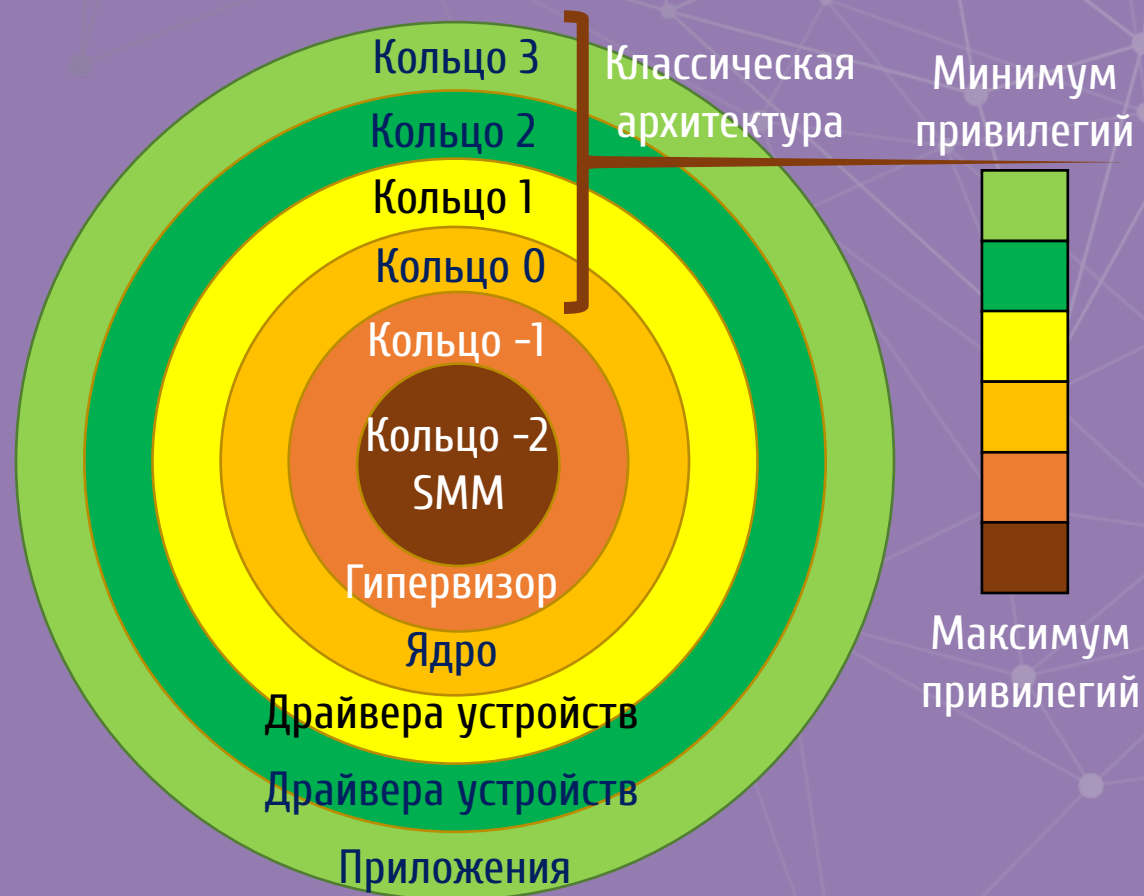


Дескриптор сегмента



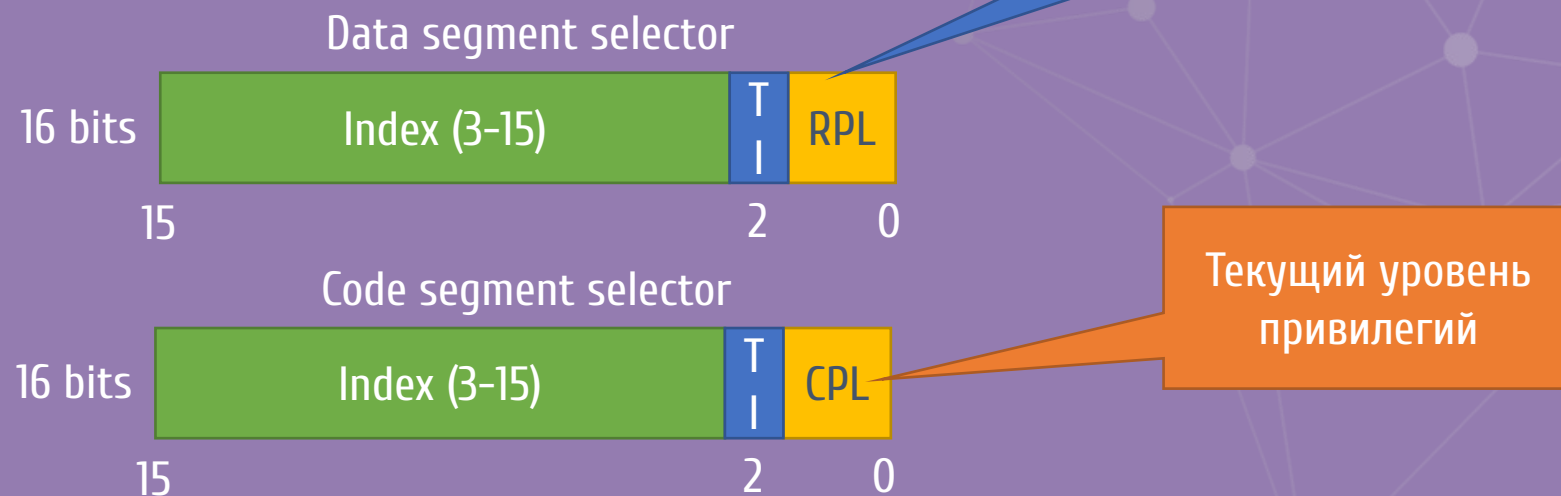
Кольца защиты

- Механизм колец строго ограничивает пути, с помощью которых управление можно передать от одного кольца к другому, а также предписывает ограничения на операции доступа к памяти, которые могут быть произведены внутри кольца.



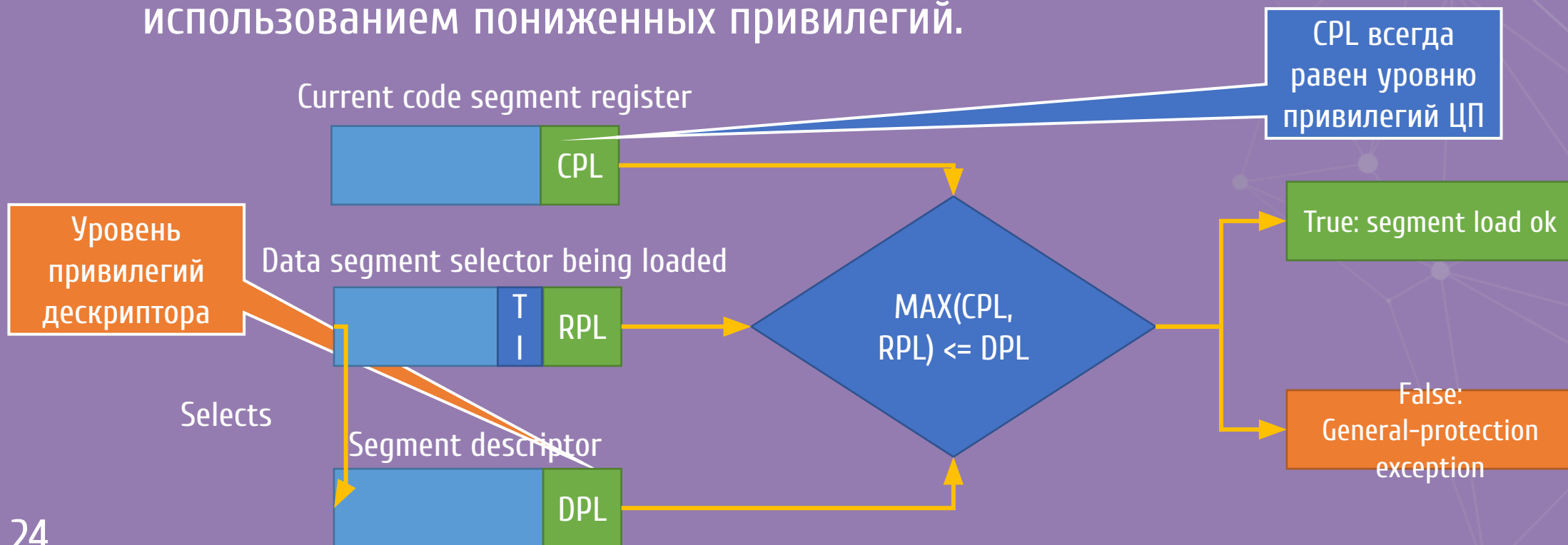
Механизм защиты

- Содержание селектора сегмента данных полностью загружается в различные сегментные регистры, такие как SS («Stack segment register») и DS («Data segment register»)



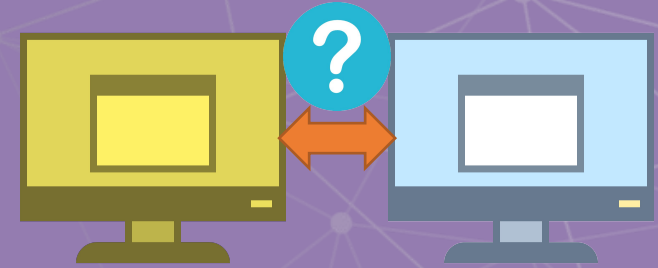
Механизм защиты

- MAX() выбирает наименее привилегированные CPL и RPL и сравнивает их с уровнем привилегий дескриптора (DPL).
- Идея RPL заключается в том, чтобы позволить коду ядра загружать сегмент с использованием пониженных привилегий.



Проблемы работы гипервизора

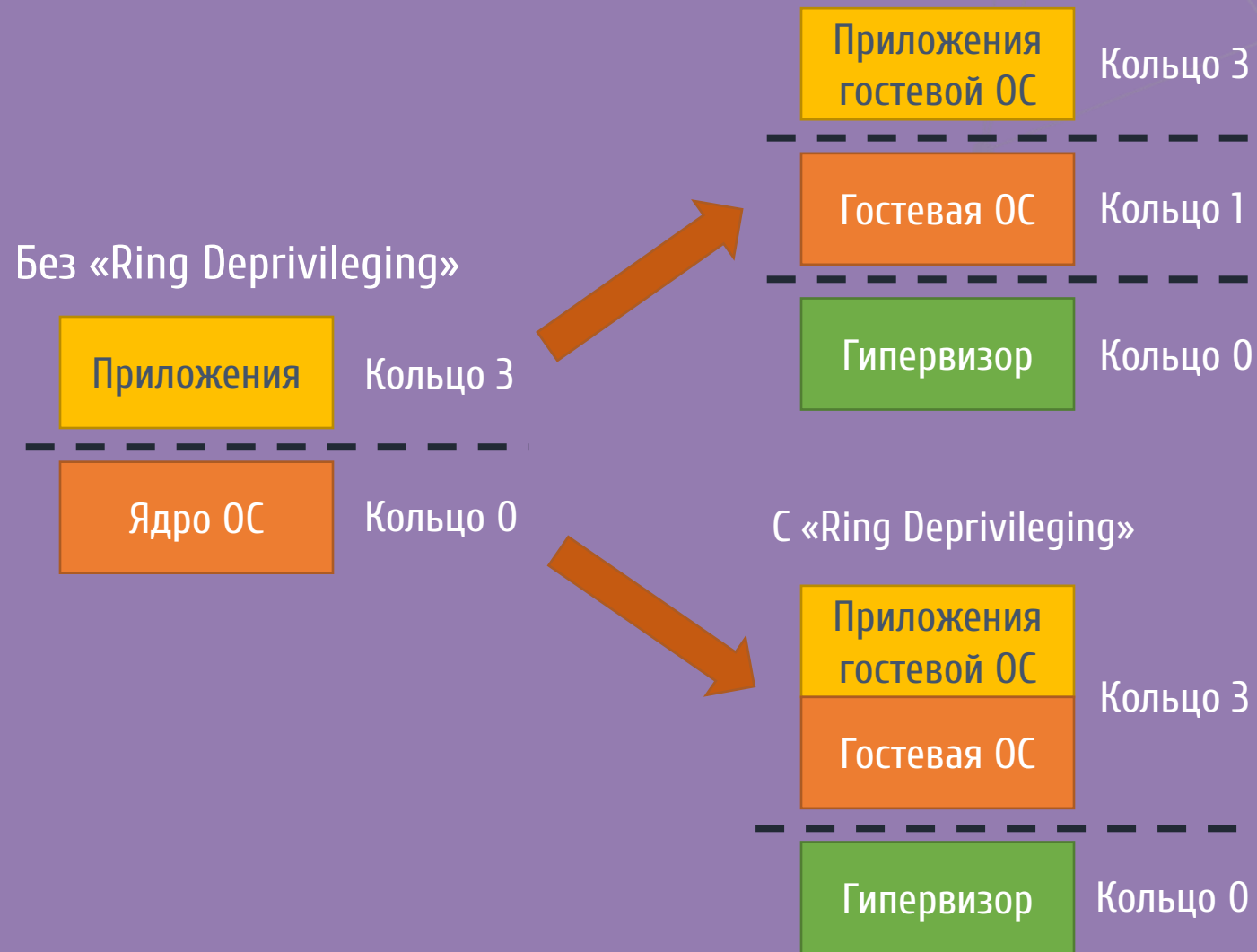
- ОС и приложения VM не должны знать, что существует гипервизор или то, что они разделяют ресурсы ЦП с другими VM
- Гипервизор должен изолировать системное ПО гостевых ОС друг от друга
- Гипервизор должен исполняться защищенно от гостевой ОС
- Гипервизор должен предоставлять платформенный интерфейс для системного ПО гостевой ОС



Классическое решение

- «Trap-and-emulate»
 - Запускать гостевую ОС в депривилегированном режиме
 - Все инструкции, требующие привилегий, исполнять в гипервизоре
 - Гипервизор эмулирует инструкции, например, использует виртуальные прерывания, а не физические

Вот так вот

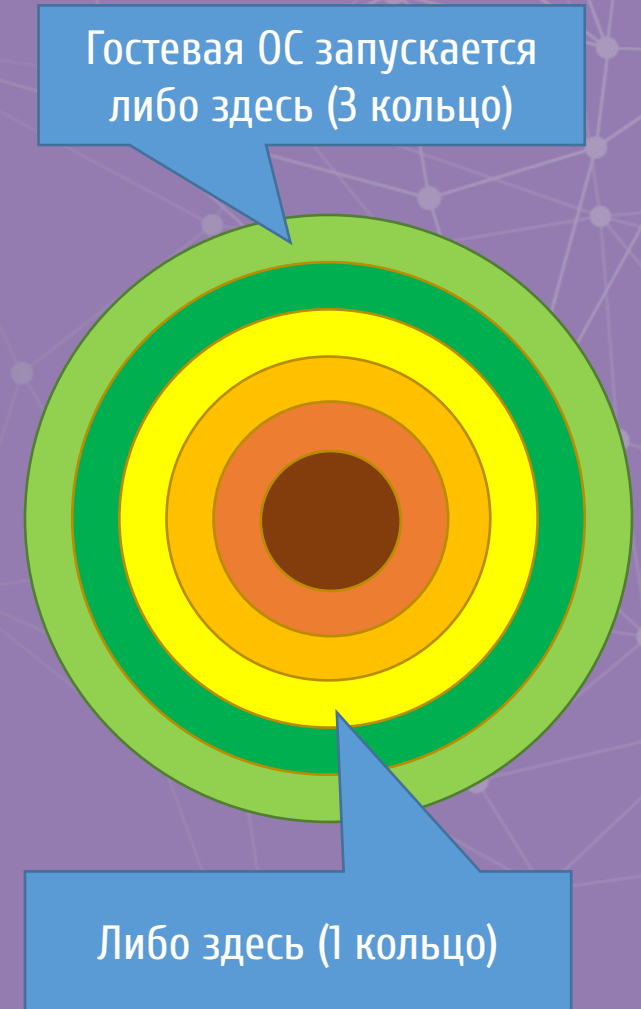


Модель «0/1/3»

Модель «0/3»

Решение для IA-32

- Любое гостевое ПО должно исполняться на кольце выше нулевого
- Привилегированные инструкции генерируют исключения => гипервизор запускается на кольце 0 для обработки ошибок
- Гостевая ОС не должна вмешиваться в работы гипервизора
- Гипервизор переводит привилегированные инструкции в форму, в которой они могут быть исполнены ОС
- Любая непривилегированная инструкция, созданная ОС или приложением, должна исполняться на виртуальной машине
- Гостевая ОС может быть лишена привилегий двумя путями:
 - Она может исполняться на кольце 1 (модель 0/1/3)
 - Или же на кольце 3 (модель 0/3/3)





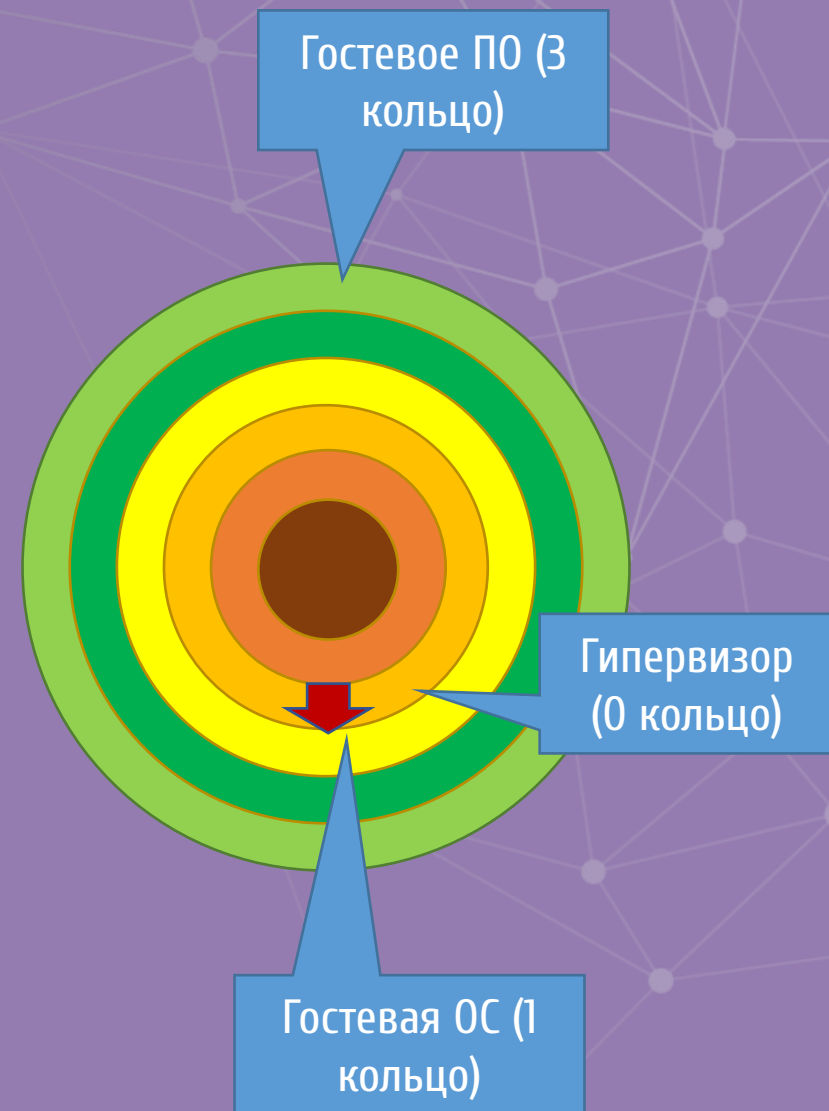
Проблемы виртуализации

Проблемы виртуализации

- Ring Aliasing
- Address-Space Compression
- Excessive Faulting
- Non-trapped instructions
- Interrupt Virtualization
- Ring Compression

Ring Aliasing

- Возникает, если ПО выполняется на уровне, отличном от том, для которого оно было написано
- Последствия:
 - Система может определить, что выполняется не на своём уровне привилегий (возвращается General Protection Exception)



Address-Space Compression

- Гипервизор может полностью работать в адресном пространстве гостевой ОС, но он будет использовать значительную его часть
- Гипервизор может работать в отдельном адресном пространстве, но он должен использовать минимальное пространство гостевой ОС для управления переходами между гостевым ОС и гипервизором (IDT и GDT для IA-32)

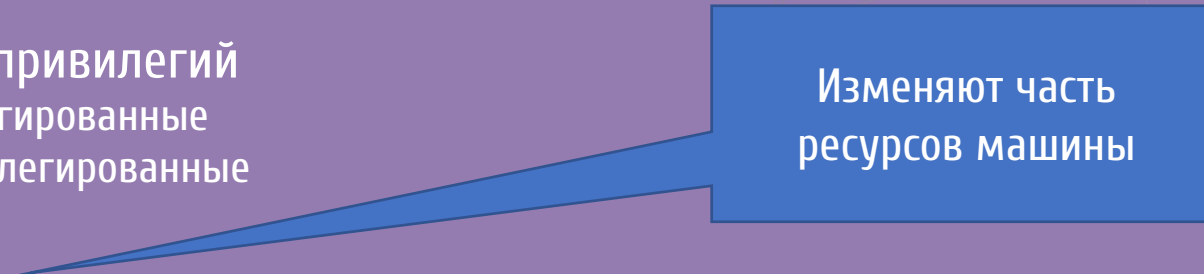
Excessive Faulting

- SYSENTER всегда выполняет переход к уровню привилегий 0, а SYSEXIT возвращает ошибку, если выполняется вне кольца 0 (General Protection Exception)
- Эмуляция SYSENTER* и SYSEXIT** вызывает серьезные проблемы с производительностью
 - *SYSENTER – механизм быстрого системного вызова, команда оптимизирована для наиболее быстрого перехода на нулевой уровень привилегий
 - **SYSEXIT – механизм быстрого системного вызова, команда оптимизирована для наиболее быстрого перехода на уровень привилегий 3 с уровня привилегий 0.

Виды инструкций

- Инструкции

- По уровню привилегий
 - Привилегированные
 - Непривилегированные
- Sensitivity
 - Sensitive
 - Non-sensitive



Изменяют часть
ресурсов машины

- При выполнении в пользовательском режиме, привилегированные команды «отлавливаются».
- «Отлавливание» («Trapping») означает, что машина принудительно переходит в системный режим, посредством чего выполняет некоторый код операционной системы, чтобы справиться с ситуацией
- В некотором смысле, отлавливание предупреждает операционную систему об исполнении команд

Non-trapping instructions

- Проблема в том, что не все sensitive-инструкции X86 являются привилегированными инструкциями. Это означает, что модификация ресурса может произойти без ведома VMM, что может быть опасным

Interrupt Virtualization

- Механизм маскировки внешних прерываний для предотвращения их вызова, когда ОС не готова – большая проблема для гипервизора
- Гипервизор должен управлять маскированием прерываний, чтобы предотвратить маскирование внешних прерываний гостевой операционной системой
- IA-32 использует флаг прерываний (IF) в регистре EFLAGS для управления прерыванием маскировки. IF = 0, если прерывания маскируются

Access to Hidden State

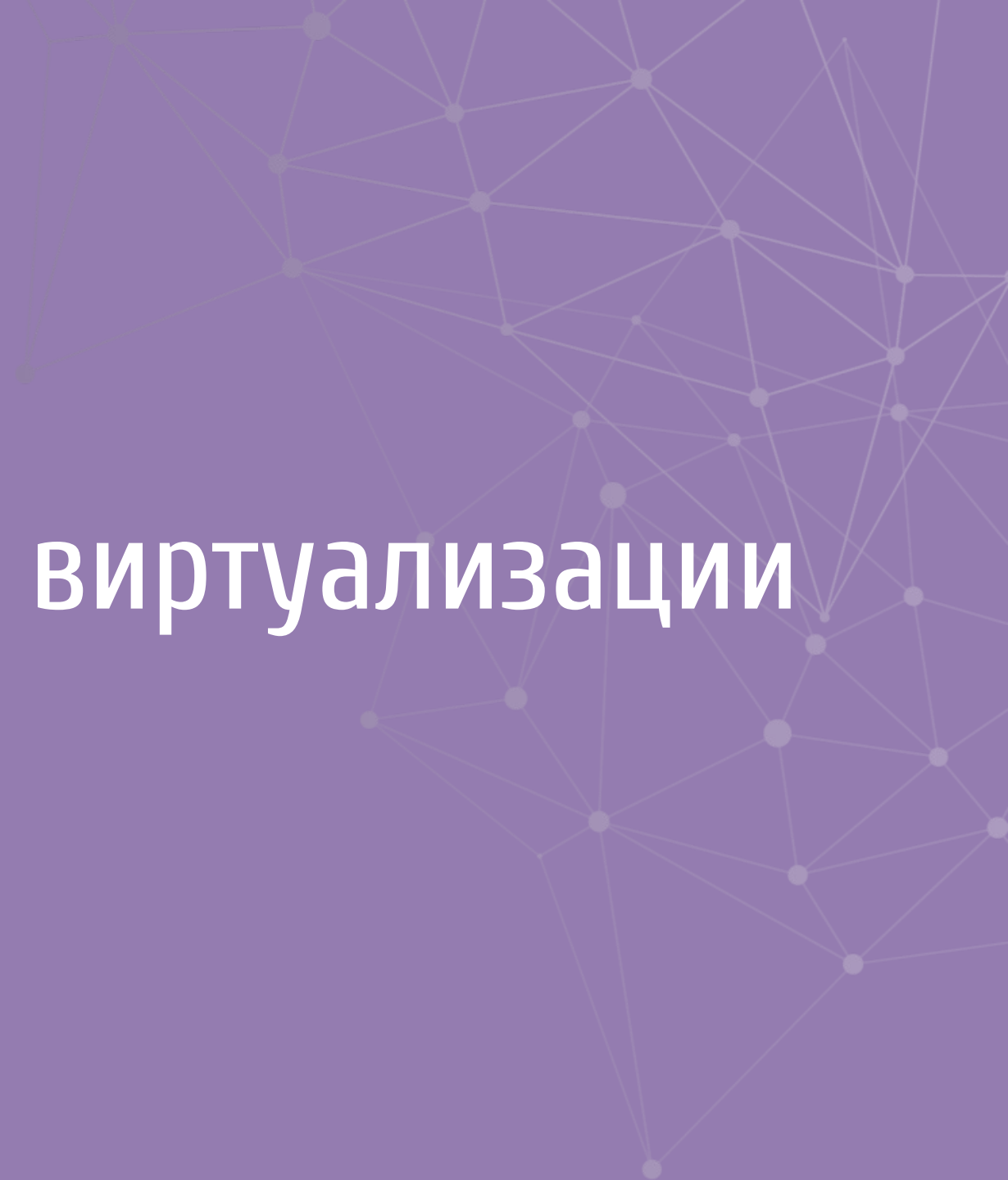
- Некоторые компоненты процессора не представлены в виде доступных системному ПО регистров
- IA-32 содержит скрытый кэш дескриптора для сегментного регистра

Ring Compression

- Механизм «Ring deprivileging» использует механизм, основанный на привилегиях, для защиты гипервизора от гостевого ПО.
- IA-32 включает два механизма: «segment limits» и «paging»:
 - «Segment limits» не применяется в 64-битном режиме
 - Подкачка страниц должна быть использована:
 - Проблема: IA-32 «paging» не отличает уровни привилегий 0-2
 - Гостевая ОС должна исполняться на уровне привилегий 3 (модель 0/3/3)
 - Гостевая ОС не защищена от гостевых приложений

Frequent Access to Privileged Resources

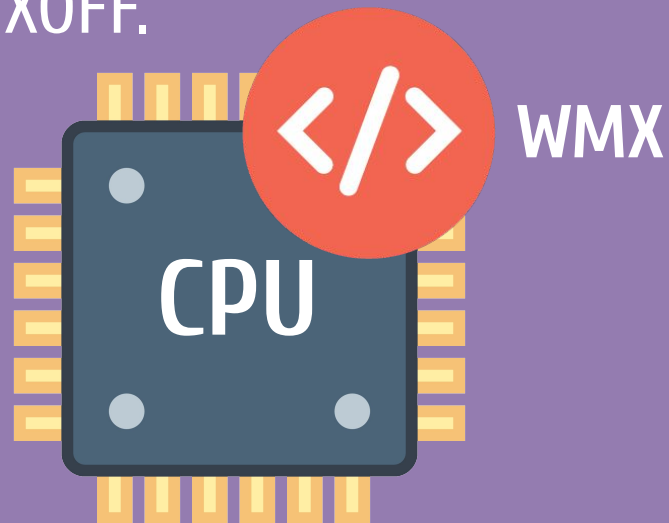
- Существует риск падения производительности, когда многократно осуществляется доступ к привилегированным ресурсам с последующей генерацией ошибок и исключений, которые должны быть обработаны гипервизором

An abstract graphic in the top right corner of the slide, consisting of a network of interconnected nodes and lines, resembling a molecular structure or a data network, rendered in a light gray color against the purple background.

Решение проблем виртуализации при помощи VT-х

Что необходимо?

- Virtual Machine eXtensions (VMX) определяют поддержку виртуальных машин на x86 –платформе на уровне процессора
- Расширенный набор инструкций:
- VMPTRLD, VMPTRST, VMCLEAR, WMREAD, WMWRITE, WMCALL, WMLAUNCH, WMRESUME, WMXON и WMXOFF.



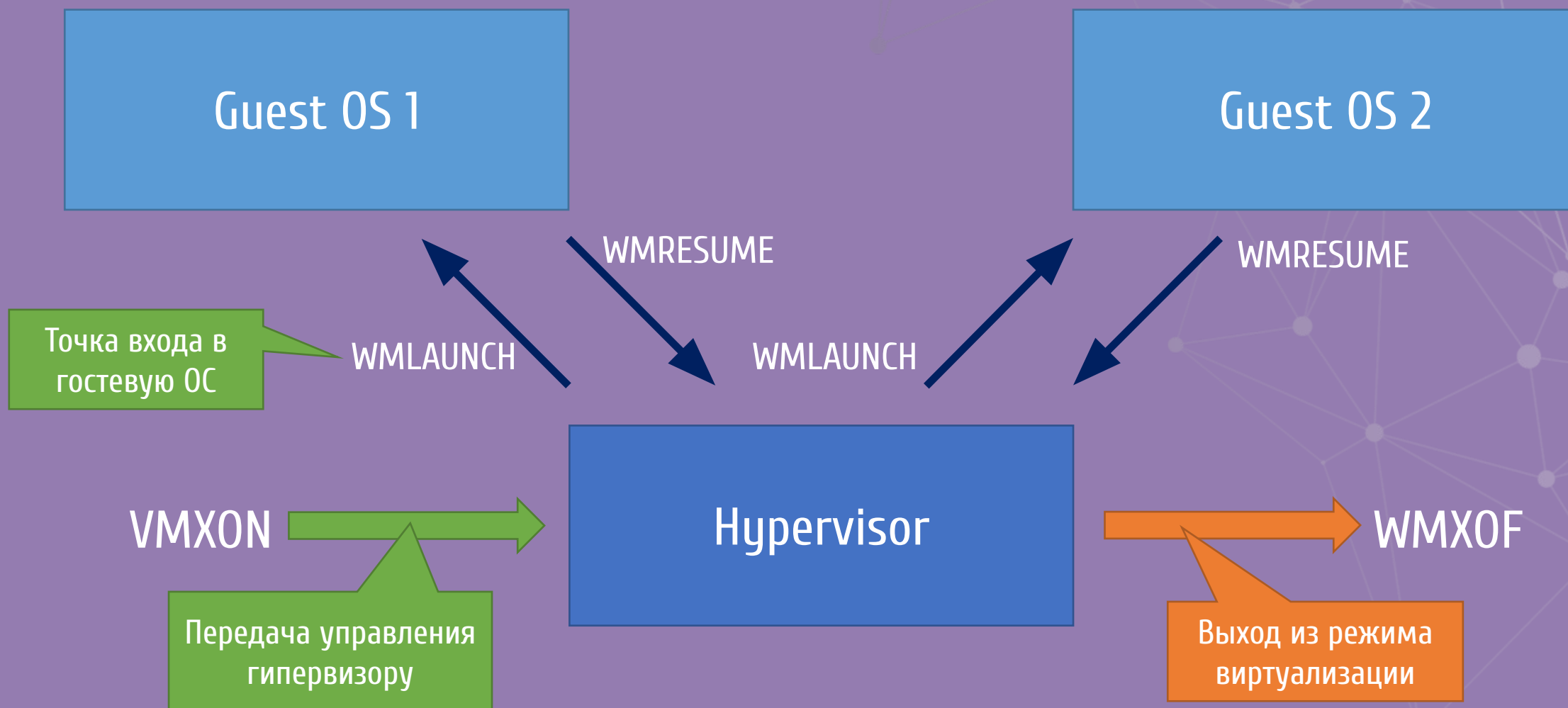
Инструкции

Инструкция	Описание
VMXON, VMXOFF	Вход и выход из режима VMW-root
WMLAUNCH	Начальный переход от гипервизора к гостевой ОС, вводит WMX в non-root режим
WMRESUME	- Используется для последующих входов - Вступает в WMX non-root режим работы - Загружает состояние гостевой ОС и критерий выхода из VMCS
WMEXIT	- Используется при переходе из гостевой ОС в гипервизор - Вступает в WMX root режим работы - Сохраняет состояние гостевой ОС в VMCS - Загружает состояние VMM из VMCS
WMPTRST, VMPTRL	Считывает и записывает указатель VMCS
WMREAD, WMWRITE, WMCLEAR	Читает из VMCS, пишет в него и очищает

VMX-операции

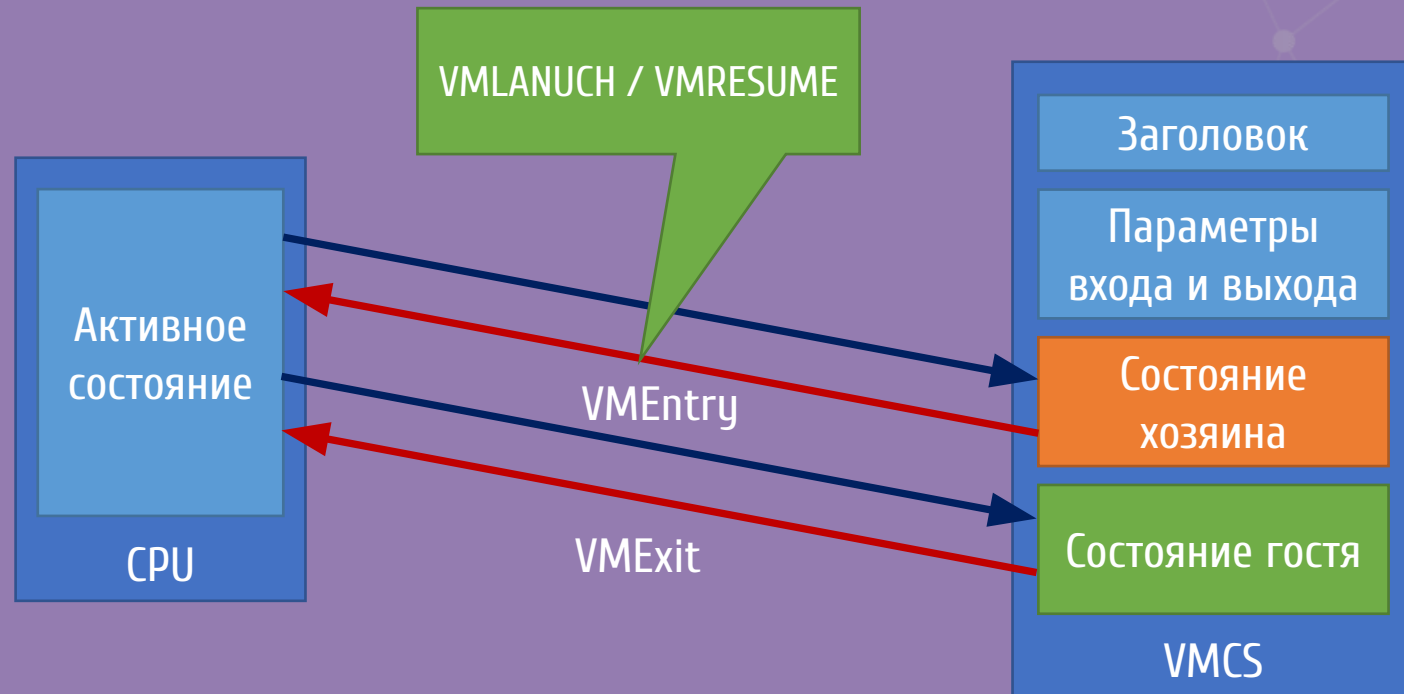
- Два режима:
 - Root – полностью привилегированный, предназначенный для VMM
 - Non-root – не полностью привилегированный, предназначенный для гостевого ПО
- Оба режима поддерживают все четыре уровня привилегий от 0 до 3

Жизненный цикл



Virtual Machine Control Structure

- Virtual Machine Control Structure (VMCS) – структура, главной целью которой является сохранение состояний «гостя» и «хозяина».



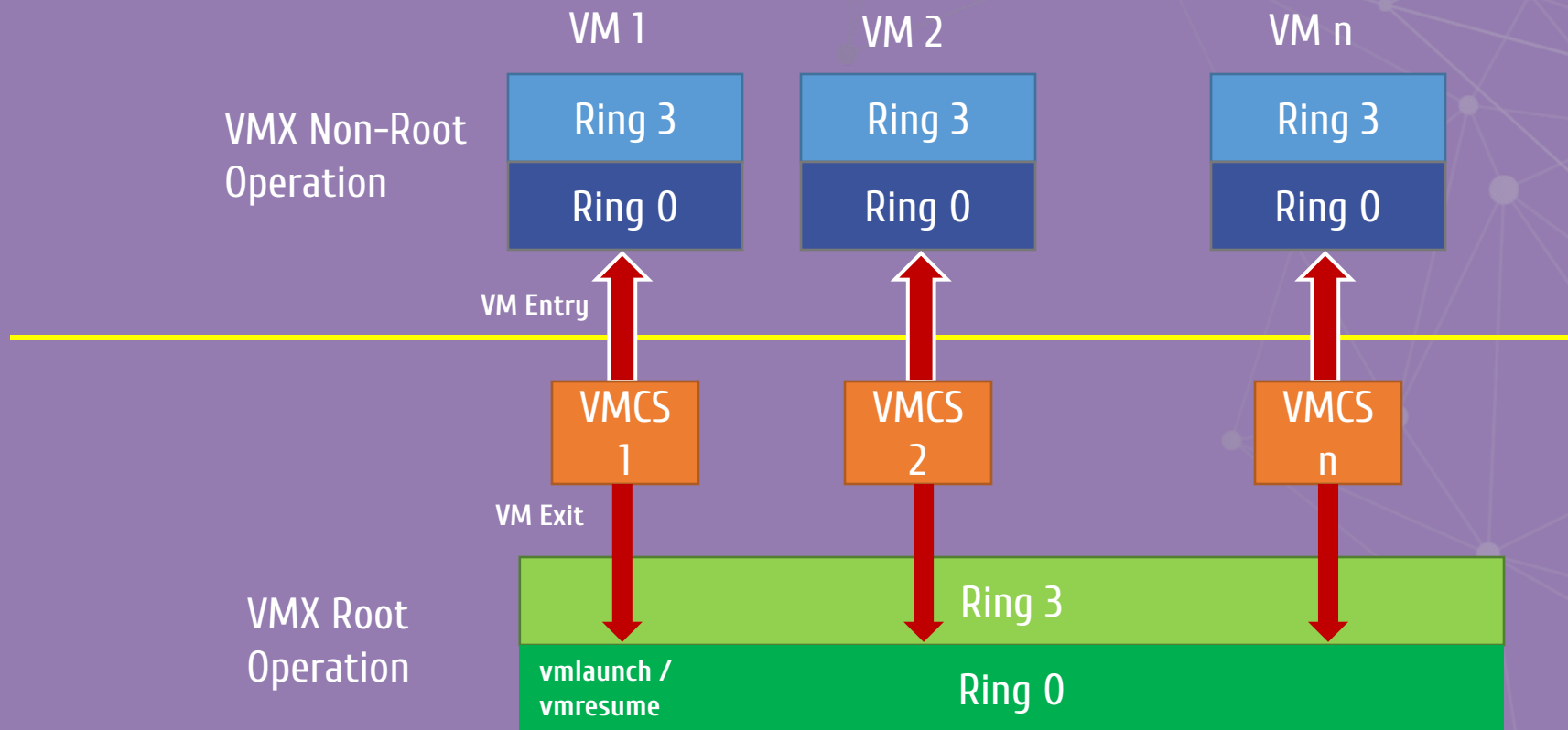
Virtual Machine Control Structure

- Управляет поведением процессора в non-root режиме и работой с VMX
- Конфигурируется гипервизором
- Управляет закрытием гостевой ОС при помощи VMCS указателя

Virtual Machine Control Structure

- Состоит из шести логических групп:
 - Guest-state area: состояние процессора сохраняется в область состояния гостя при закрытии WM из подгружается во время ее загрузки
 - Host-state area: состояние процессора подгружается из области состояния хоста при закрытии WM
 - VM-execution-fields: поля, управляющие работой процессора в режиме non-root
 - VM-exit control fields: поля, управляющие выходом WM
 - VM-entry control fields: поля, управляющие входом WM
 - VM-exit information fields: read-only поля, получающие информацию при закрытии WM, описывающие причину завершения работы WM

Переходы WMX

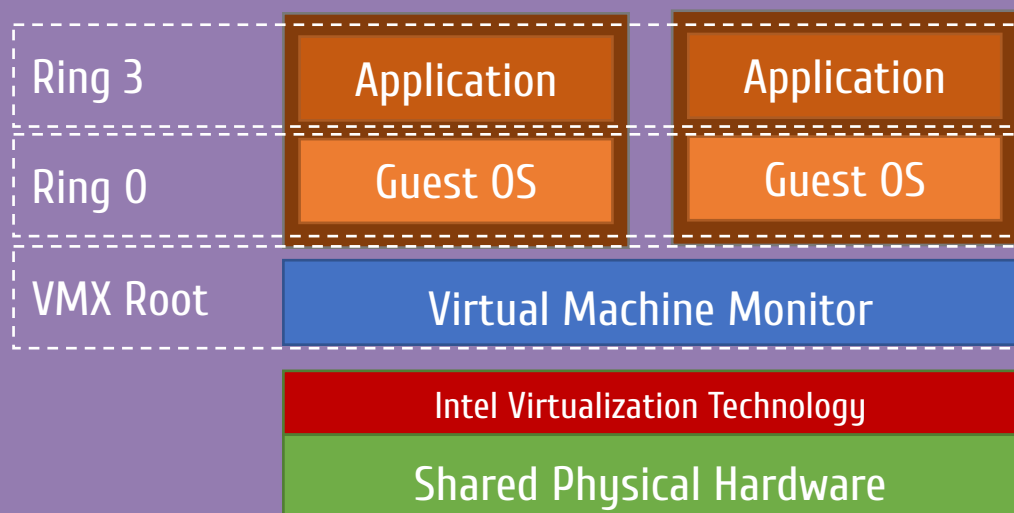


Address-Space Compression

- Каждый переход между гостевым ПО и гипервизором может изменять линейное адресное пространство, позволяя гостевому ПО полностью использовать его
- Переходы WMX управляются VMCS, который находится в физическом адресном пространстве, а не в линейном

Ring Aliasing and Ring Compression

- VT-x позволяет гипервизору запускать гостевое ПО на уровне предполагаемых привилегий:
 - Устраняет проблемы со смещением кольца – такая инструкция, как PUSH (CS регистра) не может обнаружить, что ОС запускается в виртуальной среде
 - Устраняет проблемы сжатия кольца, возникающие, когда гостевая ОС выполняется на том же уровне привилегий, что и гостевые приложения



Non-faulting Access to Privileged State

- VT-x избегает этой проблемы двумя способами:
 - Генерация VMExits во время каждого завершения
 - Обеспечивает конфигурацию прерываний и исключений

Guest System Calls

- Проблемы возникают с инструкциями SYSENTER и SYSEXIT, когда гостевая ОС выполняется вне 0 уровня привилегий. Эта проблема решена, потому что гостевая ОС может выполняться на 0 уровне.

Interrupt Virtualization

- VT-x обеспечивает поддержку виртуализации прерываний
- Он включает в себя компонент управления работой виртуальной машины, управляющий внешними прерываниями

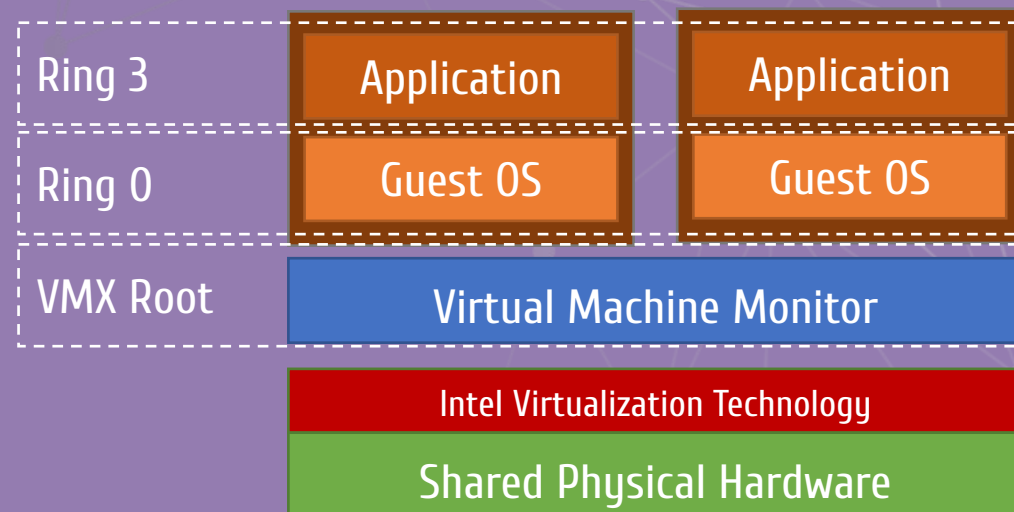
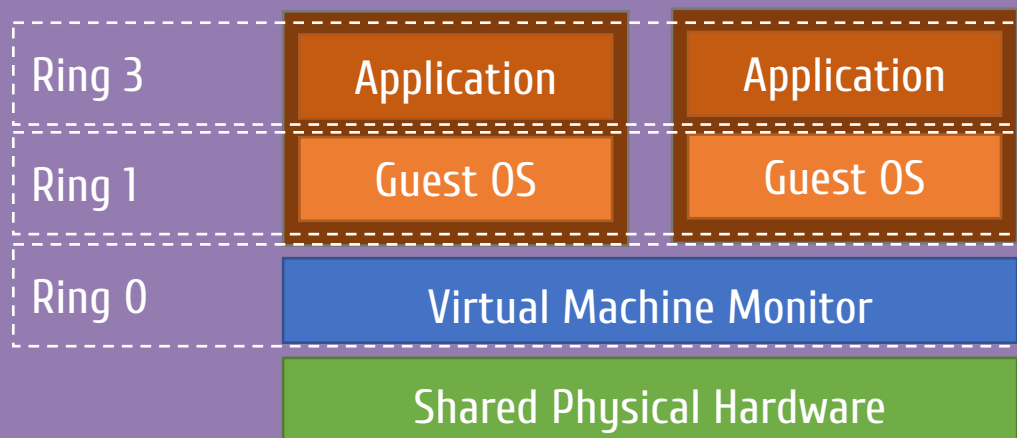
Access to Hidden State

- VT-х включает в гостевой области VMCS поля, отвечающие за состояния ЦПУ, которые не представлены в доступных программному обеспечению регистрах
 - Процессор загружает значения из этих полей при каждом входе виртуальной машины и сохраняет их при выходе

Frequent Access to Privileged Resources

- VT-x позволяет гипервизору избежать лишних расходов на частый доступ к TPR (Task Priority Register)
 - Гипервизор может настроить VMCS так, чтобы он вызывался только тогда, когда требуется

VT-x



Pre VT-x	Post VT-x
Ring 0 понижает привилегии гостевой ОС	Гипервизор выполняется в «root mode»
Гостевая ОС знает, что она выполняется не на Ring 0	Гостевая ОС выполняется прямо на «железе»
	«Ring deprivileging» для гостевых ОС убран

Заключение

- Поддержка технологий аппаратной виртуализации в процессорах открывает широкие перспективы по использованию виртуальных машин в качестве надежных, защищенных и гибких инструментов для повышения эффективности виртуальных инфраструктур



Спасибо за внимание!