ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ «ШАГ В БУДУЩЕЕ, МОСКВА»

ШМО445

регистрационный номер

Информатика и системы управления

название факультета

«Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

название кафедры

Разработка гипервизора Jinet

название работы

Автор: ***Захаров Илья Александрович***

фамилия, имя, отчество

ГБОУ Школа №1533 «ЛИТ», 11 класс

наименование учебного заведения, класс

Научный руководитель: **Байков Борис Камалевич**

фамилия, имя, отчество

ОАО «Т-Платформы»

место работы

ведущий разработчик, руководитель группы

звание, должность

_____

подпись научного руководителя

Москва - 2018

Содержание

Введение **3**

1 Аналитическая часть **4**

1.1 Актуальность 4

1.2 Требования к работе 4

1.3 Анализ предметной области 6

1.4 Аналоги 8

1.5 Выводы 8

2 Конструкторская часть **9**

2.1 Выводы 12

3 Технологическая часть **12**

3.1 Пример использования 13

3.2 Выводы 13

4 Заключение **14**

Благодарности **14**

Список литературы **14**

Иллюстрации **16**

Введение

Гипервизор – это программа, обеспечивающая разделение ресурсов компьютера на несколько виртуальных машин, и запуск на каждой виртуальной машине своей операционной системы.

Такая программа обеспечивает либо разделяемый, либо монопольный доступ виртуальных машин к каждому из аппаратных устройств компьютера. Создаются виртуальные устройства, конфигурация которых может отличаться от конфигурации устройств аппаратных.

Виртуализация позволяет эффективно и безопасно разделять работающие приложения друг от друга. Виртуальные машины не могут влиять на ход работы друг друга, и, если скомпрометирована одна виртуальная машина, все остальные остаются в безопасности. Поэтому она используется в самых разных областях IT. Зачастую виртуализацию также используют для эффективного сегментирования ресурсов компьютера на отдельные виртуальные машины.

Была поставлена задача написать минимальный учебный демонстрационный гипервизор. Исходный текст проекта выпущен под лицензией MIT. `git`-репозиторий гипервизора располагается по адресу `https://github.com/jinet-vm/vmm`.

1 Аналитическая часть

1.1 Актуальность

С каждым годом технологии виртуализации всё глубже и глубже входят в мир информационных технологий, находя применения в самых разных областях IT:

1. изоляция серверных систем для обеспечения их безопасности
2. эффективное сегментирование ресурсов компьютера
3. одновременное использование разных ОС на настольном компьютере
4. отладка гостевых систем через вывод всех выходов из ВМ

Для создания систем виртуализации требуются специалисты, способные понимать принципы и механизмы программной и аппаратной виртуализации. Для меня, как автора диплома, целью написания диплома стало погружение в мир виртуализации, вычислений и прямого программирования железа. Удалось узнать, как работает виртуализация изнутри и научиться писать код управления механизмами обслуживания виртуализации.

1.2 Требования к работе

Идея проекта была предложена руководителем с целью изучения архитектуры x86, включая механизмы изоляции приложений и виртуальных машин

друг от друга путём создания монитора виртуальных машин, он же гипервизор. Для меня это стало уникальной возможностью углубить знания архитектуры процессоров Intel и навыки системного программирования.

Цель работы – это создание минимального монитора виртуальных машин (гипервизора) с использованием механизмов аппаратной виртуализации архитектуры x86-64 (AMD64).

Были поставлены следующие задачи:

1. Обеспечить загрузку из бутсектора (bootsector)

2. Настроить режим работы видеоадаптера для отображения вывода текстовых данных

3. Настроить память в 32-битной плоской модели и осуществить переход в 32-битный защищённый режим (`Protected Mode`)

4. Настроить механизмы страничной трансляции памяти и переход в 64-битный режим (`Long Mode`)

5. Настроить обработку исключений и прерываний

6. Настроить расширения аппаратной виртуализации Intel VT-i и включение режима VMX

7. Обеспечить функционирования вызовов (VMCall) из виртуальных машин в гипервизор (монитор виртуальных машин) и возвратов обратно в виртуальные машины

1.3 Анализ предметной области

Виртуализация Виртуализация – техника предоставления исполняемой программе набора вычислительных ресурсов, абстрагированная от их аппаратной реализации. Виртуализация была предметом изучения информатики на протяжении многих лет: так, например, советские инженеры решали проблему портирования программного обеспечения с платформ имеющих другие интерфейсы, нежели физический компьютер, на котором программа исполнялась.

В рамках этой работы мы будем говорить не столько о виртуализации ресурсов, сколько о работе гипервизора – программы, занимающейся разделением работы ресурсов одной физической машины (хозяин (*англ.* host)) на множество виртуальных машин (гость (англ. guest)). Внутри каждой виртуальной машины исполняется своя ОС, ход работы которой не влияет на работу других ВМ.

Гипервизор, в отличие от эмулятора, выполняющего программную эмуляцию команд, лишь перехватывает управление у виртуальных машин в случае необходимости. Код виртуальных машины выполняется аппаратно в процессоре. Так как принцип работы гипервизора предполагает изоляцию виртуальных машин, для более эффективной его работы необходима аппаратная поддержка виртуализации. Первой в этой области была компания IBM с мэйнфреймами System/360, System/370, созданными на рубеже 60-70-х годов прошлого века. Современные процессоры Intel также поддерживают расширения аппаратной виртуализации (VT-i, VT-d), что значительно ускоряет процесс виртуализации.

После появления первых гипервизоров появилась необходимость создания формальных критериев виртуализации. В 1974 статья Джеральда Попека и Роберта Гольдберга их сформировала.

Классический критерий виртуализуемости Требования к монитору виртуальных машин (то же, что и гипервизор) состоят из трёх пунктов:

1. **Изоляция**. Каждая ВМ имеет доступ только к своим ресурсам. Виртуальные машины не влияют на работу друг друга, если одна виртуальная машина оказалась зараженной вирусом, другая продолжает работу.

2. **Эквивалентность**. Виртуальная машина ведёт себя так же, как и настоящий компьютер с аналогичными характеристиками. Единственное различие заключается в скорости исполнения, виртуальная машина работает медленнее компьютера.

3. **Эффективность**. Статистически преобладающее подмножество инструкций виртуального процессора должно исполняться напрямую хозяйским процессором, без вмешательства монитора ВМ. Управление передаётся гипервизору только в случае привилегированной операции – той, которая может нарушить изоляцию машин.

Типы гипервизоров Гипервизоры делят по их устройству на две большие группы: гипервизоры I и II типа.

- **Гипервизоры I типа** исполняются непосредственно на компьютере и имеют полный доступ к его устройствам. Компьютер загружается в софт гипервизора, и монитор ВМ, подобно операционной системе, начинает работы с устройствами. Примеры: VMWare ESXi, Xen. (см. рис. 1)

- **Гипервизоры II типа** загружаются внутри ОС и получают доступ к устройствам через её интерфейсы. Зачастую при установке гипервизор II типа устанавливает в ядро ОС свой модуль, с помощью которого он может об-

ращаться к низкоуровневым процессорным расширениям виртуализации. Примеры: KVM, VirtualBox. (см. рис. 2) Гипервизор Jinet - гипервизор I типа: так легче обращаться к ресурсам компьютера.

1.4 Аналоги

В данной работе представляется маленький гипервизор. Подобные ему большие аналоги писали десятки программистов много месяцев. Гипервизор Jinet сейчас позволяет запускать простейший код в изолируемом окружении BM виртуальной машины. У всех из предложенных ниже гипервизоров исходный текст находится в открытом доступе. Приведённые аналоги (см. таблицу 1) были написаны специалистами в области информационной безопасности и системного программирования. Некоторые из предложенных гипервизоров первого типа, а некоторые – второго.

1.5 Выводы

Была изучена архитектура процессоров Intel x86. За время работы над гипервизором пришлось изучить самые разные аспекты работы современных ПК: от работы в 16-битном режим до опыта неуспешной отладки на разработческих платах.

Была рассмотрены теоретические и практические аспекты виртуализации. Изучены критерии виртуализуемости и типы гипервизоров, исследованы аналогичные разрабатываемому минимальные и исследовательские гипервизоры.

[1]см. 1.3

Таблица 1: Аналоги

Гипервизор	Тип VMM[1]	Платформы (если II тип)	Размер исходного текста	Назначение
SimpleVisor	II	Win8.1; частично UEFI	162 KB	использование новейшего VMX
Ramooflax	I	–	2.26 MB	анализ/отладка /контроль ВМ
HyperPlatform	II	Win7; Win8.1; Win10: x86 & x64	7.64 MB	анализ работы Windows
Jinet	I	–	< 1MB	учебный гипервизор

2 Конструкторская часть

В ходе работы над проектом было реализовано множество задач (■ и + означают, что задача решена; □ и − означают, что задача ещё не решена):

Изучение Real Mode ассемблера (16bit)

+ Загрузочный сектор: написание своего загрузочного кода
+ Написание клеточного автомата в загрузочном секторе
+ BIOS - прерывания: int 13h, int 15h
+ Работа с VideoBIOS: работа с int 10h
+ Начало работы с эмуляторами Bochs, QEMU

Изучение Protected Mode (32bit)

+ Управление памятью: глобальная таблица дескрипторов (GDT)
+ Изучение Big Real Mode
+ Изучение структуры исполняемого файла ELF
+ Изучение компоновки кода на Assembler и кода на языке C
+ Начало разработки в Protected Mode на языке C
+ Чтение карты доступной физической памяти
+ Виртуальная память: настройка страничного отображения (Paging)
+ Обработка прерываний: таблица прерываний в защ. режиме (IDT)
+ Перенос кода ядра и таблиц выше 1 МБ
+ Изучение механизмов аппаратной многозадачности. Структура TSS

Изучение топологии ядер и процессоров (ACPI: SRAT, SLIT)

+ Изучение APIC, XAPIC (LAPIC, IOAPIC, ACPI: MADT)

Изучение Long Mode (64 bit, x86-64)

+ Изучение особенностей TSS и IDT в 64-битном режиме
+ Изучение особенностей страничной адресации памяти (PAE)

Изучение механизмов организации ядра

+ Сборка проекта с помощью утилиты `make`
+ Портирование проекта с собственного загрузчика на GRUB
+ Настройка VGA, VBE как выводов терминала системы
+ Управление памятью

- + Высокоуровневый доступ к структурам paging-а
- + Выделение физической памяти (binary buddy allocator)
- + Куча (heap)

+ Доступ к отладочной консоли через Intel LPSS UART[2]

+ Обработка ACPI таблиц: MADT, DBGP

+ Инициализация многоядерности: отправление SIPI, init-IPI

+ Разработка планировщика для задач на BSP (round-robin)

∗ Написание документации проекта на Doxygen [in process]

Изучение аппаратных механизмов виртуализации VMX (VT-i, VT-d)

+ Подготовка и включение VMX-режима

+ Подготовка управляющих структур для виртуальной машины

+ Создание обработчика VMCal

+ Создание обработчиков событий

+ Создание расширенных таблиц страничной трансляции (EPT)

− Создание BIOS для виртуальных машин

− Создание виртуальных устройств:

- − Клавиатура и мышь
- − Жёсткий диск (виртуальный int 13h)
- − Виртуальный текстовый дисплей
- − Виртуальный VGA дисплей

− Создание механизма проброса устройств (pass-through)

[2]определяется в DBGP, PCI номер устройства: вендор 8086h, id устройства A166h

– Создание партиции гипервизора и её структуры

– Создание инструментов управления

В ходе работы над дипломным проектом было изучено большое количество документации по процессорам Intel ([1]) и AMD (как первосоздателя x86-64: [2]), документация `gnu make` ([3]), `gnu ld` ([4]), `gcc` ([5]), `fasm` ([6]), формата ELF ([7]), таблиц ACPI ([8])

2.1 Выводы

Был получен важный опыт работы с технической документацией. Большая часть документации, изученной во время работы над дипломным проектом, была написана на техническом английском. При работе с такой низкоуровневой технологией, как аппаратная виртуализация, была важно консультироваться со справочниками и документацией Intel.

Были продуманы основные этапы разработки гиперизора Jinet. Работа над проектом начинается с изучения устройства Real Mode и заканчивается разработкой подсистем, отвечающих за непосредственное взаимодействие с Intel VMX.

3 Технологическая часть

Гипервизор Jinet был написан на языках программирования ассемблер (диалекты `fasm` и `as`) и C (компилятор `gcc`). Сборка проекта осуществляется с помощью сборщика `gnu ld` и утилиты `gnu make`. В качестве системы контроля

версий используется `git` и хостинг GitHub.

Для вёрстки данной использовался LaTeX, а презентация для защиты создавалась в Microsoft PowerPoint.

3.1 Пример использования

На рис. 3 изображена работа гипервизор Jinet. На данный момент гипервизор поддерживает работу виртуальной машины, которая выводит сообщение о своём успешном запуске с помощью vmcall: `Hello! I am VM1!`

3.2 Выводы

Был изучен инструментарий современного системного программиста. Основы статического компонования, формат исполняемых файлов `elf`, особенности встроенного ассемблера gcc – самые разные технологии системного программирования были изучены при написании данной работы. Изучались также утилиты для сборки: были написаны скрипты для сборки проекта утилитами `gnu make` и `gnu ld`, а также Python-скрипт для конфигурирования проекта.

Были изучены основы работы с языком ассемблер и работы с большими проектами на C. Были изучены три диалекта языка ассемблер: `masm`, `fasm` и `gnu as`. В работу вошёл код, написанный на последних двух. Также были изучены принципы написания интерфейсов и безопасного программирования на C.

4 Заключение

Создан гипервизор Jinet. Доказано, что можно написать гипервизор силами одного человека в короткие сроки. Многое ещё требует реализации и улучшения:

1. Доработка SeaBIOS для поддержки запуска 16-битных операционных систем

2. Поддержка ряда устройств для запуска одного из вариантов DOS

3. Поддержка гипервизором работы на нескольких ядрах, процессорах

4. Поддержка NUMA-систем

5. Поддержка технологии Intel GVT-g (представлена в 2015), делающей возможной аппаратную виртуализацию видеоадаптеров для ВМ

Благодарности

В рамках данного проекта использовался исходный код реализации функции printf из проекта tinyprintf: `https://github.com/cjlano/tinyprintf`.

Список литературы

[1] Intel. Intel® 64 and IA-32 Architectures Software Developer Manual. 2017.

[2] AMD. AMD64 Architecture Programmer's Manual, Volume 2: System Programming. 2017.

[3] GNU Project and Free Software Foundation. GNU Make Manual.

[4] GNU Project and Free Software Foundation. GNU Linker Manual.

[5] GNU Project and Free Software Foundation. GCC Manual.

[6] Grysztar Tomasz. flat assembler 1.71.

[7] Portable Formats Specification Version 1.1. Executable and Linkable Format (ELF).

[8] Toshiba HP Intel Microsoft Phoenix. Advanced Configuration and Power Interface Specification, Revision 5.0a.

[9] Popek G. J., Goldberg R. P. Formal requirements for virtualizable third generation architectures // Communications of ACM. 1974.

[10] @Atakua. Аппаратная виртуализация. Теория, реальность и поддержка в архитектурах процессоров. https://habrahabr.ru/company/intel/blog/196444/. 2013.

[11] М. Чурдакис. The Infamous Trilogy: CPU internals, Virtualization, Raw multicore programming. https://www.codeproject.com/articles/45788/the-real-protected-long-mode-assembly-tutorial-for. 2015.

[12] OSDev Wiki. http://wiki.osdev.org/Main_Page.

[13] В. Садовников. Начала программирования в защищённом режиме. http://e-zine.excode.ru/online/1/Introduction.html. 2006.

Иллюстрации

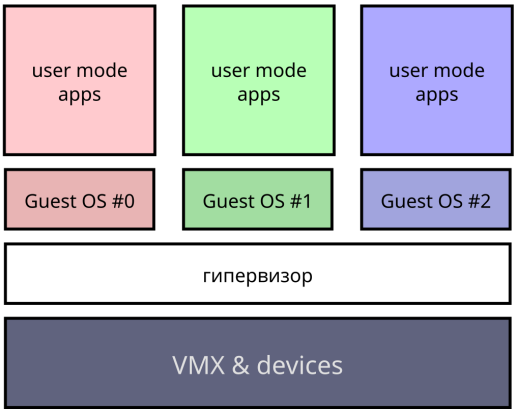

Рис. 1: I тип гипервизоров

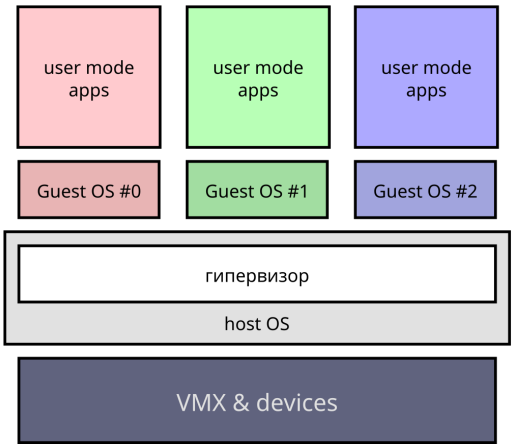

Рис. 2: II тип гипервизоров

Рис. 3: Работа программы (VESA-режим)

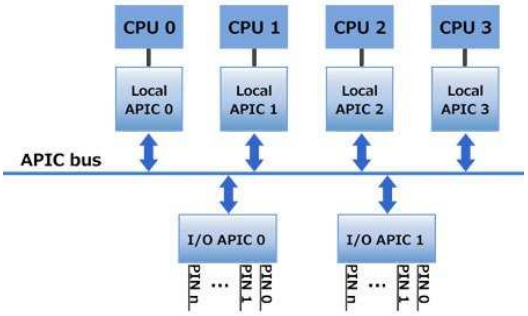

Рис. 4: Логическая схема работы APIC

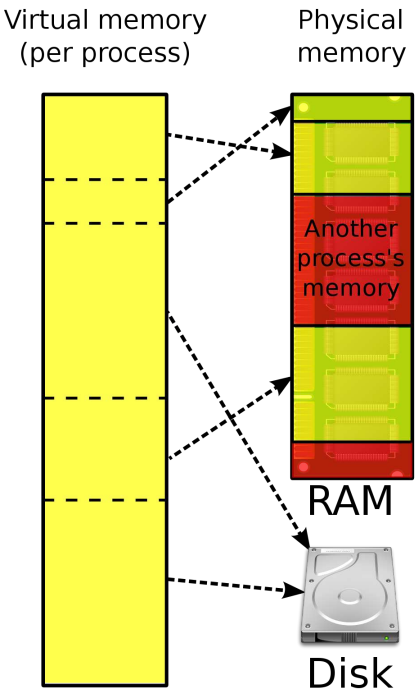

Рис. 5: Механизм работы виртуальной памяти