SCIENTIFIC JOURNAL

3(70)/2014

INFORMATSIONNO-UPRAVLIAIUSHCHIE SISTEMY (INFORMATION AND CONTROL SYSTEMS)

REFEREED EDITION

Founder
«Information and Control Systems», Ltd.

Editor-in-Chief
M. Sergeev
Dr. Sc. Tech., Professor, St.-Petersburg, Russia

Deputy Editor-in-Chief
E. Krouk
Dr. Sc. Tech., Professor, St.-Petersburg, Russia

Executive secretary
O. Muravtsova

Editorial Council
L. Chubraeva
RAS Corr. Member, Dr. Sc. Tech., Professor, St. Petersburg, Russia
L. Fortuna
PhD, Professor, Catania, Italy
A. Fradkov
Dr. Sc. Tech., Professor, St. Petersburg, Russia
V. Kozlov
Dr. Sc. Tech., Professor, St. Petersburg, Russia
C. Christodoulou
PhD, Professor, Albuquerque, New Mexico, USA
B. Meyer
PhD, Professor, Zurich, Switzerland
A. Ovodenko
Dr. Sc. Tech., Professor, St. Petersburg, Russia
Y. Podoplyokin
Dr. Sc. Tech., Professor, St. Petersburg, Russia
Yu. Shokin
RAS Academician, Dr. Sc. Phys.-Math., Novosibirsk, Russia
V. Simakov
Dr. Sc. Tech., Professor, Moscow, Russia
V. Vasilev
RAS Corr. Member, Dr. Sc. Tech., Professor, St. Petersburg, Russia
R. Yusupov
RAS Corr. Member, Dr. Sc. Tech., Professor, St. Petersburg, Russia

Editorial Board
V. Anisimov
Dr. Sc. Tech., Professor, St. Petersburg, Russia
B. Bezruchko
Dr. Sc. Phys.-Math., Saratov, Russia
N. Blaunstein
Dr. Sc. Phys.-Math., Professor, Beer-Sheva, Israel
A. Dudin
Dr. Sc. Tech., Professor, Minsk, Belarus
V. Khimenko
Dr. Sc. Tech., Professor, St. Petersburg, Russia
G. Maltsev,
Dr. Sc. Tech, Professor, St. Petersburg, Russia
V. Melekhin
Dr. Sc. Tech, Professor, St. Petersburg, Russia
A. Shalyto
Dr. Sc. Tech., Professor, St. Petersburg, Russia
A. Shepeta
Dr. Sc. Tech., Professor, St. Petersburg, Russia
A. Smirnov
Dr. Sc. Tech., Professor, St. Petersburg, Russia
Z. Yuldashev
Dr. Sc. Tech., Professor, St. Petersburg, Russia
A. Zeifman
Dr. Sc. Phys.-Math., Vologda, Russia

Editor: A. Larionova
Proofreader: T. Zvertanovskaia
Design: A. Koleshko, M. Chernenko
Layout and composition: N. Karavaeva

Contact information
The Editorial and Publishing Center, SUAI
67, B. Morskaia, 190000, St. Petersburg, Russia
Website: http://i-us.ru/en, E-mail: ius.spb@gmail.com
Tel.: +7 - 812 494 70 02

The Journal was registered in the Ministry of Press, Broadcasting and Mass Media of the Russian Federation. Registration Certificate JD № 77-12412 from April, 19, 2002. Re-registration in the Federal Service for Supervision in the Sphere of Telecom, Information Technologies and Mass Communications (ROSKOMNADZOR) due to change of the founder: «Information and Control Systems», Ltd., JD № FS77-49181 from March, 30, 2012.

The journal is distributed by subscription. Subscription can be made in the Editorial and publishing center, SUAI as well as in any post office based on «Rospechat» catalogue:
№ 48060 — annual subscript, № 15385 — semiannual subscript.

© Corporate authors, 2014

Submitted for publication 07.04.14. Passed for printing 17.06.14. Format 60×841/8.
Offset paper. Phototype SchoolBookC. Offset printing.

Layout original is made at the Editorial and Publishing Center, SUAI.
67, B. Morskaia, 190000, St. Petersburg, Russia
Printed from slides at the Editorial and Publishing Center, SUAI.
67, B. Morskaia, 190000, St. Petersburg, Russia

НАУЧНЫЙ ЖУРНАЛ

3(70)/2014

РЕЦЕНЗИРУЕМОЕ ИЗДАНИЕ

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

Учредитель
ООО «Информационно-управляющие системы»

Главный редактор
М. Б. Сергеев,
д-р техн. наук, проф., С.-Петербург, РФ

Зам. главного редактора
Е. А. Крук,
д-р техн. наук, проф., С.-Петербург, РФ

Ответственный секретарь
О. В. Муравцова

Редакционный совет:

Председатель А. А. Оводенко,
д-р техн. наук, проф., С.-Петербург, РФ
В. Н. Васильев,
чл.-корр. РАН, д-р техн. наук, проф., С.-Петербург, РФ
В. Н. Козлов,
д-р техн. наук, проф., С.-Петербург, РФ
К. Кристодолу,
д-р. наук, проф., Альбукерке, Нью-Мексико, США
Б. Мейер,
д-р наук, проф., Цюрих, Швейцария
Ю. Ф. Подоплекин,
д-р техн. наук, проф., С.-Петербург, РФ
В. В. Симаков,
д-р техн. наук, проф., Москва, РФ
Л. Фортуна,
д-р наук, проф., Катания, Италия
А. Л. Фрадков,
д-р техн. наук, проф., С.-Петербург, РФ
Л. И. Чубраева,
чл.-корр. РАН, д-р техн. наук, С.-Петербург, РФ
Ю. И. Шокин,
акад. РАН, д-р физ.-мат. наук, проф., Новосибирск, РФ
Р. М. Юсупов,
чл.-корр. РАН, д-р техн. наук, проф., С.-Петербург, РФ

Редакционная коллегия:
В. Г. Анисимов,
д-р техн. наук, проф., С.-Петербург, РФ
Б. П. Безручко,
д-р физ.-мат. наук, проф., Саратов, РФ
Н. Блаунштейн,
д-р физ.-мат. наук, проф., Беэр-Шева, Израиль
А. Н. Дудин,
д-р физ.-мат. наук, проф., Минск, Беларусь
А. И. Зейфман,
д-р физ.-мат. наук, проф., Вологда, РФ
Г. Н. Мальцев,
д-р техн. наук, проф., С.-Петербург, РФ
В. Ф. Мелехин,
д-р техн. наук, проф., С.-Петербург, РФ
А. В. Смирнов,
д-р техн. наук, проф., С.-Петербург, РФ
В. И. Хименко,
д-р техн. наук, проф., С.-Петербург, РФ
А. А. Шалыто,
д-р техн. наук, проф., С.-Петербург, РФ
А. П. Шепета,
д-р техн. наук, проф., С.-Петербург, РФ
З. М. Юлдашев,
д-р техн. наук, проф., С.-Петербург, РФ

Редактор: А. Г. Ларионова
Корректор: Т. В. Звертановская
Дизайн: А. Н. Колешко, М. Л. Черненко
Компьютерная верстка: Н. Н. Караваева

Адрес редакции: 190000, Санкт-Петербург,
Б. Морская ул., д. 67, ГУАП, РИЦ
Тел.: (812) 494-70-02, e-mail: ius.spb@gmail.com, сайт: http://i-us.ru

Журнал зарегистрирован в Министерстве РФ по делам печати, телерадиовещания и средств массовых коммуникаций. Свидетельство о регистрации ПИ № 77-12412 от 19 апреля 2002 г. Перерегистрирован в Роскомнадзоре. Свидетельство о регистрации ПИ № ФС77-49181 от 30 марта 2012 г.

Журнал входит в «Перечень ведущих рецензируемых научных журналов и изданий, в которых должны быть опубликованы основные научные результаты диссертации на соискание ученой степени доктора и кандидата наук».

Журнал распространяется по подписке. Подписку можно оформить через редакцию, а также в любом отделении связи по каталогу «Роспечать»: № 48060 — годовой индекс, № 15385 — полугодовой индекс.

© Коллектив авторов, 2014

Сдано в набор 07.04.14. Подписано в печать 17.06.14. Формат 60×84 1/8.
Бумага офсетная. Гарнитура SchoolBookC. Печать офсетная.
Усл. печ. л. 16,0. Уч.-изд. л. 20,1. Тираж 1000 экз. Заказ 258.

Оригинал-макет изготовлен в редакционно-издательском центре ГУАП.
190000, Санкт-Петербург, Б. Морская ул., 67.

Отпечатано с готовых диапозитивов в редакционно-издательском центре ГУАП.
190000, Санкт-Петербург, Б. Морская ул., 67.

УДК 004.45

АРХИТЕКТУРА АВТОНОМНОГО МИКРОГИПЕРВИЗОРА РЕАЛЬНОГО ВРЕМЕНИ И АВТОМАТИЗИРОВАННОЕ ИЗМЕРЕНИЕ ЕГО ВРЕМЕННЫХ ХАРАКТЕРИСТИК

И. В. Колчин[а], *канд. техн. наук, ведущий инженер*
С. Н. Филиппов[а, б], *младший инженер, аспирант*
[а]*ООО «Сименс», департамент корпоративных технологий, Санкт-Петербург, РФ*
[б]*Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, РФ*

Постановка проблемы: *гипервизоры и виртуальные машины приобрели популярность в последнее десятилетие благодаря своим многочисленным преимуществам. Однако есть и обратная сторона этого положения, особенно для компаний, разрабатывающих системы с особыми требованиями безопасности. Программное обеспечение становится слишком сложным, чтобы быть совместимым со всеми версиями и конфигурациями оборудования. Как следствие, подобное программное обеспечение трудно сертифицировать на соответствие требованиям стандартов безопасности, таким как IEC 61508. Целью исследования является разработка аппаратно-зависимого гипервизора на «пустом» аппаратном обеспечении без установленной операционной системы с фиксированной конфигурацией, запускающего три гостевые операционные системы.* ***Результаты:*** *написан гипервизор реального времени с микроядерной архитектурой, использующий технологию VT-d для проброса устройств в гостевые операционные системы и технологию VT-x для виртуализации процессора. Доказана возможность создания микроядерного гипервизора реального времени для жестко заданной аппаратной платформы с объемом исходных кодов менее 10 тыс. строк. Разработан и проверен метод и аппаратно-программное обеспечение для тестирования характеристик рельного времени программ.* ***Практическая значимость:*** *представленный подход к написанию гипервизора делает возможным создание компактного микрогипервизора рельного времени небольшой командой разработчиков. Предложенный метод тестирования характеристик реального времени позволяет автоматизировать этот процесс.*

Ключевые слова *— системное программное обеспечение, системы реального времени, системы с требованиями безопасности, симметричная многопроцессорность, виртуальные машины.*

Введение

Концепция виртуальной машины как совокупности ресурсов, которые эмулируют поведение реальной машины, появилась в Кембридже в конце 1960-х гг. С тех пор ее очевидные преимущества: повышение изоляции, безопасность, распределение ресурсов, постоянная доступность, повышение качества администрирования и т. д. [1] — из года в год получали дальнейшее развитие. Резкий толчок бурному развитию дала аппаратная поддержка виртуализации на самой популярной платформе x86, когда в середине 2000-х компании Intel и AMD анонсировали технологии VT-x и AMD-V соответственно. До этого события первой попыткой корпорации Intel внедрить в свои процессоры технологии аппаратной виртуализации был режим виртуального процессора 8086 в процессоре 80386, который появился в 1985 г. Возможность аппаратной виртуализации добавила к вышеупомянутым преимуществам еще целый ряд:

— упрощение разработки программных платформ виртуализации за счет предоставления аппаратных интерфейсов управления и поддержки виртуальных гостевых систем;

— возможность увеличивать быстродействие платформ виртуализации;

— улучшение защищенности, возможность переключения между несколькими запущенными независимыми платформами виртуализации на аппаратном уровне;

— запуск 64-битных гостевых систем на 32-битных хостовых системах.

К сожалению, стоит также отметить, что аппаратная виртуализация потенциально несет в себе не только положительные моменты. Возможность управлять гостевыми системами посредством гипервизора и простота написания платформы виртуализации с использованием аппаратных техник позволяют разрабатывать вредоносное программное обеспечение (ПО), которое после получения контроля над хостовой операционной системой (ОС) виртуализует ее и осуществляет все действия за ее пределами [2].

Программа, которая обеспечивает или позволяет одновременное, параллельное выполнение нескольких или даже многих ОС на одном и том же хост-компьютере, называется гипервизором, или монитором виртуальных машин. Общепринятая классификация гипервизоров [3] (рис. 1):

— автономный гипервизор (тип 1). Имеет свои встроенные драйверы устройств, модели драйверов и планировщик и поэтому не зависит от базовой ОС.

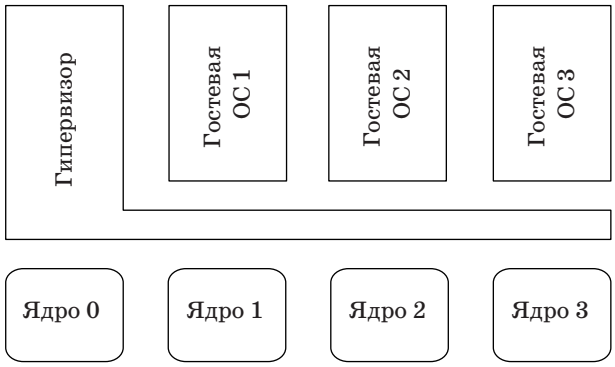
■ ***Рис. 1.*** Типы виртуализации

Так как автономный гипервизор работает непосредственно на оборудовании, то он более производителен [4]. Пример — VMware ESX;

— на основе базовой ОС (тип 2). Этот компонент работает в одном кольце с ядром основной ОС (кольцо 0). Гостевой код может выполняться прямо на физическом процессоре, но доступ к устройствам ввода-вывода компьютера из гостевой ОС осуществляется через второй компонент, обычный процесс основной ОС — монитор уровня пользователя. Примеры: Microsoft Virtual PC, VMware Workstation, QEMU, Parallels, VirtualBox;

— гибридный (тип 1+). Гибридный гипервизор состоит из двух частей: из тонкого гипервизора, контролирующего процессор и память, а также работающей под его управлением специальной сервисной ОС в кольце пониженного уровня [5]. Через сервисную ОС гостевые ОС получают доступ к физическому оборудованию. Примеры: Xen, Citrix XenServer, Microsoft Hyper-V.

В настоящее время существует, помимо упомянутых, огромное количество всевозможных гипервизоров, в том числе сертифицированных для промышленных приложений, например, с особыми требованиями к надежности, безопасности и пр. Такие разработчики, как Wind River Systems, Green Hills Software, SYSGO AG и т. д. — настоящие лидеры рынка, и угнаться за ними очень сложно. Чтобы избежать громоздкости универсальных продуктов и обеспечить возможность сертификации, было решено разработать гипервизор с фиксированной конфигурацией, запускающий три гостевые системы (рис. 2). Был выбран гипервизор 1-го типа с целью увеличить производительность системы. В качестве платформы взят четырехъядерный процессор Intel Core i5. Таким образом, мы избавились от бесчисленного множества нюансов, связанных с поддерживаемыми функциями процессоров, версиями и т. д., которые существенно усложняют ПО. Лишив продукт гибкости, мы получили возможность минимизировать объем кода гипервизора. В нашем понимании микрогипервизор



■ ***Рис. 2.*** Распределение гипервизора и гостевых ОС на выделенных ядрах для минимизации взаимного влияния на производительность

должен содержать не более 10 тыс. строк кода. Это позволяет упростить архитектуру ПО, сократить трудоемкость, срок разработки и облегчить сертификацию. Вот почему разработанный гипервизор характеризуется как микро, аппаратно-зависимый и автономный. Наша концепция направлена на то, что проще разрабатывать отдельные версии компонентного ПО гипервизора для каждой аппаратной конфигурации, нежели одну универсальную на всех.

Обзор аналогичных работ

Известны несколько продуктов, именуемых микрогипервизорами [1]: NOVA, OKL4, Codezero, XVisor. Рассмотрим некоторые из них более подробно.

NOVA (NOVA OS Virtualization Architecture) — это исследовательский проект, нацеленный на создание безопасной виртуализационной среды с малым объемом исходного кода [6]. NOVA состоит из микрогипервизора и пользовательской среды для базовых функций системы. Будучи микроядром третьего поколения, NOVA использует возможность авторизации на основе модели и предоставляет только базовые механизмы

виртуализации, пространственное и временное разграничение, планирование, коммуникацию и управление платформой ресурсов. Разделенная многосерверная среда реализует дополнительные сервисы ОС в режиме пользователя, такие как драйверы устройств, стеки протоколов и политики. На машинах с поддержкой аппаратной виртуализации NOVA может запустить несколько немодифицированных ОС одновременно. Каждая виртуальная машина имеет свой собственный монитор, который работает в качестве непривилегированного пользователя приложения поверх гипервизора.

Микровизор OKL4 разработки Open Kernel Labs [7] основан на концепции микроядра, основная идея которой состоит в том, чтобы уменьшить код ядра фундаментальных механизмов и реализации реальных системных служб на уровне пользователя серверов. Такой дизайн делает взаимодействие клиента и сервера критичным к производительности, поэтому микроядро требует очень быстрого механизма межпроцессорного взаимодействия. Микроядро должно быть достаточно общим, чтобы поддержать надстройку любых систем. Название «микровизор» отражает тот факт, что встроенный гипервизор реализуется на основе микроядра виртуализации L4 как его неотъемлемой подсистемы. Сравнение микроядерных и традиционных архитектур мониторов виртуальных машин как подходов для встраиваемых гипервизоров является предметом продолжающихся дебатов [8]. Микровизор OKL4 является микроядром L4 третьего поколения. Он широко используется в мобильных беспроводных устройствах в связи с растущим спросом на высокоэффективные платформы виртуализации во встраиваемых системах.

Встраиваемый гипервизор Codezero [1] — это новый гипервизор, основанный на архитектуре микроядра L4, но написанный «с нуля», чтобы воспользоваться преимуществами новейших исследований в микроядерной архитектуре. Он следует фундаментальным принципам микроядер в том, что реализует адресные пространства, управление потоками и межпроцессорным взаимодействием только в привилегированном микроядре, наряду с возможностями виртуализации. Codezero реализует типовой уровень абстракции над аппаратной платформой. Уровень абстракции реализует многопоточность, межпроцессорный обмен, адресное пространство управления, отображение адресного пространства, безопасность, питание и восстановление после ошибок управления.

Представляемый гипервизор вобрал в себя некоторые идеи из упомянутых выше. Однако мы подчеркиваем некоторые принципиальные отличия, одними из которых являются следующие:

— применяется экзоядро, которое допускает прямой доступ к аппаратным средствам, таким образом устраняя абстракции и сокращая издержки при обмене уровней;

— гипервизор предназначен для использования на конкретной платформе;

— каждое ядро и периферийные устройства жестко привязаны к какой-либо гостевой ОС;

— отсутствуют средства обмена между гостевыми ОС; гостевые ОС не видят друг друга — они полностью изолированы;

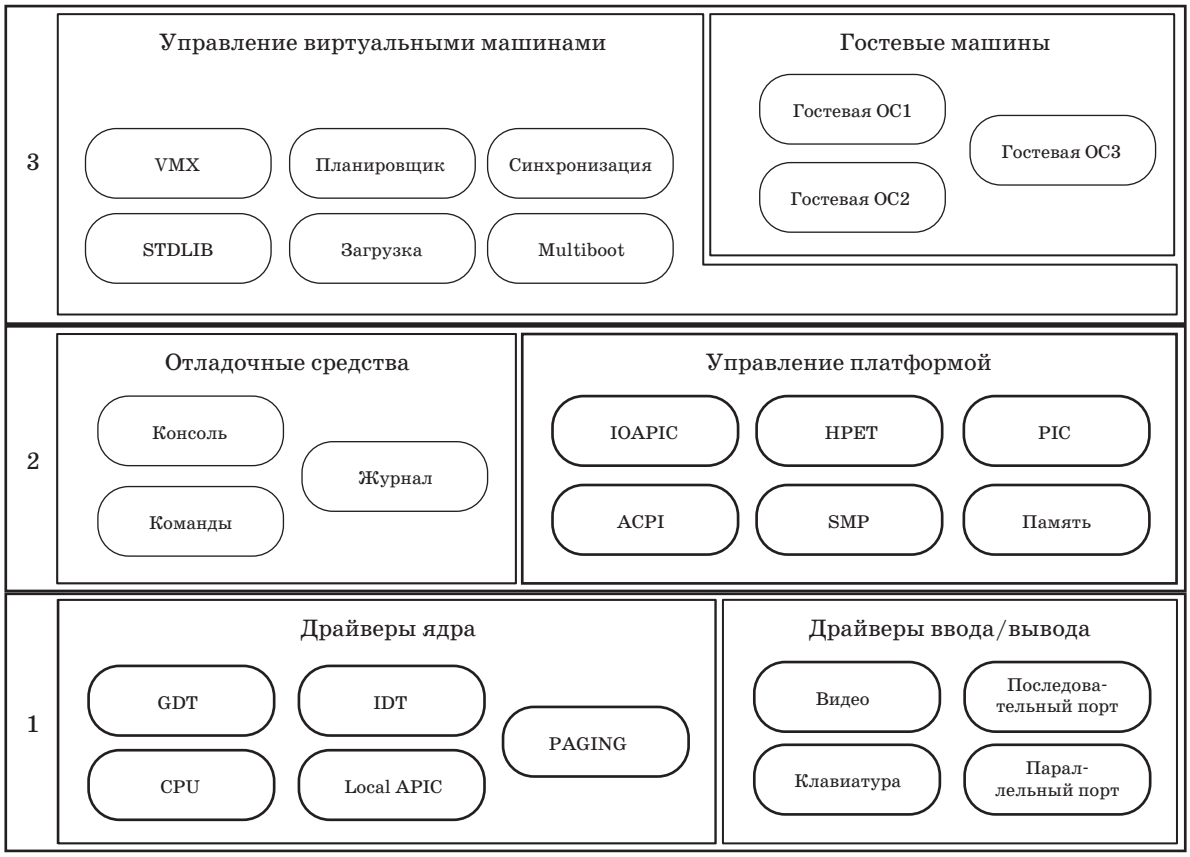
— гостевая ОС может исполняться в реальном времени, при этом ее поведение становится детерминированным.

Компоненты и принцип работы гипервизора

Архитектура

Архитектура гипервизора построена по компонентному принципу (рис. 3). Все компоненты имеют четко очерченные интерфейсы. Такой подход является очень важным, чтобы упростить модификации при переходе на другую платформу. Как уже говорилось, каждая версия нашего гипервизора работает только на определенной фиксированной аппаратной конфигурации. Это необходимо для того, чтобы ПО содержало только необходимый в текущий момент функционал. Чем проще — тем надежнее, проще верификация и сертификация.

Компоненты делятся на три слоя: *1* — низкоуровневые драйверы; *2* — управление платформой и отладочные утилиты; *3* — управление виртуализацией. Уровень драйверов подразделяется на два блока — ядро и ввод/вывод. В первый входит управление сегментированной и страничной памятью, таблицей прерываний, функциями ядра процессора и Local APIC. Ввод/вывод включает следующие устройства: графический дисплей, клавиатуру, последовательный и параллельный порты. Следующий уровень компонентов — это логическая надстройка над драйверами. HPET (high-precision event timer) используется для средств синхронизации и работы профилировщика. Модуль I/O APIC предназначается для конфигурации распределения прерываний по ядрам. Модуль ACPI позволяет получать информацию о конфигурации оборудования, реализовывать программный сброс и выключение. Модуль SMP имеет реализацию функций для запуска, приостановки и сброса ядер. Блок управления отладкой включает многооконную консоль (для каждого ядра), обработчик команд и ведения журнала (протоколирования). На самом верхнем уровне располагается слой компонентов менеджера виртуальных машин и самих гостевых машин.



■ ***Рис. 3.*** Компонентная архитектура микрогипервизора

На рисунке показаны только основные компоненты системы. На самом деле их намного больше, но даже несмотря на это система достаточно компактная за счет включения только самого необходимого. Еще раз повторим, что единственный минус такой архитектуры — полное отсутствие гибкости.

Загрузка и выполнение

Для загрузки гипервизора (рис. 4) [9], конфигурационного файла и образов гостевых систем мы используем multiboot-совместимые средства, такие как GRUB для загрузки с файловой системы компьютера или PXELINUX для загрузки по сети. Multiboot-загрузка является очень удобным средством, поскольку устраняет необходимость реализовывать файловую систему в ядре гипервизора для загрузки модулей. Также она автоматически переводит ядро в защищенный режим и выдает информацию о карте памяти. Загрузка осуществляется на так называемом boot-strap processor (BSP), назначаемом BIOS при старте системы. Как правило, это ядро с нулевым идентификатором. На BSP-ядре выполняется основной код гипервизора.

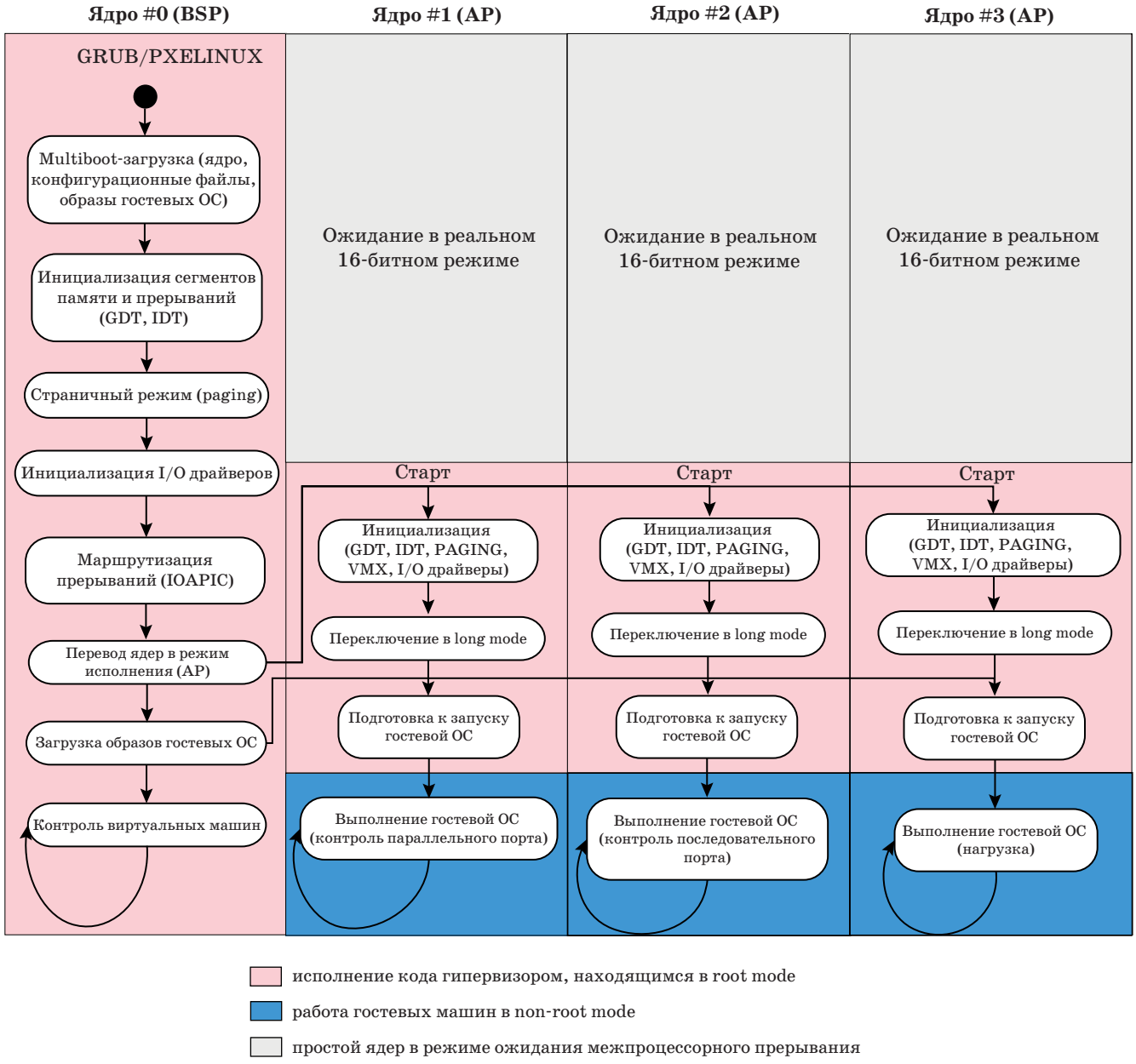
После серии инициализаций это ядро переводит остальные три ядра в режим исполнения кода, инициализирует их, переводит в 64-битный режим и запускает на виртуальных машинах загруженные образы ОС. Поскольку в SMP-системах все ядра являются равнозначными (одно ядро не может управлять другими), ПО гипервизора распределено между ядрами. После этого гипервизор переходит в режим контроля виртуальными машинами с помощью консольного терминала и средств ведения журнала.

После перехода процессора из PIC-mode в symmetric mode I/O APIC перенаправляет прерывания от параллельного и последовательного портов на 1-е и 2-е ядро соответственно. Виртуальные машины, запущенные на этих ядрах, используют интерфейсы портов в качестве внешних. Виртуальная машина на третьем ядре не имеет внешних интерфейсов и применяется только для загрузки ядра. Это необходимо для определения взаимного влияния ядер на производительность.

Карта памяти

Как известно, компьютеры x86-архитектуры имеют память с дырами (рис. 5) [10]. Это связано со свойственной ей традиционной обратной совместимостью.

Основная часть кода гипервизора лежит в области больше 1 МБ. В основной памяти располагается часть кода, необходимая для вызова 16-битных функций модуля SMP. Каждой виртуальной



■ ***Рис. 4.*** Диаграмма активности (каждая дорожка отображает временную ось ядра процессора)

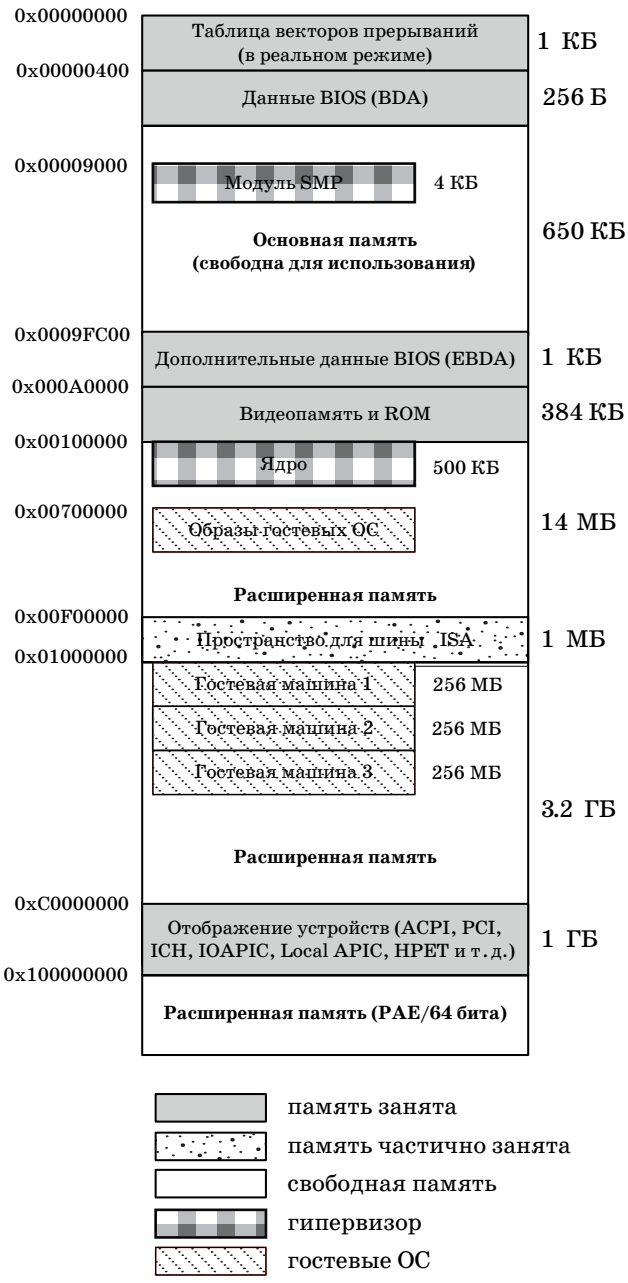
машине выделено по 256 МБ оперативной памяти. Эти области изолируются таким образом, что виртуальная машина не может получить доступ к другой области памяти. Образы виртуальных машин, загружаемые с помощью Multiboot [9], также располагаются в расширенной памяти.

Объем разработанного кода

Результирующие данные по разработанному коду показаны в табл. 1. Код содержит около сотни файлов. Основная его часть написана на Си с использованием ассемблерных функций. Код содержит обильные комментарии. И самое важное, что текущая функциональность уложилась в 8500 строк кода, что дает нам основание полагать, что код действительно можно характеризовать как микро.

■ ***Таблица 1.*** Код гипервизора

Параметр	Значение
Число файлов	96
Общее число строк	16 436
Число строк с комментариями	5094
Число пустых строк	2841
Покрытие кода комментариями	60 %
Общее число полезных строк кода	8501

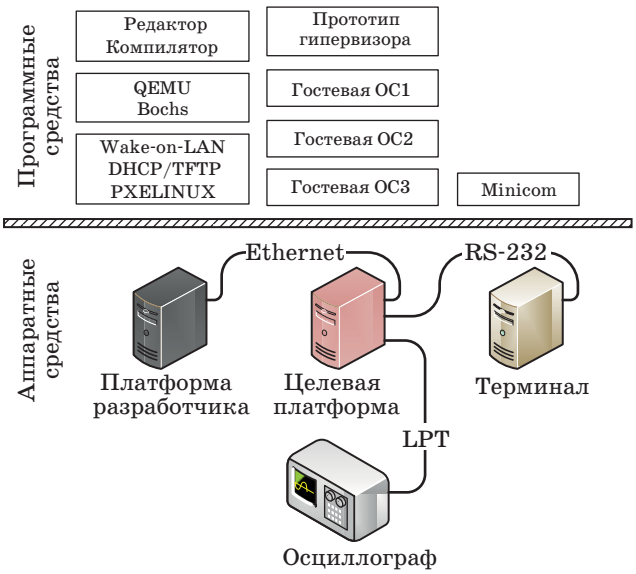


■ ***Рис. 5***. Карта памяти используемой платформы

Тестирование

Стенд для разработки и отладки

Для разработки и отладки гипервизора мы использовали стенд, схема которого представлена на рис. 6. Мы используем три компьютера: один — для разработки, второй является целевой платформой, и третий — в качестве терминала последовательного порта. Параллельный порт целевой платформы подключен к осциллографу для измерения задержки обработки прерывания. В верхней части рисунка показан перечень ПО, которое используется на соответствующих ком-



■ ***Рис. 6***. Схема стенда разработки и отладки

пьютерах. На компьютере разработчика установлены симуляторы QEMU и Bochs. Для отладки на целевой платформе используется загрузка по сети (PXE) с помощью утилиты PXELINUX.

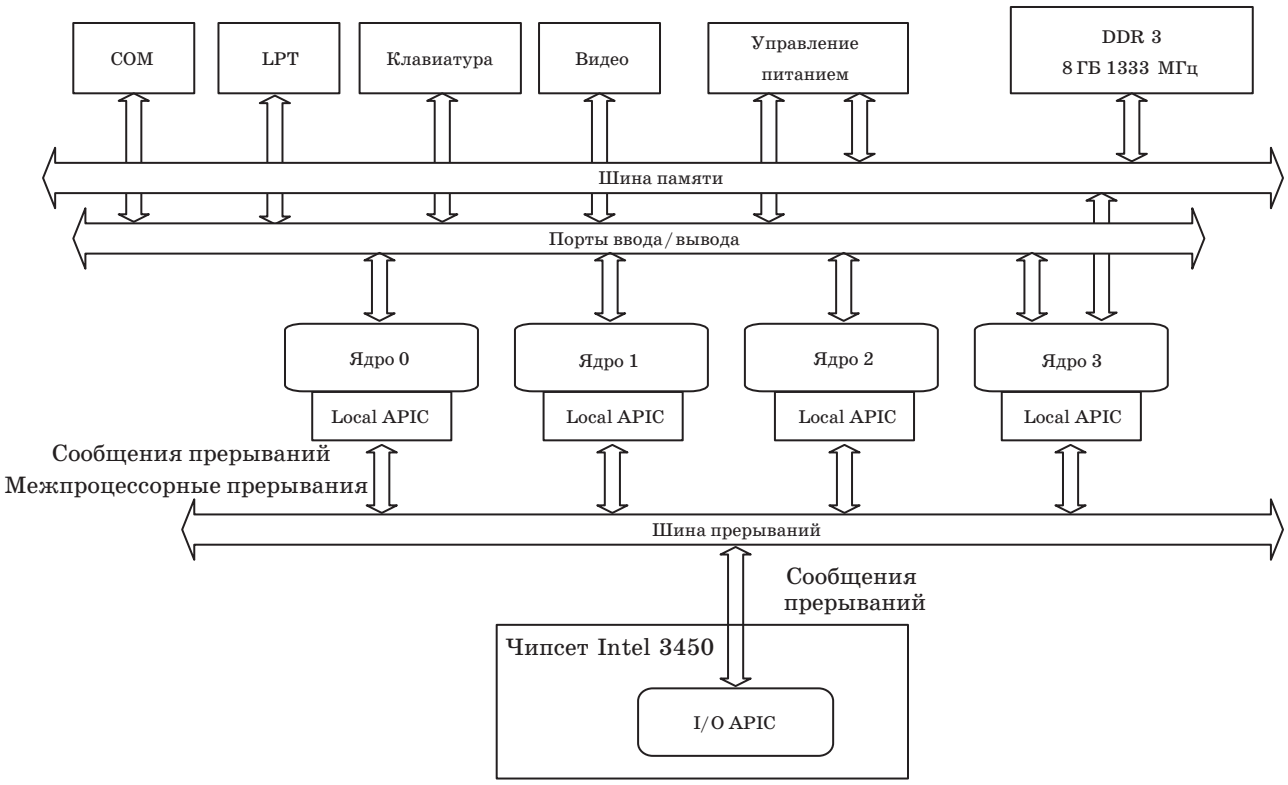
Как известно, все ядра в симметричной многопроцессорной системе равноправны (рис. 7) [11]. Единственное отличие заключается в том, что ядро, загружаемое BIOS, помечается флагом как boot-strap processor. У каждого ядра имеется встроенный local APIC [12], который соединяется со специальной шиной прерываний. По этой шине ядра получают прерывания от I/O APIC [13], отвечающего за routing прерываний, а также имеют возможность генерировать межпроцессорные прерывания. Последние используются для синхронизации, инициализации ядра и запуска на нем процедур с указанного адреса. Помимо шины прерываний, процессоры имеют соединения еще с двумя шинами — шиной памяти и шиной портов ввода/вывода. Через порты ввода/вывода процессор имеет возможность управлять периферийными устройствами. Шина памяти используется для доступа к RAM и регистрам, которые отображают в память PCI, HPET, ACPI, I/O APIC, Local APIC и другие устройства [14].

Основные характеристики тестового стенда:

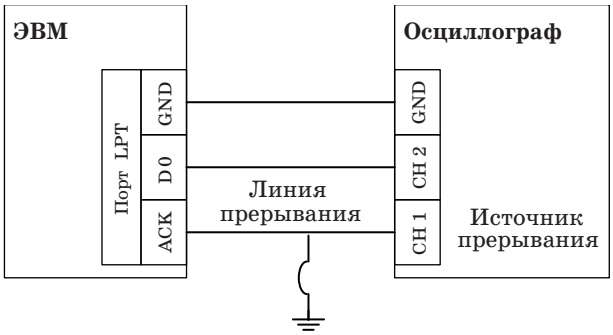
— ЦПУ: Intel Core i5 650 @ 3.20 Гц x 4;
— чипсет: Intel 3450;
— память: 8 ГБ DDR3 1333 МГц.

Задержка обработки прерывания

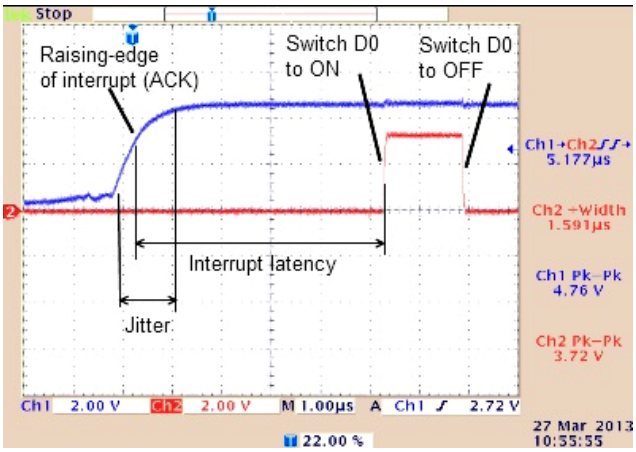
В рамках этого проекта измерение задержки обработки прерывания осуществляется с помощью параллельного порта LPT. Схема эксперимента изображена на рис. 8. Используются три вывода параллельного порта: заземление (GND),



■ ***Рис. 7.*** Симметричная многопроцессорность, используемая в работе



■ ***Рис. 8.*** Схема измерения задержки отклика на прерывание



■ ***Рис. 9.*** Экран осциллографа

данные (D0) и подтверждение (ACK). Последний служит для выдачи сигнала прерывания IRQ 7 ввода/вывода APIC в контроллере. При замыкании линии прерывания на землю (нисходящий фронт) вызывается обработчик прерывания, который последовательно выключает и включает D0. Прерывание также возникает при отсоединении контакта от земли (передний фронт). Передний фронт используется для запуска осциллографа. Все события отображены на экране осциллографа (рис. 9). В среднем достигаются задержки на прерывания 5 мкс с разбросом 0,3 мкс. Природа разброса, по-видимому, связана с некоторой емкостью в кабеле и логикой срабатывания в контроллере прерывания, что видно из плавно восходящего фронта прерывания.

Загрузка и средства отладки

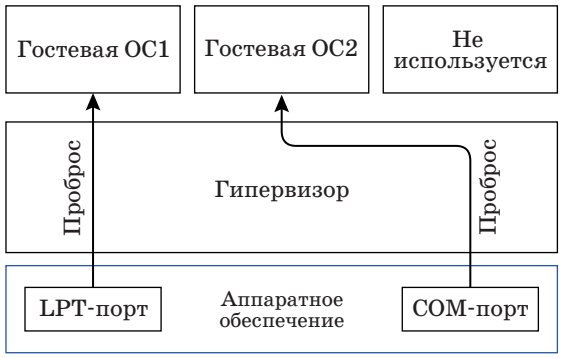
Самое сложное в данном проекте заключается в том, что ПО разрабатывается на «пустом» аппаратном обеспечении — отсутствует ОС. Одновременно возникают проблемы с тестированием потому, что многократная перезагрузка компьютера при отладке — это очень трудоемкая

работа. К счастью, в настоящее время можно до определенного этапа разработки использовать эмуляторы QEMU.

Во время исследования были опробованы различные средства для отладки (табл. 2). Как показала практика, наиболее удобным и достаточно адекватным средством оказался эмулятор QEMU. Однако, к сожалению, пользоваться им можно только до тех пор, пока можно обойтись 64-битным режимом и инструкциями VMX, поскольку они не поддерживаются. Затем на помощь приходит Bochs. Разработка и отладка с помощью симуляторов существенно ускоряются. Тем не менее требуются периодические запуски на реальном стенде. Это — процесс гораздо более медленный. Запуск программы по сети с помощью PXE в нашем случае, например, длится 35 с. Причем для следующей попытки нужна перезагрузка целевой машины.

Автоматизированное тестирование характеристик реального времени

Тестирование проводилось на машине на базе четырехъядерного Intel Core i5. Одно из ядер процессора занято выполнением программного кода гипервизора, остальные выполняют тестовые виртуальные машины. Гипервизор сконфигурирован таким образом, что прерывания поступают напрямую к ядрам. Разрешены 2 вида прерываний: прерывание на порте LPT и прерывание на порте COM. Первое сразу же направляется на 2-е ядро, а второе — на 3-е. В данной работе не рассматривается работа с COM-портом и третьим ядром. Для задач тестирования используются LPT-порт и 2-е ядро (рис. 10).



■ ***Рис. 10***. Конфигурация тестируемой системы

Аппаратный профилировщик

Для нужд профилирования было разработано специальное ПО. Аппаратный профилировщик базируется на стандартной плате отладки фирмы Xilinx. Плата построена на основе ПЛИС Virtex-5. На ней размещены различные коммуникационные интерфейсы. В данной работе использовались только Ethernet, COM- и LPT-порты платы. Плату можно смонтировать в корпус тестируемой системы посредством порта PCI-e, что даст возможность обращаться к плате как к стандартному RAM-контроллеру. Смонтированный таким образом профилировщик может генерировать прерывания и различную активность на портах ввода/вывода. В свою очередь, данная активность может перенаправляться одному из ядер для обработки в ПО. То же самое может быть получено при использовании LPT-порта.

Сценарий тестирования состоит из трех фаз: 1-я фаза — профилировщик генерирует прерывание и запускает высокоточный таймер; 2-я фа-

■ ***Таблица 2***. Сравнение средств симуляции и загрузки

Средство	Время загрузки, с	Преимущества	Недостатки
PXE — средство для загрузки компьютера по сети	35	Использование реального аппаратного обеспечения	Долгое время загрузки; требуется поддержка в PXE на загружаемой машине; поддерживается только TFTP; требуется дополнительный компьютер
iPXE — полноценный загрузчик по сети	45	Использование рельного аппаратного обеспечения; не требуется поддержка PXE; поддерживаются многие сетевые протоколы	Долгое время загрузки; требуется дополнительный компьютер
QEMU — монитор виртуальных машин уровня пользователя (поддерживается KVM)	1	Возможность отладки эмулируемой системы; быстрая загрузка; не требуется дополнительный компьютер	Long mode и VMX не поддерживаются; неправильная эмуляция некоторых операций
Bochs — эмулятор и отладчик архитектуры x86 и x86_64	10	Возможность отладки эмулируемой системы; не требуется дополнительный компьютер; VMX и Long mode поддерживаются	Долгое время загрузки; сложность использования; неправильная эмуляция некоторых операций

за — профилировщик переходит в состояние ожидания реакции тестируемой системы; в момент прихода отклика тестируемой системы таймер на профилировщике останавливается и время реакции записывается — 3-я фаза. Затем тест повторяется для получения статистически обоснованного результата. Интервал между тестами может быть произвольным.

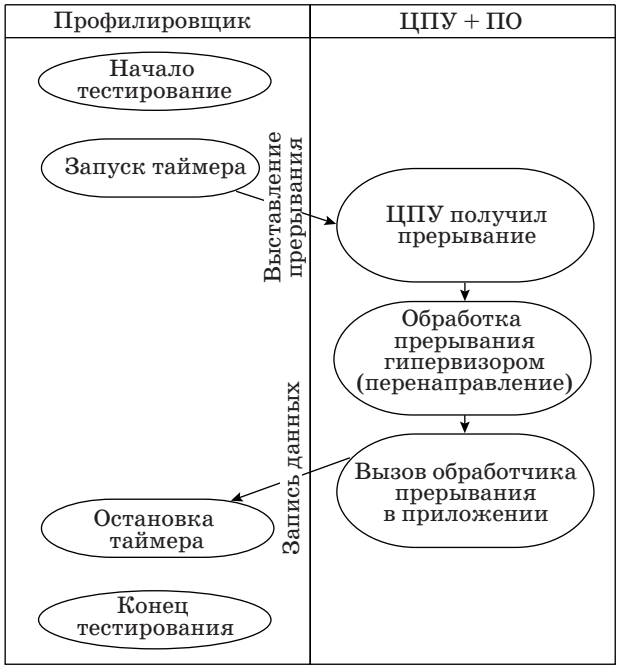
Разрешение измерений таймера — 50 нс; максимальное время измерения — 1 мкс. Для построения гистограммы профилировщик хранит во внутренней памяти последние 256 событий с временем реакции на каждое. Также профилировщик считает максимальное и минимальное время реакции для каждого типа события.

Через присутствующий на плате профилировщика сетевой порт Ethernet пользователь может получить доступ ко всем хранящимся данным измерений.

Последовательность тестирования

Согласно конфигурации и сценарию тестирования (рис. 11), профилировщик генерирует прерывание на LPT-порте тестируемой системы. На это прерывание реагирует программный обработчик, запущенный на тестируемой системе. Его ответ выражается в изменении состояния одного из битов порта LPT, что вызовет изменение уровня сигнала на соответствующем контакте данного порта. Время от момента генерирования прерывания до получения «ответа» называется временем отклика.

Профилировщик хранит два вида событий — соответствующих переднему и заднему фронту сигнала. При ответе обработчик сигнала инвертирует 1 бит порта LPT.



■ ***Рис. 11.*** Этапы тестирования

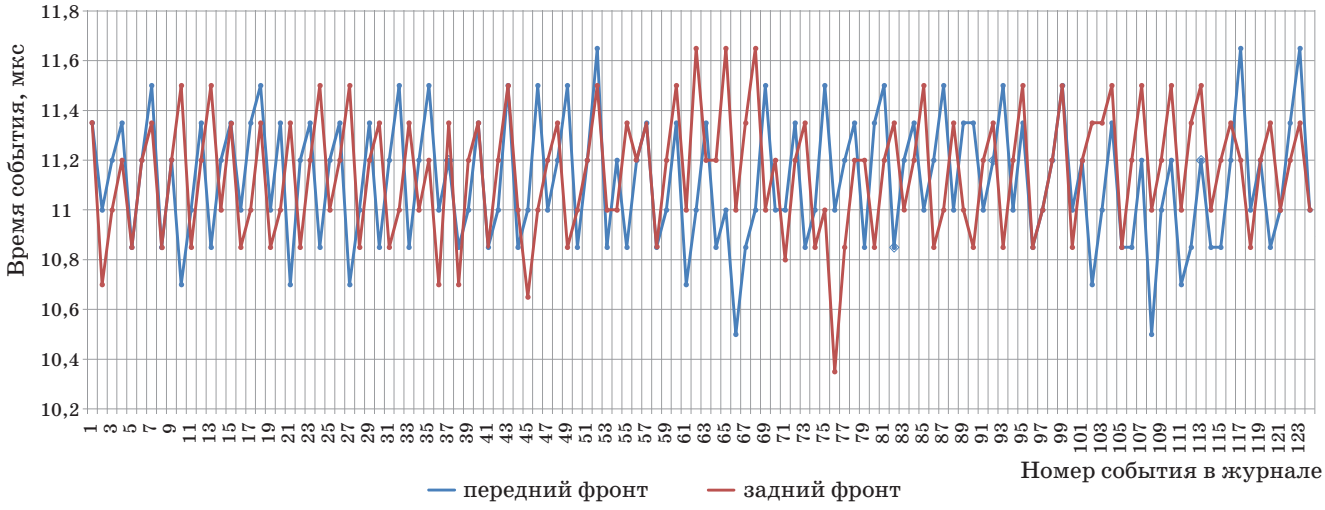
Результаты тестирования

Записанные профилировщиком события показаны на рис. 12.

Максимальное и минимальное время отклика составило, соответственно, 11,65 и 10,7 мкс для обоих типов событий, среднее время отклика — 11,1 мкс.

Заключение

В данной работе обоснована необходимость разработки аппаратно-зависимого автономного мик-



■ ***Рис. 12.*** События по времени отклика

рогипервизора, позволяющего запускать одновременно три виртуальные машины на отдельных ядрах четырехъядерного PC с x86-архитектурой. Показано, как компонентная модель архитектуры микрогипервизора позволяет уложиться в 8500 строк кода. Также замерен один из основных показателей систем реального времени — задержка обработки прерывания — для ISA-устройства, которая равна 11,1 мкс. В настоящее время идут работы над реализацией технологии виртуализации PCI-устройств VT-d; планируется адаптация кода аппаратно-зависимого микрогипервизора для восьмиядерной платформы Intel Core i7.

Литература

1. **Jones M. T.** Virtualization for embedded systems. The how and why of small-device hypervisors // Developer Works. 2011. N 4. http://www.ibm.com/developerworks/library/l-embedded-virtualization/ (дата обращения: 13.01.2014).
2. **King S. T.** et al. SubVirt: Implementing Malware with Virtual Machines // Proc. of the 2006 IEEE Symp. on Security and Privacy SP '06. Washington, DC, USA: IEEE Computer Society, 2006. P. 314–327.
3. **Hypervisor.** http://en.wikipedia.org/wiki/Hypervisor (дата обращения: 11.01.2014).
4. **Iqbal A., Sadeque N., Mutia R.** An Overview of Microkernel, Hypervisor and Microvisor Virtualization Approaches for Embedded Systems // DEITLU. 2009. N 5. P. 1–15.
5. **VMMs** versus hypervisors. http://blogs.msdn.com/b/virtual_pc_guy/archive/2006/07/10/661958.aspx (дата обращения: 14.01.2014).
6. **Steinberg U., Kauer B.** A Microhypervisor-based Secure Virtualization Architecture // Proc. of the 5th European Conf. on Computer Systems, EuroSys '10. N. Y., USA: ACM, 2010. P. 209–222.
7. **Heiser G., Leslie B.** The OKL4 Microvisor: Convergence Point of Microkernels and Hypervisors // Proc. of the First ACM Asia-Pacific Workshop on Workshop on Systems, APSys '10. N. Y., USA: ACM, 2010. P. 19–24.
8. **Hand S.** et al. Are Virtual Machine Monitors Microkernels Done Right? // Proc. of the 10th Conf. on Hot Topics in Operating Systems — Volume 10, HOTOS'05. Berkeley, CA, USA: USENIX Association, 2005. P. 1–2.
9. **Multiboot** Specification version 0.6.96. http://www.gnu.org/software/grub/manual/multiboot/multiboot.html (дата обращения: 15.11.2013).
10. **Combined** Volume Set of Intel® 64 and IA-32 Architectures Software Developer's Manuals. http://download.intel.com/products/processor/manual/325462.pdf (дата обращения: 05.11.2013).
11. **MultiProcessor** Specification v1.4. http://developer.intel.com/design/pentium/datashts/24201606.pdf (дата обращения: 05.11.2013).
12. **Advanced** Configuration and Power Interface Specification, Revision 5.0. http://www.acpi.info/DOWNLOADS/ACPIspec50.pdf (дата обращения: 05.11.2013).
13. **Intel** 82093AA I/O Advanced Programmable Interrupt Controller (I/O APIC). http://download.intel.com/design/chipsets/datashts/29056601.pdf (дата обращения: 05.11.2013).
14. **IA-PC** HPET (High Precision Event Timers) Specification, revision 1.0a. http://www.intel.ua/content/dam/www/public/us/en/documents/technical-specifications/software-developers-hpet-spec-1-0a.pdf (дата обращения: 07.11.2013).

UDC 004.45

The Architecture of Bare-Metal Real-Time Microhypervisor and Automated Measurement of Time Response

Kolchin I. V.[a], PhD, Tech., Leading Engineer, ivan.kolchin@siemens.ru
Filippov S. N.[a, b], Post-Graduate Student, Junior Engineer, filippov_sergey@lenta.ru
[a]Siemens LLC, Corporate Technology, 3A, Volynskii St., 191186, Saint-Petersburg, Russian Federation
[b] Saint-Petersburg State University of Aerospace Instrumentation, 67, B. Morskaia St., 190000, Saint-Petersburg, Russian Federation

Purpose: Hypervisors and virtual machines have become popular in the recent decade due to their indisputable advantages. But there is a reverse side of this achievement especially for industrial companies which are engaged into development of safety-critical systems. Software becomes too complicated to be compatible with all possible versions and configurations of hardware. As a result it is difficult to certify this software for its compliance with safety standards such as IEC 61508. The purpose of the research is to develop a hardware-dependent bare-metal hypervisor which can lunch 3 guest operating systems. **Results:** There has been developed a real-time hypervisor with microkernel architecture which uses VT-d technology to pass through devices to guest operating systems and VT-x technology to virtualize a processor. There has been proven a possibility to develop a real-time microhypervisor for a given hardware platform with a source code comprising less than 10000 lines. There has been developed and checked a method and software/hardware for testing real-

time characteristics of software. **Practical relevance:** The given method to develop a hypervisor makes it possible to work out a bare-metal hardware specific real-time portative microhypervisor in a short time period employing a small team of developers. The proposed method for testing real-time characteristics allows conducting this process automatically.

Keywords — Bare Metal Software, Microhypervisor, Real-Time Systems, Safety-Critical Systems, Symmetric Multiprocessing, Virtual Machines.

References

1. Jones M. T. Virtualization for Embedded Systems. The How and Why of Small-Device Hypervisors. *Developer Works,* 2011, no. 4. Available at: http://www.ibm.com/developerworks/library/l-embedded-virtualization/ (accessed 13 January 2014).
2. King S. T., Chen P. M., Wang Y. M., Verbowski C., Wang H. J., Lorch J. R. SubVirt: Implementing Malware with Virtual Machines. *Proc. of the 2006 IEEE Symp. on Security and Privacy SP '06*. Washington, DC, USA, IEEE Computer Society, 2006, pp. 314–327.
3. *Hypervisor.* Available at: http://en.wikipedia.org/wiki/Hypervisor (accessed 11 January 2014).
4. Iqbal A., Sadeque N., Mutia R. An Overview of Microkernel, Hypervisor and Microvisor Virtualization Approaches for Embedded Systems. *DEITLU*, 2009, vol. 5, pp. 1–15.
5. *VMMs versus hypervisors*. Available at: http://blogs.msdn.com/b/virtual_pc_guy/archive/2006/07/10/661958.aspx (accessed 14 January 2014).
6. Steinberg U., Kauer B. A. Microhypervisor-based Secure Virtualization Architecture. *Proc. of the 5th European Conf. on Computer Systems EuroSys '10*. New York, NY, USA, ACM, 2010, pp. 209–222.
7. Heiser G., Leslie B. The OKL4 Microvisor: Convergence Point of Microkernels and Hypervisors. *Proc. of the First ACM Asia-pacific Workshop on Workshop on Systems APSys '10*. New York, NY, USA, ACM, 2010, pp. 19–24.
8. Hand S., Wareld A., Fraser K., Kotsovinos E., Magenheimer D. Are Virtual Machine Monitors Microkernels Done Right? *Proc. of the 10th Conf. on Hot Topics in Operating Systems — Volume 10 HOTOS'05*. Berkeley, CA, USA, USENIX Association, 2005, pp. 1–2.
9. *Multiboot Specification version 0.6.96*. Available at: http://www.gnu.org/software/grub/manual/multiboot/multiboot.html (accessed 15 November 2013).
10. *Combined Volume Set of Intel® 64 and IA-32 Architectures Software Developer's Manuals*. Available at: http://download.intel.com/products/processor/manual/325462.pdf (accessed 05 November 2013).
11. *MultiProcessor Specification v1.4*. Available at: http://developer.intel.com/design/pentium/datashts/24201606.pdf (accessed 05 November 2013).
12. *Advanced Configuration and Power Interface Specification, Revision 5.0*. Available at: http://www.acpi.info/DOWNLOADS/ACPIspec50.pdf (accessed 05 November 2013).
13. *Intel 82093AA I/O Advanced Programmable Interrupt Controller (I/O APIC)*. Available at: http://download.intel.com/design/chipsets/datashts/29056601.pdf (accessed 05 November 2013).
14. *IA-PC HPET (High Precision Event Timers) Specification, revision 1.0a*. Available at: http://www.intel.ua/content/dam/www/public/us/en/documents/technical-specifications/software-developers-hpet-spec-1-0a.pdf (accessed 07 November 2013).