УДК 004.414.23

МОДЕЛИРОВАНИЕ РАБОТЫ УПРАВЛЯЮЩИХ МНОГОМАШИННЫХ КОМПЛЕКСОВ В СИСТЕМЕ ВИРТУАЛЬНЫХ МАШИН

Р.Э. Асратян

Рассмотрен подход к построению действующих программных макетов управляющих многомашинных вычислительных комплексов (МВК) на основе технологии систем виртуальных машин. Суть подхода заключается в представлении МВК в форме совокупности параллельно работающих и взаимодействующих между собой виртуальных машин, организованных в достаточно мощной хост-машине. Цель построения макета — создание инструментальной среды для отладки и тестирования программного обеспечения МВК. Описаны принципы организации взаимодействия виртуальных машин в макете и параллелизма их работы в модельном времени.

Ключевые слова: системы виртуальных машин, моделирование вычислительных систем.

ВВЕДЕНИЕ

В последние годы наблюдается несомненный рост интереса пользователей и разработчиков к технологии виртуальных машин, принципы которой были разработаны и реализованы еще в 1980-х гг. [1]. Подтверждением тому служит появление на рынке ряда продуктов [2—4], открывающих для пользователей современных ЭВМ уникальные возможности этой технологии:

— организации в одной реальной машине («хост-машине») сразу нескольких виртуальных машин — логических образов реальных компьютеров, с высокой степенью точности отображающих архитектуру хост-машины;

— загрузки в каждую виртуальную машину собственной операционной системы («гостевой» системы), которая может отличаться от операционной системы хост-машины («хост-системы»);

— запуска в виртуальных машинах приложений, совместимых с гостевыми системами.

Из публикаций, посвященных данной теме [2—7], можно заключить, что наиболее важными направлениями применения этой технологии обычно считаются:

— преодоление несовместимости приложений с хост-системами;

— экспериментирование с различными гостевыми системами;

— объединение преимуществ различных гостевых систем в одной машине;

— защита хост-системы от опасных приложений и вирусных атак.

В данной статье рассматривается еще одно применение технологии виртуальных машин — создание действующих макетов многомашинных вычислительных комплексов (МВК) в одной хост-машине. Под действующим макетом МВК понимается его логический образ, воспроизводящий его архитектуру с такой степенью точности, которая дает возможность проводить разработку и отладку программного обеспечения (системного и прикладного) для МВК в этом макете.

Данная статья написана на основе опыта автора, полученного при разработке программного обеспечения для специализированных управляющих мультимикромашинных комплексов на базе отечественных микропроцессоров 1801 (архитектура PDP-11) и 1839 (архитектура VAX-11) в конце 1980-х и начале 1990-х гг. Важной составной частью этой работы было создание двух версий инструментальной системы виртуальных машин (ИСВМ) для PDP-11 и VAX-11 соответственно, предназначенных для проведения отладки программного обеспечения целевых МВК в одной достаточно мощной хост-машине [8, 9].

1. НАЗНАЧЕНИЕ И ПРИНЦИПЫ ОРГАНИЗАЦИИ ИНСТРУМЕНТАЛЬНОЙ СИСТЕМЫ ВИРТУАЛЬНЫХ МАШИН

Инструментальная система виртуальных машин предназначалась для моделирования работы специализированных управляющих микропроцессорных МВК реального времени, обеспечивающих управление внешним оборудованием в «безлюдном» режиме. Такие МВК часто составляют единое целое с внешним оборудованием в конструктивном плане и, поэтому, называются «встроенными». Центральная часть их структуры обычно включает несколько микро-ЭВМ, соединенных одним или несколькими высоконадежными магистральными каналами (МК) со строго регулируемым доступом (например, в стандарте MIL-1553 или MIL-1773) [10]. Аппаратная архитектура отдельных микро-ЭВМ включает в себя центральный процессор, несколько модулей памяти (в том числе модуль постоянной энергонезависимой памяти для хранения программ) и один или несколько адаптеров МК.

Главная архитектурная особенность встроенных МВК заключается в составе периферийных устройств. Он практически «не пересекается» с составом периферийных устройств универсальных компьютеров. Такие привычные для всех компоненты, как клавиатура, видео и Ethernet-адаптеры, мышь и даже жесткие диски и их контроллеры, обычно отсутствуют во встроенных МВК. Верно и обратное: входящее в состав встроенных МВК периферийное оборудование (датчики, исполнительные механизмы, адаптеры МК), как правило, отсутствует в универсальных компьютерах. Именно для моделирования таких МВК была разработана и применялась ИСВМ.

Программное обеспечение (ПО) для встроенных МВК часто разрабатывается «с нуля» и включает в себя компактную встроенную операционную систему (ОС) реального времени и фиксированный набор циклически решаемых прикладных задач.

Одной из главных в подобных разработках является проблема комплексной отладки целевого ПО, вернее, проблема создания вычислительной среды, в которой можно было бы проводить отладку и тестирование программ. Это связано с тем, что стоимость и время подготовки натурных макетов (или отладочных стендов) МВК весьма велики, а набор отладочных инструментов в таких макетах обычно весьма ограничен. Если учесть высокие требования к качеству программ для управляющих МВК, то можно заключить, что создание отладочной среды представляет собой едва ли не самое «узкое место» в их разработках.

Инструментальная система виртуальных машин позволяет создавать действующие макеты МВК в форме совокупности виртуальных машин, функционирующих в одной хост-машине на принципах разделения времени. Под виртуальной машиной (ВМ) понимается программный «образ» (макет) реальной ЭВМ. Он включает в себя совокупность виртуальных ресурсов, отображающих все существенные для программиста ресурсы реальной ЭВМ, а также набор программ-обработчиков этих ресурсов. Главное требование к ВМ можно сформулировать так: целевое ПО (системное и прикладное) должно «чувствовать себя» в ней так же, как в реальной машине.

Существует определенная аналогия между ВМ и «задачей» в многозадачной ОС, а также между ВМ и машиной-эмулятором (например, JAVA-машиной): все они представляют собой среду для выполнения программ. Однако существует и ряд фундаментальных различий.

- С понятием «задачи» не связывается ряд ресурсов и механизмов реальной ЭВМ: система прерываний, адресное пространство ввода-вывода, привилегированные регистры процессора и т. п. Поэтому бессмысленно пытаться, например, запустить ОС в задачной среде.
- В машинах-эмуляторах выполнение программ полностью «изолировано» от аппаратуры хост-машины: каждая инструкция кода интерпретируется программно. В ВМ же лишь механизмы обработки виртуальных ресурсов эмулируются программно (прерывания, выполнение привилегированных инструкций и т. п.), в то время как выполнение большинства инструкций реализуется непосредственно в процессоре хост-машины. Разумеется, необходимое условие применимости данного подхода заключается в идентичности или близости архитектур хост-машины и микро-ЭВМ, входящих в МВК.

Сходство между задачей и ВМ обусловливается еще и механизмом разделения времени в хост-системе. С каждой задачей в многозадачной ОС связывается управляющий блок, в котором сохраняется программный «контекст» и состояние задачи. Очереди (списки) этих блоков служат основой для механизма попеременной загрузки активных задач в процессор — так называемой диспетчеризации задач. Точно так же каждой ВМ в ИСВМ соответствует ее управляющий блок (VMCB — Virtual Machine Control Block), в котором также сохраняется программный «контекст» и состояние ВМ. Разница заключается в том, что в VMCB содержатся еще и образы ресурсов реальной ЭВМ (или указатели на эти образы). По аналогии с диспетчером задач диспетчер ВМ постоянно поддерживает указатель на VMCB «текущей» ВМ, т. е. ВМ, загруженной в хост-систему в данный момент.

2. ПРИНЦИПЫ ТЕХНОЛОГИИ ВИРТУАЛЬНЫХ МАШИН

В самой общей форме соотношение реальных ресурсов хост-машины и виртуальных ресурсов ВМ проиллюстрировано на рис. 1, из которого видно, что множество ресурсов реальной ЭВМ условно разбивается на две группы: «пользовательских» и «системных» ресурсов. К первой из них относят «обычные» ресурсы машины (регистры общего назначения, счетчик команд, слово состояния процессора), состояния которых легко сохраняются и восстанавливаются в программном «контексте» (например, с помощью инструкций LDPCTX и SVPCTX). Наоборот, к группе «системных» относят ресурсы, состояния которых или невозможно, или слишком сложно оперативно сохранять и восстанавливать (векторы прерываний, регистры внешних устройств из области ввода-вывода, ряд «привилегированных» регистров процессора и т. п.). В контексте ВМ организуются «образы» этих ресурсов (виртуальные ресурсы), а все операции с ними реализуются в режиме программной эмуляции.

Поясним это на следующем примере. Одним из важнейших системных механизмов реальной ЭВМ является механизм прерываний. В нем задействованы такие ресурсы, как векторы прерываний, слово состояния процессора, системный стек и указатель системного стека. Для отображения этого механизма в адресном пространстве каждой ВМ должен быть организован собственный массив векторов прерываний и область системного стека. Образы регистра-указателя системного стека, слова состояния программы и, возможно, привилегированного регистра, хранящего адрес массива векторов прерываний (SCBR), должны содержаться в VMCB. При возникновении прерывания в хост-машине программа обработки прерывания должна провести эмуляцию механизма прерывания в текущей ВМ. Эта эмуляция включает запись в системный стек ВМ слова состояния процессора и значения счетчика команд (с коррекцией значения образа регистра-указателя системного стека), формирование образа нового слова состояния и «отбрасывание» управления в ВМ по адресу, содержащемуся в соответствующем векторе прерывания ВМ.

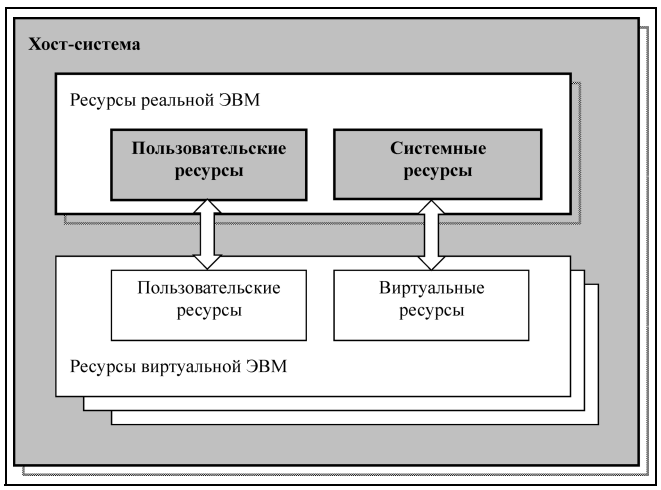

Рис. 1. Ресурсы реальной и виртуальной ЭВМ

В данной работе мы не будем подробно останавливаться на вопросах организации работы отдельных ВМ в хост-системе, но попытаемся рассмотреть механизмы отображения параллельной работы ЭВМ в МВК и взаимодействия между ВМ.

3. ВЕДЕНИЕ МОДЕЛЬНОГО ВРЕМЕНИ

Все ВМ в макете МВК подразделяются на два типа:

— рабочие ВМ, представляющие собой образы реальных микроЭВМ в макете МВК, в которые загружается и в которых отлаживается и тестируется целевое ПО;

— служебные ВМ, представляющие собой среду, в которую загружаются и в которой выполняются эмуляторы внешних устройств МВК — важнейшая составная часть макета.

Несмотря на то, что рабочие и служебные ВМ решают совершенно разные задачи, они обрабатываются общим механизмом диспетчеризации, обеспечивающим поочередную передачу управления всем активным ВМ и равномерное «продвижение», всех ВМ в модельном времени.

Модельное время — важнейшая характеристика ИСВМ. Необходимость отображения параллельной работы нескольких микро-ЭВМ в МВК, а также эмуляторов специфических внешних устройств (датчиков, исполнительных механизмов, адаптеров МК), попросту отсутствующих в хост-машине, однозначно приводит к необходимости ведения модельного времени (как и во всех системах программного моделирования реальных процессов).

Механизм ведения модельного времени в ИСВМ основан на следующих принципах.

- В специальной переменной IsvmTime ведется общий счетчик модельного времени в сотых долях микросекунды, начиная с момента начала работы макета МВК.
- В каждой ВМ ведется аналогичный собственный счетчик модельного времени VmTime, размещенный в VMCB.
- Инкрементация собственного счетчика VmTime в рабочих ВМ производится автоматически после выполнения каждой инструкции в ВМ с помощью

механизма покомандного прерывания (Т-прерывания) в хост-системе.

• Инкрементация собственного счетчика VmTime в служебных ВМ производится после выполнения специальной макрокоманды IsvmSleep, позволяющей перевести служебную ВМ в состояние ожидания на указанный интервал модельного времени. Автоматическая инкрементация счетчика VmTime после выполнения каждой инструкции в служебных ВМ не производится (их работа выполняется в режиме отключенного Т-прерывания в хост-машине).

• Инкрементация общего счетчика модельного времени IsvmTime производится в тот момент, когда собственные счетчики всех ВМ начинают превышать его значение.

Механизм ведения модельного времени неотделим от механизма диспетчеризации ВМ, который базируется на двух списках VMCB, первый из которых соответствует активным ВМ, а второй — ВМ, находящимся в состоянии ожидания окончания интервала модельного времени (VMCB в этом списке упорядочены по возрастанию момента окончания интервала, т. е. по возрастанию значения VmTime). Диспетчеризация активных ВМ ведется по «круговому» принципу: каждая ВМ получает управление на фиксированный квант модельного времени, после чего управление передается следующей ВМ.

Размер кванта (Tquant) служит параметром настройки макета и выбирается как результат компромисса между быстродействием макета и точностью отображения параллелизма. Время завершения очередного кванта времени поддерживается в переменной IsvmTime. В переменной IsvmCurrVmcb диспетчер поддерживает указатель на VMCB той ВМ, которая в данный момент загружена в процессор хост-машины («текущей» ВМ).

Процедура диспетчера ВМ инициируется в ИСВМ после выполнения каждой инструкции в каждой рабочей ВМ (с помощью механизма Т-прерывания) или же после выполнения макрокоманды IsvmSleep в служебной ВМ. Она выполняется по следующему алгоритму.

Шаг 1. Если список активных ВМ пуст, то перейти к шагу 7. Если для текущей ВМ выполняется соотношение VmTime > IsvmTime, то перейти к шагу 2. Если текущая ВМ является служебной, то перейти к шагу 8. В противном случае выполнить анализ следующей инструкции в текущей ВМ и получить оценку времени ее выполнения в модельном времени (Tcomm), нарастить VmTime на величину Tcomm и перейти к шагу 8.

Шаг 2. Сохранить состояние текущей ВМ в ее VMCB. Если текущая ВМ не является последней в списке активных ВМ, то перенести указатель IsvmCurrVmcb на VMCB следующей ВМ, загрузить состояние следующей ВМ в процессор хост-машины и перейти к шагу 1. В противном случае перейти к шагу 3.

Шаг 3. Нарастить общий счетчик модельного времени IsvmTime на величину Tquant.

Шаг 4. Если список ожидающих ВМ пуст, то перейти к шагу 6, в противном случае, перейти к шагу 5.

Шаг 5. Если для первой ВМ из списка ожидающих ВМ выполняется соотношение VmTime $\leqslant$ $\leqslant$ IsvmTime, то перенести VMCB этой ВМ из списка ожидающих ВМ в начало списка активных ВМ и перейти к шагу 4. В противном случае перейти к шагу 6.

Шаг 6. Перенести указатель IsvmCurrVmcb на VMCB первой ВМ в списке активных ВМ, загрузить состояние данной ВМ в процессор хост-машины и перейти к шагу 1.

Шаг 7. Сохранить состояние текущей ВМ в ее VMCB. Если список ожидающих ВМ пуст, то прекратить работу макета МВК. В противном случае присвоить IsvmTime значение VmTime первой ВМ в списке ожидающих и перейти к шагу 5.

Шаг 8. Вернуть управление в текущую ВМ.

Как видно из приведенного алгоритма, в ИСВМ используется сочетание пошагового (поквантового) и событийного методов ведения модельного времени. Пока имеются активные ВМ, модельное время IsvmTime наращивается на размер кванта Tquant после того, как все они «отработают» предыдущий квант времени (шаг 3). Если же активных ВМ нет, то модельное время скачкообразно увеличивается до ближайшего момента окончания интервала времени в списке ожидающих ВМ (шаг 7).

В смысле быстродействия макета МВК самым критичным является шаг 1, так как он осуществляется после выполнения каждой инструкции в рабочей ВМ. Время выполнения очередной инструкции оценивается на основе анализа кода операции и методов адресации операндов с помощью статических таблиц, отражающих скоростные характеристики микро-ЭВМ. Принципиальное преимущество в быстродействии виртуального макета перед программным эмулятором МВК заключается в том, что время выполнения инструкции все же оценивается в несколько раз быстрее, чем ее программная эмуляция.

Разумеется, существенное влияние на быстродействие макета оказывают быстродействие хост-машины, число ВМ в макете и величина Tquant (чем она меньше, тем больше накладные расходы на диспетчеризацию ВМ, но тем точнее отображается параллелизм работы компонентов МВК). На рис. 2 приведен график, отражающий характерную зависимость быстродействия макета от величины

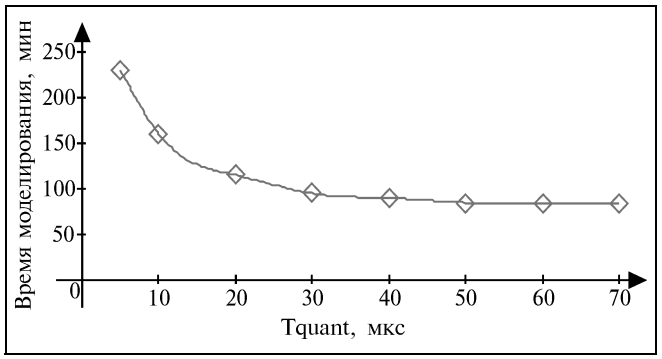

Рис. 2. Зависимость быстродействия макета от величины Tquant

Tquant. График получен в результате тестирования реального целевого ПО в виртуальном макете из четырех рабочих ВМ на хост-машине PDP-11/45. По вертикальной оси отложено физическое время работы макета, потребное для отображения одной минуты модельного времени. Этот и целый ряд подобных экспериментов позволили выбрать верхний предел для значения Tquant: 20 — 30 мкс модельного времени, так как большие значения практически не увеличивают быстродействия макета. Конкретное же значение Tquant в значительной степени зависит от организации вычислительного процесса в МВК. Оно должно выбираться с учетом степени «связности» задач, решаемых на различных микро-ЭВМ (чем интенсивнее взаимодействуют такие задачи, тем меньше должно быть это значение) и естественных ограничений на длительность экспериментов.

4. ВЗАИМОДЕЙСТВИЕ ВИРТУАЛЬНЫХ МАШИН

Средства взаимодействия ВМ в ИСВМ предназначены для отображения взаимодействия компонентов МВК: микро-ЭВМ и внешних устройств. К основным формам взаимодействия ВМ относятся:

— доступ (по чтению и записи) одной ВМ к любой области адресного пространства другой ВМ;

— прерывание другой рабочей ВМ с использованием любого вектора прерывания в ее адресном пространстве;

— возбуждение асинхронной подпрограммы в другой служебной ВМ;

— активизация и деактивизация другой ВМ;

— постановка одной ВМ в очередь на обслуживание к другой ВМ (с передачей параметров обращения и переходом в неактивное состояние до окончания обслуживания);

— последовательная выборка других ВМ из собственной очереди на обслуживание (с получением параметров обращения и последующей активизацией).

Разумеется, право одной ВМ влиять на работу другой определяется при создании макета МВК в описании каждой ВМ (см. далее). Разработчики программ-эмуляторов внешних устройств, загружаемых в служебные ВМ, обеспечиваются набором специальных макрокоманд, инициирующих любую из перечисленных форм взаимодействия. Например, программа эмуляции устройства внешнего таймера, обеспечивающего периодическое прерывание всех микро-ЭВМ через фиксированный интервал времени, могла бы быть устроена в форме цикла, содержащего:

— макрокоманды последовательного прерывания всех рабочих ВМ макета МВК;

— уже упомянутую макрокоманду IsvmSleep, обеспечивающую задержку в отдельном времени на длительность требуемого временного интервала.

Разумеется, в рабочих ВМ, в которые загружается целевое ПО, нельзя использовать подобные макрокоманды. Поэтому взаимодействие с другими ВМ в данном случае реализуется через особые области в адресном пространстве рабочей ВМ, отображающие области ввода-вывода (или области регистров внешних устройств) реальной микро-ЭВМ. С каждой такой областью может быть ассоциирована одна из перечисленных форм взаимодействия, направленного к одной из остальных ВМ. В этом случае взаимодействие может быть инициировано автоматически при выполнении в рабочей ВМ обычной инструкции, адресующей область ввода-вывода. Например, старт-стопный регистр упомянутого внешнего таймера мог бы быть размещен в адресном пространстве одной из рабочих ВМ с возможностью активизации/деактивизации таймера путем простой записи определенных кодов по соответствующему адресу.

5. ИНИЦИАЛИЗАЦИЯ ИСВМ И СОЗДАНИЕ МАКЕТА МВК

Основным компонентом ИСВМ служит монитор виртуальных машин (МВМ) — программный модуль, в котором сосредоточены все процедуры управления виртуальными машинами (диспетчеризация, ведение модельного времени, эмуляция виртуальных ресурсов, управление взаимодействием ВМ и др.). Работа ИСВМ всегда начинается с модуля «инициатор-терминатор», который обеспечивает встраивание МВМ в ядро хост-системы (первая версия ИСВМ была разработана для RSX-11, вторая — для VAX/VMS). Это встраивание заключается в загрузке МВМ в системную область памяти и перенастройке векторов прерывания хост-системы на специальные обработчики

прерываний в составе МВМ (и инициатор-терминатор, и МВМ функционируют в режиме ядра системы). Вторая функция инициатора-терминатора — создание макета МВК по его описанию.

Одной из важных характеристик ИСВМ является ее способность настраиваться под структуру целевого МВК. Эта настройка осуществляется путем подготовки файла описания макета, содержащего описание целевой структуры на простом декларативном языке.

Описание макета должно содержать следующие характеристики:

— потребное число ВМ;

— тип и начальное состояние каждой ВМ;

— состав и структуру адресного пространства каждой ВМ;

— структуру связей между ВМ (задающую правила взаимодействия ВМ).

Чтобы дать представление о языке описания структуры МВК, приведем простое описание структуры одной микро-ЭВМ:

```
VM5 PROCESSOR ACTIVE
AREA 0:100000/RO EXMOD
AREA 100000:40000/RW
AREA 160000:10000/IO
LINK VM3:160000:10/WQ/RQ.
```

Первое предложение этого описания открывает описание ВМ и задает ее номер (5), тип (PROCESSOR — рабочая ВМ) и начальное состояние (активна). Три следующие предложения (AREA) описывают три области памяти данной ВМ. Для каждой области указываются ее начальный адрес, длина и тип: RO — область только для чтения (ПЗУ), RW — область для чтения и записи (ОЗУ), IO — область ввода-вывода. Отметим, что первое из предложений AREA содержит еще и имя двоичного программного модуля (EXMOD), который должен быть загружен в соответствующую область до начала работы макета — это тот самый модуль, который должен быть «прошит» в ПЗУ реальной микро-ЭВМ. В данном простом примере предполагается, что в модуле EXMOD содержатся компактная ОС (с собственными векторами и обработчиками прерываний) и прикладные задачи.

Последнее предложение (LINK) задает связь с другой ВМ (VM3). В данном случае эта связь предполагает, что при попытке обращения (чтения или записи) к диапазону адресов с 160000 по 160010 в области ввода-вывода описываемая ВМ должна быть поставлена в очередь на обслуживание к другой ВМ (VM3) с передачей ей параметров обращения (адреса, записываемого значения и т. д.).

Инициатор-терминатор обеспечивает просмотр файла описания макета, построение и инициализацию структур данных МВМ (в первую очередь, VMCB для всех ВМ), резервирование областей памяти ВМ в памяти хост-системы и загрузку требуемых программных модулей в эти области.

При завершении работы ИСВМ инициатор-терминатор освобождает все занятые ресурсы и восстанавливает векторы прерываний хост-системы в первоначальное состояние.

6. СРЕДСТВА ОТЛАДКИ ЦЕЛЕВОГО ПО

В составе ИСВМ имеется специальный компонент (диалоговый монитор), позволяющий оператору управлять работой макета и отслеживать состояние каждой ВМ:

— запускать макет МВК и останавливать его;

— задавать точки останова (breakpoints) в адресном пространстве каждой ВМ;

— задавать моменты останова макета в модельном времени;

— задавать режимы трассировки и пошагового выполнения в каждой ВМ;

— вызывать на экран состояние любой ВМ, включая значения в регистрах процессора, слово состояния программы, содержимого областей памяти и др., а также вносить изменения в состояние ВМ;

— сохранять состояние всего макета в файле и загружать состояние макета из файла (что дает возможность разбиения работы макета на несколько отдельных сеансов).

Технически диалоговый монитор реализуется как служебная ВМ, что позволяет ему конкурировать за процессорное время хост-машины и пользоваться всеми услугами МВМ.

Особо подчеркнем, что все отладочные режимы не вносят возмущения в ход вычислительного процесса в макете МВК. Так, трассировка или пошаговое выполнение программ в отдельной или нескольких ВМ с неизбежностью выполняются «на фоне» работы всех остальных в едином модельном времени, а останов какой-либо ВМ в точке останова вызывает останов всего макета и модельного времени. Немаловажно и то, что все эти режимы не «отнимают» у ВМ ни адресного пространства, ни каких-либо других ресурсов.

Диалоговый монитор получает управление в начале и конце работы макета, а также при любых событиях, вызывающих останов работы макета.

ОБСУЖДЕНИЕ (вместо заключения)

В наши дни мало кто помнит систему PDP-11, и даже блестящая архитектура VAX-11 почти не упоминается в литературе. Архитектура компьютеров претерпела существенные изменения с начала 1990-х гг. Поэтому вопрос о возможности применения описанного подхода к современным архи-

тектурам, безусловно, важен для оценки его перспективности. Хотя автор не может сослаться на конкретный опыт такого применения, он хотел бы высказать несколько соображений по этому поводу.

Рассмотренные в данной работе функции ИСВМ, касающиеся отображения параллельной работы и взаимодействия между ВМ, можно рассматривать как «надстройку» над функциями создания отдельных ВМ и обеспечения их функционирования. Главным архитектурным механизмом, поддерживающим эту «надстройку», является механизм пошагового прерывания программы (Т-прерывания), который имеется во всех современных компьютерных архитектурах, что обеспечивает хотя бы принципиальную возможность применения подхода. Что же касается функций поддержки функционирования отдельных ВМ с учетом всех особенностей современных архитектур (сегментно-страничная организация памяти, отделение адресного пространства ввода-вывода от адресного пространства памяти, новая организация системы команд и системы прерываний и т. д.), то решение этой задачи в значительной степени может быть «заимствовано» у современных систем виртуальных машин общего назначения.

Важная архитектурная черта современных микропроцессоров состоит в наличии механизма кеширования программ и данных, который является одним из основных источников их высокого быстродействия, но приводит к высокой вариабельности длительностей инструкций. Так как моделирование работы этого механизма привело бы к огромному замедлению работы модели, ИСВМ может ориентироваться только на усредненные статические оценки длительностей. Разумеется, это не может не привести к временному искажению вычислительного процесса в модели по сравнению с реальной средой. (Эта проблема в значительной степени касается и программных эмуляторов МВК.) Но даже и в этом случае возможность проведения отладочных «прогонов» ПО уже на ранних этапах разработки (параллельно с разработкой аппаратуры и натурных макетов) может оказаться весьма полезной. В первую очередь это касается проверки правильности стратегических и алгоритмических решений, заложенных в основу функционирования встроенной ОС и, в частности, в организацию межмашинных взаимодействий. Окончательная же отладка может быть проведена, разумеется, только в реальных условиях на натурном макете. Опыт применения ИСВМ показал, что описываемый подход действительно позволяет повысить степень готовности целевого ПО на ранних этапах проекта.

ЛИТЕРАТУРА

1. *Система* виртуальных машин для ЕС ЭВМ / И.М. Булко, Н.Н. Дорожко, Л.И. Дудкин и др. — М.: Финансы и статистика, 1985. — 374 с.
2. *Mann A.T.* The rational guide to Microsoft Virtual PC. — Greenland, NH, USA: Rational Press, 2004. 112 p.
3. *Гультяев А.К.* Виртуальные машины — несколько компьютеров в одном. — СПб.: Питер, 2006. — 224с.
4. *Костромин В.А.* Linux для пользователя. — СПб.: БХВ-Петербург, 2003. — С. 672.
5. *Костромин В.А.* Две системы на одном компьютере // Открытые системы. — 2001. — № 7. — С. 23—28.
6. *Костромин В.А.* Linux вместе с Windows // Там же. — № 3. — С. 24—31.
7. *Елманова Н.* Виртуальные машины — 2006. Новые продукты компании VMware // Компьютер Пресс. — 2006. — № 5. — С. 132—136.
8. *Асратян Р.Э., Волков А.Ф.* Мультипроцессное моделирование вычислительных систем // Тез. докл. всесоюзн. конф. «Проблемы развития аппаратных и программных средств вычислительной техники для машинного моделирования». — М.: Радио и связь, 1987. — С. 71—72.
9. *Принципы* виртуального макетирования мультимикромашинных систем / Р.Э. Асратян, В.А. Бугаев, О.В. Гронда, Н.В. Дагурова // Тез. докл. XI Всесоюзн. совещания по проблемам управления. — М.: НКАУ СССР, 1989. — С. 195—196.
10. *Мячев А.А.* Интерфейсы средств вычислительной техники. — М: Радио и связь, 1993. — 352 с.

Статья представлена к публикации руководителем РРС В.Ю. Столбовым.

Асратян Рубен Эзрасович — канд. техн. наук,
вед. науч. сотрудник, Институт проблем управления
им. В.А. Трапезникова РАН, г. Москва,
☎(495) 334-88-61, ✉ rea@ipu.ru.