Лабораторный практикум по программному моделированию

Учебное пособие

Версия 1.3β
6 июля 2014 г.

Содержание

Предисловие

В настоящем практикуме описываются лабораторные работы по курсу «Основы программного моделирования ЭВМ», проводившиеся в Московском физико-техническом институте на базе вычислительного кластера лаборатории суперкомпьютерных технологий для биомедицины, фармакологии и малоразмерных структур iScalare.

Если вы обнаружили опечатку, стилистическую, фактическую ошибку, которые, более чем вероятно, встречаются в тексте, имеете замечания по содержанию или предложения по тому, как можно улучшить данный материал, то просим сообщить об этом по e-mail `grigory.rechistov@phystech.edu` — нам очень важно ваше мнение.

Отметим, что текст данной работы постоянно обновляется, и поэтому в версиях, имеющих в своём номере пометку «бета» (β), незаконченные места обозначаются символом TODO .

Введение

1. Компьютерная симуляция

Использование компьютерного моделирования в процессе проектирования позволяет заметно сократить среднее время, проходящее от момента предложения концепции новой системы до поступления на рынок первых образцов готовой продукции. Это происходит благодаря т. н. «сдвигу влево» (*англ.* shift left) всей существующей методологии, что позволяет совмещать многие процессы параллельно во времени, так как часть из них может быть начата гораздо раньше, чем это было возможно ранее, и эффективно сокращать цикл разработки (рис. 1).

Программное обеспечение для имитационного моделирования используется для тестирования и исследования производительности, функциональных и иных свойств вычислительных систем на стадиях их раннего проектирования, когда реальные образцы соответствующей аппаратуры ещё недоступны. Кроме того, оно позволяет писать приложения для таких систем заранее.

Задача данного цикла лабораторных работ — познакомить слушателей с новейшими достижениями в области компьютерной симуляции, связанными с эффективным созданием моделей, максимально точно представляющих аппаратные средства и при этом имеющих приемлемую скорость работы, получаемую благодаря эффективному задействованию имеющихся вычислительных ресурсов. Изучение проводится на программном продукте Wind River® Simics (в дальнейшем просто Simics), который в настоящее время является одним из самых современных инструментов разработки, тестирования и исследования цифровых компьютерных систем и используется как в промышленности, так и в научной среде. Несмотря на это, почти все рассматриваемые вопросы

включают в себя идеи и концепции, общие для разных симуляторов.

2. Предварительные замечания

Для максимально эффективного усвоения материала данного пособия читателю рекомендуется иметь начальные знания по архитектуре ЭВМ. Желательно понимание общих принципов работы операционных систем, а также знакомство как минимум с одним языком программирования.

3. Обозначения

При первом использовании терминов, заимствованных из английского языка и не имеющих известных авторам общепринятых переводов на русский язык, в скобках после них будут указываться оригинальные термины.

Всюду в тексте данной работы будут использованы следующие шрифтовые выделения и обозначения.

- Обычный текст используется для основного материала.
- `Моноширинный текст` вводится для исходных текстов программ на различных (псевдо) языках программирования и их ключевых слов, имён регистров устройств, листингов машинного кода.
- *Наклонный текст* используется для выделения новых понятий.
- **Полужирный текст** используется для обозначения элементов графического интерфейса: имён окон, пунктов меню и т.п.
- Числа в шестнадцатеричной системе счисления имеют префикс **0x** (например, 0x12345abcd), в двоичной системе счисления — суффикс **b** (например, 10010011b).
- Команды, которые необходимо вводить в строку приглашения Simics, имеют префикс `simics>`:

```
simics> list-objects
```

- Команды, которые необходимо вводить в строку приглашения командной строки Bash, имеют префикс $ для обычного пользователя или # для команд, выполняемых суперпользователем root:

```
$ ./simics targets/x86-x58-ich10/viper.simics
# mount /dev/sdb /mnt/disk
```

- Обязательные аргументы команд указаны в угловых скобках, необязательные — в квадратных, например:

```
$ command <mandatory argument> [optional argument]
```

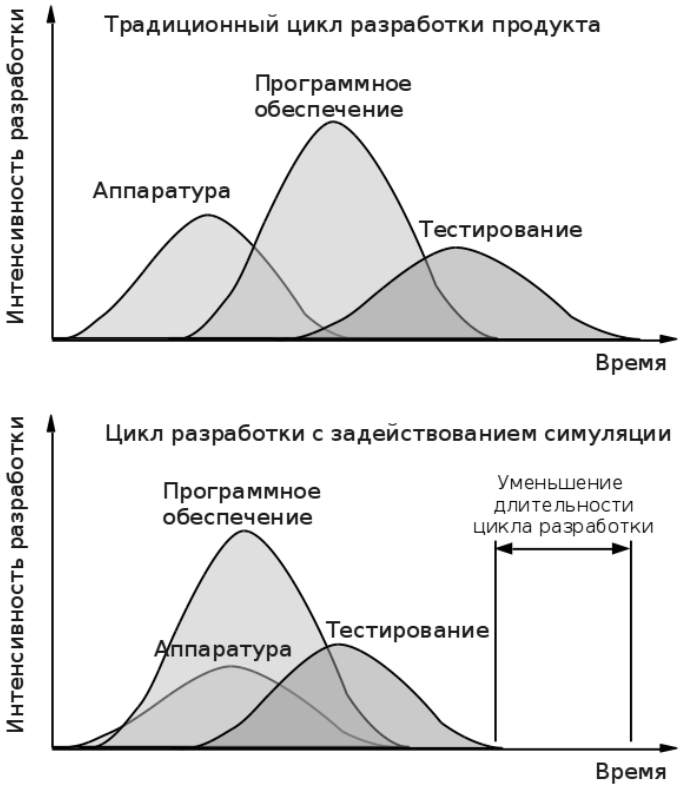

Рис. 1. Сдвиг влево — возможность совместить моменты начала отдельных стадий проектирования новых цифровых систем, таким образом сокращая цикл разработки и уменьшая время вывода их на рынок

1. Моделирование процессора архитектуры OpenRISC 1000

Данная индивидуальная работа посвящена реализации модели компьютерной платформы, основанной на спецификации OpenRISC 10000 [1]. Модель строится на основе API Simics и оформляется как набор модулей и сценариев для данного симулятора.

1.1. Спецификации OpenRISC 1000

OpenRISC 1000 — дизайн процессора, построенный по философии Open Source [3] как для программного кода, так и для аппаратуры. Спецификация определяет 32- или 64-битный RISC-процессор общего назначения, с пятью стадиями конвеера, необязательной поддержкой MMU (*англ.* memory management unit), кэшей, управлением энергопотреблением и др. Важная для данной лабораторной работы черат спецификации OpenRISC 1000 — это опциональность многих частей функциональности, что позволяет ограничиваться только минимально необходимыми для конкретной реализации элементами.

Существуют реализации OpenRISC 1000 в виде программных моделей, спецификаций для FPGA и описаний RTL на языке Verilog. Программная поддержка существует со стороны компиляторов GCC и LLVM, адаптированных библиотек LibC и GNU toolchain.

Для создания рабочей функциональной модели ЦПУ требуется спецификация на следующие его элементы: архитектурное состояние (число и типы регистров), набор инструкций (кодировка и семантика), а также интерфейсы к внешним элементам системы (память и периферийные устройства). Детали, относящиеся

к особенностям организации конвеера, длительностям работы отдельных инструкций, устройству кэшей и т.п. не являются необходимыми на данном уровне точности моделирования.

1.1.1. Набор регистров

Классификация регистров OpenRISC 1000 (рис. 1.1), требующих реализации в ходе работы.

- 32 регистра общего назначения шириной в 64 бита, доступные из пользовательского и супервизорного режимов: `R0` – `R31`. Регистр R0 всегда должен содержать ноль.
- Специфичные для модулей регистры (*англ.* special purpose registers, SPR). В случае, если некоторая опциональная часть спецификации реализована, с ней могут идти дополнительные регистры для индикации статуса и управления состоянием. Каждый из них имеет собственное имя и функциональность. SPR разделены по группам. Группа 0 — регистры супервизора шириной 32 бита. Некоторые из них могут быть доступны на чтение из пользовательского режима.

1.1.2. Набор инструкций

Набор инструкций состоит из нескольких классов (рис. 1.2), соответсвующих различным задачам и типам обрабатываемых данных.

- ORBIS32 — 32-битные команды общего назначения; единственный класс, обязательный для реализации;
- ORBIS64 — 64-битные команды общего назначения;
- ORFPX32 — 32-битные команды для работы над числами с плавающей запятой;
- ORFPX64 — 64-битные команды для работы над числами с плавающей запятой;

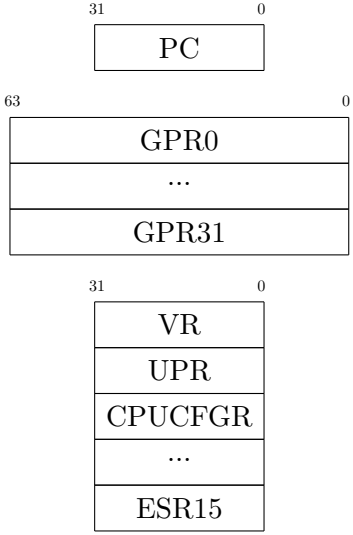

Рис. 1.1. Регистры OpenRISC 1000

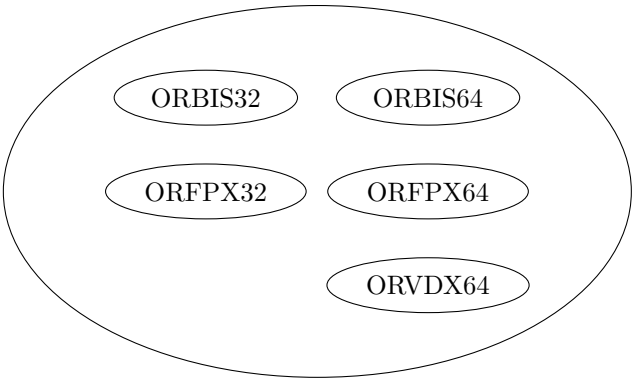

Рис. 1.2. Классы инструкций набора команд OpenRISC 1000

- ORVDX64 — команды для операции над векторами данных с элементами шириной 8, 16, 32 или 64 бита;
- Частные команды, специфичные для конкретной модели.

Ширина отдельных инструкций фиксирована и равна 32 битам. ORBIS32 состоит из следующих подклассов команд:

- арифметические операции;
- базовые инструкции для обработки сигналов (*англ.* digital signal processing, DSP);
- загрузка и запись данных из памяти;
- управление порядком исполнения;
- прочие.

1.2. Кодовая база Simics для создания моделей процессоров

Основа модели ЦПУ — код `sample-risc`, модифицированный для нужд учебного проекта:

- убрана мноноядерность;
- декодер упрощён до одной инструкции.

Основа модели таймера — sample-device.

Конфигурация платформы (ЦПУ, память, опционально таймер) — основана на сценарии `queues-in-cosimulator.simics`.

1.3. Тестирование моделей

Для тестирования корректности моделирования отдельных инструкций применяется юнит-тестирование на основе фреймворка test-runner. Для проверки последовательностей команд код гостевой код может быть написан вручную в машинных кодах или же с помощью GNU toolchain [2].

1.4. Порядок работы

1. Подготовить workspace Simics.
2. Собрать код sample-risc.
3. Реализовать набор регистров базового архитектурного состояния.
4. Реализовать декодер команд.
5. Интегрировать модель openrisc в сценарий платформы.
6. Реализовать модель таймера.
7. Интегрировать модель таймера в сценарий платформы.

1.4.1. Дополнительные шаги

- Реализовать набор инструкций ORBIS64.
- Реализовать функциональность TLB и/или MMU.
- Реализовать устройство PIC.
- Реализовать модель кэша данных.
- Реализовать набор инструций ORFPX32.

Список литературы к занятию

1. *Open Cores* OpenRISC 1000 Architecture Manual. Architecture Version 1.0 Document Revision 0. — 2012. — URL: `http://opencores.org/websvn,filedetails?repname=openrisc&path=/openrisc/trunk/docs/openrisc-arch-1.0-rev0.pdf`.
2. *Open Cores* OpenRISC GNU tool chain. — 2014. — URL: `http://opencores.org/or1k/OpenRISC_GNU_tool_chain`.
3. The Open Source Definition (Annotated). — Open Source Initiative. — URL: `http://opensource.org/osd-annotated`.

Приложения

A. Дополнительная информация по работе с Simics

В данное приложение включены сведения о различных приёмах, используемых при ежедневном использовании Simics, не описанные в главах, посвящённых индивидуальным лабораторным работам.

A.1. Обновление workspace

Для получения последних исправлений ошибок в моделях необходимо использовать самую свежую версию базового пакета Simics из установленных на кластере. Номер версии определить по файлам, установленным в папке `/share/simics/simics-4.6/`. В тексте данной работы последней версией будет считаться **4.6.32**, при этом 4.6 — это основная версия, а последняя цифра — номер минорной версии обновления. Он будет использован позже.

Каждая копия workspace характеризуется версиями пакетов, в ней используемых. Номер пакета Simics Base определяет настройки версий остальных пакетов, установленных одновременно с ним. Для того чтобы увидеть список пакетов с их версиями, используйте команду:

```
$ ./simics -v
Simics Base                                          1000
    4.6.32    (4051)
Model Library: Intel Core i7 with X58 and ICH10      2075
    4.6.21    (4051)
Model Builder                                        1010
    4.6.14    (4051)
Extension Builder                                    1012
    4.6.6     (4042)
```

В примере сверху базовый пакет имеет версию 4.6.32. Новые пакеты будут периодически ставиться на кластере для исправления ошибок в базовых моделях. Для обновления своего workspace используйте команду `workspace-setup`, находящуюся внутри новой версии базового пакета (версии 4.6.<**minor**>), выполненную внутри workspace, который вы хотите обновить.

```
$ /share/simics/simics-4.6/simics-4.6.<minor>/bin/workspace-
    setup
```

Также версию Simics можно узнать из командной строки любой запущенной симуляции:

```
simics> version
```

A.2. Список часто используемых команд Simics

Команда	Синонимы	Выполняемая функция
`help <topic>`	`man`	Справка по команде, классу или слову topic
`win-help`		Открыть окно индексируемой справки
`continue`	`c, r, run`	Начать или продолжить симуляцию
`stop`		Остановить симуляцию
`step-cycle [count]`	`sc`	Исполнить count циклов, печатаю следующую инструкцию
`exit`	`quit, q`	Выйти из симулятора
`run-command-file <script.simics>`		Выполнить скрипт Simics
`pregs [-all]`		Распечатать содержимое регистров текущего процессора
`print-time [-all]`	`ptime`	Вывести значение виртуального времени процессора
`win-control`		Открыть окно **Simics Control**
`%<register name>`	`read-reg`	Прочитать содержимое регистра текущего процессора
`%<register name> = <val>`	`write-reg`	Записать значение в регистр текущего процессора
`output-radix <10|16>`		Изменить основание используемой для вывода чисел системы счисления
`break <address>`		Поставить точку останова по адресу
`delete [id]`		Удалить точку останова по её номеру

B. Скрипт для отладки

```
name_prefix = cli.simenv.host_name
if name_prefix != '':
    name_prefix = name_prefix + '_'

sample_risc0 = pre_conf_object(name_prefix + 'sample_risc0',
    'sample-risc')
sample_risc0.queue = sample_risc0

ram_image0 = pre_conf_object(name_prefix + 'ram_image0', '
    image')
ram_image0.queue = sample_risc0
ram_image0.size = 0x800000

ram0 = pre_conf_object(name_prefix + 'ram0', 'ram')
ram0.image = ram_image0

phys_mem0 = pre_conf_object(name_prefix + 'phys_mem0', '
    memory-space')
phys_mem0.queue = sample_risc0
phys_mem0.map = [[     0x0, ram0,                  0, 0, 0
    x800000],
                ]

ctx0 = pre_conf_object(name_prefix + 'ctx0', 'context')
ctx0.queue = sample_risc0

sample_core0 = pre_conf_object(name_prefix + 'sample_core0',
    'sample-risc-core')
sample_core0.queue = sample_risc0
sample_core0.sample_risc = sample_risc0
sample_core0.physical_memory_space = phys_mem0
sample_core0.current_context = ctx0

#cosim_cell = pre_conf_object(name_prefix + 'cosim_cell', '
    cell')
#cosim_cell.current_processor = sample_core0
```

```
#cosim_cell.current_step_obj = sample_risc0
#cosim_cell.current_cycle_obj = sample_risc0
#cosim_cell.scheduled_object = sample_risc0
#sample_risc0.cell = cosim_cell

SIM_add_configuration([sample_risc0, ctx0, ram_image0, ram0,
     phys_mem0, sample_core0, sample_core1], None)
```

C. Код целевого скрипта practicum.simics

```
# Script for mipt practicum
load-module pci-components
load-module std-components
load-module x86-comp
load-module x86-nehalem-comp
load-module x58-ich10-comp
load-module memory-comp

add-directory "%simics%/targets/x86-x58-ich10/images/"

$disk_image          = "/share_debian/hpc-images/debian-
    master-2012-05-12.craff"
$cpu_class           = core-i7-single
$freq_mhz            = 3300
$cpi                 = 1
$disk_size           = 20496236544
$rtc_time            = "2008-06-05 23:52:01 UTC"
$memory_megs         = 2048
$text_console        = TURE
$use_acpi            = TRUE
$gpu                 = "accel-vga"
$bios                = "seabios-simics-x58-ich10
    -0.6.0-20110324.bin"
$break_on_reboot     = TRUE
$apic_freq_mhz       = 133
$use_vmp             = TRUE
$spi_flash           = "spi-flash.bin"
$mac_address         = "00:19:A0:E1:1C:9F"
$host_name           = "practicum"

$system = (create-x86-chassis name = $host_name)

### motherboard
$motherboard = (create-motherboard-x58-ich10 $system.mb
```

```
        rtc_time = $rtc_time
        acpi = $use_acpi
        break_on_reboot = $break_on_reboot
        bios = $bios
            mac_address = $mac_address
        spi_flash = $spi_flash)
$southbridge = $motherboard.sb
$northbridge = $motherboard.nb

### processor
$create_processor = ''create-processor-'' + $cpu_class
$create_processor_command = (
        $create_processor
        + '' $motherboard.cpu0''
        + '' freq_mhz = $freq_mhz''
        + '' apic_freq_mhz = $apic_freq_mhz''
        + '' use_vmp = $use_vmp''
        + '' cpi = $cpi'')
$cpu0 = (exec $create_processor_command)
connect $motherboard.socket[0] $cpu0.socket

### memory
$dimm = (create-simple-memory-module $motherboard.memory
                                     memory_megs =
    $memory_megs)
connect $motherboard.dimm[0] $dimm.mem_bus

### GPU
$vga = (create-pci-accel-vga $motherboard.gpu)
connect $northbridge.gpu $vga.connector_pci_bus

### consoles
$console = (create-std-text-graphics-console $system.console
    )
$console.connect keyboard $southbridge
$console.connect $vga

### disk
if not (lookup-file $disk_image) {
    interrupt-script ''Disk image file not found: '' +
    $disk_image
}
$disk = (create-std-ide-disk $system.disk size = $disk_size
    file = $disk_image)
$southbridge.connect ''ide_slot[0]'' $disk
```

```
instantiate-components

#SimicsFS support (add SimicsFS pseudo device)
$hostfs = python ''SIM_create_object('hostfs', 'hfs0', [])''
practicum.mb.phys_mem.add-map $hostfs 0xfed2_0000 16

try {
    win-command-line
} except { echo ''Failed to create GUI''}

script-branch { # Automatize GRUB and login
    local $con = $host_name.console.con
    $con.wait-for-string ''automatically in 5s''
    $con.input ''\n''
    $con.wait-for-string ''login:''
    $con.input ''user\n''
    $con.wait-for-string ''Password:''
    $con.input ''user\n''
}
```

D. Программа debug_example.c

```
/*
 * This program  reads input and converts it to uppercase.
 * It has an intentional bug included that makes it crash on
     certain inputs.
 *
 * Usage: stdin - input string.
 * Compile with gcc -static -g debug_example.c -o
    debug_example
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void read_input(char* in) {
    char symbol;
    in[0] = 0; // initialize the string with zero length
    while((symbol = getchar()) != EOF) {
        *in++ = symbol;
    }
    *in = 0; // close the string
}

void convert_to_uppercase(char * in, char *out) {
    int i;
    for (i =0; i <= strlen(in); i++) {
        if (isalpha(in[i]))
            out[i] = toupper(in[i]);
        else
            out[i] = in[i];
    }
}

int main(int argc, char** argv) {
    char input[32], *output;
```

```
    read_input(input);
    convert_to_uppercase(input, output);

    printf(''%s\n'', output);
    return 0;
}
```

E. Установка и лицензирование Simics

В данное приложение включена информация по лицензированию и установке Simics в учебной лаборатории. Наиболее полная информация по данному вопросу содержится в документе «Simics Installation Guide» [1], который идёт в поставке со всеми пакетами Simics (файл `installation-guide.pdf`).

Приводимые ниже инструкции были собраны для Simics версии 4.6 для хозяйской системы Linux 64 бит, рекомендуемой для всех пользователей. Для ОС Linux 32 бит инструкции изменяются незначительно; для ОС Windows они применимы после учёта особенностей графического процесса инсталляции.

E.1. Академическая программа Wind River Simics

Компания Intel предлагает Simics бесплатно для некоммерческих исследований и обучения в выбранных университетах через академическую программу Intel Simics Academic Program. Для включения нового университета в эту программу необходимо согласие на поддержку начинания одного ментора — сотрудника Intel.

Дополнительная информация об истории и статусе академической программы Simics [.]

E.1.1. Условия использования

Использование Simics по академической программе должно строго соответствовать условиям соглашения, т.е. быть ограничено учебной и/или некоммерческой научно-исследовательской

деятельностью. В случае возникновения необходимости проведения коммерческих исследований или разработок необходимо обратиться к представителям Wind River для получения нового соглашения и другого типа лицензии.

Держатель лицензии от участвующего университета обязан донести эту информацию до всех пользователей инсталляции и контролировать выполнение ими условий соглашения, в том числе с помощью административных и технических мер.

Подробные детали об условиях и ограничениях академической программы содержатся в Intel Academic SLA, поставляемом с копией Simics для университетов.

E.2. Установка файлов и запрос лицензии

E.2.1. Пакеты

Simics распространяется в формате пакетов — набора файлов, реализующих одну или несколько типов моделируемых систем или функциональность самого симулятора. Каждый пакет имеет свой фиксированный номер. Пакет №1000 — это Simics Base, содержащий базовую функциональность симулятора. Все остальные пакеты являются дополнениями к нему.

Дистрибутив пакета — это файл с именем вида `simics-pkg-1000-4.6.34-linux64.tar`. Здесь 1000 — номер пакета, 4.6.34 — версия пакета, linux64 — архитектура хозяйской системы. Каждый дистрибутив каждого пакета зашифрован собственным ключом, состоящим из 32 символов. Дистрибутивы и их ключи получите у спонсора вашей академической программы.

Для установки всех необходимых файлов выполняется следующая процедура. Часть команд может потребовать наличия прав администратора.

1. Разархивируйте все пакеты *.tar:

   ```
   $for f in simics-pkg-*.tar; do tar xf $f; done
   ```

2. В созданной директории `simics-4.6-install` запустите скрипт установки:

```
$ cd simics-4.6-install
# ./install-simics.pl
```

3. Введите ключи шифрования для каждого номера пакета, который планируется установить:

```
-> Looking for Simics packages in current directory...
Enter a decryption key for package-1000-4.6.34-linux64.
    tar.gz.tf,
or Enter to [Abort]: Ключ[]
```

4. На вопрос, какие из пакетов требуется установить, ответьте «All packages»:

```
install-simics can install the following packages:
 Number  Name          Type    Version  Host
    Package
   1     Simics-Base   simics  4.6.34   linux64
    package-1000
   2     Eclipse       addon   4.6.16   linux64
    package-1001
   3     All packages
Please enter the numbers of the packages you want to
    install, as in "1 4 3"
Package numbers, or Enter to [Abort]:  3
```

5. На вопрос о директории назначения введите абсолютный путь или оставьте значение по умолчанию:

```
Enter a destination directory for installation, or
    Enter
for [/opt/simics]:  Путь[ установки или Enter]
```

6. Подтвердите начало установки, выбрав «y».

7. При введении правильных ключей дистрибутивы будут расшифрованы и установлены в указанную при установке директорию — в ней должны появиться подпапки с файлами из пакетов Simics.

```
-> Decrypting package-1000-4.6.34-linux64.tar.gz.tf
-> Testing package-1000-4.6.34-linux64.tar.gz
-> Installing package-1000-4.6.34-linux64.tar.gz
```

```
-> Decrypting package-1001-4.6.16-linux64.tar.gz.tf
-> Testing package-1001-4.6.16-linux64.tar.gz
-> Installing package-1001-4.6.16-linux64.tar.gz

================================

install-simics has finished installing the packages and
     will now
configure them.

No previous Simics installation was found. If you wish
    to configure
the newly installed Simics from a previous installation
     not found by
this script, you can do so by running the 'addon-
    manager' script in
the Simics installation with the option --upgrade-from:
    ./bin/addon-manager --upgrade-from /previous/
    install/

install-simics has installed the following add-on
    package:
   Eclipse  4.6.16  /opt/simics/simics-eclipse-4.8.26
```

8. На вопрос о регистрации расширений (*англ.* add-on) ответьте «y»:

```
Do you wish to make these add-on packages available in
Simics-Base 4.6.34? (y, n) [y]:  y
```

После успешного завершения файлы Simics были скопированы на ваш диск. Следующий шаг — получение лицензии для их запуска. Он описывается далее.

E.2.2. Получение lmhostid

Для получения файла лицензии необходимо сгенерировать и передать число, так называемый lmhostid

Об именовании сетевых интерфейсов. На момент написания данного материала утилиты из состава Simics не поддерживали получение корректного lmhostid на системах, использующих

схему «стабильного именования» сетевых интерфейсов. Вместо традиционных для Linux имён `eth0`, `eth1` и т.д. сетевым картам выдаются имена, зависящие от производителя и физического расположения в системе.

Для обеспечения правильного именования TODO .

1. Установите пакет `lsb-core` на системе. Для Debian и Ubuntu это выполняется командой:

   ```
   # apt-get install lsb-core
   ```

2. Для получения lmhostid на сервере, *который будет использоваться для запуска демона лицензий*, выполните команду:

   ```
   $ /opt/simics/simics-4.6.34/flexnet/linux64/bin/lmutil
       lmhostid
   lmutil - Copyright (c) 1989-2011 Flexera Software, Inc.
        All Rights Reserved.
   The FlexNet host ID of this machine is ""602fe934a369
       422fe934a36c ""
   Only use ONE from the list of hostids.
   ```

   Выданное число (в примере выше «602fe934a369») — это lmhostid. Если чисел выдано несколько, то используйте только одно из них.

Возможные решения:
TODO Команда

E.2.3. Заполнение заявки

E.3. Настройка сервера лицензий

E.3.1. Файл лицензии

Получаемый от производителя файл лицензии — это текстовый документ, содержащий информацию о сроке действия, ограничениях количества одновременно запускаемых копий и поддерживаемых расширениях приложения. Пример содержимого для начала этого файла:

```
# Simics 4.6 licence for the Simics Academic Program
#
# University:         Moscow Institute of Physics and
    Technology
# Contact:            academic.contact@university.edu
# Sponsor:            sponsor.contact@sponsor.com
# Licensing Contact: licencing.contact@licencer.com
#
SERVER  lic.university.edu lmhostid
VENDOR simics /home/Incoming/simics-4.6/simics-4.6.100/
    flexnet/linux64/bin
#
FEATURE simics simics 4.6 28-feb-2014 50 BD47D265FA68 \
        VENDOR_STRING=intel;academic HOSTID=ANY BORROW TS_OK
     \
        SIGN="0441 6AFA 450C BDBE E4D7 E125 1042 EEFF 04B5
    767A ABCD \
        5088 80DB D912 292E 4FD5 22DD 22D0 D55F 5B25 4818"
<...>
```

Не изменяйте никаких строчек этого файла, кроме имени сервера лицензий. Сохраните копию файла в надёжном месте. Запишите дату окончания действия лицензии для последующей своевременной инициации процедуры обновления.

E.3.2. Запуск сервера

Для запуска серевера лицензий используется программа lmgrd, поставляемая с базовым пакетом[1]. Её расположение: `<simics-base>/flexnet/linux64/bin/lmgrd`.

Пример последовательности команд для ручного запуска lmgrd:

```
cd /opt/simics/simics-4.6/simics-4.6.100
./lmgrd -c /opt/simics/simics-4.6/simics-4.6.100/licenses/
    simics.lic
```

В данном случае процесс остаётся в консоли (не уходит в фоновый режим) и печатает диагностику в консоль. Для остановки достаточно убить его с помощью Ctrl-C.

[1]Варианты этой программы, полученные из других источников, не рекомендуются и не поддерживаются

Для автоматического запуска и остановки процесса lmgrd при включении и выключении системы рекомендуется использовать init-скрипт в стиле инициализации SysV. Пример такого скрипта: `https://gist.github.com/grigory-rechistov/11142235`, также он приведён в секции E.6.

E.3.3. Проверка работоспособности

E.4. Расположение файлов и сервера лицензий при подключении по сети

По умолчанию все файлы Simics размещаются в директории `/opt/simics`. Если необходимо обеспечить запуск симулятора на нескольких компьютерах, подсоединённых по сети, рекомендуется разместить эти файлы установки в файловой системе, доступной по сети, например, по протоколам NFS или CIFS. Таким образом, клиентские машины смогут переиспользовать структуру инсталляции без необходимости её копирования на локальные диски, что упростит поддержку и обновления. Настройка сетевой файловой системы выходит за рамки данного руководства; необходимую информацию можно найти, например, в [2].

E.4.1. Финальный вид инсталляции

На рис. E.1 приведена рекомендуемая схема соединения систем и расположения служб для работы Simics на всех компьютерах учебного класса или лаборатории. В данном примере сервер для запуска демона лицензий отделён от сервера общих файлов; на практике они могут быть одной и той же системой.

E.4.2. Решение возникших проблем

Следует отметить, что процесс lmgrd рекомендуется запускать из-под непривилегированного пользователя, т.е. не root. Кроме того, он должен быть в состоянии найти файл с т.н. программой vendor-daemon, которая для Simics называется simics и находит-

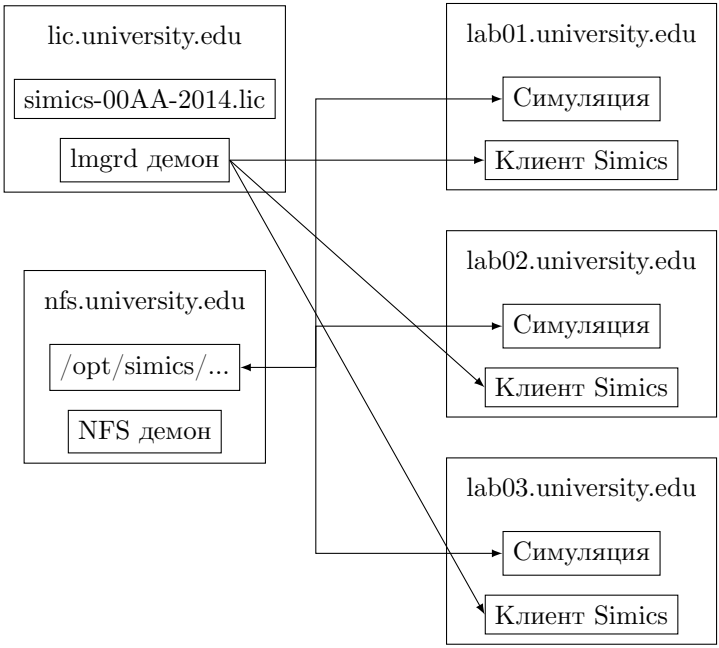

Рис. E.1. Расположение и функции узлов инсталляции Simics

ся в той же директории, что и lmgrd. Поэтому запуск должен происходить из этой же директории.

Невозможно стартовать lmgrd. Проверьте флаги файла lmgrd на признак исполняемости.

lmgrd выходит сразу после запуска. Возможные причины: 1) копия lmgrd уже запущена; 2) не найден файл лицензии; 3) не найден файл с вендор-демоном simics; 4) запуск на системе с неправильным lmhostid; 5) Файл блокироки /tmp/locksimics существует и недоступен на запись.

Невозможно запустить Simics.

Нет подключения к демону лицензии с того же сервера, на котором запу

Нет подключения к демону лицензии по сети

Демон лицензий не работает после перезагрузки сервера

Исчерпано число лицензий

E.5. Обновление пакетов существующей инсталляции

TODO

E.6. init скрипт для старта и остановки демона лицензий

Данный скрипт `lmgrd-simics` должен быть размещён в /etc/inid.d с правом исполнения, затем для Debian-систем должен быть включён с помощью команды:

```
# update-rc.d lmgrd-simics defaults
```

Он доступен по ссылке `https://gist.github.com/grigory-rechistov/11142235`.

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          lmgrd-simics
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Control Flexera lmgrd license daemon
    for Simics installation
# Description:       Control start/stop of lmgrd entry for
    Simics
#
### END INIT INFO

# Author: Grigory Rechistov (<grigory.rechistov@phystech.edu
    >)
#

# Do NOT ''set -e''

# PATH should only include /usr/* if it runs after the
    mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC=''lmgrd for Simics''
SIMICSDIR=/opt/simics/simics-4.6/simics-4.6.100
LICENSEFILE=/opt/simics/simics-4.6/simics-4.6.100/licenses/
    simics.lic # change to your license file
NAME=lmgrd
VENDORDAEMON=simics
DAEMONDIR=$SIMICSDIR/flexnet/linux64/bin
SCRIPTNAME=/etc/init.d/$NAME
DAEMON=$DAEMONDIR/$NAME

PIDFILE=/var/tmp/$NAME.pid
LOCKFILE=/var/tmp/locksimics
LOGFILE=/var/tmp/lmgrd-simics.log

# Exit if the package is not installed
[ -x ''$DAEMON'' ] || exit 0

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Load the VERBOSE setting and other rcS variables
. /lib/init/vars.sh
```

```
# Define LSB log_* functions.
# Depend on lsb-base (>= 3.2-14) to ensure that this file is
    present
# and status_of_proc is working.
. /lib/lsb/init-functions

#
# Function that starts the daemon/service
#
do_start()
{
  # Return
  #   0 if daemon has been started
  #   1 if daemon was already running
  #   2 if daemon could not be started
  start-stop-daemon --start -c daemon:daemon --make-pidfile
    --pidfile $PIDFILE -d $DAEMONDIR --exec $DAEMON -- -c
    $LICENSEFILE -l +$LOGFILE || return 2
  pidof $NAME > $PIDFILE # This is lame; but lmgrd about
    itself does not create anything.
}

#
# Function that stops the daemon/service
#
do_stop()
{
  # Return
  #   0 if daemon has been stopped
  #   1 if daemon was already stopped
  #   2 if daemon could not be stopped
  #   other if a failure occurred
  start-stop-daemon --stop --retry=TERM/30/KILL/5 --pidfile
    $PIDFILE --name $NAME
  RETVAL="$?"
  [ "$RETVAL" = 2 ] && return 2
  # Many daemons don't delete their pidfiles when they exit.
  rm -f $PIDFILE
        rm -f $LOCKFILE
  return "$RETVAL"
}

#
# Function that sends a SIGHUP to the daemon/service
```

```
#
do_reload() {
  #
  # If the daemon can reload its configuration without
  # restarting (for example, when it is sent a SIGHUP),
  # then implement that here.
  #
  start-stop-daemon --stop --signal 1 --quiet --pidfile
    $PIDFILE --name $NAME
  return 0
}

case "$1" in
  start)
  [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "
    $NAME"
  do_start
  case "$?" in
    0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
    2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
  esac
  ;;
  stop)
  [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "
    $NAME"
  do_stop
  case "$?" in
    0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
    2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
  esac
  ;;
  status)
       status_of_proc "$DAEMON" "$NAME" && exit 0 || exit $?
       ;;
  #reload|force-reload)
  #
  # If do_reload() is not implemented then leave this
    commented out
  # and leave 'force-reload' as an alias for 'restart'.
  #
  #log_daemon_msg "Reloading $DESC" "$NAME"
  #do_reload
  #log_end_msg $?
  #;;
  restart|force-reload)
```

```
  #
  # If the ''reload'' option is implemented then remove the
  # 'force-reload' alias
  #
  log_daemon_msg ''Restarting $DESC'' ''$NAME''
  do_stop
  case ''$?'' in
    0|1)
    do_start
    case ''$?'' in
      0) log_end_msg 0 ;;
      1) log_end_msg 1 ;; # Old process is still running
      *) log_end_msg 1 ;; # Failed to start
    esac
    ;;
    *)
      # Failed to stop
    log_end_msg 1
    ;;
  esac
  ;;
  *)
  #echo ''Usage: $SCRIPTNAME {start|stop|restart|reload|force
    -reload}'' >&2
  echo ''Usage: $SCRIPTNAME {start|stop|status|restart|force-
    reload}'' >&2
  exit 3
  ;;
esac
:
```

Литература

1. Simics Installation User Guide 4.8 / Wind River. — 2013.
2. *Сгибнев М.* Настройка NFS сервера и клиента в Debian Lenny. — 2009. — URL: `http://www.opennet.ru/tips/info/2061.shtml` (дата обр. 22.03.2014).

Список TODO

Даннаяя секция предназначена для напоминания авторам, какие задачи по улучшению содержимого книги необходимо выполнить. Всем остальным просьба не обращать внимания.

- Исправить форматирование таблиц.
- Описать установку лицензии Simics в приложении.
- Отформатировать блоки кода: `http://mydebianblog.blogspot.ru/2012/12/latex.html`.

Платформы для симуляции

- DCPU-16 (Mojang)
- CHIP8, CHIP16 (game)
- Z-machine (Zork)
- MIX/MMIX (Knuth)