УДК 658.512.2, ББК 32.81я43 © В. К. Разгоняев
DOI: 10.24411/2225-8264-2020-10022

В. К. Разгоняев

ИСПОЛЬЗОВАНИЕ ВИРТУАЛЬНЫХ КОМПЬЮТЕРОВ ПРИ ПРЕПОДАВАНИИ ДИСЦИПЛИН, СВЯЗАННЫХ С ИНФОРМАЦИОННЫМИ ТЕХНОЛОГИЯМИ

Целью настоящей статьи является обоснование необходимости внедрения технологий виртуализации при обучении программированию. Особенно это полезно при изучении элементов низкоуровневого программирования, что позволяет дать студентам основу для освоения высокоэффективных методов программной разработки. Для достижения поставленной цели решаются следующие задачи: в качестве надежного инструментария для низкоуровневого программирования используется хорошо зарекомендовавшее себя, унаследованное программное обеспечение; основной базой для этого служит использование современных технологий виртуализации, в частности, использование виртуальных компьютеров. В данной работе рассматривается методика применения виртуальных компьютеров в процессе прохождения курсов, связанных с компьютерными технологиями. В качестве иллюстрации использования рассматриваемой методики приводится практический пример изложения темы, связанной с рассмотрением архитектуры микропроцессоров и программированием на языке ассемблера. Чтобы не использовать дорогостоящее программное обеспечение, которое, к тому же, как правило, требует лицензии, применяется унаследованная утилита DEBUG. Данная утилита входит в состав свободно распространяемой операционной системы FreeDOS. Работа в данной системе позволяет учащимся одновременно усвоить необходимые навыки работы с командной строкой. Наиболее эффективным способом работы в операционной системе FreeDOS является установка ее на виртуальный компьютер. В данной работе для этой цели используется виртуальный компьютер Oracle VM VirtualBox. Все используемое программное обеспечение находится в свободном доступе, является бесплатным, не требует лицензии и не занимает больших объемов памяти на компьютере, что очень удобно для образовательных учреждений. На начальном этапе изучения архитектуры процессора наиболее полезна утилита DEBUG, которую можно использовать для ассемблирования и изучения работы небольших фрагментов программ, демонстрирующих программирование простых арифметических выражений. В то же время изучение этих фрагментов позволяет продемонстрировать основные методы использования разных типов регистров на этапе исполнения программ. Использование трассировки программ дает возможность детально изучить, к чему приводит действие каждой команды, и дает наглядное представление о работе микропроцессора. В работе отмечается также наличие двух разных типов виртуальных компьютеров, используемых в образовательном процессе. Одни из них обеспечивают эмуляцию реальных процессоров и компьютерных элементов, другие являются лишь программной реализацией процессоров стекового типа и используются для компиляции высокоуровневых программ в программы для процессоров определенного типа. Приведенное исследование позволяет сделать следующие выводы и дать соответствующие рекомендации. Использование низкоуровневых элементов программирования позволяет заложить прочную основу глубокого понимания сущности разработки программ и возможности реализации высокоэффективных методов программирования. В наиболее полном объеме использовать имеющийся инструментарий программных средств позволяет практическое внедрение виртуальных компьютеров. Исходя из этого рекомендуется как можно шире внедрять в практику использование технологий виртуализации и низкоуровневого программирования.

Ключевые слова: виртуализация, виртуальный компьютер, операционная система, утилита DEBUG, язык Ассемблера, архитектура микропроцессора, регистры.

Интенсивное развитие технологий виртуализации достигло к настоящему времени такого уровня, что позволяет создавать виртуальные компьютеры, применение которых становится экономически выгодным в самых разных областях науки и техники. Особенно это справедливо в сфере образования. Например, если требуется дать учащимся навыки работы в различных операционных системах, то для этого раньше необходимо было иметь несколько компьютеров, на которых установлены разные операционные системы. В настоящее время уже нет такой необходимости. Достаточно на один компьютер установить несколько виртуальных компьютеров с требуемыми операционными системами [4, p.79-98]. В качестве другого примера можно привести изучение компьютерных сетей. Вместо того чтобы закупать несколько компьютеров для создания компьютерной сети, сегодня достаточно установить несколько виртуальных компьютеров на один физический компьютер и изучать сетевые технологии с их помощью. Еще один пример, приводимый ниже, позволил ознакомить учащихся с машинным и ассемблерным кодами, используя утилиту DEBUG, поддержка которой прекращена фирмой Microsoft.

Необходимо отметить два разных направления создания виртуальных компьютеров. Во-первых, использование программной эмуляции для разработки виртуальных компьютеров воспроизводящих работу реальных компьютеров. Среди машин этого типа наиболее известными являются: Oracle VM VirtualBox [6], VMWare Workstation [7] и Windows Virtual PC [8]. Ниже более

подробно будет рассмотрено применение виртуального компьютера Oracle VM VirtualBox. К другому направлению создания виртуальных компьютеров относится создание специализированных виртуальных компьютеров, используемых в качестве промежуточного звена при компиляции программного кода, записанного на высокоуровневом языке, в программу на машинном языке реального микропроцессора. Здесь, в качестве примера, можно привести две основных реализации таких компьютеров: Java Virtual Machine (JVM) [5] и Common Language RunTime (CLR) [1]. Это компьютеры стекового типа, архитектура которых разработана специально таким образом, чтобы облегчить компиляцию программ, созданных с помощью высокоуровневых языков программирования в исполнимый код для специализированных микропроцессоров.

Для лучшего понимания причин и методов использования виртуальных компьютеров в образовании рассмотрим практический пример применения виртуального компьютера Oracle VM VirtualBox при прохождении тем, связанных с низкоуровневым программированием. В процессе подготовки высококвалифицированных ИТ-специалистов, особенно в области разработки программного обеспечения, необходимым условием является овладение ими фундаментальных основ программирования. Наилучшим средством для достижения этого является знакомство учащихся с элементами машинного языка и ассемблерного программирования [2, с.5-9]. На начальных этапах обучения наилучшим инструментом для этого является применение утилиты DEBUG. Данная утилита была создана в период первого появления персональных компьютеров и включалась в состав их базового программного обеспечения, будучи удобным средством для написания простых и средней степени сложности программ для компьютеров на базе микропроцессоров с архитектурой Intel. Кроме того, эта утилита являлась удобным инструментом для знакомства с данной архитектурой и экспериментирования на ее основе в процессе изучения и приобретения навыков работы с ней. Интенсивное развитие компьютерных технологий и быстрая смена поколений персональных компьютеров привело к тому, что фирма Microsoft перестала включать ее в состав современных операционных систем и прекратила ее поддержку. Действительно, современные микропроцессоры Intel являются более мощными и обладают большим набором команд, чем их первые предшественники. Особенно это касается 64-х битных микропроцессоров. В связи с этим оказалось сложным поддерживать совместимость программ, созданных для первых микропроцессоров, при их работе на современных платформах, что и привело к отказу от поддержки этих программ в современных операционных системах. Однако это ни в коей мере не умаляет значение этих программ и, в частности, утилиты DEBUG для применения их в учебных курсах, особенно, если учесть, что фирмы Intel и Microsoft обеспечивают обратную совместимость своих продуктов и базовый набор команд первых процессоров выполняется и современными процессорами. Поэтому изучение команд и методов программирования с помощью утилиты DEBUG является надежным фундаментом для освоения архитектуры современных микропроцессоров Intel.

Поскольку утилита DEBUG предназначена для работы в операционной системе DOS, для ее применения нецелесообразно использовать тяжеловесную операционную систему Windows. К сожалению, все официальные версии Microsoft DOS являются платными, кроме того, их поддержка на сегодняшний день прекращена. С другой стороны, в настоящее время имеется официальная версия операционной системы FreeDOS, которая, к тому же, распространяется бесплатно. Эта версия включает большой набор разнообразных утилит, включая интересующую нас утилиту DEBUG. Операционную систему FreeDOS можно бесплатно скачать с ее официального сайта [3]. Установка ее на виртуальную машину VirtualBox осуществляется стандартным образом. Основные параметры установки видны на рисунке 1.

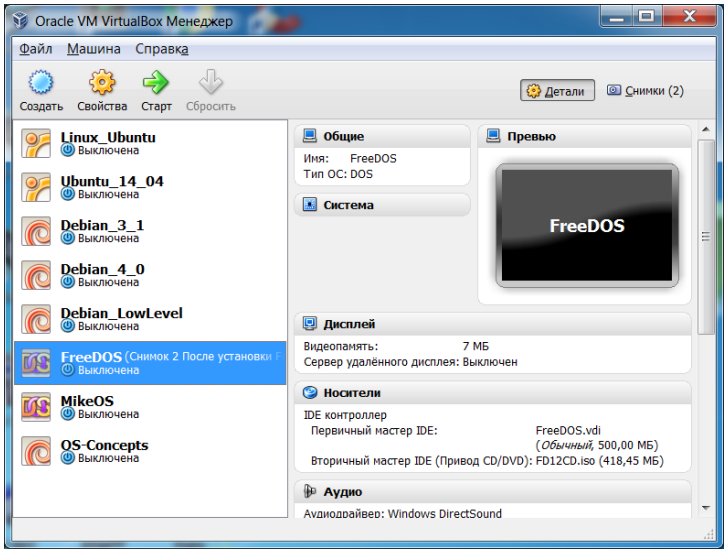

Рис.1. Операционная система FreeDOS, установленная на виртуальную машину VirtualBox.

При создании практико-ориентированных курсов рекомендуется рассматривать как можно большее количество небольших программ, иллюстрирующих фундаментальные основы изучаемого языка и/или технологии. Приведем пример одной из таких программ, который можно использовать на начальных этапах изучения языка Ассемблера для 32-х разрядной архитектуры микропроцессоров Intel семейства x86.

```
mov  ax, 2
mov  bx, 3
add  ax, bx
```

Эта простая программа на языке Ассемблера предназначена для сложения двух целых чисел, но для нас ее ценность состоит в том, чтобы с ее помощью пояснить основные понятия языка Ассемблера и принципы разработки таких программ. Необходимо сразу пояснить, что если писать программу не для утилиты DEBUG, а для Ассемблера, т. е. программы транслирующей исходный код в объектный код, то перед началом программы необходимо вставить ряд директив для настройки программы, а также несколько директив в конце программы. Утилита DEBUG позволяет пропустить такие настройки, что значительно облегчает начинающим освоение языка Ассемблера. Для нас основная ценность данной программы заключается в том, что с ее помощью можно пояснить основные компоненты архитектуры микропроцессора Intel и структуру ассемблерных программ.

Дадим некоторые пояснения к приведенной программе. Программа состоит из трех команд. Первая команда «mov ax, 2» осуществляет загрузку числа 2 в регистр общего назначения ax. Команда с кодом mov (от английского move – переместить) служит для копирования данных между регистрами или между регистром и ячейкой памяти. Отметим, что с точки зрения программиста архитектура микропроцессора представляется набором разнообразных регистров, выполняющих те или иные специфические для них функции. Например, регистры общего назначения ax, bx, cx и dx служат для хранения данных, к которым процессор имеет непосредственный доступ, вследствие чего операции с этими данными выполняются наиболее быстро. Существуют и другие специализированные группы регистров, о которых мы скажем потом. Отметим, что после данной команды следуют два параметра, разделенные запятой: обозначение регистра, в который помещается обрабатываемое значение, и само число, которое необходимо занести в регистр. Поэтому команды такого типа называются двухоперандными командами с непосредственной адресацией. Система команд микропроцессора Intel включает также команды с одним операндом и без операндов, кроме того команды могут использовать разные типы адресации.

Следующая команда «mov bx, 3» осуществляет загрузку числа 3 в регистр общего назначения bx. И, наконец, последняя команда «add ax, bx» (add – с английского сложить) приводит к сложению чисел, находившихся в регистрах ax и bx и, согласно правилам языка Ассемблера, загрузке полученного значения в регистр ax. Отметим, что предыдущее число, находившееся в регистре ax, оказывается утерянным. Поэтому если требуется сохранить данное число для дальнейших вычислений, необходимо предусмотреть в программе дополнительные команды для сохранения числа в каком-либо другом регистре или в ячейке памяти.

Приведенная программа, состоящая из трех команд, записанных на языке Ассемблера, прежде чем быть выполненной должна быть переведена в исполнимый код, состоящий из команд в машинном коде, которые могут быть выполнены непосредственно микропроцессором. Наиболее просто это сделать с помощью утилиты DEBUG, которая в интерактивном режиме позволяет выполнить ряд команд, предназначенных для выполнения в ассемблерных программах. На начальном этапе можно ограничиться лишь некоторыми, наиболее интересными для нас командами.

Такими командами, с нашей точки зрения, являются:

- команда «a» (assemble – перевод). После ввода данной команды утилита DEBUG переходит в интерактивный режим ввода ассемблерных команд. При этом ввод каждой команды должен заканчиваться нажатием клавиши «Enter», после чего, при отсутствии ошибок во введенной команде, она автоматически транслируется в машинный код. Чтобы закончить ввод ассемблерных команд необходимо нажать клавишу «Enter» два раза.
- команда «t» (trace – трассировка). Данная команда приводит к выполнению одной ассемблерной команды, после чего выводится содержимое регистров процессора. Использование данной команды позволяет проследить шаг за шагом процесс выполнения программы и увидеть результат действия каждой команды.
- команда «r» (register – регистр). Данная команда показывает содержимое регистров. Кроме того, с ее помощью можно менять содержимое требуемых регистров.
- команда «q» (quit – выход). Данная команда приводит к окончанию работы с утилитой DEBUG.

На рисунке 2 показан снимок экрана, иллюстрирующий сеанс работы с утилитой DEBUG. Напомним, что все действия выполняются на виртуальном компьютере Oracle VM VirtualBox в операционной системе FreeDOS.

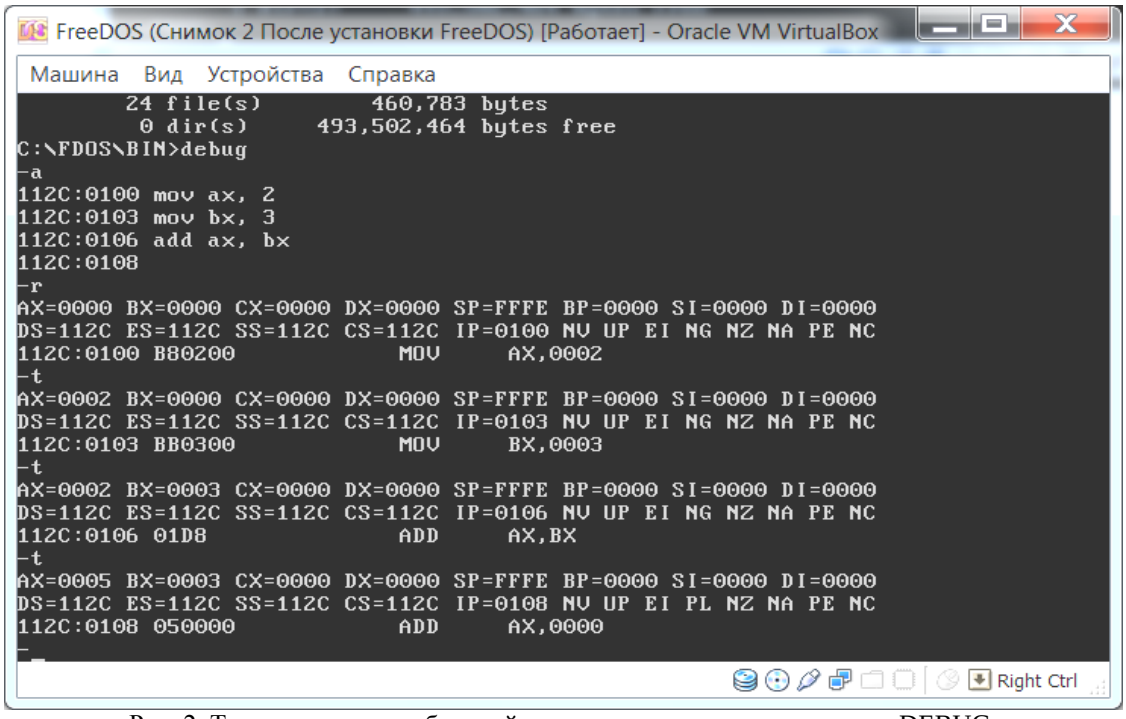

Рис. 2. Трассировка ассемблерной программы с помощью утилиты DEBUG.

На рисунке видно, что после выдачи команды «a», означающей начало ввода ассемблерных команд, вводимые команды располагаются последовательно по адресам, представленным в левой части строки. Необходимо отметить, что каждый адрес состоит из двух частей, разделенных двоеточием. Первая часть адреса идентифицирует начальный адрес сегмента кода, который записан в сегментном регистре кода CS (code segment – сегмент кода). Вторая часть адреса идентифицирует смещение команды от начала сегмента, которое записано в регистре IP (instruction pointer - указатель команд), предназначенном для адресации текущей исполняемой команды.

Кроме того, видно, что по окончании действия команд «r» и «t» печатается строка, содержащая адрес текущей ячейки памяти, содержащийся в ней машинный код команды и соответствующий ей ассемблерный код. Данная информация позволяет наглядно показать связь между ассемблерным и машинным кодами используемых команд.

При выполнении трассировки программы с помощью утилиты DEBUG наглядно видно как меняется содержимое регистра IP при переходе от одной команды к другой.

Необходимо также отметить наличие целого ряда регистров, содержимое которых показывает утилита DEBUG. Кроме упомянутых выше регистров общего назначения или регистров данных имеются также сегментные регистры: CS (code segment – сегмент кода), DS (data segment – сегмент данных), SS (stack segment – сегмент стека) и ES (extended segment – сегмент дополнительных данных) и целый ряд других.

Наличие сегментных регистров указывает на то, что архитектура процессора ориентирована на сегментированную память. Использование сегментированной памяти изначально было обусловлено тем, что в период разработки процессора 8086 персональные компьютеры уже работали с оперативной памятью до 1Мбайта, для адресации которой требовалось 20 адресных линий. Однако чтобы сохранить низкую конкурентную цену, разработчики процессора не хотели увеличивать разрядность регистров свыше 16 бит. Поэтому было решено для формирования двадцати битного адреса использовать два регистра по шестнадцать битов. Один из этих регистров, называемый сегментным, содержал значение, определяющее начало сегмента. Другой регистр содержал величину смещения от начала сегмента. Полный адрес формировался добавлением к шестнадцати битам сегментного регистра четырех двоичных нулей справа и сложением полученного двадцати битного числа с шестнадцати битной величиной смещения. Полученное двадцати битное число позволяло адресовать любую ячейку в пространстве памяти в 1Мбайт.

При изучении данного материала рекомендуется, для лучшего усвоения материала, дать учащимся задание на составление аналогичных программ, иллюстрирующих другие арифметические операции, а для более подготовленных учащихся дать задания на вычисление выражений, содержащих две или три арифметических операции.

Таким образом, использование виртуальных компьютеров является эффективным и недорогим средством решения задачи об установке унаследованного программного обеспечения, необходимого для прохождения отдельных тем в различных курсах информатики. Несмотря на прекращение поддержки, эти программы часто продолжают находить применение в различных областях науки и техники по сегодняшний день. Особенно это справедливо в области образования, где

в процессе обучения необходимо поддерживать и соблюдать преемственность технологий и этапов обучения.

Приводимый пример показывает, как можно эффективно и без дополнительных затрат дать учащимся возможность освоить принципы работы с командной строкой, основы микропроцессорной архитектуры и элементы программирования на языке ассемблера. Дальнейшее применение этих технологий позволяет изучить процесс загрузки операционной системы, структуру загрузочного диска и другие важные и полезные разделы, входящие в курсы низкоуровневого программирования и операционных систем.

Библиографический список

1. Обзор среды CLR [Электронный ресурс] / – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/standard/clr, свободный.

2. Рудаков, П. И. Язык Ассемблера: уроки программирования. [Текст] / П. И. Рудаков, К. Г. Финогенов. – М.: ДИАЛОГ-МИФИ, 2001. – 640 с.

3. FreeDOS [Электронный ресурс] / – Режим доступа: https://www.freedos.org/, свободный.

4. Holcombe, J. Survey of operating systems —3rd ed. / J. Holcombe, C. Holcombe− NY: McGraw-Hill, 2012. – 432 p.

5. Java Virtual Machine (JVM) PC [Электронный ресурс] / – Режим доступа: https://www.java.com/en/download/, свободный.

6. VirtualBox [Электронный ресурс] / – Режим доступа: https://www.virtualbox.org/, свободный.

7.VMware Workstation [Электронный ресурс] / – Режим доступа: https://www.vmware.com/ru/products/workstation-pro/workstation-pro-evaluation.html, свободный.

8.Windows Virtual PC [Электронный ресурс] / – Режим доступа:https://www.microsoft.com/ru-RU/download/details.aspx?id=3702, свободный.

References

1. *Obzor sredi CLR* [Modern Russian: Review of environment CLR] / – Access mode: https://docs.microsoft.com/ru-ru/dotnet/standard/clr

2. Rudakov P. I., Finogenov K. G. *Yazik Assemblera: uroki programirovaniya* [Modern Russian: Assembler language: programming lessons]. 2001. – 640 p.

3. FreeDOS – Access mode: https://www.freedos.org/

4. Holcombe, J. Survey of operating systems —3rd ed. / J. Holcombe, C. Holcombe− NY: McGraw-Hill, 2012. – 432 p.

5. Java Virtual Machine (JVM) PC / – Access mode: https://www.java.com/en/download/

6. VirtualBox – Access mode: https://www.virtualbox.org/

7. VMware Workstation – Access mode: https://www.vmware.com/ru/products/workstation-pro/workstation-pro-evaluation.html

8. Windows Virtual PC – Access mode: https://www.microsoft.com/ru-RU/download/details.aspx?id=3702

USING VIRTUAL COMPUTERS IN TEACHING COURSES OF INFORMATION TECHNOLOGIES

Vladimir C. Razgonyaev,
docent, Omsk, Siberian Institute of Business and Information Technologies

Abstract. In the article is considered methodic of using virtual computers in the process of teaching courses linked with computer technologies. For illustration of using considered methodic the practical example is given lessons about microprocessors' architectures and assembler programming. To avoid using too expensive software there is used legacy utility DEBUG. This utility is found in free distributing operating system FreeDOS. Working in this system can give students the possibility to learn the methods of working with command line interface. The most effective method of working in the operating system FreeDOS is installing it on virtual computer. To reach this aim in this work it is used virtual computer Oracle VM VirtualBox. All used software is found in free access doesn't need any paying and doesn't use much memory. These properties are very useful in education offices. On the first steps of learning microprocessors' architectures is very useful utility DEBUG which may be used for translating small programs' fragments using arithmetic expressions. Using and exploring these fragments makes it possible to demonstrate most important methods of using registers of different types in running programs. Using the tracing of programs makes it possible to learn the performing of each command and get detailed views of microprocessors' working. In the work is also noted that there

are two different types of virtual computers used in the education. There are some emulating real microprocessors and computers, and others used for compiling high-level language programs into programs for microprocessors.

Key words: virtualization, virtual computer, operating system, utility DEBUG, language Assembler, microprocessor's architecture, registries.

Сведения об авторе:

Разгоняев Владимир Константинович – кандидат технических наук, доцент НОУ ВПО «Сибирский институт бизнеса и информационных технологий» (644116, Российская Федерация, г. Омск, ул. 24 Северная, 196/1), e-mail: razgonvc@mail.ru

Статья поступила в редакцию 18.02.2020 г.