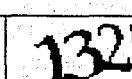
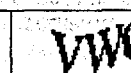
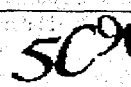
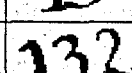
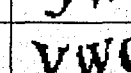
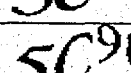
ме собственно вычислительных затрат, требует также дополнительных ресурсов машинной памяти.

3. Апробация алгоритма. Выводы

Разработанный алгоритм был реализован на MS .NET-платформе с использованием объектно-ориентированного языка программирования C# и технологий на базе указателей [3], обеспечивающих необходимую производительность обработки графических изображений. Апробация алгоритма осуществлялась в составе системы, включающей модуль распознавания типа MLP на основе многослойного персептрона, а также модули предварительной и постобработки изображений (масштабирование, удаление шума, бинаризация, определение замкнутых контуров и др.), освещение которых не входит в основную задачу данного исследования.

В таблице приведены примеры сегментации тестовых графических образов, представляющих собой искаженные изображения алфавитно-цифровых последовательностей.

Исходное изображение			
Сегменты			
Результат распознавания	13250	ywcfg	5C9GV
Вероятность соответствия	0,5608	0,9592	0,4847

Невысокие, в ряде случаев, значения итоговой вероятности соответствия распознанных сегментов выходной последовательности символов, по-видимому, связаны с особенностями функционирования (диапазоном выходных значений вероятности) модуля распознавания, так как собственно распознавание во всех приведенных примерах очевидно безошибочное.

Таким образом, предлагаемый алгоритм сегментации имеет приемлемую вычислительную сложность, эффективен и практически пригоден для обработки искаженных символьных последовательностей, а потому перспективен для построения систем обработки растровых изображений без использования векторной графики.

ЛИТЕРАТУРА

1. Прэтт У. Цифровая обработка изображений. – М.: Мир, 1982. – Кн. 2 – 480 с.
2. Абламейко С.В., Лагуновский Д.М. Обработка изображений: технология, методы, применение. – Мн.: Амалфея, 2000. – 304 с.
3. Троелсен Э. C# и платформа .NET 3.0. – С-Пб.: Питер, 2008. – 1456 с.

УДК 004.514.62
Жук А.М.
Научный руководитель: к.т.н., доцент Костюк Д.А.

АДАПТАЦИЯ ЛАБОРАТОРНОГО ПРАКТИКУМА ПО ОСНОВАМ ЯЗЫКА АССЕМБЛЕРА ДЛЯ ОС GNU/LINUX

Все больше различных организаций, в т.ч. в странах СНГ, используют ОС семейства GNU/Linux. В России начался перевод учреждений образования на эту ОС. Главными достоинствами GNU/Linux, относящимися к образовательным учреждениям, являются: полная бесплатность, частые обновления дистрибутивов, обеспечивающие высокий

уровень безопасности программных продуктов, хорошая документированность и поддержка Интернет-сообществом, свободный доступ к исходному коду ПО.

В работе представлено исследование возможности перевода на ОС GNU/Linux начальное обучение студентов низкоуровневому системному программированию на примере языка ассемблер. Проанализированы сложности, связанные с архитектурными отличиями платформы, выполнен анализ имеющихся средств разработки и предложен комплекс программного обеспечения для выполнения лабораторного практикума.

По данным проекта Ohloh [1], занимающегося сбором статистики по исходным кодам программного обеспечения и содержащего на сегодняшний день данные о приблизительно трех миллиардах строк кода двухсот тысяч разработчиков, код на ассемблере содержится в 1 352 активных проектах (по данным на апрель 2009 г.). Активным в данном случае считается проект, частота изменения ассемблерного кода в котором в течение года составляет не менее 1 строки в месяц.

При этом общее число строк на ассемблере в рассмотренных проектах равняется 40 223 602 (31 593 665 без учета комментариев).

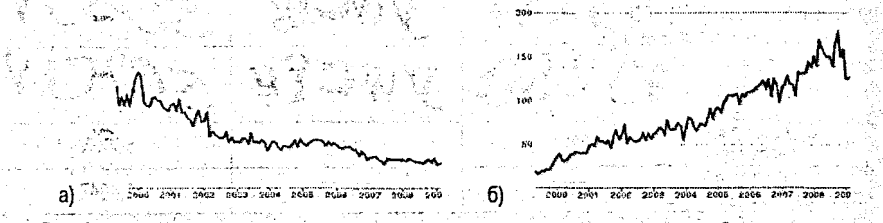


Рис. 1 – Динамика числа программных продуктов с активно изменяемым ассемблерным кодом в процентах от общего количества наблюдаемых проектов (а) и динамика изменения их количества (б)

В целом, как видно из рис. 1-а, доля проектов, использующих ассемблерный код, в последнее десятилетие устойчиво снижается. Данная тенденция может быть объяснена в первую очередь более активным использованием языков программирования высокого уровня в сфере микроконтроллеров для встраиваемых приложений, а также меньшей ролью оптимизации кода в современных прикладных программах. Тем не менее, за счет экстенсивного роста информационных технологий, общее число проектов, активно использующих ассемблер, также демонстрирует неуклонный рост, хоть и не такой быстрый, как для языков высокого уровня (соответствующую зависимость демонстрирует рис. 1-б).

В ряде случаев роль ассемблера остается по-прежнему важной: при разработке драйверов устройств и некоторых других аппаратно-зависимых частей операционных систем, при создании компиляторов и интерпретаторов языков программирования, при решении задач, требующих жесткой оптимизации критических участков кода, и др.

Неверным было бы забывать и о неизменной важности образовательной миссии языка ассемблер. Помимо востребованности низкоуровневых программистов на рынке, навыки программирования на ассемблере важны для понимания внутренней структуры и принципов работы микропроцессорной техники. Составление и отладка ассемблерных программ является единственным способом отслеживания особенностей внутреннего устройства и функционирования процессоров, знание основ ассемблера увеличивает способность программиста генерировать качественный и эффективный код на языках высокого уровня, избавляет производимые им программные продукты от ряда «узких мест» производительности и, иногда, уменьшает число потенциальных уязвимостей в коде [2, 3].

В настоящее время программирование на ассемблере рассматривается в учебных программах в основном на примере ОС DOS. Такой выбор программной платформы вызван историческими причинами (в т.ч. большим числом учебных пособий по ассемблеру,

написанных во время популярности данной ОС), а также заметно возросшей сложностью низкоуровневого программирования для сменившей ее ОС Windows [4].

Вместе с тем нельзя не отметить, что в настоящее время различные версии DOS используются в основном на компьютерах, произведенных более 15 лет назад. Прикладное программное обеспечение, разработанное для DOS и сохранившее актуальность до настоящего времени, обычно применяется на современных операционных системах в режиме эмуляции DOS либо в виртуальных окружениях. Определить число компьютеров, использующих DOS в качестве основной ОС, довольно сложно. Но, т.к. данная ОС не присутствует в статистике, собираемой поисковыми серверами среди компьютеров, подключенных к Интернет, можно предполагать, что доля таких вычислительных систем составляет мене 0.01%. В связи с этим поиск альтернативных программных платформ для преподавания языка ассемблера остается актуальной задачей.

В качестве наиболее подходящей платформы для перевода лабораторных практикумов по основам программирования на языке ассемблер нами выбрана ОС GNU/Linux. Выбор платформы обоснован как факторами роста ее популярности, так и простотой программирования. Рассмотрим подробнее оба фактора.

Открытый доступ к исходному коду и связанные с ним технические и экономические преимущества значительно облегчают адаптацию ОС для самых различных платформ. Этим объясняются сильные позиции GNU/Linux на рынке специализированных устройств. Данная ОС используется в потребительской электронике, телекоммуникационном оборудовании, медицинских системах, автомобилях, вооружении и др. По данным Venture Development Corporation, Linux является наиболее популярной системой для встраиваемых устройств, занимая более 15% рынка по состоянию на январь 2007 г., существенно опережая ближайшего конкурента, vxWorks, доля которого составляет 10.3%. Позиции ОС на рынке мобильных устройств также достаточно сильны и еще более укрепились с появлением Linux-платформы фирмы Google, объявившей в апреле 2009 г. о продаже миллионного телефона со своей версией ОС. В целом, по данным компании ABI Research, к 2012 г. прогнозируется увеличение числа проданных портативных устройств, работающих под управлением GNU/Linux, до 203 млн. Наибольшее распространение ОС на данном рынке наблюдается в Азиатско-Тихоокеанском регионе, по различным причинам: как из-за богатых функциональных возможностей ОС, актуальных для японских пользователей, так и из-за ценовых преимуществ и государственной политики в случае Китая.

По данным Gartner [5], общий объем рыночных прибылей от продаж компьютерного оборудования и ПО для Linux в 2008 г. – порядка 35.7 млрд. долл. при ежегодном темпе роста в 26%. Рассматриваемая ОС занимает более прочные позиции на рынке серверов, однако рост присутствует также и на рынке настольных ПК.

В ряде стран, включая Евросоюз, выполняются государственные программы, направленные на замещение GNU/Linux и сопутствующей инфраструктурой ПО аналогичных проприетарных продуктов в госсекторе, обоснованные как экономическими преимуществами, так и простотой выполнения аудита кода на предмет выявления потенциальных уязвимостей либо нежелательных в данной сфере недокументированных возможностей. Таким образом, во всех сферах применения наблюдается и прогнозируется рост доли платформы, предлагаемой в качестве замены для обучения низкоуровневому программированию.

Перевод существующих лабораторных практикумов на новую платформу по определению сопряжен с трудностями из-за возникающих архитектурных различий. Так, лабораторные работы, связанные с использованием директив управления сегментами и функций BIOS, а также построенные на применении резидентных технологий, задействуют особенности 16-битной архитектуры Intel и разработанных для нее в 80-е годы про-

граммных продуктов компании Microsoft. На 32-битных и, тем более, 64-битных системах данные работы не могут выполняться в естественных условиях (т.н. native mode) вне зависимости от того, идет ли речь о GNU/Linux или о любой другой из современных ОС. Напротив, лабораторные работы, связанные с использованием системных вызовов ОС, могут быть расширены и разбиты на несколько частей в связи с тем, что на данной платформе на это делается упор при программировании.

Пользовательская программа не имеет возможности получить полный доступ к аппаратным ресурсам компьютера, наподобие программирования на ассемблере в DOS и в ранних версиях Windows. Это связано с архитектурными отличиями ОС семейства GNU/Linux. Самым низким уровнем работы прикладной программы является обращение напрямую к ядру ОС. Именно на этом уровне и работают программы, написанные на ассемблере в Linux.

В отличие от служб DOS, доступных по прерыванию 21h, и функций BIOS прерываний 10h и 16h, широко применяемых при низкоуровневом программировании для DOS, все системные вызовы Linux доступны по прерыванию 80h. При этом их значительно меньше, чем в WinAPI, т.к. в отличие от Windows, ядро Linux не включает в себя значительные объемы кода, связанные с графическим интерфейсом пользователя, отводя эту роль сторонним программным средствам [6]. В самом ядре Linux ассемблер используется для ряда критичных участков кода, а также в значительном числе драйверов, однако большая часть ОС написана на языке С, часто уже знакомом студентам к моменту изучения ассемблера. Использование базового языка С, унаследованные традиции ОС Unix и стандарта POSIX значительно облегчают изучение системных вызовов Linux.

DOS	Linux
proc fileWrite☐ mov ah,40h☐ mov bx,[filehandle] ☐ mov cl,[*buffLen*]☐ mov dx,offset *buff*☐ int 21h☐ ret☐endp fileWrite ... call fileWrite	fileWrite:☐ mov eax, 4 ☐ mov ebx, [*desc*]☐ mov ecx, *buf*☐ mov edx, [*bufLen*]☐ int 80h☐ ret☐ ... call fileWrite

Рис. 2 – Сравнение ассемблерного кода платформ DOS и GNU/Linux

Обучение программированию на выбранной платформе облегчает также то, что интерфейс обращения к системным вызовам Linux строго унифицирован. В системе определен один единственный порядок размещения аргументов в регистрах, обязательный для всех вызовов функций ядра. В этом отношении Linux является современным ядром ОС, ориентированным на простоту освоения и доработки и хорошо документированным (на английском языке). Сравнительный пример участка ассемблерной программы для платформ DOS и GNU/Linux приведен на рис. 2 (в обоих случаях приведена одна и та же процедура, выполняющая вывод в файл).

При выборе программного инструментария для выполнения лабораторных работ был проанализирован ряд ассемблеров, доступных на платформе Linux, включая gas, as86, NASM и FASM. Критериями выбора являлись степень документированности, качество поддержки, удобство синтаксиса и использования утилит, портируемость программного кода. Анализ показал преимущества ассемблера NASM по совокупности приведенных показателей. Это 32- и 64-битный ассемблер, созданный с расчетом на переносимость и расширяемость. Он поддерживает широкий спектр форматов объектных файлов, включая стандартные для Linux и *BSD a.out, ELF, COFF, Mach-O, 16-битные OBJ ОС Майкрософт, а также Win32 и Win64. В качестве выходного файла может быть использован простой бинарный файл. Синтаксис NASM ориентирован на традиционный

синтаксис ассемблеров фирмы Intel. NASM позволяет создавать программы на всех известных на сегодня платформах на базе архитектуры x86 и имеет хорошую поддержку макросов. Из отличий, как правило, выделяют то, что синтаксис ассемблера NASM является регистрочувствительным – директивы набираются прописными литерами, а инструкции – строчными.

Из-за своей специфики, а также по традиции, для программирования на языке ассемблера существует мало интерактивных сред программирования, но есть дополнительные модули к универсальным средам – Eclipse, NetBeans и пр., облегчающие программирование и отладку на этом языке. Традиционно, однако, в лабораторных работах по программированию на языке ассемблера используются автономный текстовый редактор и утилиты командной строки.

При выборе текстового редактора был проанализирован ряд программных продуктов и сделан выбор в пользу программы mcedit благодаря минимальным отличиям от консольных редакторов, привычных для студентов, и возможности автоматической подсветкой синтаксиса программ.

В качестве средства визуальной отладки выбран отладчик DDD (Data Display Debugger), разрабатывавшийся длительное время различными организациями, компаниями и университетами. Он является графической оболочкой (front-end) к различным консольным отладчикам и использует их для работы. В качестве последнего был выбран отладчик GDB (Gnu DeBugger), активно применяемый при разработке программного обеспечения в сфере встраиваемых систем и микроконтроллеров [7].

ЛИТЕРАТУРА

1. Assembly programming language statistics – Ohloh.http://www.ohloh.net/languages/19. – Доступ 28.04.2009.

2. Пирогово В.Ю. Assembler для Windows. – М.: Изд-во Молгачева, 2005. – 552 с.

3. Магда Ю.С. Использование ассемблера для оптимизации программ на C++. СПб.: БХВ-Петербург, 2004. – 492 с.

4. Магда Ю.С. Ассемблер. Разработка и оптимизация Windows-приложений. СПб.: БХВ-Петербург, 2003. – 544 с.

5. Brown E. Linux spending will defy recession, IDC claims. http://www.linuxfoundation.org/sites/main/files/publications/Linux_in_New_Economy.pdf. – Доступ 13.04.2009.

6. Таненбаум Э. Современные операционные системы, – СПб.: Питер, 2002. – 1040 с.

7. Debugging with DDD. 15/01/2004. http://www.gnu.org/manual/ddd/

УДК 004.514.62

Жук А.М.

Научный руководитель: к.т.н., доцент Костюк Д.А.

АНАЛИЗ АРХИТЕКТУР МОДУЛЕЙ ЯДРА СОВРЕМЕННЫХ ОС ДЛЯ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ АССЕМБЛЕР

В настоящее время программирование на ассемблере практически повсеместно преподается на примере ОС DOS. Такой выбор платформы вызван наличием большого числа учебных пособий по ассемблеру, написанных во время популярности DOS, а также ее относительной простотой по сравнению с более современными ОС.

Вместе с тем в настоящее время программное обеспечение, написанное для DOS, применяется все реже, и практически всегда – на более современных операционных системах в режиме совместимости или эмуляции. Определить число компьютеров, использующих DOS в качестве основной ОС, представляется крайне затруднительным, но т.к. данная ОС не присутствует в статистике, собираемой среди компьютеров, подклю-