

Министерство сельского хозяйства РФ
ФГБОУ ВПО «КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ
АГРАРНЫЙ УНИВЕРСИТЕТ»

Факультет прикладной информатики
Кафедра компьютерных технологий и систем

МИКРОПРОЦЕССОРЫ

Методические рекомендации
к выполнению курсовой работы для бакалавров специальности
230400.62 – «Информационные системы и технологии»

Краснодар
КубГАУ
2015

Составители: А. В. Параскевов, А. Г. Дмитриева, А. Н. Бардак, В. И. Лойко, С. А. Курносов

Микропроцессоры: метод. рекомендации к выполнению курсовой работы / сост. А. В. Параскевов [и др.]. – Краснодар: КубГАУ, 2015. – 57 с.

В методических указаниях изложены цели и задачи подготовки курсовой работы, основы составления плана, подбора литературных источников, примерная тематика курсовых работ, требования к оформлению курсовой работы и библиографическому описанию используемой литературы, а также содержится другая информация, необходимая для подготовки работы, отвечающей предъявляемым к ней требованиям.

Методические указания предназначены для бакалавров специальности 230400.62 – «Информационные системы и технологии».

Рассмотрено и одобрено методической комиссией факультета прикладной информатики Кубанского госагроуниверситета, протокол №3 от 24.11.2014.

Председатель
методической комиссии

Е. А. Иванова

© ФГБОУ ВПО «Кубанский
государственный аграрный
университет», 2015

Оглавление

Введение	4
Сбор научной информации по теме исследования, ее предварительный анализ и составление плана курсовой работы	7
Анализ собранного материала, изложение темы	9
Структура и объем пояснительной записки курсовой работы	11
Содержание курсовой работы и требования к ее изложению	12
Примерный постраничный объем составных частей курсовой работы	15
Оформление курсовой работы.....	16
Критерии оценки курсовой работы	21
Глоссарий.....	23
Примерная тематика курсовых работ по дисциплине «Микро-процессоры»	28
Список литературы	30
Приложение 1 Титульный лист	32
Приложение 2 Лист задания	33
Приложение 3 Реферат	34
Приложение 4 Содержание	35
Приложение 5 Пример заполнения курсовой работы.....	36

Введение

Курсовая работа — это самостоятельная учебно-исследовательская работа студента, выполненная под руководством преподавателя, одна из основных форм учебных занятий и форм контроля учебной работы студентов.

Важным условием подготовки высококвалифицированных специалистов является их самостоятельная творческая работа, к которой следует отнести написание курсовых работ. Курсовая работа — один из важнейших показателей результатов обучения, в определенной мере свидетельствующий о степени овладения учебным материалом, об уровне подготовленности обучающегося по той или иной дисциплине, его кругозора, грамотности и общей культуры.

Курсовая работа должна выполняться в тесной связи с реальной практикой, базироваться на конкретном фактическом материале, носить исследовательский характер и способствовать развитию творческого потенциала обучаемых (студентов).

Выполнение курсовой работы является очень важным компонентом учебного процесса. Ее написание способствует формированию у студента навыков самостоятельного научного исследования, интереса к углубленному изучению предмета — в данном случае, дисциплины «Микропроцессоры». Благодаря подобной научной деятельности углубляются и закрепляются уже полученные знания, приобретаются новые, развивается творческое мышление, вырабатываются навыки письменного изложения студентом своих мыслей с использованием специальной технической терминологии.

Тема курсовой работы определяется по выбору самого студента, желательно, после консультации с преподавателем, ведущим практические занятия или читающим лекции: это

может быть как одна из предлагаемых в примерном перечне тем, так и иная (обязательно – в рамках учебной программы), в последнем случае необходимо согласие преподавателя.

При выполнении и защите курсовой работы студент должен выполнить следующие пункты:

- *выбрать тему и определить руководителя курсовой работы;*
- *получить задание на выполнение курсовой работы;*
- *выполнить подбор литературы, ее систематизацию для анализа;*
- *составить план и сформировать структуру курсовой работы;*
- *провести необходимые исследования, и реализацию результатов курсовой работы, обоснование своей точки зрения по исследуемому вопросу;*
- *оформить работу, составить библиографический список использованных источников и литературы;*
- *выполнить и оформить пояснительную записку к курсовой работе;*
- *представить работу на проверку руководителю курсовой работы;*
- *зарегистрировать курсовую работу на кафедре;*
- *защитить курсовую работу у руководителя.*

Особо рекомендуется обратить внимание на новейшие взгляды на анализируемую проблему, для чего нужно ознакомиться с последними вышедшими в свет монографиями, журнальными статьями и т. д.

Курсовая работа должна быть написана студентом лично, самостоятельно. Категорически запрещено переписывание первоисточников без ссылки на них: при выявлении преподавателем плагиата в работе она возвращается студенту для повторного написания.

Тема курсовой работы должна быть раскрыта, причем сам по себе размер работы не является показателем этого. Студен-

ту необходимо грамотно, логически верно, с использованием необходимой литературы и материалов практики рассмотреть все поставленные в плане вопросы.

При выборе темы учитывается ее актуальность, интерес студента к данной теме, наличие необходимой литературы. Выбор одной и той же темы студентами в пределах одной подгруппы не допускается. Студент вправе предложить свою собственную тему исследования, отсутствующую в предложенном перечне, согласовав её с руководителем.

Сбор научной информации по теме исследования, ее предварительный анализ и составление плана курсовой работы

Выполнение курсовой работы необходимо начинать с предварительного подбора научной информации по теме исследования (учебной и монографической литературы, статей, опубликованных в периодической печати и т.д.). В начале, целесообразно ознакомиться с соответствующим разделом учебников по дисциплине, понять содержание темы, определить её место и значение в изучаемом курсе. Далее нужно ознакомиться с литературой, имеющейся в доступных студенту библиотеках и сетевых ресурсах. При этом не следует ограничиваться поиском только заранее намеченной литературы, а необходимо полностью просмотреть соответствующий раздел каталога библиотеки.

Список использованной литературы должен быть полным и включать основополагающие монографические работы, учебники и учебные пособия. После предварительного ознакомления с литературой по теме исследования студент составляет план курсовой работы.

Правильно составленный план является одной из важнейших составляющих успешного написания курсовой работы. Наличие плана курсовой работы позволяет осветить в ней только те вопросы, которые относятся к теме, обеспечить чёткость и последовательность в изложении материала, избежать пробелов и повторений, научно организовать самостоятельный труд, сэкономить время.

План должен состоять из введения, нескольких глав и заключения. Для более чёткого определения круга вопросов, которые необходимо рассмотреть, главы работы можно разделить на параграфы (подпункты). При этом следует помнить,

что включение в план большого количества вопросов может привести к перегруженности работы, чрезмерному увеличению её объёма, к повторениям или поверхностному раскрытию вопросов.

Студенту рекомендуется согласовать план с научным руководителем, так как неудачно составленный план может свести на нет всю последующую работу. После уточнения плана необходимо оценить достаточность подобранной учебной и монографической литературы, статей, опубликованных в периодической печати и в случае необходимости скорректировать данный список, исключив ненужные источники либо добавив дополнительную литературу.

Анализ собранного материала, изложение темы

После подбора литературы и составления плана студент приступает к самой важной стадии выполнения курсовой работы — анализу собранного материала и изложению темы. Содержательная часть курсовой работы состоит из введения, основной части и заключения. Во введении автор курсовой работы должен: обосновать актуальность темы, её теоретическую и практическую значимость, сформулировать цель и задачи курсовой работы. Цель курсовой работы должна вытекать из названия работы. Она состоит в рассмотрении наиболее важных дискуссионных вопросов отдельной темы, недостаточно изученных проблем.

Задачи производны от цели курсовой работы. При определении задач курсовой работы следует отметить составные части рассматриваемой темы, анализ которых необходим и достаточен для достижения цели. Объём введения не должен превышать 2-4 листов. В основной части работы последовательно раскрываются поставленные вопросы. В работе должно быть показано глубокое понимание сущности избранной темы, знание используемых источников, умение сопоставлять различные мнения и делать необходимые личные обобщения и выводы. Курсовая работа обязательно должна содержать примеры, используемые для иллюстрации теоретических положений.

Написание курсовой работы — самостоятельное, систематизированное, логически завершённое, творческое изложение студентом в соответствии с планом основных сведений по избранной теме, отражающее его понимание определенных научных проблем. *Заимствование текста из чужих источников без соответствующей ссылки на них не допускается и влечет возвращение курсовой работы на доработку.*

В заключении автор подводит итоги сделанной работы, формулирует основные выводы и предложения. Обобщения и выводы необходимо излагать кратко и своими словами. Заключение должно быть связано с целью и задачами работы. Выводы рекомендуется оформить в виде пронумерованных абзацев. Объём заключения не должен превышать 2-3 листов. Объём курсовой работы должен составлять 23-40 листов (без учёта приложений, если таковые имеются).

Структура и объем пояснительной записки курсовой работы

В пояснительную записку курсовой работы должны быть включены, в заданной последовательности, следующие структурные элементы:

- титульный лист (Приложение 1);
- задание на выполнение курсовой работы (Приложение 2);
- реферат (Приложение 3).

Реферат должен содержать:

- Сведения об объеме курсовой работы, количестве иллюстраций, таблиц, приложений, количестве использованных источников.

- Перечень ключевых слов:

Перечень ключевых слов должен включать от 5 до 15 слов или словосочетаний из текста курсовой работы, которые в наибольшей мере характеризуют ее содержание. Ключевые слова приводятся в именительном падеже и печатаются строчными буквами в строку через запятые.

- Текст реферата:

Текст реферата должен отражать цель исследования, методы исследования, полученные результаты курсовой работы, область применения результатов курсовой работы.

Содержание курсовой работы и требования к ее изложению

- **Содержание:**

Содержание должно включать наименования структурных элементов (введение; заключение; список использованных источников, приложения и др.), номера и заголовки разделов, подразделов и пунктов основной части с указанием номеров страниц, с которых начинаются эти структурные элементы, разделы, подразделы, пункты курсовой работы (Приложение 4).

- **Введение:**

Введение должно содержать:

- оценку современного состояния и актуальности исследуемой темы;
- определение цели и задач исследования. Цель работы должна быть сформулирована четко и лаконично, соответствовать выбранной теме исследования, отражать те действия, которые студент должен предпринять для написания курсовой работы;
- основные исходные данные для выполнения курсовой работы (см. Приложение 5).

.

- **Основная часть:**

1. Постановка исследуемой темы.
2. Выбор, анализ и оценка возможных способов исследования темы курсовой работы.
3. Содержание исследования, проблемные характеристики функционирования разработанной программы.

Основная часть должна состоять из 2-3 логически связанных и соподчиненных глав (разделов), каждая из которых подделяется на несколько частей (подразделов). В этих главах на основе изучения работ отечественных и зарубежных авторов излагается сущность исследуемой проблемы, раскрывается ее содержание. Далее в основной части курсовой работы выполняется квалифицированный анализ состояния исследуемой проблемы на настоящее время, рассматриваются различные подходы к ее решению, дается их оценка. При этом студент не ограничивается констатацией фактов, а выявляет тенденции развития, вскрывает недостатки и причины, их обусловившие, намечает пути их возможного устранения. На основе теоретических положений и обобщений существующих точек зрения автор курсовой работы должен выразить свое отношение, обосновав собственную точку зрения по данному вопросу (или принять чью-либо точку зрения с обоснованием такого решения). В основной части курсовой работы желательно приводить взятые из периодической печати примеры, подтверждающие актуальность рассматриваемой темы (см. Приложение 5).

- **Заключение:**

Заключение содержит выводы и предложения, к которым приходит автор в результате проведенного исследования. Поэтому заключение пишется после того, как написана вся работа. Оно не является продолжением основного исследования, а служит кратким, но логичным изложением взглядов автора на важность рассматриваемой проблемы, новые определения, идеи, пути решения анализируемых проблем, предложения.

В заключении должны найти отражение основные результаты решенных задач, заявленных во введении, а также выводы и предложения по всей работе в целом (см. Приложение 5).

- **Список используемых источников:**

Список используемых источников представляет собой перечень использованных книг, учебников, самоучителей, мето-

дических пособий статей электронных ресурсов, конспектов лекций, которые были использованы при работе над курсовой работой. Фамилии авторов приводятся в алфавитном порядке, при этом все источники даются под общей нумерацией литературы. В исходных данных источника указываются фамилия и инициалы автора, название работы, место и год издания (см. Приложение 5).

- Приложения:

В приложения рекомендуется включать материалы, связанные с выполненной дипломной работой, которые не могут быть включены в основную часть:

- части прокомментированных листингов программ, блок-схемы алгоритма работы программы и др.

Примерный постраничный объем составных частей курсовой работы

Таблица 1 – Структура курсовой работы

№ п/п	Структурные элементы работы	Объем структурных эле- ментов курсовой работы (листов)
1	Титульный лист	1
2	Задание на выполнение курсовой работы	1 - 2
3	Реферат	1 – 2
4	Содержание	1 – 2
5	Введение	2 – 4
6	Раздел 1	3 – 7
7	Раздел 2	5 – 8
8	Раздел 3	7 – 10
9	Заключение	1 – 2
10	Список используемых источников	1 – 2
11	Приложения	не более 5

Оформление курсовой работы

Текстовая часть курсовой работы должна быть напечатана на одной стороне листа белой бумаги формата А4 с размерами полей: правое – 10 мм; верхнее, нижнее – 20мм; левое 30мм.

Текст набирается на компьютере с применением программных средств Microsoft Office, нежирным шрифтом Times New Roman размером 14 пунктов через полуторный межстрочный интервал, с размером абзацного отступа: 15мм. Цвет шрифта: черный.

Математические формулы должны быть записаны с использованием редактора формул Microsoft Equation.

Иллюстрации (схемы, рисунки, диаграммы и др.) и таблицы, помещенные в текстовой части, должны быть подготовлены программными средствами MS Word или Microsoft Visio.

Основная часть курсовой работы подразделяется на разделы и подразделы.

Разделы должны иметь порядковую нумерацию в пределах основной части курсовой работы и должны иметь свои заголовки. Номер раздела обозначается прописной арабской цифрой, ставится перед заголовком раздела без точки между номером и заголовком раздела. Заголовок пишется с прописной буквы, после заголовка раздела точка не ставится. Номер и заголовок раздела следует набирать нежирным шрифтом и начинать с абзацного отступа по ширине поля листа без подчеркивания.

Подразделы должны иметь порядковую нумерацию в пределах одного раздела и должны иметь свои заголовки. Номер подраздела включает номер раздела и порядковый номер подраздела в разделе, обозначается прописными арабскими цифрами, разделенными точкой. Номер и заголовок подраздела следует печатать также как номер и заголовок раздела.

Структурные элементы (Титульный лист, Реферат, Содержание, Введение, Заключение, Список используемых источников, Приложения) пояснительной записки не нумеруются, после заголовков точки не ставятся.

Структурные элементы (Титульный лист, Реферат, Содержание, Введение, Заключение, Список используемых источников, Приложения) и главы курсовой работы должны начинаться с нового листа (при этом настоятельно рекомендуется использовать команду «Вставка»-«Разрыв страницы», чтобы текст не смещался во время последующей правки).

Заголовки структурных элементов, номера и заголовки разделов должны быть напечатаны с абзацным отступом по верхнему полю листа.

Между заголовком структурного элемента и следующим за ним текстом пропускается одна строка.

Формулы и уравнения следует выделять из текста в отдельную строку и выравнивать по ширине. Выше и ниже каждой формулы или уравнения должно быть оставлено не менее одной свободной строки. Если уравнение не умещается в одну строку, то оно должно быть перенесено после математических знаков, причем знак в начале следующей строки повторяют.

Пояснение значений символов и числовых коэффициентов следует приводить непосредственно под формулой в той же последовательности, в которой они приведены в формуле.

Формулы и уравнения, помещаемые в курсовой работе, должны нумероваться в пределах каждого раздела. Номер формулы и уравнения должен включать номер раздела и порядковый номер формулы или уравнения в разделе, должен быть записан прописными арабскими цифрами с точкой между номером раздела и порядковым номером формулы или уравнения, помещен в скобках с выравниванием по правому полю листа на уровне середины формулы или уравнения.

Иллюстрации (рисунки, графики, схемы, диаграммы и др.), помещенные в курсовой работе, должны иметь обозначения, нумерацию и наименования.

Обозначение всех видов иллюстраций осуществляется словом «Рисунок». Рисунки подписываются снизу.

Нумерация иллюстраций должна осуществляться в пределах каждого раздела. Номер иллюстрации должен включать номер раздела и порядковый номер иллюстрации в разделе, должен быть напечатан прописными арабскими цифрами с точкой между номером раздела и порядковым номером иллюстрации в разделе. Наименование иллюстрации должно отражать его вид и содержание. Иллюстрации следует располагать непосредственно после текста, в котором они впервые упоминаются. Таблицы, помещаемые в иллюстрационной записке, должны иметь обозначения, номера и названия.

Например:

Рисунок 1.1 – Блок-схема алгоритма SPT.

Данный рисунок будет являться первым рисунком в первой главе и должен отображать блок-схему работы алгоритма SPT.

Обозначение всех видов таблиц осуществляется словом «Таблица».

Нумерация таблиц должна осуществляться в пределах каждого раздела. Номер таблицы должен включать номер раздела и порядковый номер таблицы в разделе, должен быть напечатан арабскими цифрами с точкой между номером раздела и порядковым номером таблицы в разделе. Название таблицы должно отображать её содержание.

Таблицы следует располагать непосредственно после текста, в котором она упоминается в первые, или на следующей странице.

Таблицу с большим количеством строк допускается переносить на следующий лист. При переносе части таблицы на другой лист справа над продолжением таблицы пишут слово «Продолжение» и номер таблицы.

Например:

Таблица 2.3 – Описание основных элементов объектной модели.

Таблицы подписываются сверху.

Список используемых источников должен содержать сведения о применяемых, цитируемых и обсуждаемых в пояснительной записке законодательных, нормативных, научных, учебных и периодических изданиях. Сведения о каждом издании вносятся на основе его издательских данных.

Перечень сведений обо всех использованных изданиях оформляется в виде списка использованных источников.

Для оформления списка использованных источников необходимо выполнить следующее:

- определить перечень всех использованных источников;
- распределить и упорядочить перечень использованных источников на следующие группы:
 - государственные законодательные акты, упорядоченные по их важности и по датам их принятия;
 - государственные стандарты, упорядоченные по возрастанию номеров их обозначений;
 - ведомственные нормативные документы, упорядоченные по датам их принятия;
 - научные, учебные и периодические издания, упорядоченные в алфавитном порядке;
- провести сквозную последовательную порядковую нумерацию использованных источников всех выделенных групп, начиная с государственных законодательных актов.

Ссылки на использованные источники выполняются по их присвоенным порядковым номерам.

Страницы курсовой работы следует нумеровать в следующем порядке:

- Титульный лист, Содержание, Задание на выполнение курсовой работы, Реферат, Содержание, Разделы основной части, Заключение, Список используемых источников и Приложения входят в общую нумерацию страниц пояснительной записки;

- номера страниц курсовой работы проставляются начиная со второй страницы Введения и далее нумеруются сквозной нумерацией все листы и приложения к курсовой работе;

- страницы курсовой работы следует нумеровать арабскими цифрами, номер страницы проставляют в центре нижней части листа без точки.

Критерии оценки курсовой работы

Оценка **«отлично»** выставляется, если тема курсовой работы раскрыта в полной мере, работа выполнена самостоятельно, содержит анализ практических проблем. Представленный материал работы свидетельствует о глубоком понимании автором рассматриваемых вопросов. Изложение материала работы отличается логической последовательностью, наличием иллюстративно-аналитического материала (таблицы, диаграммы, блок-схемы и т.д.), ссылок на литературные источники, завершается конкретными выводами.

Курсовая работа оформлена аккуратно, в соответствии с предъявленными требованиями.

Оценка **«хорошо»** выставляется, если раскрыто основное содержание темы, работа выполнена преимущественно самостоятельно, содержит анализ практических проблем. Представленный в ней материал свидетельствует о достаточно глубоком понимании автором рассматриваемых вопросов. Изложение материала работы отличается логической последовательностью, наличием иллюстративно-аналитического материала (таблицы, диаграммы, схемы и т.д.), ссылок на литературные и нормативные источники, завершается конкретными выводами. Имеются недостатки, не носящие принципиального характера. Курсовая работа оформлена аккуратно, в соответствии с предъявленными требованиями. Имеются незначительные ошибки в оформлении работы.

Оценка **«удовлетворительно»** выставляется, если тема курсовой работы раскрыта частично, работа выполнена в основном самостоятельно, содержит элементы анализа реальных проблем. Не все рассматриваемые вопросы изложены достаточно глубоко, есть нарушения логической последовательности, ограниченно применяется иллюстративно-аналитический

материал (таблицы, схемы и т.д.), отсутствуют ссылки на литературные источники. В работе допущено большое количество ошибок и опечаток.

Оценка **«неудовлетворительно»** выставляется, если не раскрыта тема курсовой работы. Материал изложен неграмотно, без логической последовательности.

Глоссарий

Адресация памяти (addressing mode) – осуществление ссылки (обращение) к устройству или элементу данных по его адресу; установление соответствия между множеством однотипных объектов и множеством их адресов; метод идентификации местоположения объекта.

Активационная запись (activation record) – область стека, заполняемая при вызове процедуры.

Ассемблер (assembly language) – язык программирования низкого уровня.

Ассемблер (assembler) – компилятор с языка ассемблера.

Байт (byte) – тип данных, имеющий размер 8 бит, минимальная адресуемая единица памяти.

Бит (bit) – минимальная единица измерения информации.

«Всплывающая» программа (popup program) – резидентная программа, активирующаяся по нажатию определенной «горячей» клавиши.

«Горячая» клавиша (hotkey) – клавиша или комбинация клавиш, используемая не для ввода символов, а для вызова программ и подобных необычных действий.

Двойное слово (double word) – тип данных, имеющий размер 32 бита.

Дескриптор (descriptor) – восьмибайтная структура, хранящаяся в одной из таблиц GDT, LDT или IDT и описывающая сегмент или шлюз.

Директива (directive) – команда ассемблеру, которая не соответствует командам процессора.

Драйвер (driver) – служебная программа, выполняющая функции посредника между операционной системой и внешним устройством.

Защищенный режим (protected mode) – режим процессора, в котором действуют механизмы защиты, сегментная адресация с дескрипторами и селекторами и страничная адресация.

Задача (task) – программа, модуль или другой участок кода программы, который можно запустить, выполнять, отложить и завершить.

Идентификатор (handle или identifier) – число (если handle) или переменная другого типа, используемая для идентификации того или иного ресурса.

Исключение (exception) – событие, при котором выполнение программы прекращается и управление передается обработчику исключения.

Итерация (iteration) — организация обработки данных, при которой действия повторяются многократно, не приводя при этом к вызовам самих себя (не путать с рекурсией).

Код (code) – исполнимая часть программы (обычная программа состоит из кода, данных и стека).

Команда перехода (branch)– команда процессора, которая нарушает естественный порядок исполнения команд, вынуждая выбирать и исполнять последующие команды с произвольно заданного адреса.

Компилятор (compiler) – программа, преобразующая текст, написанный на понятном человеку языке программирования, в исполнимый файл.

Конвейер (pipe) – последовательность блоков процессора, которая задействуется при выполнении команды.

Конвенция (convention) – договоренность о передаче параметров между процедурами.

Конечный автомат (finite state machine) – программа, которая может переключаться между различными состояниями и выполнять в разных состояниях разные действия.

Кэш (cache) – быстрая память, используемая для буферизации обращений к основной памяти.

Лимит (limit) – поле дескриптора (равно размеру сегмента минус 1).

Линейный адрес (linear address) — адрес, получаемый сложением смещения и базы сегмента.

Ловушка (trap) – исключение, происходящее после вызвавшей его команды.

Локальная метка (local label) – это метка, которая известна только внутри того оператора Asm, где она была определена.

Метка (label) – идентификатор, связанный с адресом в программе.

Нить (thread) – процесс, данные и код которого совпадают с данными и кодом других процессов.

Нереальный режим (unreal mode) – реальный режим с границами сегментов по 4 Гб.

Операнд (operand) – параметр, передаваемый команде процессора.

Описатель носителя (media descriptor) – байт, используемый DOS для идентификации типа носителя (обычно не используется).

Останов (abort) – исключение, происходящее асинхронно.

Отложенное вычисление (lazy evaluation) – вычисление, которое выполняется, только если реально требуется его результат.

Очередь предвыборки (prefetch queue) – буфер, из которого команды передаются на расшифровку и выполнение.

Ошибка (fault) – исключение, происходящее перед вызвавшей его командой.

Пиксель (pixel) – минимальный элемент растрового изображения.

Повторная входимость (reentrancy) – возможность запуска процедуры из обработчика прерывания, прервавшего выполнение этой же процедуры.

Подчиненный сегмент (conforming segment) – сегмент, на который можно передавать управление программам с более низким уровнем привилегий.

Препроцессор (preprocessor) – это компьютерная программа, принимающая данные на входе и выдающая данные, предназначенные для входа другой программы (например, компилятора). О данных на выходе препроцессора говорят, что они находятся в препроцессированной форме, пригодной для обработки последующими программами (компилятор).

Прерывание (interrupt) – сигнал от внешнего устройства, приводящий к прерыванию выполнения текущей программы и передаче управления специальной программе-обработчику.

Разворачивание циклов (loop unrolling) – превращение циклов, выполняющихся известное число раз, в линейный участок кода.

Реальный режим (real mode) – режим, в котором процессор ведет себя идентично 8086 – адресация не выше одного мегабайта памяти, размер всех сегментов ограничен и равен 64 Кб, только 16-битный режим.

Резидентная программа (resident program).– программа, остающаяся в памяти после возврата управления в DOS

Сегмент (segment) – элемент сегментной адресации в памяти или участок программы для DOS/Windows.

Селектор (selector) – число, хранящееся в сегментном регистре.

Секция (section) – участок программы для UNIX.

Скан-код (scan-code) – любой код, посылаемый клавиатурой.

Слово (word) – тип данных, имеющий размер 16 бит.

Смещение (offset) – относительный адрес, отсчитываемый от начала сегмента.

Стековый кадр (stack frame) – область стека, занимаемая параметрами процедуры, активационной записью и локальными переменными или только локальными переменными.

Страничная адресация (pagination) – механизм адресации, в котором линейное адресное пространство разделяется на страницы, которые могут располагаться в разных областях памяти или вообще отсутствовать.

Таблица переходов (jumptable) – массив адресов процедур для косвенного перехода на процедуру с известным номером.

Шлюз (gate) – структура данных, позволяющая осуществлять передачу управления между разными уровнями привилегий в защищенном режиме.

Примерная тематика курсовых работ по дисциплине «Микропроцессоры»

1. Исследование и сравнение реализации 2-мерных массивов и стеков на языке ассемблера.
2. Исследование и сравнение реализации очередей и записей на языке ассемблера.
3. Исследование реализации MMX-технологии микропроцессоров.
4. Исследование методов и способов обработки всех видов прерываний на языке ассемблера.
5. Исследование методов реализации работы с процедурами (организация интерфейса с процедурами).
6. Исследование работы с логическими командами (логические данные; команды сдвига; работа с битовыми строками; пересылка битов).
7. Исследование цепочечных команд (пересылка цепочек; команды пересылки байтов, слов, двойных слов; сравнение цепочек; сканирование цепочек; загрузка элемента цепочки в аккумулятор; перенос элемента из аккумулятора в цепочку; ввод элемента цепочки из порта ввода/вывода).
8. Создание Windows-приложения «Калькулятор» на ассемблере.
9. Исследование обработки прерываний в защищенном режиме (обработка прерываний в защищенном режиме; шлюз ловушки; шлюз прерывания; шлюз задачи; инициализация таблицы IDT; обработчики прерываний; программирование контроллера прерываний).
10. Исследование команд работы с портом ввода/вывода.
11. Исследование и сравнение реализации объединений и записей на языке ассемблера.

12. Исследование связи ассемблера с языками высокого уровня (связь Pascal-Assembler).
13. Исследование связи ассемблера с языками высокого уровня (связь C-Assembler).
14. Исследование и сравнение реализации объединений и стеков на языке ассемблера.
15. Создание Windows-приложения «Мировое время» на ассемблере.

Список литературы

- 1 Александров Е., Грушвицкий Р., Куприянов М. Микропроцессорные системы. – М.: – Политехника, 2008. – 543 с.
- 2 Пильщиков В. Язык макроассемблера IBM PC. – М.: – Академия, 2008. – 412 с.
- 3 Пирогов В. Ассемблер для Windows. – М.: – Вильямс, 2007. – 522 с.
- 4 Юров В. Ю. Ассемблер. – СПб.: Питер, 2007. – 624 с.
- 5 Григорьев В.Л. Архитектура и программирование арифметического сопроцессора. М., 1991. –326 с.
- 6 Лямлин Л.В. Макроассемблер MASM. М.,1994. – 254с.
- 7 К. Ирвин. Язык Ассемблера для процессоров Intel. М.: – Вильямс, 2006. 616 с.
- 8 Корнеев В., Киселев А. Современные микропроцессоры. – СПб.: – ВHV-СПб, 2007. – 448 с.
- 9 Новиков Ю. Основы микропроцессорной техники: Курс лекций. – М.: – Интернет-университет информационных технологий, 2007. – 436 с.
- 10 Финогенов К.Г., Рудаков П.И. Язык Ассемблера: уроки программирования. М.: – Диалог-МИФИ, 2005. – 235 с.
- 11 Белов А.В. Самоучитель по микропроцессорной технике. – М.: – Наука и Техника, 2007. – 224 с.
- 12 Кузин А.В., Жаворонков М.А. Микропроцессорная техника. Учебник. – М.: – Академия, 2008. – 314 с.

ПРИЛОЖЕНИЯ

Титульный лист

ФГБОУ ВПО
КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ
УНИВЕРСИТЕТ

кафедра компьютерных технологий и систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе

по дисциплине:	«Микропроцессоры»
на тему:	Исследование и сравнение реализации очередей и записей на языке ассемблера
выполнил студент:	Петров А. А.
группа:	ПИ-1202
Допущен к защите:	05.11.2014
Руководитель работы:	к.т.н., доцент Иванов И.И.

Защищён _____
(дата)

Оценка _____

Члены комиссии

(подпись, дата, расшифровка подписи)
Краснодар, 20__ г.

Лист задания

КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИ-
ТЕТ

Кафедра компьютерных технологий и систем

УТВЕРЖДАЮ:

Зав. кафедрой _____

ЗАДАНИЕ

на курсовую работу

Студенту: **Петрову А.А.** группы ПИ-1202 2 курса
Факультета **прикладной информатики**
специальности **230400.62 «Информационные системы и технологии»**
Тема проекта: **Исследование и сравнение реализации очередей и записей на языке ассемблера**
Содержание задания: **Исследовать и сравнить реализацию очередей и записей на языке ассемблера**

Объем работы:

а) пояснительная записка к проекту **25 страниц**

б) графическая часть **2 рисунка**

Рекомендуемая литература: **Программирование на языке ассемблера для микроконтроллеров семейства i8051 - Каспер Эрни**

Срок выполнения проекта: с _____ по _____

Срок защиты: с _____ по _____

Дата выдачи задания: «____» _____ 20__ года

Дата сдачи проекта на кафедру: «____» _____ 20__ года

Руководитель проекта _____

(подпись, Ф.И.О., звание, степень)

Задание принял студент _____

(подпись, дата)

Краснодар, 20__ г.

Реферат

РЕФЕРАТ

Работа содержит: 25 страниц, 2 рисунка, 1 таблица, 11 использованных источников, 1 блок-схема, 2 приложения.

СТРУКТУРЫ ДАННЫХ, РЕГИСТР, ПАМЯТЬ, ЭФФЕКТИВНОСТЬ, РЕАЛИЗАЦИЯ, ДИЗАССЕМБЛИРОВАНИЕ.

Краткая характеристика работы:

Исследован метод реализации.... в результате можно сделать обоснованный вывод о том, что... Написана программа, реализующая..., также было проведено сравнение по следующим показателям....

Содержание

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1. ИССЛЕДОВАНИЕ АЛГОРИТМОВ

1.1 Алгоритм

1.2 Алгоритм

2. РЕАЛИЗАЦИЯ АЛГОРИТМОВ НА ЯЗЫКЕ АССЕМБЛЕРА

2.1 Изучение структур входных/выходных данных

2.2 Алгоритмизация задачи

2.3 Особенности реализации алгоритмов ... на языке ассем- блера

3. ОПИСАНИЕ ПРОГРАММЫ

3.1 Системные требования

3.2 Руководство пользователя

3.3 Сравнительный анализ алгоритмов

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ 1 ЛИСТИНГ ПРОГРАММЫ

ПРИЛОЖЕНИЕ 2 ТАБЛИЦЫ СРАВНИТЕЛЬНОГО АНАЛИЗА АЛГОРИТМОВ

Пример наполнения курсовой работы

ВВЕДЕНИЕ

Представление алгоритма в виде последовательности двоичных кодов называется программой. Программа записывается в отдельном исполняемом файле (например .exe для windows-систем). Таким образом, программа – это алгоритм, предназначенный для выполнения компьютером. Двоичное представление команд компьютера называется машинным кодом. Довольно скоро стало понятно, что процесс формирования машинного кода можно автоматизировать. Уже в 1950 году для записи программ начали применять мнемонический язык – язык assembly. Язык ассемблера позволил представить машинный код в более удобной для человека форме: для обозначения команд и объектов, над которыми эти команды выполняются, вместо двоичных кодов использовались буквы или сокращенные слова, которые отражали суть команды. Теперь вместо рябящих в глазах нулей и единиц, они могли писать программу командами, состоящими из символов приближенных к обычному языку. Для того времени этот язык был новшеством и пользовался популярностью т.к. позволял писать программы небольшого размера, что при тех машинах критерий значительный. Но сложность разработки в нём больших программных комплексов привела к появлению языков третьего поколения – языков высокого уровня. Но на этом жизнь ассемблера не закончилась, он жив и по сей день и не только жив, но и пользуется популярностью в узких кругах. Сейчас его используют в написании отдельных фрагментов программ или иногда в написании самих программ. Примеров может

быть много, но самые яркие это использование ассемблера в написании драйверов, игр и загрузчиков ОС.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Сведения из теории

Обоснование необходимости разработки Windows-приложений на ассемблере:

1. Язык ассемблера позволяет программисту полностью контролировать создаваемый им программный код и оптимизировать его по своему усмотрению.

2. Компиляторы языков высокого уровня помещают в загрузочный модуль программы избыточную информацию. Эквивалентные исполняемые модули, исходный текст которых написан на языке ассемблера, имеют в несколько раз меньший размер.

3. При программировании на ассемблере сохраняется полный доступ к аппаратным ресурсам компьютера.

4. Приложение, написанное на языке ассемблера, как правило, быстрее загружается в оперативную память компьютера.

5. Приложение, написанное на языке ассемблера, как правило, обладает более высокой скоростью работы и реактивностью ответа на действия пользователя.

Минимальное приложение Windows состоит из трех частей:

1. Главная функция.
2. Цикл обработки сообщений.
3. Оконная функция.

Имея исходный файл Windows-приложения на языке C/C++, можно получить текст на языке ассемблера. На его основе впоследствии можно сформировать функционально эквивалентный исполняемый модуль. Необходимо дизассемблировать исполняемый модуль программы. Причем сделать это нужно тем дизассемблером, который понимает интерфейс

Win32 API. Дизассемблированный файл можно сохранить как листинг (расширение .lst) и как исходный текст ассемблера (расширение .asm). Файл листинга в своей левой части содержит колонку с адресами смещения команд. Все метки и символические имена в дизассемблированном тексте формируются с использованием этих смещений.

Одним из главных критериев выбора языка разработки Windows-приложения является наличие в нем средств, способных поддержать строго определенную последовательность шагов. Каркасное Windows-приложение на ассемблере содержит один сегмент данных (.data) и один сегмент кода (.code). Сегмент стека в исходных текстах Windows-приложений непосредственно описывать не нужно. Windows выделяет для стека объем памяти, размер которого задан программистом в файле с расширением .def.

Все символические имена в программе на ассемблере по умолчанию являются глобальными. Задание директивы locals включает в трансляторе механизм контроля областей видимости имен и позволяет использовать в программе локальные имена. Символическим именам, которые необходимо сделать локальными, должна предшествовать определенная последовательность не менее чем из двух символов. Эти символы задаются как параметры директивы locals. Если этого не сделать, то по умолчанию используется последовательность из двух символов @@ . Блоком, в пределах которого можно объявить локальные имена, может быть не только функция, но и участок программы между двумя метками.

Директива .model задает модель сегментации flat и стиль генерации кода при входе в процедуры программы и выходе из них – STDCALL. Код загрузочного модуля, генерируемый с опцией flat, будет работать на микропроцессорах .386 и старше. По этой причине директиве .model должна предшествовать одна из директив .386, .486 или .586. Указание этой модели памяти заставляет компоновщик создавать исполняемые

файл с расширением .exe. В программе с плоской моделью памяти используется адресация программного кода типа near. Параметр STDCALL определяет порядок передачи параметров через стек, справа налево. Функции Win32API, используемые в программе, должны быть объявлены внешними с помощью директивы extrn. Это необходимо сделать для того, чтобы компилятор мог сгенерировать правильный код, так как тела функций Win32API содержатся в dll-библиотеках системы Windows. В соответствии с соглашениями операционной системы Windows оконная функция приложения должна быть видимой за пределами приложения, в котором она написана. Это связано с тем, что оконная функция вызывается самой операционной системой Windows при поступлении сообщения для данного приложения. Загрузчик Windows самостоятельно загружает сегментные регистры, при этом учитывается требуемая модель памяти.

Существуют следующие формы комбинирования программ на языках высокого уровня с ассемблером:

- Использование ассемблерных вставок (встроенный ассемблер, режим inline). Ассемблерные коды в виде команд ассемблера вставляются в текст программы на языке высокого уровня. Компилятор языка распознает их как команды ассемблера и без изменений включает в формируемый им объектный код. Эта форма удобна, если надо вставить небольшой фрагмент.

- Использование внешних процедур и функций. Это более универсальная форма комбинирования. У нее есть ряд преимуществ:

- написание и отладку программ можно производить независимо;

- написанные подпрограммы можно использовать в других проектах;

- облегчаются модификация и сопровождение подпрограмм.

Встроенный ассемблер

При написании ассемблерных вставок используется следующий синтаксис:

```
_asm КодОперации операнды ; // комментарии  
КодОперации задает команду ассемблера,  
операнды – это операнды команды.
```

В конце записывается ;, как и в любой команде языка Си.

Комментарии записываются в той форме, которая принята для языка Си.

Если требуется в тексте программы на языке Си вставить несколько идущих подряд команд ассемблера, то их объединяют в блок:

```
_asm  
{  
    текст программы на ассемблере ; комментарии  
}
```

Внутри блока текст программы пишется с использованием синтаксиса ассемблера, при необходимости можно использовать метки и идентификаторы. Комментарии в этом случае можно записывать как после ;, так и после //.

Использование внешних процедур

Для связи посредством внешних процедур в общем случае возможны два варианта вызова:

- программа на языке высокого уровня вызывает процедуру на языке ассемблера;
- программа на языке ассемблера вызывает процедуру на языке высокого уровня.

В программах, написанных на языке ассемблера, используется соглашение передачи параметров `stdcall`. Однако по сути получение и передача параметров в языке ассемблера производится явно, без помощи транслятора.

Конвенция C используется, в первую очередь, в языках C и C++, а также в PROLOG и других. Параметры поме-

щаются в стек в обратном порядке, и, в противоположность PASCAL-конвенции, удаление параметров из стека выполняет вызывающая процедура.

Смешанные конвенции

Существует конвенция передачи параметров STDCALL, отличающаяся и от C, и от PASCAL-конвенций, которая применяется для всех системных функций Win32 API. Здесь параметры помещаются в стек в обратном порядке, как в C, но процедуры должны очищать стек сами, как в PASCAL.

Еще одно отличие от C-конвенции – это быстрое или регистровое соглашение FASTCALL. В этом случае параметры в функции также передаются по возможности через регистры.

1.2 Постановка задачи

Предметной областью решаемой задачи является калькулятор простых чисел и инженерными функциями, реализующий операции сложения, умножения, вычитания, деления, нахождение корня квадратного, а также $\sin()$, $\cos()$ и тд..

Программа включает в себя набор алгоритмов для выполнения арифметических операций.

Входными данными являются числа, которые вводятся в окна TEdit. После этого они передаются в машинно-ориентированный язык программирования Assembler, для дальнейшей операций.

Выходными данными являются число, которые являются результатом арифметических действий над входными данными. Результат выводится с использованием элементов типа TLabel.

Для написания данной программы был использован язык Assembler и C++.

В качестве инструмента разработки была использована среда разработки C++ Builder от компании Embarcadero. Причиной выбора именно этой IDE стал удобный реактор форм, а также ее создания и интеграция C++ с языком Assembler.

Приложение «Исследование связи языка C с Ассемблером» имеет расширение .exe и, следовательно, работает только в ОС Windows.

Требования к системе:

- 1) ОС Windows XP или более поздние версии;
- 2) Direct X 9;
- 3) Наличие в системе следующих библиотек:borldmm.dll, cc32100mt.dll, rtl140.bpl, vcl140.bpl.
- 4) Требования к оборудованию:
- 5) 10 Мб свободного места на жестком диске;
- 6) Устройства ввода (мышь, клавиатура).

2 ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

2.1 Алгоритм решения

В самом начале выполнения программы появляется форма, где пользователю предлагается заполнить соответствующие поля необходимыми для расчета данными и выбрать какую-либо операцию.

Затем, в ходе выполнения программы производится проверка полноты и корректности введенных начальных данных. Если исходные данные не прошли проверку – выводится соответствующее уведомление.

После успешно пройденной проверки, компилятор *Borland C++ Builder интегрирует языки C++ и Assembler*.

Ключевое слово в данном случае `__asm`, которое вызывает встроенный ассемблер и может отображаться везде, где допустим оператор C или C++. Он не может отображаться самостоятельно. За ним должна следовать инструкция по сборке, группа инструкций, заключенная в круглые скобки, либо, в крайнем случае, пустая пара круглых скобок. Термин "блок `__asm`" в этом разделе относится к любой инструкции или группе инструкций, в скобках или без них. Синтаксис:

```
__asm assembly-instruction [ ; ]
```

```
__asm { assembly-instruction-list } [ ; ]
```

При использовании без круглых скобок ключевое слово `__asm` означает, что остальная часть строки — это оператор на языке сборки. При использовании с фигурными скобками оно означает, что каждая строка между скобками — это оператор на языке сборки. Для обеспечения совместимости с предыдущими версиями `__asm` является синонимом `__asm`.

Пример:

Следующий фрагмент кода — это простой блок `__asm`, заключенный в фигурные скобки:

```
__asm {  
    mov al, 2  
    mov dx, 0xD007  
    out dx, al  
}
```

Кроме того, можно поставить `__asm` перед каждой инструкцией по сборке.

```
__asm mov al, 2  
__asm mov dx, 0xD007  
__asm out dx, al
```

Все три примера создают один и тот же код, но первый стиль (где блок `__asm` заключен в фигурные скобки) имеет некоторые преимущества. Фигурные скобки четко отделяют код сборки от кода C++ и позволяют избежать лишнего повторения ключевого слова `__asm`. Скобки также помогают избежать неоднозначности. Если требуется поместить оператор C++ на одной строке в виде блока `__asm`, необходимо заключить блок в фигурные скобки. Без фигурных скобок компилятор не может определить, где прекращается код сборки и начинаются операторы C++. Наконец, поскольку текст в фигурных скобках имеет тот же формат, что и обычный текст MASM, можно легко вырезать и вставить текст из существующих исходных файлов MASM.

В отличие от фигурных скобок C++ фигурные скобки, в которые заключается блок `__asm`, не влияют на область видимости переменной. Можно также разместить блоки `__asm` в виде вложения, вложение не влияет на область видимости переменной.

После расчётов из кода ассемблера данные передаются в C++ и результаты выводятся в специально отведенное поле, а выполнение программы прекращается.

2.2 Макет приложения

На рисунке 2.1 показан макет приложения

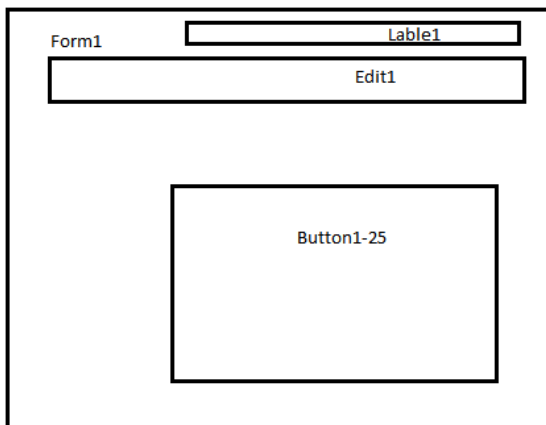


Рисунок 2.1 – Макет приложения

- Button0-9Click – кнопка предназначена для ввода цифр.
- Button10Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Разделитель».
- Button11Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Плюс».
- Button12Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Минус».
- Button13Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Умножить».

- Button14Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Разделить».
- Button15Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Равно»
- Button16Click – кнопка, необходимая для обнуления.
- Button17Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции « $1/X$ ».
- Button18Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Корень».
- Button19Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Число Π ».
- Button20Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Интеграл».
- Button21Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Модуль».
- Button22Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Синус».
- Button23Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Косинус».
- Button24Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Тангенс».

- Button25Click – кнопка необходимая для указания в программе необходимости выполнения арифметической операции «Котангенс».
- Edit1 – поле для ввода.
- Label – поле для отображения результата.
- Форма “о проекте”.

2.3 Описание программы

Используемые директивы

```
#include <vcl.h>
```

Используемые элементы

button

edit

label

Обработчики событий

```
void __fastcall Button1Click(TObject *Sender);
void __fastcall Button10Click(TObject *Sender);
void __fastcall Button11Click(TObject *Sender);
void __fastcall Button15Click(TObject *Sender);
void __fastcall Button16Click(TObject *Sender);
void __fastcall Button17Click(TObject *Sender);
void __fastcall Button18Click(TObject *Sender);
void __fastcall Button19Click(TObject *Sender);
void __fastcall Button20Click(TObject *Sender);
void __fastcall Button21Click(TObject *Sender);
void __fastcall Button22Click(TObject *Sender);
void __fastcall Button23Click(TObject *Sender);
void __fastcall Button24Click(TObject *Sender);
void __fastcall Button25Click(TObject *Sender);
```

Для написания данного проекта были использованы следующие библиотеки:

ASM.h – библиотека, необходимая для подключения Языка ассемблер;

string.h – библиотека, необходимая для работы со строками;

После ввода чисел в окна программы и нажатия на одну из клавиш, запускается обработчик событий, соответствующий

одной из кнопок:

Обработчик событий соответствует кнопке Button11, имеющей свойство Caption равное «+». Сначала обработчик событий копирует данные из полей Edit1 в переменные ап типа Double. Затем он вызывает встроенный ассемблер через ключевое слово __asm, и передает в него значения. Далее производятся расчёты над введенными переменными:

```
__asm
{
    fld a
    fld b
    fadd
    fstp res
}
```

Далее производится передача результата из ассемблера в C++, значение буфера заносится в Label1.

Обработчики событий соответствующие кнопкам Button12, Button13, Button14 интересных особенностей не имеют, их структура схожа с вышеописанными обработчиками событий.

При произведении операций каждое число и операция над ним заносится в lable1.

Обработчик событий соответствующий кнопке Button12, посредством ранее сохранённых значений в Label1, выводит готовый результат в Label1.

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

3.1 Результат работы программ

Программа была разработана таким образом, что пользователь не может ввести ничего с клавиатуры, вследствие этого исключается ошибка неверного ввода данных.

Входные данные представлены на рисунке 3.1:

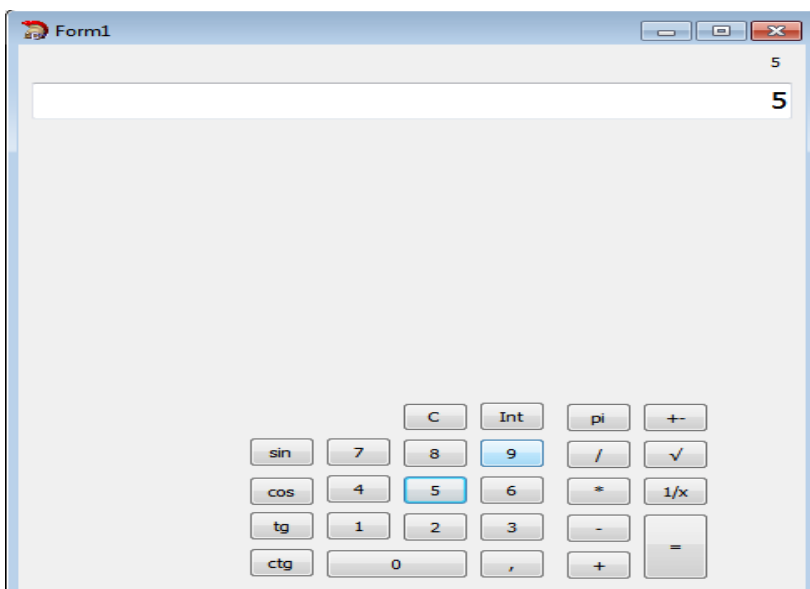


Рисунок 3.1 – входные данные

Выходные данные (по нажатию на кнопку «cos»)
представлены на рисунке 3.2:

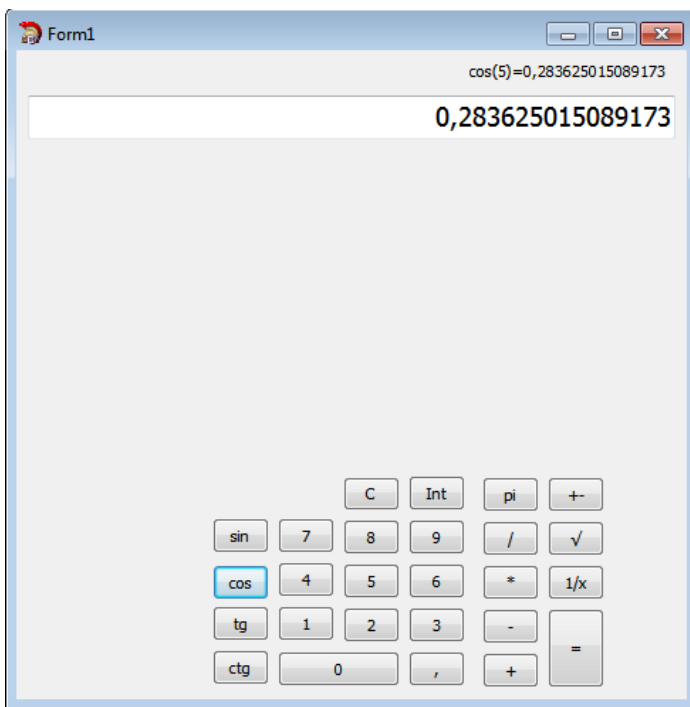


Рисунок 3.2 – выходные данные

3.2 Руководство пользователя

Запускаемым файлом программы является файл C+Assembler.exe.

Данная программа предназначена для вычисления как простых алгебраических вычислений с числами в двоичной системе счисления, так и сложных тригонометрических функций. Основные требования к операционной системе не предъ-

являются, так как не используются современные графические технологии, а лишь встроенный графический язык.

Запуск программы осуществляется по клику на одну из кнопок Button0-Button9.

После выбора действия информация показывается в полях Label1 , показанных на рисунке 2.1.

При вычислении простых значений требуется не только набрать числа и методы их исчисления, но также нажать кнопку «=».

Выходные данные представлены на рисунке 3.3:

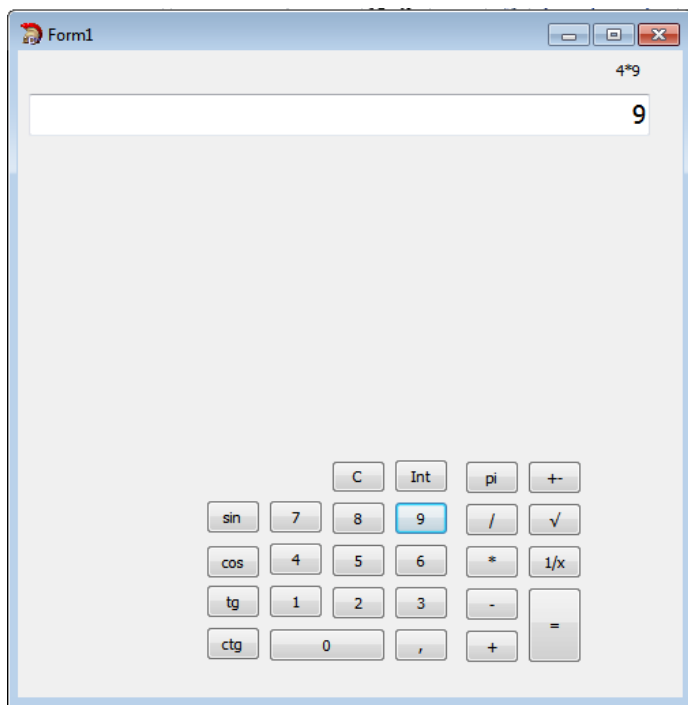


Рисунок 3.3 - Вычисление простых операций

Выходные данные представлены на рисунке 3.4:

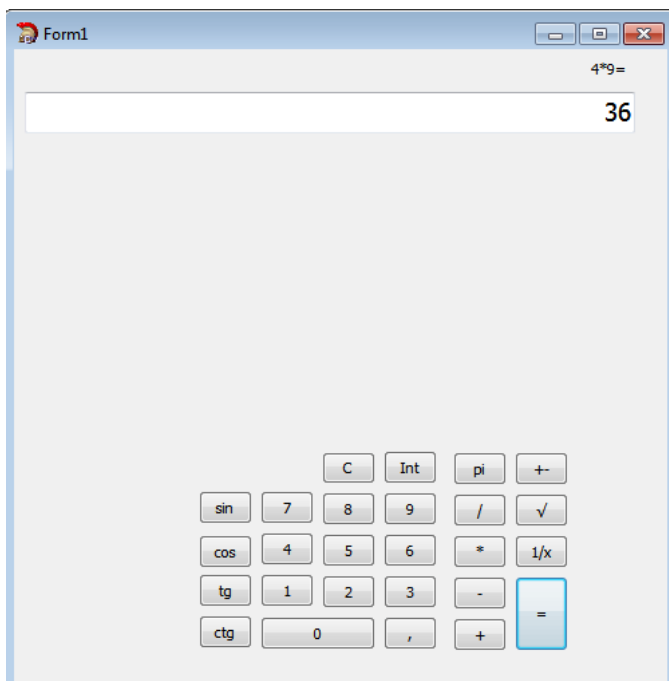


Рисунок 3.4 - Вычисление простых операций

Тригонометрические и сложные функции не требуют нажатия кнопки «=»

Выходные данные представлены на рисунке 3.5:

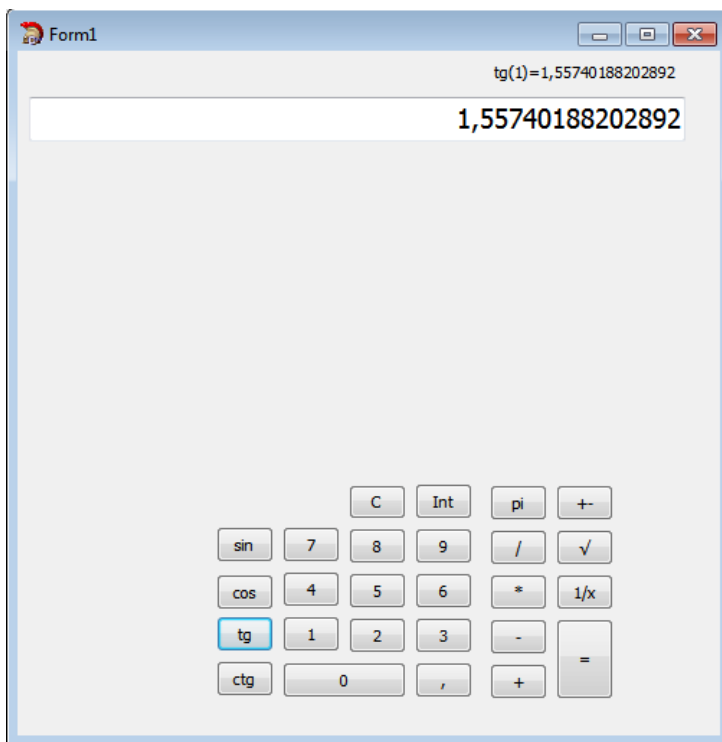


Рисунок 3.5 - Вычисление сложных операций

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были выполнены все поставленные задачи разработки приложений на языке C++ и Assembler и опыт работы со средой разработки C++ *Builder*.

Результатом выполнения является работоспособная программа, способная выполнять простые алгебраические операции, а также сложные тригонометрические функции с числами.

Были максимально предусмотрены всевозможные ошибки, которые могут возникнуть при использовании данной программы, однако это не исключает возможность их появления.

Программа производит вычисления с высокой точностью и большими значениями, требуя относительно не высоких затрат ресурсов используемого ПК. Программа имеет простой пользовательский интерфейс и выполняет все поставленные ей задачи.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Калашников О.А. «Ассемблер? Это просто! Учимся программировать» / Калашников О. А. — БХВ-Петербург, 2011. — С. 336. — ISBN 978-5-9775-0591-8.
- 2) Аблязов Р. З. «Программирование на ассемблере на платформе x86-64» / Аблязов Р. З. — М.: ДМК Пресс, 2011. — С. 304. — ISBN 978-5-94074-676-8.
- 3) Герберт Шилдт. «Полный справочник по C++» / Герберт Шилдт. — ООО «И.Д. Вильямс», 2011. — 1056 с.
- 4) Айвор Хортон. «Visual C++ 2010: полный курс» / Айвор Хортон — Москва: Диалектика, 2010.
- 5) Бьёрн Страуструп «Язык программирования C++». / Бьёрн Страуструп — Специальное издание. Москва: Бином-Пресс, 2010. — 1104 с.
- 6) Владислав Пирогов. «Ассемблер для Windows» / Владислав Пирогов — СПб.: БХВ-Петербург, 2010. — 896 с. — ISBN 978-5-9775-0084-5.

МИКРОПРОЦЕССОРЫ

Методические рекомендации

Параскевов Александр Владимирович, **Бардак** Алексей Николаевич
Дмитриева Анна Геннадьевна и др.

Подписано в печать _____. Формат $60 \times 84 \frac{1}{16}$.

Усл. печ. л. – 3,3. Уч.-изд. л. – 2,6.

Тираж 75 экз. Заказ № ____.

Типография Кубанского государственного аграрного университета,
350044, г. Краснодар, ул. Калинина, 13