МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Вятский государственный университет»
(ФГБОУ ВО «ВятГУ»)
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Допущено к защите
Руководитель работы
_____/ Караваева О.В./
(подпись) (Ф.И.О)
«__»_____2020г.

РАЗРАБОТКА ПРОГРАММЫ «ДИЗАССЕМБЛИРОВАНИЕ КОМАНД»

Пояснительная записка курсовой работы по дисциплине
«Комплекс знаний бакалавра»
ТПЖА.09.03.01.014 ПЗ

Разработал студент группы ИВТ-31 _____/Седов М.Д./

Руководитель
Преподаватель кафедры ЭВМ _____/ Караваева О.В./

Проект защищен с оценкой «_____» _____
(оценка) (дата)

Члены комиссии _____/ _____/
(подпись) (Ф.И.О)

_____ /_____/

Киров 2020

Реферат

Седов М.Д. Разработка программы «Дизассемблирование команд»: ТПЖА.090301014 ПЗ: Курс. работа / ВятГУ, каф. ЭВМ; рук. Караваева О.В. - Киров, 2020. – ПЗ 42 с, 7 рис., 2 табл., 9 источников.

ДИЗАССМБЛИРОВАНИЕ, ФОРМАТ КОМАНД, ГЕНЕРАТОР ЗАДАНИЙ, ТЕСТИРОВАНИЕ, PYTHON, CMAKE

Цель курсового проекта — повышение качества обучения за счет использования программного комплекса для дизассемблирования команд исполняемого файла при выполнении лабораторных работ по дисциплине «Системное программное обеспечение».

Программное обеспечение, разработанное в рамках данного курсового проекта — лабораторная установка «Дизассемблирование команд».

В ходе выполнения курсового проекта был выполнен анализ предметной области, проектирование и разработка программного обеспечения.

Содержание

Введение

В настоящее время в области образования все больше производится автоматизация контроля знаний учащихся и освоения нового материала.

Одним из способов автоматизации являются специальные программные модели, эмулирующие работу какой-либо системы. С их помощью студенты имеют возможность достаточно подробно изучить ее работу, принципы и особенности. В совокупности с теоретическим материалом это позволяет увеличить степень освоения новых знаний по данной дисциплине и повысить качество обучения в целом

Такие программные модели досточно широко используются при выполнении лабораторных работ по дисциплине «Системное программное обеспечение». К сожалению, качество некоторых приложений оставляет желать лучшего, что затрудняет изучение нового материала. Поэтому было принято решение выполнить анализ одной из программной модели — дизассемблирование команд исполняемого файла.

1. **Анализ предметной области**

На данном этапе работы необходимо провести обзор текущей программной модели дизассемблирования команд, выяснить ее недостатки, обосновать актуальность разработки новой модели и сформировать требования к программному обеспечению в виде технического задания.

1.1 Обзор текущей программной модели

Текущая программная модель была разработана в 1999 году студентами Вятского государственного университета А. В. Перминовым и Д. М. Чопуком.

1.1.1 Дизассемблирование машинных команд

Дизассемблирование — процесс перевода (трансляции) машинного кода, обьектного файла или библиотечного модуля в текст программы на языке ассемблера.

Чаще всего дизассемблирование необходимо для анализа программы (или ее части), исходный код которой неизвестен — с целью модификации, копирования или взлома; реже — для поиска ошибок (багов) в программах и компиляторах,а также для анализа и оптимизации создаваемого компилятором машинного кода.

При работе с исполняемым кодом или байт-кодом, созданным на некоторых языках высокого уровня (например, java) имеется возможность восстановить не только текст на языке ассемблера, но даже и структуру классов программы, а если при компиляции исполняемого файла не была отключена отладочная информация — то и исходный текст программы.

Машинная команда – это набор кодов операций, выполняемых определенной машиной, или коды, специфичный для конкретной машины.

Компьютерная программа, записанная на машинном языке, состоит из машинных инструкций, каждая из которых представлена в машинном коде в виде

т. н. кода операции(КОП) — двоичного кода отдельной операции из системы команд машины. Для удобства программирования вместо числовых КОП, которые только и понимает процессор, обычно используют их условные буквенные мнемоники. Набор таких мнемоник, вместе с некоторыми дополнительными возможностями (например, некоторыми макрокомандами, директивами), называется языком ассемблера.

Каждая машинная инструкция выполняет определённое действие, такое как операция с данными (например, сложение или копирование машинного слова в регистре или в памяти) или переход к другому участку кода (изменение порядка исполнения; при этом переход может быть безусловным или условным, зависящим от результатов предыдущих инструкций). Любая исполнимая программа состоит из последовательности таких атомарных машинных операций.

1.1.2 Интерфейс пользователям

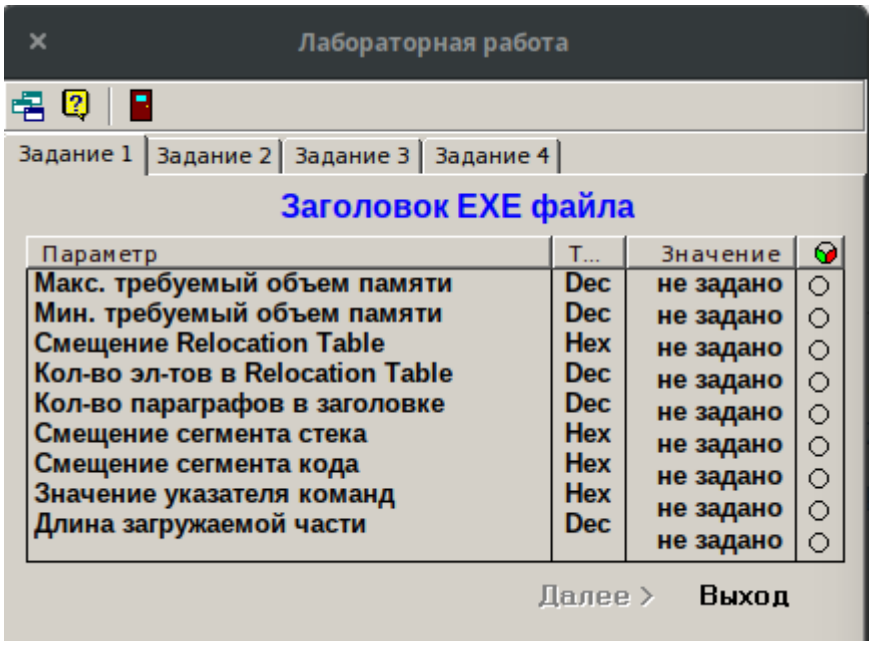


Рисунок 1 — Основное окно программы

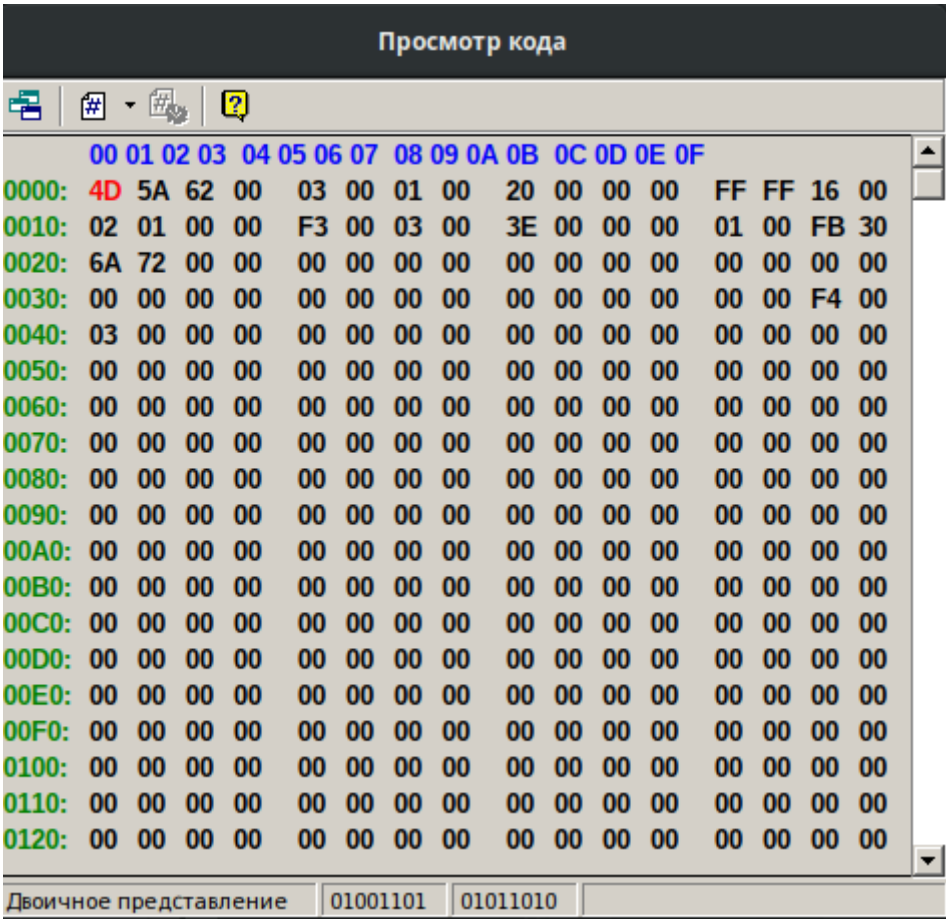


Рисунок 2 — Окно с просмотром кода

Главное окно программы разделено на следующие области:

- панель с тремя кнопками: «выход», «справка», «переход к окну с кодом»
- четыре вкладки с номерами заданий
- таблица с списком параметров заголовка, типом представления данных, полем ввода ответов, индикатором правильности ответа
- кнопка «далее»
- кнопка «выход»

Окно с просмотром кода имеет таблицу шириною в 15 символов, а в данном случае, - байтов. Каждая строка и каждый столбец таблицы имеет шестнадцатеричный адрес представления. В нижнем панели располагается двоичное представление выделенной ячейки таблицы и следующей за ней ячейки.

После успешного выполнения первого задания открывается вкладка со вторым заданием, на котором располагается список команд. Справа от списка команд

располагаются кнопки «добавить», «изменить», «удалить», а также две кнопки в виде стрелок вверх и вниз, и кнопка «проверить». В нижней части располагаются кнопки «далее» и «выход».

Пользователь имеет возможности выполнить следующие действия:

- ввести соответствующие значения о заголовке EXE модуля
- добавить описание команды на языке ассемблера
- изменить описание команды на языке ассемблера
- удалить выбранную команду
- переместить вверх или вниз выбранную команду относительно остальных команд
- проверить правильность ввода команд
- выделить произвольную ячейку в окне просмотра кода
- переместится во все четыре стороны относительно выбранной ячейки в окне с кодом
- подтвердить действия, нажав кнопку «далее»
- получить «справку»
- отменить все действия и завершить программу, нажав кнопку «выход»

1.1.3 Виды машинных команд и их представления

Машинная команда должна содержать следующую информацию:

- код операции. Помещается в первом байте команды. Код предполагает работу без операндов, с одним или двумя операндами.
- способ адресации и собственный адрес. В команде должно быть закодировано, содержатся ли операнды в регистрах, или это непосредственные данные, или операнд находится в памяти.

Форматы команд МП в языке Ассемблера строятся в соответствии двоичной внутренней машинной формой представления, по которой обобщенный формат использует базовую двухбайтную структуру для построения основной части команды, имеющей вид

Код операции - это обязательный элемент, описывающий операцию, выполняемую командой. Многим командам соответствует несколько кодов операций, каждый из которых определяет нюансы выполнения операции. Последующие поля машинной команды определяют местоположение операндов, участвующих в операции, и особенности их использования.

Поле размера равно нулю, если операнды имеют размер 1 байт. Единичное значение указывает на слово.

Направление обозначает операнд-приемник. При нулевом значении результат присваивается правому операнду, при единичном – левому. Размер данных и количество следующих байтов команды определяется конкретным кодом операции и наличием в команде фиксированных байтов префиксов переключения разрядности данных и адресации.

Байт режима адресации modr/m (пост-байт). Значения этого байта определяет используемую форму адреса операндов. Операнды могут находиться в памяти в одном или двух регистрах. Если операнд находится в памяти, то байт modr/m определяет компоненты (смещение, базовый и индексный регистры), используемые для вычисления его эффективного адреса . В защищенном режиме для определения местоположения операнда в памяти может дополнительно использоваться байт sib (Scale-Index-Base — масштаб-индекс- база). Байт modr/m состоит из трех полей.

Поле mod определяет количество байт, занимаемых в команде адресом операнда. Поле mod используется совместно с полем r/m, которое указывает способ мо- дификации адреса операнда смещение в команде.

К примеру, если mod = 00, это означает, что поле смещение в команде отсутствует, и адрес операнда определяется содержимым базового и (или) индексного регистра. Какие именно регистры будут использоваться для вычисления эффективного адреса, определяется значением этого байта.

Если mod = 01, это означает, что поле смещение в команде присутствует, занимает один байт и модифицируется содержимым базового и (или) индексного регистра.

Если mod = 10, это означает, что поле смещение в команде присутствует, занимает два или четыре байта (в зависимости от действующего по умолчанию или определяемого префиксом размера адреса) и модифицируется содержимым базового и (или) индексного регистра.

Если mod = 11, это означает, что операндов в памяти нет: они находятся в регистрах. Это же значение байта mod используется в случае, когда в коман- де применяется непосредственный операнд.

1.1.4 Недостатки

В текущей программной модели были найдены следующие недостатки:

- приложение доступно только для ОС Windows. Пользователи других операционных систем должны запускать Windows в виртуальной машине либо использовать другие средства по запуску Windows — приложений в других ОС;
- нестабильность. В ходе выполнения лабораторной работы программная модель несколько раз аварийно завершалась, из-за чего результаты выполненной работы безвозвратно терялись;
- ошибки при проверке пользовательских действий. В некоторых случаях было замечено, что программная модель принимала правильную последовательность действий как ошибочную. Это в совокупности с нестабильностью программы усложняет изучение студентами программной модели и, как следствие, увеличивает время на выполнение лабораторной работы;
- нечеткость, «размытость» интерфейса на дисплеях со сверхвысоким разрешением экрана (HiDPI);
- в связи с утерей исходного кода программы и сложностью дизассемблирования определить формат файла задания не представляется возможным, что препятствует разработке новых вариантов заданий;

- отсутствует возможность сохранения состояние выполнения работы, которое отличается от финального. Тем самым студентам приходится выполнять полностью работу с целью сохранить результаты.

1.2 Актуальность разработки

Лабораторная работа по данной теме (дизассемблирование) имеет важную роль в закреплении лекционного материала и предоставляет студентам возможность изучить более подробно форматы команд x86 процессора и способ их дизассемблирования.

Из-за того, что исходный код программной модели утерян, исправить ошибки и недостатки в текущей модели не предоставляется возможным. Поэтому было принято решение разработать новую программную модель, повторяющую функционал текущей, в которой будут исправленные вышеописанные ошибки и недостатки.

1.3 Техническое задание

1.3.1 Наименование программы

Наименование программы - «Дизассемблирование команд».

1.3.2 Краткая характеристика области применения

Программа предназначена для закрепления студентами лекционного материала по дисциплине «Системное программное обеспечение», а именно: дизассемблирование машинных команд.

1.3.3 Назначение разработки

Функциональным назначением программы является предоставление студентам возможности изучить процесс дизассемблирования машинных команд во время выполнения лабораторных работ.

Программа должна эксплуатироваться на ПК студентов, преподавателей и на ПК, установленных в учебных аудиториях Вятского государственного университета. Особые требования к конечному пользователю не предъявляются.

1.3.4 Требования к программе

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

1) функции генерации задания;
2) функции сохранения текущего прогресса выполнения задания в файл;
3) функции загрузки задания с сохраненным прогрессом выполнения из файла;
4) функции подсчета количества ошибок, сделанных в ходе выполнения задания;
5) функции просмотра действий, выполненных в ходе прохождения задания.

Надежное (устойчивое) выполнение программы должно быть обеспечено выполнением пользователем совокупности организационно- технических мероприятий, перечень которых приведен ниже:

1) организацией бесперебойного питания технических средств;
2) использованием лицензионного программного обеспечения.

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных прав.

В состав технических средств должен входить IBM — совместимый персональный компьютер, включающий в себя:

1) x86 — совместимый процессор с тактовой частотой не меньше 1.0 ГГц;
2) дисплей с разрешением не меньше, чем 1024x768;
3) не менее 500 мегабайта оперативной памяти;
4) не менее 100 мегабайт свободного дискового пространства;

5) клавиатура, мышь.

Системные программные средства, используемые программой, должны быть представлены следующими операционными системами:

- 64 — разрядная ОС Window 7/8/8.1/10;
- 64 — разрядная ОС Ubuntu 18.04 и выше

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса и предоставлять возможность выполнять наиболее частые операции с помощью сочетаний клавиш на клавиатуре.

1.3.5 Требования к программной документации

Состав программной документации должен включать в себя:

1) техническое задание;
2) программу и методики испытаний;
3) руководство пользователя;
4) техническую документацию;
5) исходный код.

1.3.6 Стадии и этапы разработки

Разработка должна быть проведена в три стадии:

1) разработка технического задания;
2) рабочее проектирование;
3) внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласование и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1) разработка программы;
2) разработка программной документации;

3) испытания программы.

На стадии внедрения должен быть выполнен этап подготовки и передачи программы заказчику.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1) постановка задачи;

2) определение и уточнение требований к техническим средствам;

3) определение требований к программе;

4) определение стадий, этапов и сроков разработки программы и документации на нее;

5) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77 с требованиями п. Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные виды работ:

1) разработка, согласование, утверждение программы и методики испытаний;

2) проведение приемо — сдаточных испытаний;

3) корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена подготовка и передача программы и программной документации в эксплуатацию заказчику.

2. Разработка структуры приложения

На данном этапе работы необходимо в соответствии с требованиями, поставленными в техническом задании, разработать алгоритмы функционирования и модульную структуру приложения.

2.1 Разработка алгоритмов функционирования

На основе обозначенных в техническом задании требований к программному обеспечению можно построить алгоритмы, описывающие структуру и поведение разработки.

2.1.2 Алгоритм генерации заданий

Основная идея алгоритма генерации задания заключается в том, что каждая команда в списке для дизассемблирования выбирается случайным образом, при этом команды делятся на две группы: диаграммы с условным(безусловным) переходом, команды, выполняющие последовательные действия.

Алгоритм генерации состоит из следующих шагов:

1) заполнить случайными данными заголовок EXE файла

2) создать пустой стек

3) создать пустой список для команд

4) сгенерировать команду для добавления

5) если команда условная, то перейти к п 8

6) добавить команду в список команд

7) если число сгенерированных команд равно 10, перейти к п 11, иначе перейти к п 4

8) сгенерировать команду из группы последовательных команд

9) добавить команду в список командам, увеличить счетчик команд внутри условия на 1

10) если число команд стало больше 3, перейти к п 4, иначе перейти к п 8

11) записать полученные результаты и создать EXE файл

Схема алгоритма генерации задания представлена на рисунке 3.

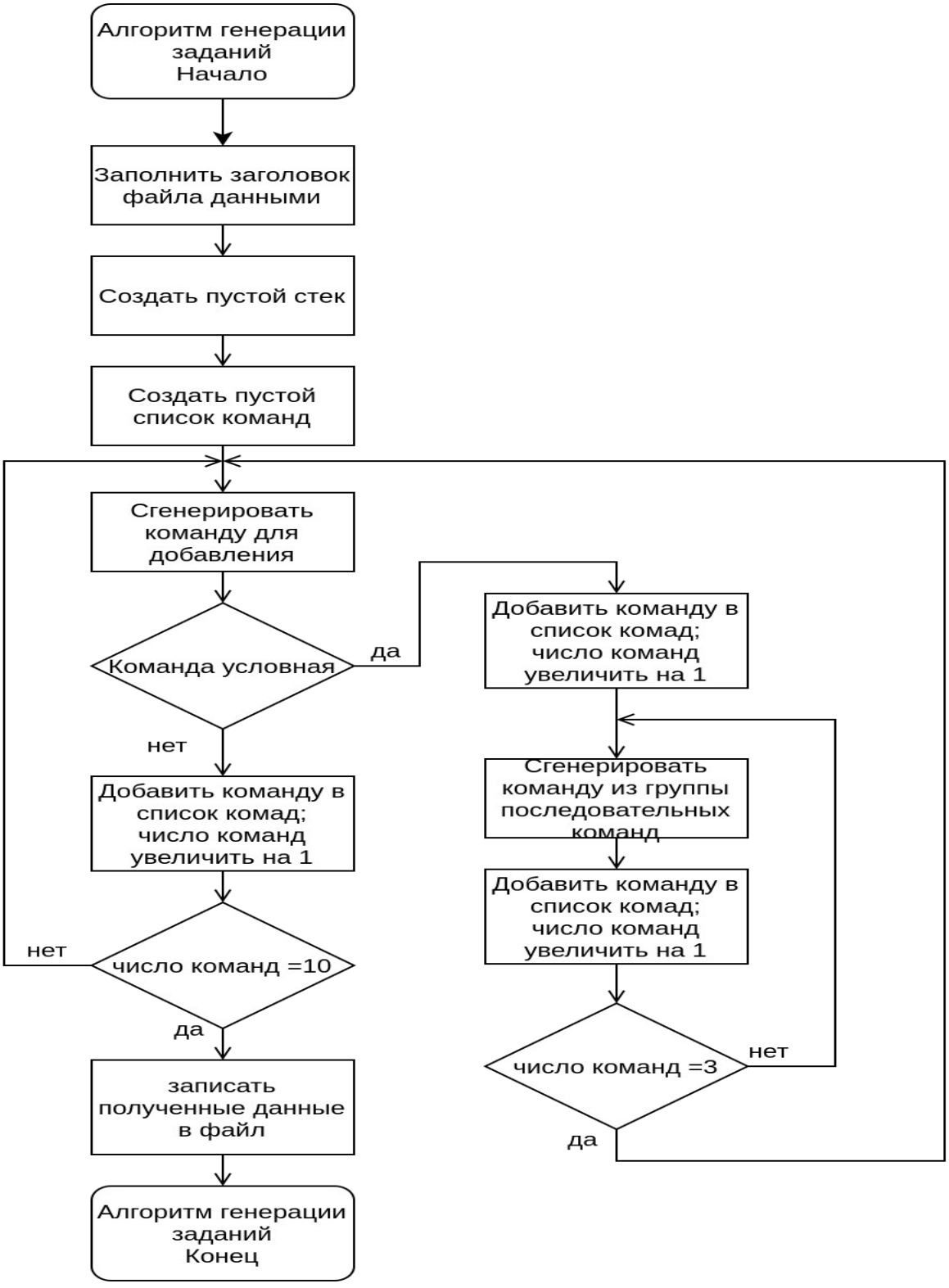


Рисунок 3 — Схема алгоритма генерации задания

2.2 Разработка модульной структуры приложения

Для обеспечения функционирования системы разработана обобщенная структура программного продукта, представляющая собой набор взаимосвязанных модулей, которые реализуют используемые алгоритмы функционирования. Модульная структура приложения представлена на рисунке 4.

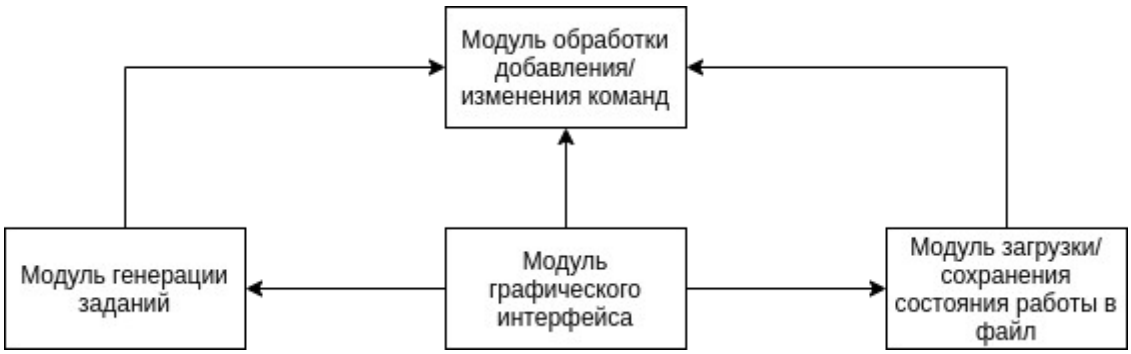


Рисунок 4 — Модульная структура приложения

Каждый из модулей имеет следующие назначения и функционал:

- модуль обработки добавления или изменения команд. Определяет алгоритмы обработки команд и проверки выполненных пользователем действий, а также определяет необходимые типы данных машинных команд, и самого задания на лабораторную работу;
- модуль-генератор заданий. Определяет алгоритмы генерации заданий. Обеспечивает достаточную степень уникальности задания для каждого пользователя без необходимости разработки заданий вручную;
- модуль загрузки и сохранения заданий в файл. Предоставляет возможность сохранять текущее состояние выполняемого задания между запусками приложения, а также обеспечивает защиту от непредвиденного изменения структуры файла;
- модуль графического интерфейса. Является связующим звеном между приложением и пользователем; отображает данные о ходе выполнения задания в текстовом и графическом виде.

3 Программная реализация

На данном этапе работы необходимо выбрать инструменты для кодирования приложения (язык программирования, библиотеки), выполнить реализацию разработанных ранее модулей и алгоритмов, а также разработать графический интерфейс пользователя.

3.1 Выбор инструментов разработки

Так приложение должно запускаться на различных ОС, а также быть простым в установке и запуске, необходимо выбрать компилируемый ЯП с возможностью создания «родных» для платформы исполняемых файлов. Альтернативным вариантом будет использование интерпретируемого ЯП, который будет иметь возможность запускаться на многих ОС. Помимо прочего, для реализации графического интерфейса необходимо выбрать кроссплатформенную библиотеку для его построения.

Среди доступных ЯП имеются:

- Golang;
- C++;
- Rust;
- C#;
- Python

Среди кроссплатформенных библиотек для построения графического интерфейса имеются:

- GTK (от сокращения GIMP Toolkit) – кроссплатформенная библиотека элементов интерфейса. Написана на C, доступны интерфейсы для других языков программирования, в том числе для C++, C#, Python. Для проектирования графического интерфейса доступно приложение Glade;
- Qt - кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов

графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Комплектуется визуальной средой разработки графического интерфейса Qt Designer, позволяющей создавать диалоги и формы в режиме WYSIWYG;

- Tkinter (**Tk inter**face) - кросс-платформенная графическая библиотека на основе средств Tk (широко распространённая в мире GNU/Linux и других UNIX-подобных систем, портирована также и на Microsoft Windows). Tkinter входит в стандартный дистрибутив Python. Имеет множество готовых графических компонентов, которые могут понадобиться при создании приложения. Имеет небольшой порог входа в силу простоты создания приложений на языке Python.
- wxWidgets - кроссплатформенная библиотека инструментов с открытым исходным кодом для разработки кроссплатформенных на уровне исходного кода приложений. Основным применением wxWidgets является построение графического интерфейса пользователя, однако библиотека включает большое количество других функций и используется для создания весьма разнообразного ПО. Важная особенность wxWidgets: в отличие от некоторых других библиотек (GTK, Qt и др.), она максимально использует «родные» графические элементы интерфейса операционной системы всюду, где это возможно. Это существенное преимущество для многих пользователей, поскольку они привыкают работать в конкретной среде, и изменения интерфейса программ часто вызывают затруднения в их работе. Написана на C++. Для проектирования графического интерфейса доступно приложение wxGlade;
- AvaloniaUI - кроссплатформенный XAML-фреймворк для построения графических интерфейсов пользователя для платформ .NET Framework, .NET Core и Mono.

При этом нужно учитывать, что не все комбинации языков программирования с библиотеками графического интерфейса возможны, удобны в разработке и достаточно стабильны. Для языков Golang и Rust не имеются стабильные версии перечисленных библиотек, для GTK сборка C++- приложений под Windows

требует нетривиальной настройки окружения вплоть до эмуляции Linux-окружения в Windows, а AvaloniaUI доступна только под C# и не имеет еще стабильной версии. Среди доступных вариантов остаются:

- C++ и wxWidgets;
- C# и GTK;
- C++ и Qt;
- Python и Tkinter.

В ходе ознакомления с библиотекой wxWidgets были выявлены следующие недостатки: недостаточно подробная документация, нетривиальная настройка режима Drag’n’Drop (необходим для перемещения элементов ГПИ, представляющих блоки памяти), малообъемный функционал конструктора форм wxGlade.

У GTK графические элементы выглядят достаточно непривычно в сравнении с родными для Windows приложениями; имеет те же недостатки, что и wxWidgets.

В силу простоты, мощности и популярности был выбран язык Python и билиотека Tkinter.

В качестве системы сборки выбрана Cmake — одна из наиболее популярных среди проектов, написанных на Python.

3.2 Реализация модулей приложения

3.2.1 Модуль добавления, изменения команд

В данном модуле должны быть реализованы алгоритмы добавления, изменения команд пользователя.

Диаграмма классов данного модуля представлена на рисунке 5.

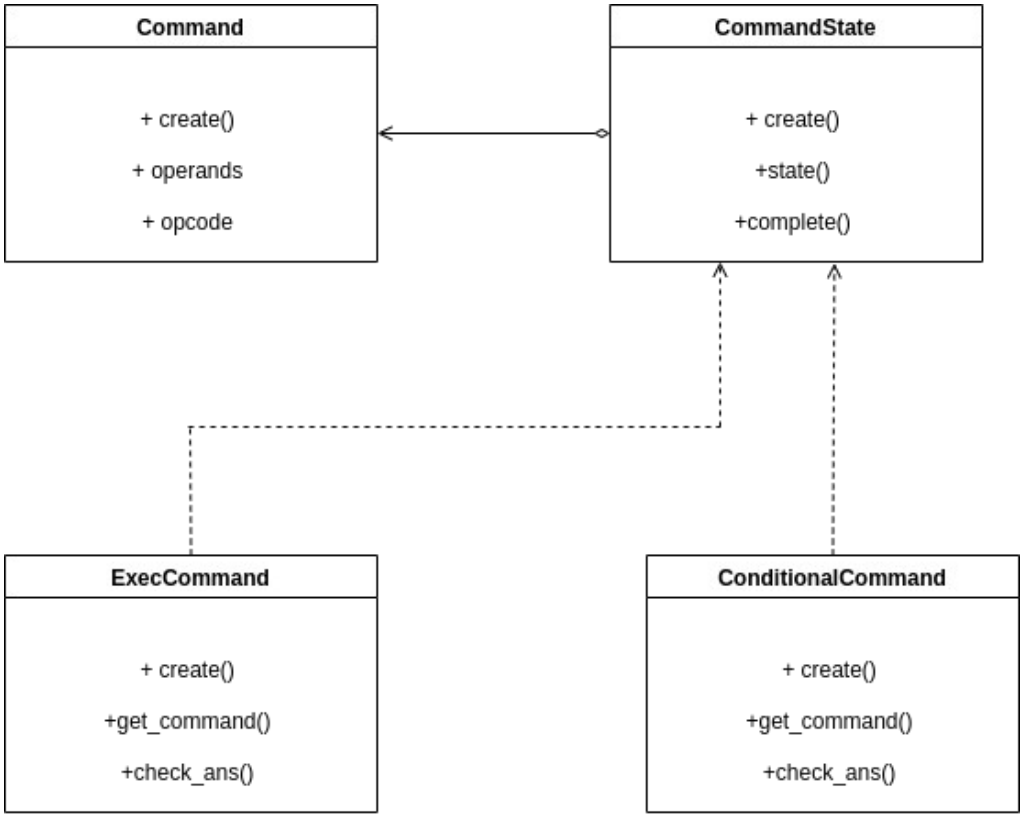


Рисунок 5 — Диаграмма классов модуля обраотки команд

3.2.2 Модуль загрузки и сохранения заданий в файл

Данный модуль реализует функционал сохранения текущего прогресса выполнения задания в файл, загрузку задания из файла с защитой от непредвиденных изменений.

Состояние задание определяется такими параметрами как:

- список всех заданий;
- количество уже выполненных заданий;
- количество допущенных ошибок пользователя;
- список введенных команд;

Все перечисленные выше параметры должны быть сохранены в файл.

В качестве формата файла задания был выбран JSON, потому что он имеет синтаксис, достаточно удобный как для человека, так и для машины. JSON является текстовым форматом, а не бинарным, поэтому файлы в формате JSON можно просматривать и редактировать в любом текстовом редакторе. Задания в файле сохранены в виде массива JSON-объектов. Структура объекта задания представлена в таблице 1.

Таблица 1 — Структура обьекта задания

Поле	Тип	Описание
type	number	Тип задания
completed	number	Выполнено ли задание
fails	number	Количество допущенных ошибок
commands	array	Массив команд, которые ввел пользователь

Структура обьекта, включащего в себя представление EXE программы представлена в таблице 2.

Таблица 2 — Структура EXE программы

Поле	Тип	Описание
header	array	Массив, включающий в себя байты заголовка
stack	array	Массив, включающий в себя значения стека
main	array	Массив, включающий в себя значения главной части программы

3.2.3 Модуль генератор-заданий

Данный модуль исполнен в виде набора функций и классов, реализующих алгоритм генерации EXE файла для выполнения заданий, содержащий следующие методы:

- rand_command() - выбирает случайны образом команду
- generate() - используется для генерации задания
- create_header() - создает описание заголовка EXE файла
- create_main() - создает описание главной части EXE файла

3.2.4 Модуль графического интерфейса

Проанализировав интерфейс предыдущей установки, было принято решение внести в него несколько небольших изменений:

- в окне первого задания после успешного ввода значения в строку, запретить редактирование данных с целью защиты от ненужных действий пользователя
- была добавлена возможность перемещения между полями вводы при помощи нажатии кнопки «Tab», что существенно увеличивает скорость ввода ответов
- была добавлена возможность смены фокуса после нажатия кнопки «Enter»
- при попытке закрыть главное окно программы будет выводится диалог подтверждения, чтобы исключить потерю результатов из-за случайного закрытия программы
- на основные действия пользователя были добавлены горячие клавиши

Экранная форма основного окна программы представлена на рисунке 6.

Рисунок 6 - Экранная форма основного окна программы

Диаграмма классов представлена на рисунке 7.

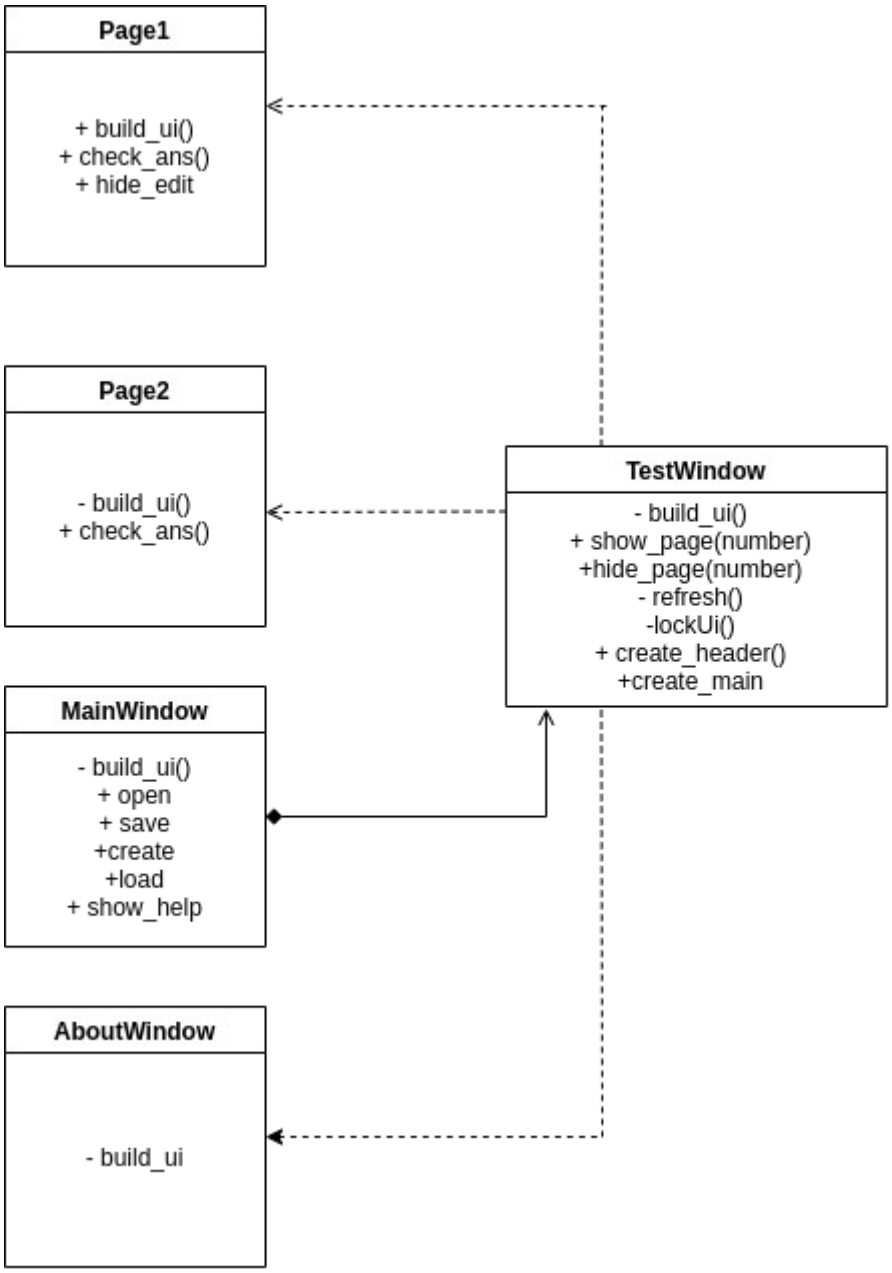


Рисунок 7 — Диаграмма классов модуля графического интерфейса

На основе разработанной структуры приложения и алгоритмов функционирования была выполнена программная реализация приложения. Модули приложения были реализованы в виде классов и функций; в графический интерфейс пользователя были внесены небольшие улучшения, а также был разработан формат файлов заданий. Диаграмма разработанных классов представлена в приложении А.

Заключение

В ходе выполнения курсового проекта было разработано программное обеспечение – лабораторная установка «Дизассемблирование». Изначально предполагалось исправление и доработка существующей установки, однако такая задача оказалось крайне сложной в сравнении с разработкой нового приложения.

В качестве направления дальнейшего развития можно выбрать разработку отдельного конструктора заданий для преподавателя и обеспечение шифрования файлов заданий.

Перечень сокращений

ЯП — язык программирования

ГПИ — графический пользовательский интерфейса

ОС — операционная системам

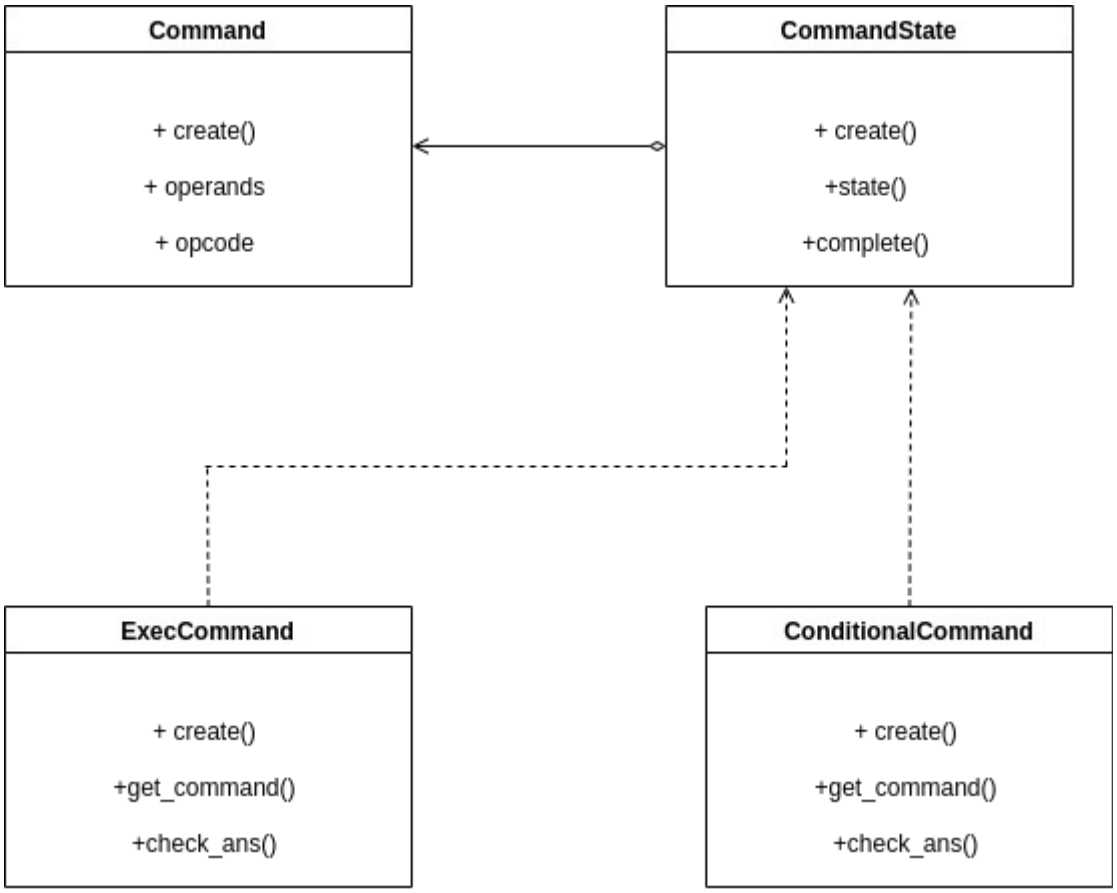
ПК — персональный компьютер

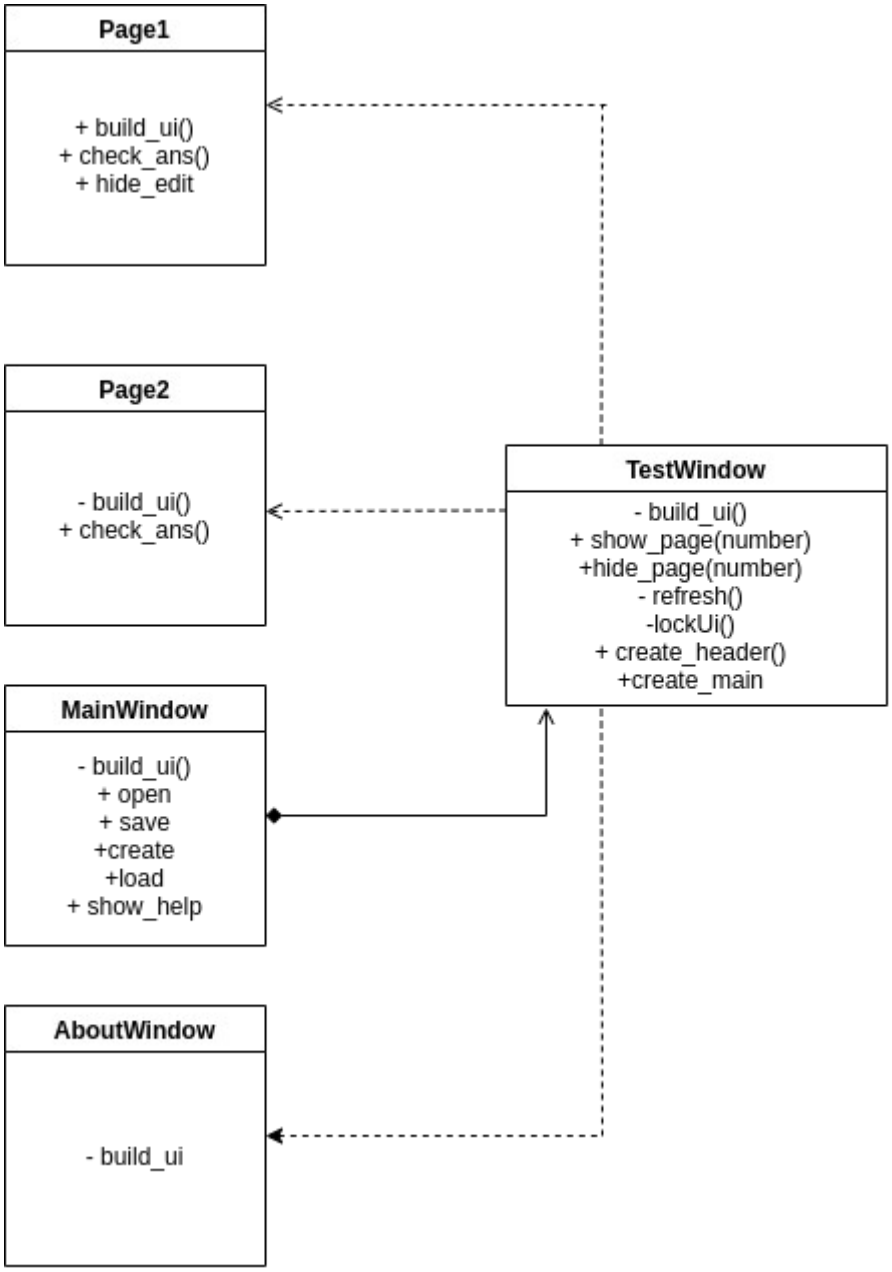
КОП — код операции

Приложение А

(обязательное)

Диаграммы классов

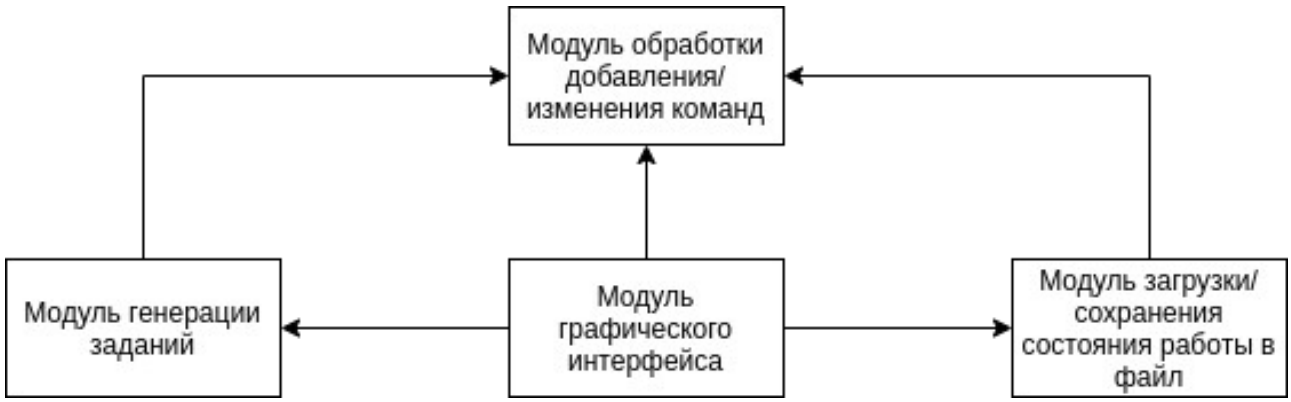



Page1
+ build_ui()
+ check_ans()
+ hide_edit
Page2
- build_ui()
+ check_ans()
TestWindow
- build_ui()
+ show_page(number)
+hide_page(number)
- refresh()
-lockUi()
+ create_header()
+create_main
MainWindow
- build_ui()
+ open
+ save
+create
+load
+ show_help
AboutWindow
- build_ui

Приложение Б

(обязательное)

Модульная структура приложения



Приложение В

(справочное)

Список использованных источников и материалов

- https://docs.python.org/3/library/tkinter.html
- https://ru.wikipedia.org/wiki/Tkinter
- http://effbot.org/tkinterbook/
- https://ru.wikipedia.org/wiki/WxWidgets
- https://ru.wikipedia.org/wiki/Qt
- https://ru.wikipedia.org/wiki/GTK
- https://www.python.org/

Приложение Г

(обязательное)

Библиографический список

1. ГОСТ 19.701–90 Единая система программной документации. М.: Изд-во стандартов, 1990.

2. Караваева О.В. Ассемблеры и компиляторы: Изд-во ВятГУ, 2011. – 82 с.

Приложение Д

(обязательное)

Листинг программы

```
from mainwindow import MainWindow

if __name__ == "__main__":
    app = MainWindow()
    app.mainloop()

import tkinter as tk
from random import randint

class HexEditor(tk.Toplevel):
    def __init__(self, parent=None):
        super().__init__(master=parent)
        self.screen_height = self.winfo_screenheight()
        self.resizable(False, False)
        self.prev_selected = (None, None)
        self.header = []
        self.dump = []
        self.create_header()
        self.add_zero_bytes(28*16+14)
        self._build_ui()


    def add_zero_bytes(self, count):
        self.dump += tuple('00' for _ in range(count))

    def generate_ans_p1(self):
        ans = []
        ans += (65535, 0)
```

```
            ans.append([self.header[0x19] + self.header[0x18] + 'H', \
                                '0X' + self.header[0x19] + self.header[0x18]])
            ans += (1, 0x20)
            ans.append([self.header[0xF] + self.header[0xE] + 'H', \
                                '0X' + self.header[0xF] + self.header[0xE]])
            ans.append([self.header[0x17] + self.header[0x16] + 'H', \
                                '0X' + self.header[0x17] + self.header[0x16]])
            ans.append([self.header[0x15] + self.header[0x14] + 'H', \
                                '0X' + self.header[0x15] + self.header[0x14]])
            print(self.header[0x3] + self.header[0x2])
            ans.append(1024 - int(self.header[0x3] + self.header[0x2], 16))
            return ans

    def _build_ui(self):
            self.title("Просмотр кода")
            self.geometry("465x475+600+{0}".format(self.screen_height // 2 - 150))


            canvas = tk.Canvas(self)
            canvas.pack(side='top', expand="yes")
            table = tk.Frame(canvas)
            table.pack(side="left")


            scrollbar = tk.Scrollbar(self, orient='vertical')
            scrollbar.pack(side='right', fill='y')
            scrollbar['command'] = canvas.yview
            canvas['yscrollcommand'] = scrollbar.set


            self._widgets = []
```

```
            for row in range(len(self.dump) // 16):
                current_row = []
                #labels
                for column in range(16):
                    label = tk.Label(table, text=self.dump[row * 16 + column],
borderwidth=0, anchor=tk.W, justify=tk.LEFT)
                    label.grid(row=row, column=column, sticky="nsew", padx=5)
                    current_row.append(label)

                self._widgets.append(current_row)

            for row in range(len(self.dump) // 16):
                for column in range(16):
                    self._widgets[row][column].bind("<Button-1>", lambda event,
q=row, p=column: self._select_byte(q, p))
                    self._widgets[row][column].bind("<Left>", lambda event,
q=row, p=column: self._move_left(q, p))

            toolbar = tk.Frame(self, bd=1, relief=tk.RAISED)
            toolbar.pack(side=tk.BOTTOM, fill=tk.X)

            binary_text = tk.Label(toolbar, text='Двоичное представление', padx=15)
            binary_text.pack(side=tk.LEFT)
            self.binary_byte = tk.Label(toolbar)
            self.binary_byte.pack(side=tk.LEFT)

        def _move_left(self, q, p):
            print(q, p)
            if p > 0:
                self._widgets[q][p - 1].config(fg='red')
                _q = q
```

```
                    _p = p - 1
                else:
                    self._widgets[q - 1][15].config(fg='red')
                    _q = q - 1
                    _p = 15
                self._widgets[_q][_p].focus_set()
                if self.prev_selected != (None, None):
                    self._widgets[self.prev_selected[0]]
[self.prev_selected[1]].config(fg='black')
                self.prev_selected = (q, p)
                print(self.prev_selected)

        def _select_byte(self, q, p):
                self._widgets[q][p].focus_set()
                self._widgets[q][p].config(fg='red')
                #self.binary_byte['text'] = format(self._widgets[q][p]['text'], '08b')
                if self.prev_selected != (None, None):
                    self._widgets[self.prev_selected[0]]
[self.prev_selected[1]].config(fg='black')
                self.prev_selected = (q, p)

        def create_header(self):
                self.header += ('4D', '5A')
                self.header += (format(randint(50, 200), 'X'), '00')
                self.header += ('03', '00')
                self.header += ('01', '00')
                self.header += ('20', '00')
                self.header += ('00', '00')
                self.header += ('FF', 'FF')
                self.header += (format(randint(100, 200), 'X'), '00')
                self.header += (format(randint(16, 255), 'X'), '01')
```

```
            self.header += ('00', '00')
            self.header += (format(randint(16, 255), 'X'), '01')
            self.header += ('01', '00')
            self.header += ('3E', '00')
            self.header += ('00', '00')
            self.header += ('01', '00')
            self.header += ('FB', '50')
            self.header += ('6A', '72')
            self.header += tuple('00' for _ in range(28))
            self.header += ('35', '01')
            self.header += ('01', '00')

            self.dump += self.header

import tkinter as tk
from tkinter.messagebox import showinfo
from hex_editor import HexEditor

class Page(tk.Frame):
    def __init__(self, *args, **kwargs):
        tk.Frame.__init__(self, *args, **kwargs)

    def show(self):
        self.lift()

class Page2(Page):
    def __init__(self, parent=None):
        self.parent = parent
        Page.__init__(self)

    def _build_ui(self):
```

```
            pass


class Page1(Page):
    def __init__(self, parent=None):
            self.parent = parent
            Page.__init__(self)
            #label = tk.Label(self, text="asd")
            self.params = [["Макс. требуемый объем памяти", "Dec"],
            ["Мин. требуемый объем памяти", "Dec"],
            ["Смещение Relocation Table", "Hex"],
            ["Кол-во эл-тов в Relocation Table", "Dec"],
            ["Кол-во параграфов в заголовке", "Dec"],
            ["Смещение сегмента стека", "Hex"],
            ["Смещение сегмента кода", "Hex"],
            ["Значение указателя команд", "Hex"],
            ["Длина загружаемой части", "Dec"]]
            self._completed_asks = 0
            self._build_ui()

    def _check_ans(self, text, row):
            text = text.upper()
            if row == 0 or row == 1 or row == 3 or row == 4 or row == 8:
                try:
                    if int(text) != self.parent._ans_p1[row]:
                        showinfo(title="Ошибка", message="Неправильно
введенное число")
                        self._canvases[row].itemconfig(1, fill="red")
                    else:
                        self._canvases[row].itemconfig(1, fill="green")
                        self._completed_asks += 1
                        self._widgets[row][2].config(state='disabled')
```

```
                    self._widgets[row][2].config(font=('Helvetica', 10,
'bold'))
                    if row < 8:
                        self._widgets[row+1][2].focus_set()
                except ValueError:
                    showinfo(title="Ошибка", message="Неправильно
введенное число")
            else:
                if text not in self.parent._ans_p1[row]:
                    showinfo(title="Ошибка", message="Неправильно
введенное число")
                    self._canvases[row].itemconfig(1, fill="red")
                else:
                    self._canvases[row].itemconfig(1, fill="green")
                    self._completed_asks += 1
                    self._widgets[row][2].config(state='disabled')
                    self._widgets[row][2].config(font=('Helvetica', 10, 'bold'))
                    if row < 8:
                        self._widgets[row+1][2].focus_set()

            if self._completed_asks == len(self.params):
                self.next_button.config(state="normal")

        def _build_ui(self):
            nameLabel = tk.Label(self, text="Заголовок EXE файла")
            nameLabel.pack(side="top")

            table = tk.Frame(self)
            self._widgets = []
            self._canvases = []
            for row in range(len(self.params)):
```

```
                current_row = []
                #labels
                for column in range(len(self.params[0])):
                        label = tk.Label(table, text=self.params[row][column],
borderwidth=0, anchor=tk.W, justify=tk.LEFT)
                        label.grid(row=row, column=column, sticky="nsew",
padx=10)
                        current_row.append(label)

                #entry
                entry = tk.Entry(table, highlightcolor="blue")
                entry.insert(0, "Не задано")
                entry.grid(row=row, column=2, sticky="nsew")
                current_row.append(entry)

                #draw circle for check correct answers
                canvas = tk.Canvas(table, width="20", height="20")
                canvas.create_oval(4, 4, entry["width"] - 6, entry["width"] - 6,
width=2)
                canvas.grid(row=row, column=3)
                current_row.append(canvas)
                self._widgets.append(current_row)
                self._canvases.append(canvas)

        for row in range(len(self.params)):
                self._widgets[row][2].bind("<Return>", lambda event, q=row:
self._check_ans(self._widgets[q][2].get(), q))
                self._widgets[row][2].bind("<FocusIn>", lambda event, q=row:
self._widgets[q][2].delete(0, tk.END))
                self._widgets[row][2].bind("<FocusOut>", lambda event, q=row:
self._widgets[q][2].insert(0, "Не задано"))
```

```
            table.pack(side="top")

            #buttons next and exit
            exit_button = tk.Button(self, text="Выход", borderwidth=0,
activeforeground="red", command=self.parent.parent.destroy)
            exit_button.pack(side="right", padx=10)

            self.next_button = tk.Button(self, text="Далее", state="disabled",
borderwidth=0, activeforeground="red")
            self.next_button.pack(side="right")


class TestFrame(tk.Frame):
    def __init__(self, parent=None):
            super().__init__(master=parent)
            self.parent = parent
            self._build_ui()

            self._ans_p1 = self.parent._hexeditor.generate_ans_p1()

    def _build_ui(self):
            self.p2 = Page2(self)
            self.p1 = Page1(self)

            self.p1.pack(side="bottom",fill="x")


class MainWindow(tk.Tk):
    def __init__(self):
            super().__init__()
```

```
        screen_height = self.winfo_screenheight()
        self.geometry("430x280+{0}+{1}".format(150, screen_height // 2 - 100))
        self.resizable(False, False)
        self._build_ui()

    def _build_ui(self):
        self.title("Лабораторная работа по СисПО")

        self._hexeditor = HexEditor(self)

        self._testFrame = TestFrame(self)
        self._testFrame.pack(fill=tk.BOTH, expand=1)
```