**СОДЕРЖАНИЕ**

[СПИСОК СОКРАЩЕНИЙ И ОПЕРЕДЕЛЕНИЙ 2](#_Toc406618953)

[ВВЕДЕНИЕ 3](#_Toc406618954)

[1. Постановка задачи 4](#_Toc406618955)

[2. Разработка эмулятора 5](#_Toc406618956)

[2.1. Анализ набора команд и их характеристика 5](#_Toc406618957)

[2.2. Выбор формата ассемблерных команд и допустимых операций 6](#_Toc406618958)

[2.3. Составление возможных ошибок записи команд 8](#_Toc406618959)

[2.4. Формирование задач, связанных с преобразованием команд 8](#_Toc406618960)

[2.5. Выбор формы и представления исходных данных и результатов 10](#_Toc406618961)

[2.6. Выбор и обоснование методов решения задачи 12](#_Toc406618962)

[2.7. Разработка подпрограммы функций для решения подзадач 12](#_Toc406618963)

[2.8. Разработка и обоснование основной программы 14](#_Toc406618964)

[2.9. Методика отладки и тестирования 15](#_Toc406618965)

[ЗАКЛЮЧЕНИЕ 20](#_Toc406618966)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 21](#_Toc406618967)

[ПРИЛОЖЕНИЕ 1 22](#_Toc406618968)

[ПРИЛОЖЕНИЕ 2 43](#_Toc406618969)

[ПРИЛОЖЕНИЕ 3 53](#_Toc406618970)

[ПРИЛОЖЕНИЕ 4 54](#_Toc406618971)

# СПИСОК СОКРАЩЕНИЙ И ОПЕРЕДЕЛЕНИЙ

**ЭВМ** – Электронно-вычислительная машина

**АЛУ** – Арифметико-логическое устройство  
**ОЗУ** – Оперативное запоминающее устройство

**РОН** – Регистр общего назначения

# ВВЕДЕНИЕ

В настоящее время, не смотря на изобилие различного рода языков высокого уровня, которые подчас представляют возможности визуальной разработки программ, все еще находит своё применение язык ассемблер. В отличие от высокоуровневых языков, где оператору сопоставлено множество машинных кодов, в ассемблере существует однозначное соответствие между командой и машинным кодом. В этом заключается преимущество низкоуровневых языков перед высокоуровневыми. Во-первых, создавая программу на ассемблере, не обязательно использовать многие модули и дополнительные операции, которые добавляются автоматически в высокоуровневых программах. Это приводит как к уменьшению длины кода программы, так и к увеличению ее быстродействия. Естественно, всё преимущества могут быть достигнуты лишь при корректном использовании ресурсов ЭВМ. Во-вторых, существует возможность решения обратной задачи: перевода машинного кода в программу.

Целью данного курсового проекта является разработка и реализация программной модели вычислительной системы и эмулятора ассемблерных команд (согласно заданию), а также составление данной пояснительной записки в соответствии с ГОСТ (а именно единой системой программной документации – ЕСПД).

Поставленная задача даёт возможность понять работу некоторых команд такого низкоуровневого языка, как ассемблер, принцип обработки команд процессором, что даст возможность узнать о типах команд, адресации и о методах реализации хранения данных, а также работы с ними.

В свою очередь данный курсовой проект позволит студенту закрепить (а местами и приобрести) навыки по алгоритмизации типовых задач, ознакомиться в общих чертах с внутренней структурой ЭВМ, а также получить необходимые навыки для оформления проектов, технических заданий и прочей программной документации.

# Постановка задачи

Необходимо разработать эмулятор ассемблерных команд. Согласно варианту задания есть ряд исходных данных:

1. Система команд:
   * Чтение.
   * Запись.
   * Сложение.
   * Вычитание.
   * Логическое “И”.
   * Переход.
2. Объём оперативной памяти: 256 байт.
3. Длина команды: 1 байт.
4. Операнды: регистр-регистр.

Для эффективной реализации курсового проекта необходимо разбить общую задачу на подзадачи, отсюда можно выделить этапы, формирующие следующее техническое задание:

1. Анализ набора команд и их характеристика

* Необходимо проанализировать данную систему команд, определить действия, выполняемые этими командами.

1. Выбор формата ассемблерных команд и допустимых операций

* После анализа необходимо определиться с форматом команд и операций, определить назначение битов и тип адресации.

1. Составление возможных ошибок записи команд

* После выполнения пункта 2 может возникнуть ряд проблем и ошибок, следовательно – необходимо проанализировать каждую команду и описать все возможные ошибки.

1. Формирование задач, связанных с преобразованием команд

* Исходя из пунктов 1-3 нужно составить алгоритм преобразования команд из мнемокода.

1. Выбор формы и представления исходных данных и результатов
   * Для полноценной работы программы ей необходимо оперировать данными, в этом пункте следует определиться с представлением исходных данных и результатов.
2. Выбор и обоснование методов решения задачи
   * Описание и обоснование используемых для решения задачи методов.
3. Разработка подпрограммы функций для решения подзадач
   * Реализация методов из пункта 6.
   * Разработка графической оболочки для удобства управления программой
4. Разработка и обоснование основной программы
   * Разработка основной программы, подключение «модулей» - методов из пунктов 6-7.
   * Реализации программы для ОС Android
5. Методика отладки и тестирования
   * Поверка работоспособности программы, тестирование, анализ на соответствие требованиям задания, а также пример работы.

# Разработка эмулятора

## Анализ набора команд и их характеристика

Предусмотрим работу только с целыми числами размером в 1 байт, принадлежащих интервалу [-128;127]. Число -128 зарезервируем для служебных целей, которые будут описаны далее. Таким образом пользователю предоставляется симметричный диапазон чисел [-127;127]. Данные необходимо вводить в шестнадцатеричной форме, таким образом имеем диапазон [81;7F].

На входе программы необходимо иметь мнемокод – для этого необходимо предусмотреть возможность выбирать текстовый файл. Соответственно, на выходе необходимо получить листинг и объектный код, который можно будет вывести на экран.

Входной файл можно создавать в любом редакторе, но он должен соответствовать определённым стандартам, если соответствия нет – автор не несёт ответственности за корректность выполнения программы. Для каждой корректно введённой команды в листинге необходимо формировать строку трансляции, а в объектный код добавлять соответствующие ей байты.

Представим данные и программу в виде простой схемы (Рис. 1).

Строки мнемокода

Ассемблер

листинг

Объектный код

*Рис. 1. Примитивная схема данных и программы*

Введём ещё две дополнительные команды, необходимость ввода которых будет описана в дальнейшем – реверсировать, дублировать.

Для данной системы команд можем произвести краткое описание:

* Чтение – чтение данных из памяти для дальнейшего использования (данные помещаются в регистр).
* Запись – запись данных непосредственно из команды в память.
* Запись – запись данных из регистра в область памяти для дальнейшего хранения.
* Сложение.
* Вычитание.
* Логическое “И”.

Для трёх операций, указанных выше, соответственно сложение, вычитание и логическое умножение, операции производятся с данными, содержащимися в регистрах.

* Реверсировать – меняет местами два верхних значения стека.
* Дублировать – копирует верхнее значение стека и записывает его.
* Переход – переход к установленной метке.

## Выбор формата ассемблерных команд и допустимых операций

Для всех команд, кроме чтения и записи будем использовать безадресные команды. Память реализуем в виде стека. Для упрощения работы со стеком и вводятся команды дублировать и реверсировать.

Все команды в соответствии с заданием имеют длину 1 байт, кроме команды запись (когда запись идёт непосредственно из команды), которая имеет длину 2 байта (так как в команде должны присутствовать и данные, длина которых 1 байт)

Запись может быть пользовательская (указываем, что записать), а может быть запись данных из регистра.

Для удобства составим таблицу с подробным описанием команд (Таблица 1).

**Таблица 1. Описание системы команд**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Код** | **Название** | **Мнемокод** | **Описание** | **Бит t** | **Рис.** |
| 000 | Чтение | LOAD regN | Загрузка данных из памяти mem в регистр reg1 или reg2 | 0 | 3 |
| 001 | Запись | STORE regN | Запись данных из регистра reg1 или reg2 в память mem | 0 | 3 |
| 001 | Запись | STORE reg0 DATA | Запись данных в память mem | 1 | 4 |
| 010 | Сложение | ADD | reg1 = reg1 + reg2 | 0 | 2 |
| 011 | Вычитание | SUB | reg1 = reg1 - reg2 | 0 | 2 |
| 100 | Логическое И | AND | reg1 = reg1 & reg2 | 0 | 2 |
| 101 | Переход | JO | Завершение выполнения при переполнении. | 0 | 2 |
| 110 | Реверсировать | REV | A B B A | 0 | 2 |
| 111 | Дублировать | DUP | A A  A | 0 | 2 |

Начальные данные содержат текст со строками длинной не более 255 символов. Так же количество строк не должно быть более 255. Формат исходной строки имеет следующий вид:

команда (пробел\-ы) параметр\_1 (пробел\-ы) парметр\_2 (пробел\-ы) (точка с запятой); комментарий

Обязательными является только команда, параметр\_1 и параметр\_2 могут быть опущены (в зависимости от команды), комментарий также не является необходимым.

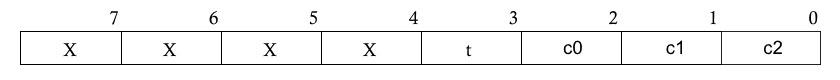
команда – это слово, соответствующее в ассемблере некоторому действию (см. Таблицу 1).

параметр\_1 – это regN – номер регистра (1, 2 или 0).

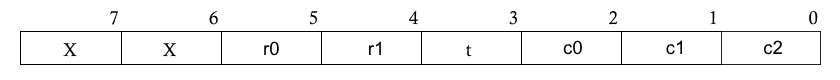
параметр\_2 – это данные для записи в память в шестнадцатеричной системе счисления

В таблицу бит t – автоматически выставляется при интерпретации, он отвечает за длину команды (1 или 2 байта).

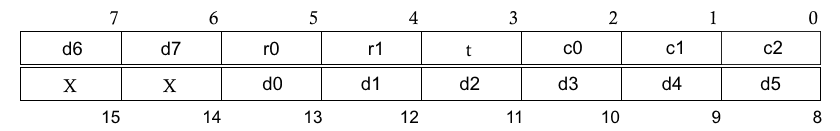
Схематически интерпретированную команду можно представить в виде трёх рисунков (Рис. 2, Рис. 3 и Рис. 4).



*Рис. 2. Структура команды при t=0 и без параметров*



*Рис. 3. Структура команды при t=0 и одним параметром*



*Рис. 4. Структура команды при t=1 и двумя параметрами*

Где c0-c2 – это биты кода операции, бит t отвечает за дальнейшую структуру команды, r0-r1 отвечают за номер регистра (00 – отсутствует, 01 – первый, 10 – второй, 11 – не приемлемо), d0-d7 – непосредственно данные, X – значит, что данные биты не рассматриваются (будем считать их условно нулевыми).

Разделителем считается только пробел, поэтому его может быть неограниченное количество. Строка, идущая после символа «точка с запятой», так называемый комментарий, опускается при «ассемблировании».

В задачу программы входит перевод заданного во входном тексте мнемокода в соответствующий ему объектный код, сохранение листинга перевода. Если же синтаксические конструкции не были распознаны, то она, конструкция, интерпретируется как ошибка, классифицируется и сообщается пользователю. В результате мы должны получить объектный код и листинг.

Объектный код является последовательностью байт, сопоставленной с каждой из строк ассемблера. Объектный код записывается по ходу преобразования исходного текста.

Листинг является текстом и соответствует преобразованию каждой из строк исходного текста. Каждая из успешно преобразованных команд в листинге соответствует строке следующего формата:

КОД\_ОПЕР\_ДАННЫЕ КОМАНДА

Где КОД – это трёхразрядное двоичное число, соответствующее коду команды.

ОПЕР – операнд (при наличии) – номер регистра

ДАННЫЕ – непосредственные данные – двухразрядное шестнадцатеричное число

КОМАНДА – команда в мнемокоде.

Не преобразованная команда в листинге соответствует следующей форме:

ИСХОДНАЯ\_СТРОКА; > ОШИБКА

Где ОШИБКА – соответствующая причина остановки программы.

## Составление возможных ошибок записи команд

Первая ошибка, которая может возникнуть при подаче на вход исходного файла – это некорректное расширение. Данная программа поддерживает только 2 расширения – \*.txt и \*.w2f – второе расширение является искусственным, по сути это тот же текстовой документ. Так же пользователю необходимо учитывать ограничения длины строки и их количестве, описанные в предыдущем пункте.

Что касается конкретно записи команд, то можно выделить следующие ошибки:

* Неизвестная команда – в исходном файле обнаружена неподдерживаемая команда – в листинге выводится как “UNKNOWN COMMAND”
* Недопустимое количество операндов – в листинге выводится как “INCORRECT OPERANDS NUM”
* Неизвестные операнды – в листинге выводится как “UNKNOWN OPERANDS”

Так же могут возникнуть такие программные ошибки, как ошибка при закрытии файла и внезапное исчезновение файла, что будет отображено пользователю, как «ошибка при закрытии файла» и «файл на найден».

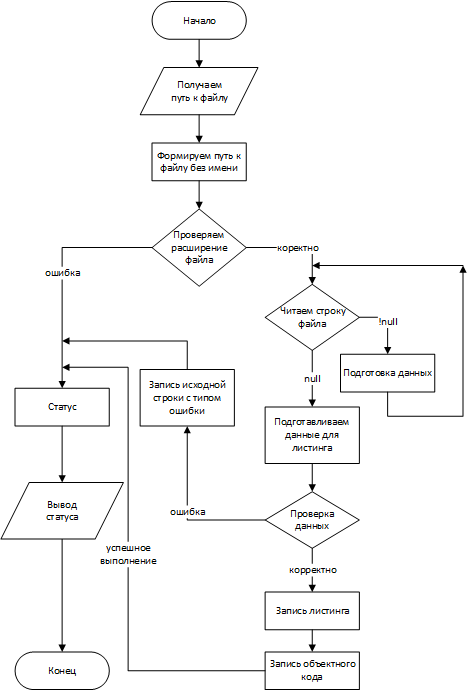
Если пользователь не укажет в конце какой-то строки символ «;», то программа не будет преобразована в объектный код.

Далее в соответствующем пункте будут приведены примеры с ошибками.

## Формирование задач, связанных с преобразованием команд

На вход программы подаётся исходный файл с мнемокодом. Ставится задача преобразовать данный файл в объектный код и листинг, попутно проверив его на возможные ошибки, описанные в предыдущем пункте.

За данное преобразование файла отвечает класс Assembler. На Рис.5 покажем алгоритм, используемый для преобразования.



*Рис. 5. Алгоритм преобразования команд в объектный код*

В классе содержаться следующие функции:

* public String toAssembl() – основная функция класса, непосредственно выполняющая ассемблирование
* private boolean checkName(String name) – функция проверки расширения файла
* private boolean makeListingFile(String[] data) – функция создания листинга
* private String prepareDataForListingFile(String data) – функция подготовки данных для листинга
* private String mnemoCodeToBinaryCode(String mnemo) – функция для преобразования кода операции из мнемокода в двоичный
* private boolean checkAreOperandsCorect(String[] operand, int num, String mnemo) – функция проверки корректности операндов
* private boolean closeFile(BufferedReader bf\_reader) и private boolean closeFile(BufferedWriter bf\_writer) – функции для закрытия потока чтения/записи
* private boolean makeObjectCodeFile(String[] data) – функция создания файла объектного кода
* private String prepareDataForObjectCodeFile(String data) – функция подготовки данных для объектного кода

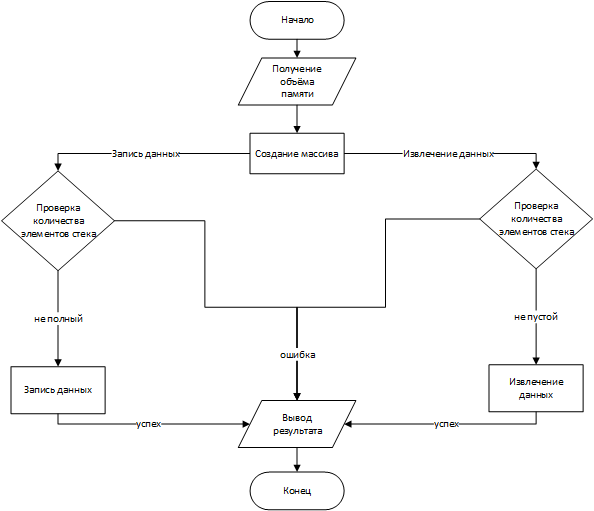
## Выбор формы и представления исходных данных и результатов

Как уже было описано ранее, исходные данные подаются в текстовом файле. Далее возникает вопрос в реализации хранения данных во время выполнения программы, а также способ хранения результатов.

Память в данной программе реализуется в виде стека. Стек – организация памяти, при которой используется принцип LIFO (Last In First Out), что дословно переводится как последним зашёл, первый вышел. Из недостатков можно выделить то, что мы не можем обращаться к стеку не линейно, то есть, мы можем получить значение, хранящееся в вершине стека и для того, чтобы, например, получить пятый элемент нам необходимо «извлечь» четыре вышестоящих.

Но не смотря на столь, казалось бы, существенный недостаток использование стека оправдывается тем, что мы упрощаем структуру команды. У нас пропадает необходимость указывать в команде область памяти куда следует записать данные, или же откуда следует извлечь.

Реализация памяти находится в классе Stack, алгоритм которого предоставлен на Рис. 6.



*Рис. 6. Алгоритм работы памяти*

В классе содержатся следующие функции:

* public boolean push(byte data) – функция записи данных в стек
* private boolean checkPush(byte[] st) – функция проверки стека на возможность записи данных
* public byte pop() – функция извлечения данных из стека
* private boolean checkPop(byte[] st) – функция проверки стека на возможность извлечения данных
* public int returnStackSizeValue() – функция возврата фактического размера стека (количества элементов)
* public byte[] showLeftData(int size) – функция для возврата массива с копией данных, хранящихся в стеке

В программе используются два регистра общего назначения (РОН) длиной в один байт и один регистр флагов, значения которого устанавливаются после каждой операции регистрами (сложение, вычитания, логическое ‘И’).

Регистры реализованы в классе Alu, структура которого будет рассмотрена позже. Для регистров использовались, соответственно, следующие переменные:

* private byte reg1
* private byte reg2
* private byte reg\_flag

Переменные имеют идентификатор доступа «private» потому как любой доступ к ним запрещён, кроме как из класса Alu (запись в регистры может осуществляться только арифметико-логическим устройством). Для получения значений регистров извне класса Alu были созданные соответствующие методы, возвращающие значения оных.

## Выбор и обоснование методов решения задачи

Из анализа условия задачи следует, что необходимо создать эмулятор работы микропроцессора с памятью, ассемблер и дизассемблер. Ассемблер будет преобразовывать исходный файл в листинг и объектный код, в свою очередь дизассемблер будет передавать объектный код эмулятору попутно изменяя его в понятный для него (эмулятора) код, а сам эмулятор будет эмулировать пошаговое выполнение команд, изменяя значения регистров.

Таким образом, мы выделили подзадачи:

1. Чтение данных из исходного файла, ассемблирование и дизассемблирование, вывод мнемоники пользователю. Заполнение памяти данными из исходного файла.

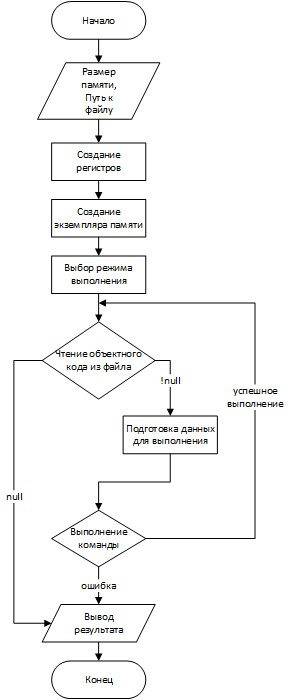
2.Пошаговое либо автоматическое последовательное выполнение команд, изменение значений регистров в соответствии с результатами работы программы:

- первый байт операции содержит код команды, количество операндов и их типы, все 3 значения выделяются из байта с помощью сдвигов и логического умножения на специальные маски. Затем из памяти считываются один или два следующих за командой байта, содержащие операнды.

3.Если ошибок не обнаружено, данная команда выполняется согласно условию, иначе выводится сообщение пользователю, и программа завершает выполнение, в случае неверного задания операндов пользователь получит сообщение об ошибке.

## Разработка подпрограммы функций для решения подзадач

После того как мы преобразовали исходный файл и определились с хранением данных становится задача выполнения команд. Для этого используется класс Alu, алгоритм которого приведён на Рис. 7.



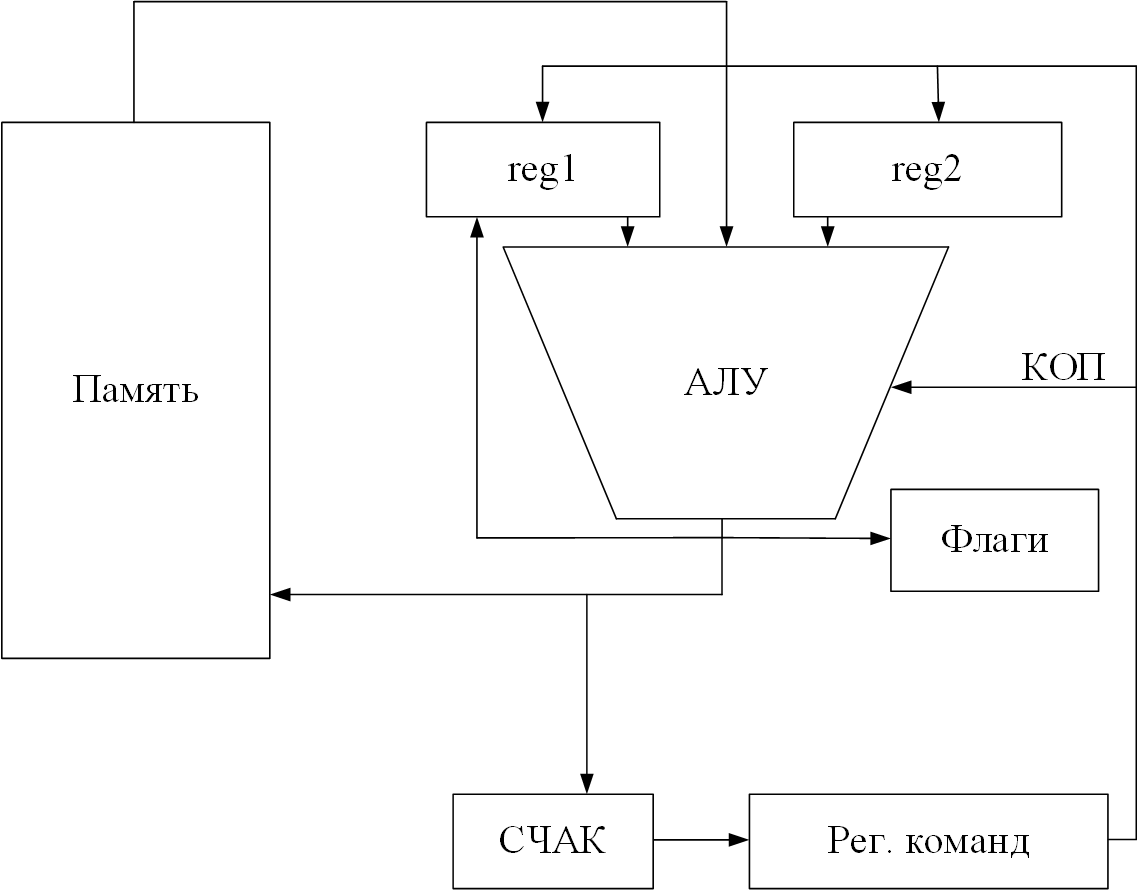
*Рис. 7. Алгоритм работы АЛУ*

В классе содержатся следующие функции:

* private boolean prepareDataFormFile() – функция подготовки данных для выполнения
* private boolean closeFile(BufferedReader bf\_reader) – функция закрытия потока чтения файла
* private short getComandFromString(String string) – функция возвращает код операции
* private boolean executeComand(int index) – функция запускает выполнение определённой команды в зависимости от кода операции
* private void setFlags(short data) – функция устанавливает флаги
* private boolean comandRead(String raw\_comand) – функция для команды чтения из памяти
* private boolean comandWrite(String raw\_comand) – функция для команды записи в память или регистр
* private boolean comandAdd() – функция для команды сложение
* private boolean comandSup() – функция для команды вычитание
* private boolean comandAnd() – функция для команды логическое ‘И’
* private boolean comandJump() – функция для перехода при переполнении
* private boolean comandRev() – функция для реверсирования данных в памяти
* private boolean comandDup() – функция для дублирования данных в памяти
* public String runAutoMode() – выполнение программы в автоматическом режиме
* public String runManualMode() – выполнение программы в ручном режиме

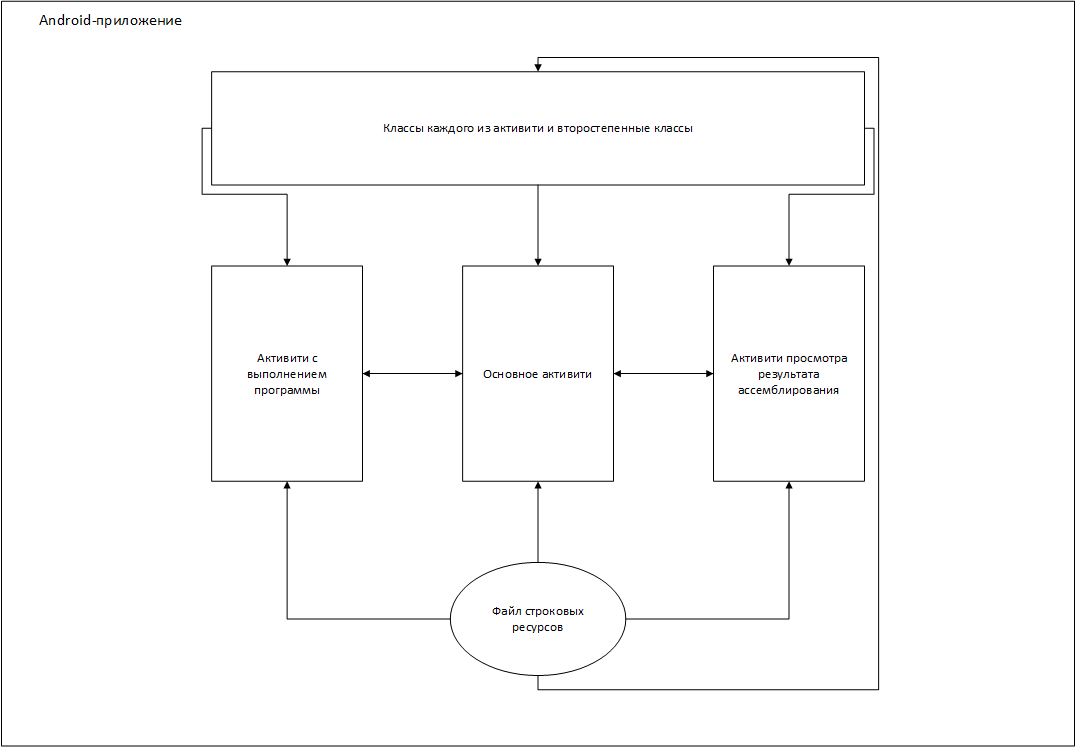
## Разработка и обоснование основной программы

Схематически эмулятор можно изобразить с помощью схемы (Рис. 8).



*Рис. 8. Схема эмулятора*

Теперь, когда все так называемые модули готовы можно собрать основную программу. Так как программа была написана в качестве Android-приложения на Рис. 8 графически проиллюстрируем работу приложения.



*Рис. 8. Принцип работы приложения*

Данная схема очень общая и упрощённая. Из неё мы можем увидеть, что строковые данные хранятся в отдельном файле (имеющем расширение .xml). Хранение всех строк там не обязательно, но является признаком хорошего тона. Доступ к ресурсам может быть осуществлён из любой точки программы.

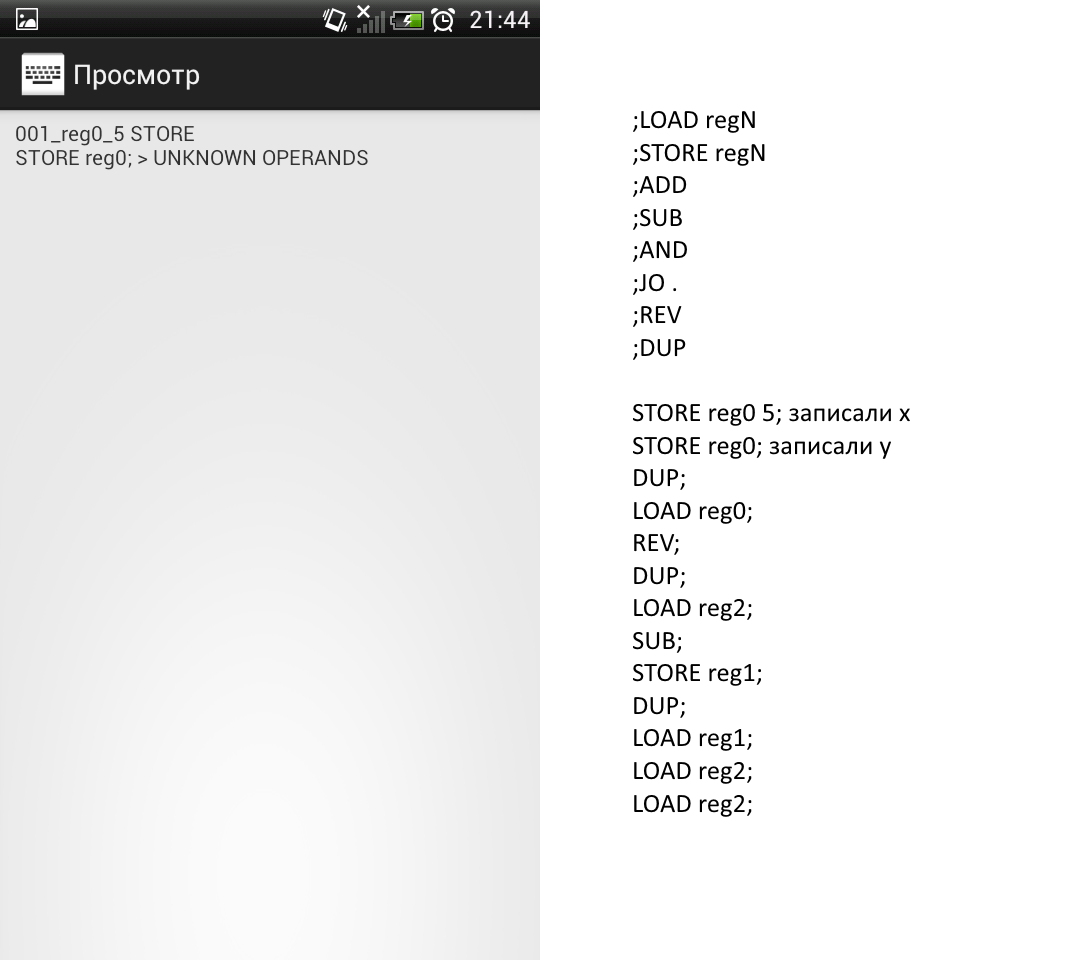
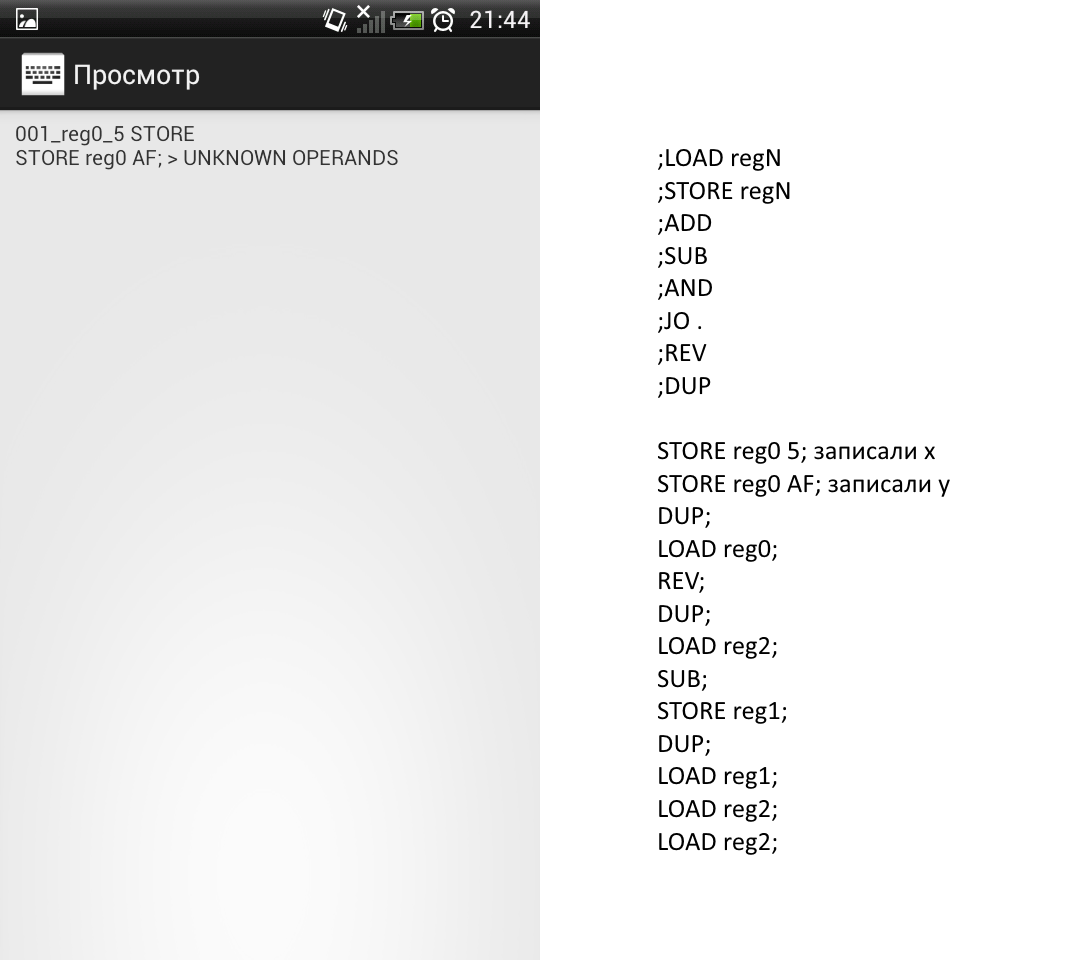
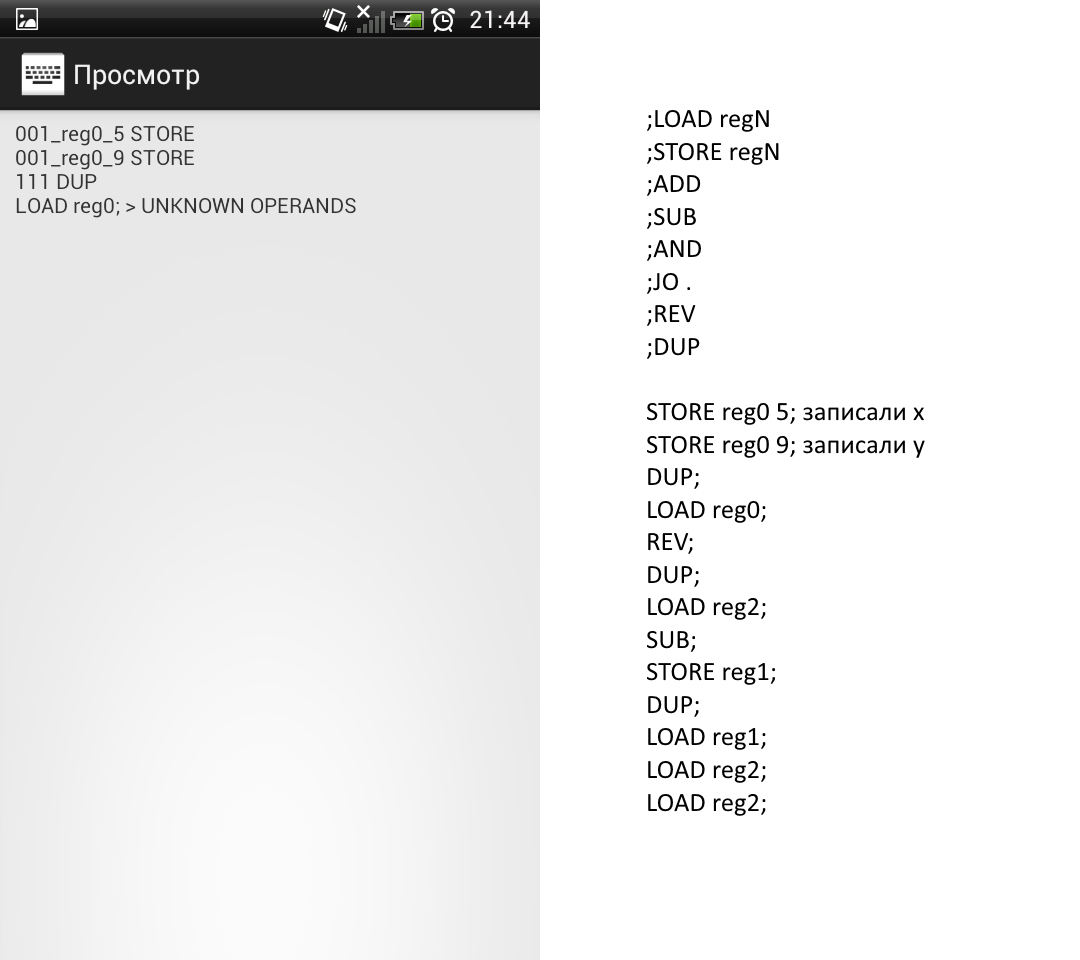
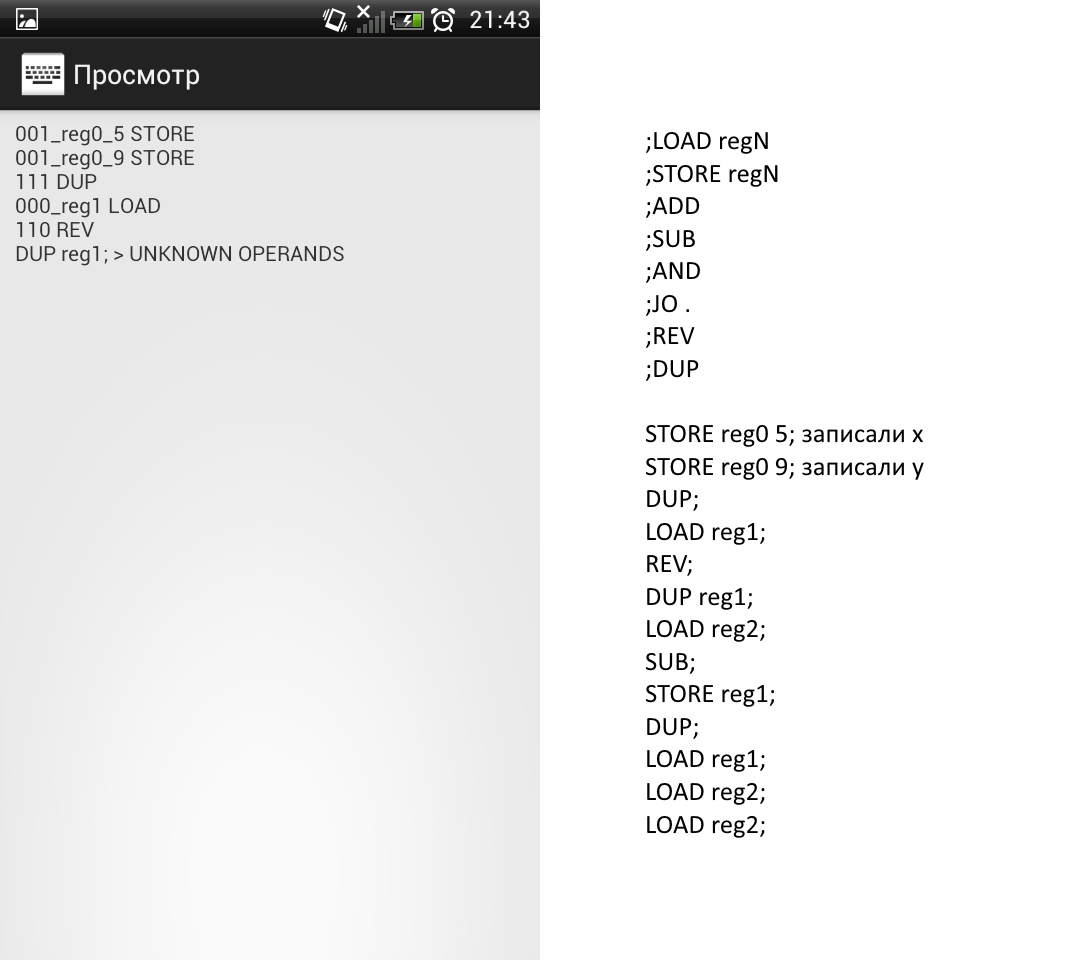
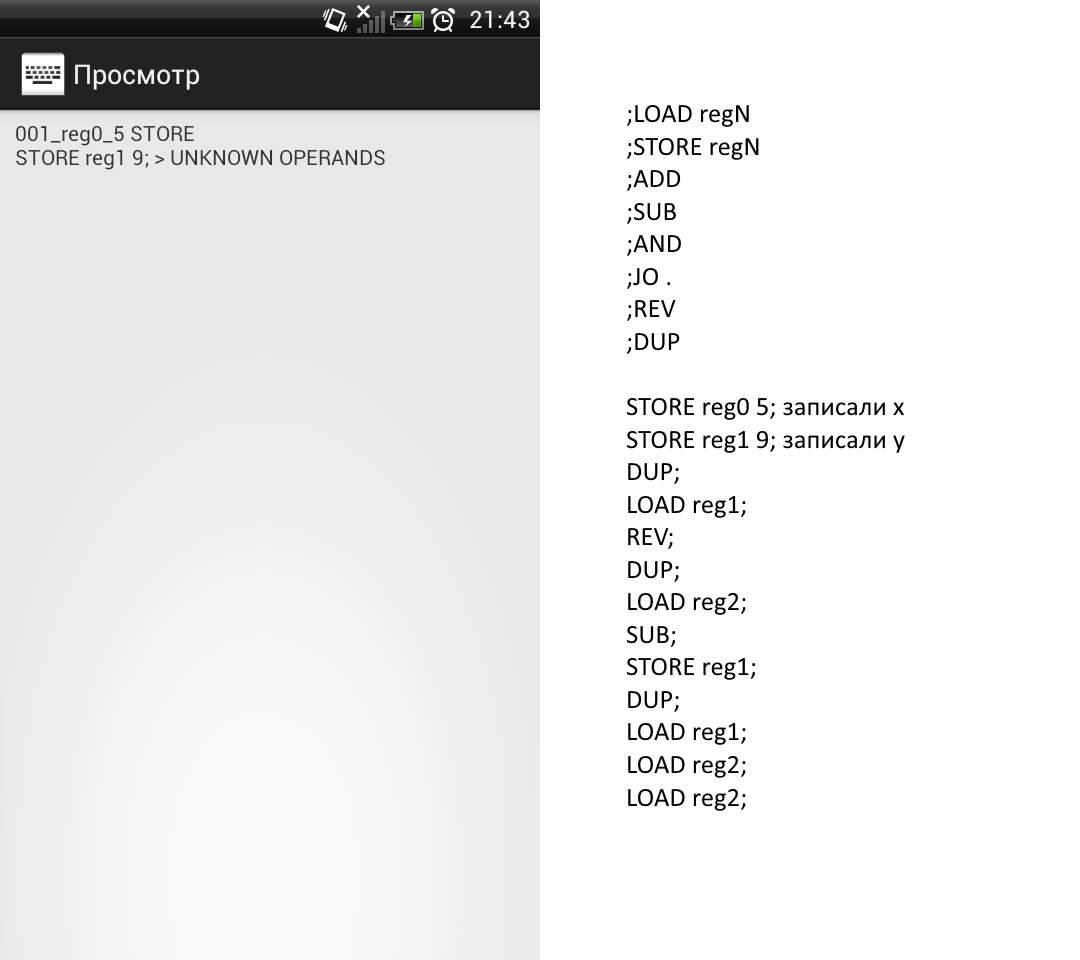
Активити представляют собой .xml файлы с разметкой, для каждого активити должен быть создан свой класс, создающий непосредственно активити по файлу разметки.

В Android приложении есть и другие не мало важные файлы, например, AndroidManifest.xml (в этом файле описываются все разрешения приложения, порядковый номер версии, минимальный и максимальный уровень API, перечисляются активити и др.). Содержимое этих файлов будет предоставлено в приложении Б.

## Методика отладки и тестирования

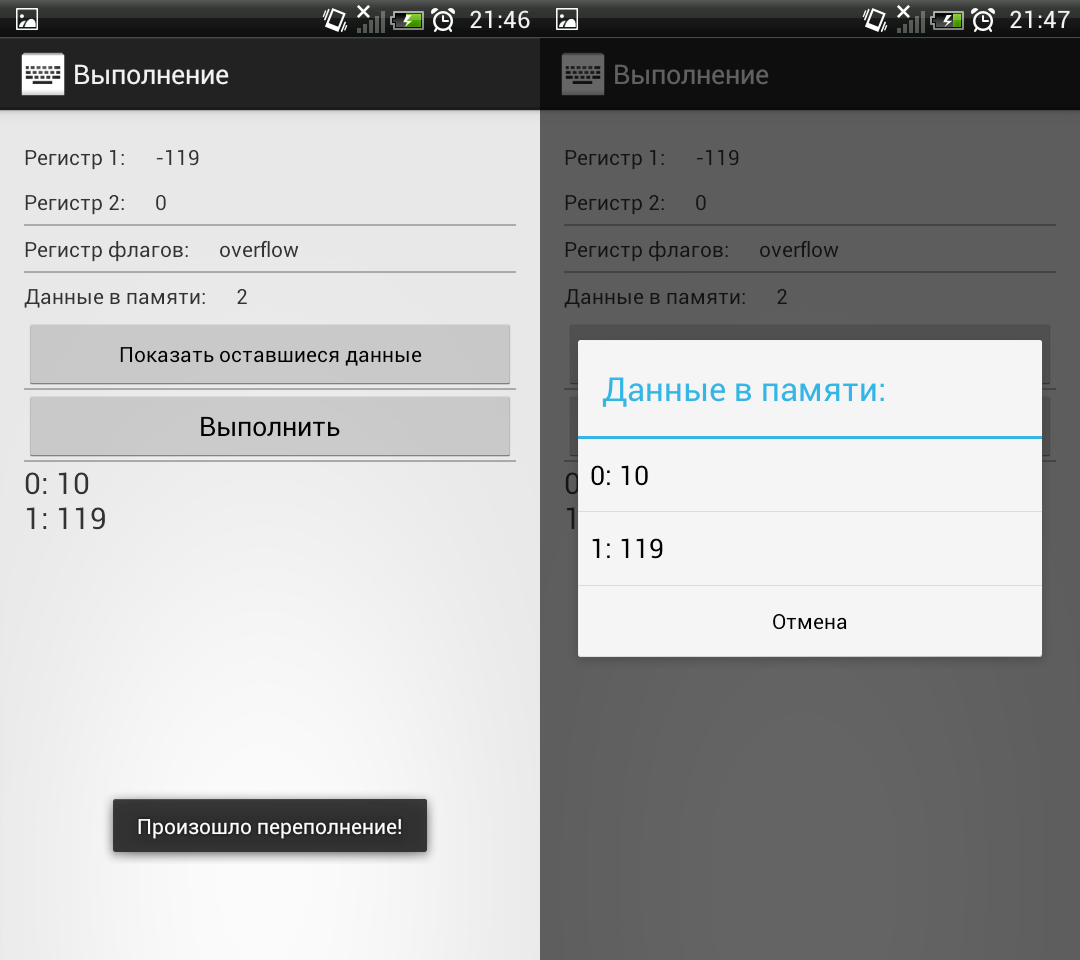
Продемонстрируем работу программы на примерах.

Примеры ошибок при записи исходного файла:



На скриншотах отчётливо видно возможные ошибки. В левой части выводится листинг с ошибкой, в правой же исходный файл.

Пример работы без ошибок:



# ЗАКЛЮЧЕНИЕ

В данной курсовой работе был разработан рабочий эмулятор ассемблерных команд для мобильной ОС Android. Разработка велась в среде Eclipse. С помощью данной программы можно наглядно продемонстрировать работу процессора. В данной работе использовались безадресные команды (кроме чтения и записи), а память была реализована в виде программного стека.

# СПИОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Ассемблер. [Электронный ресурс] <http://ru.wikipedia.org/wiki/Ассемблер>
2. Бек Л. Системное программное обеспечение. М.: Мир, 1990. – 425 с.
3. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. - 26 с.
4. ГОСТ 2.105-95 ЕСКД. Общие требования к текстовым документам. - 36 с.
5. Каган Б.М. Электронные вычислительные машины и системы: Уч. пос. для вузов. – М.: Энергоатомиздат, 1991. – 592 с.
6. Ларионов А. М. и др. Вычислительные комплексы, системы и сети: Учебник для вузов. Л.: Энергоатомиздат. Ленингр. отд-ние, 1987. – 288с.
7. Микропроцессорные системы. Под ред. Пузанкова Д.В. Уч. пос.- СПб.: Политехника, 2002.-547 с.
8. Ноутон П. Java 2: [пер. с англ.]/П.Ноутон, Г.Шилдт. – СПб.: БХВ-Петербург, 2007. – 1072 с.
9. Основы построения цифровых электронных вычислительных машин комплексов вооружения ПВО. Под ред. А.Х.Ганитулина. – ПВУРЭ ПВО 1986.
10. Полин Е. Л. Архитектура ЭВМ – конспект лекций, 2013
11. Семененко В.А и др. Электронные вычислительные машины. – М.: Высшая школа. 1991. – 288 с.
12. Шевкопляс Б.В. Микропроцессорные структуры. Инженерные решения: Справочник. - М.: Радио и связь, 1986. - 264с.

# ПРИЛОЖЕНИЕ 1

Класс MainActivity:

**package** com.worm2fed.kursach\_prog;

**import** com.worm2fed.kursach\_prog.OpenFileDialog.OpenDialogListener;

**import** android.app.Activity;

**import** android.app.AlertDialog;

**import** android.content.DialogInterface;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.os.Environment;

**import** android.view.View;

**import** android.widget.TextView;

**import** android.widget.Toast;

**public** **class** MainActivity **extends** Activity {

**private** String file\_path = "", path = "";

Intent asembler\_result\_activity, run\_activity;

String[] nm;

**int** mem\_size, mode\_id = 0;

// хранит результат ассемблирования

**private** **boolean** asm\_res = **false**;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

// создаём активити

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***activity\_main***);

// размер памяти

mem\_size = 256;

}

// обработчик нажатия на кнопку 'выбрать файл'

**public** **void** onOpenFileClick(View view) {

String message = "";

// проверка на доступность SD-карты

**if** (!Environment.*getExternalStorageState*().equals(

Environment.***MEDIA\_MOUNTED***)) {

message = getApplicationContext().getResources().getString(

R.string.***sdcard\_error***);

Toast.*makeText*(getApplicationContext(), message, Toast.***LENGTH\_SHORT***)

.show();

} **else** {

// создаём список с файлами для выбора

OpenFileDialog fileDialog = **new** OpenFileDialog(**this**);

fileDialog.show();

// создаём слушатель для получения путя к файлу при его выборе из

// спсика

fileDialog.setOpenDialogListener(**new** OpenDialogListener() {

@Override

**public** **void** OnSelectedFile(String fileName) {

// **TODO** Auto-generated method stub

// Связываем объект GUI с переменной

TextView filePath = (TextView) findViewById(R.id.***file\_path***);

// выводим путь файла рядом с кнопкой

filePath.setText(fileName);

// переменная, хранящаяя путь к файлу

file\_path = fileName;

path = "";

nm = file\_path.split("/");

**for** (**int** i = 0; i < (nm.length - 1); i++)

path = path + nm[i] + "/";

}

});

}

}

// обработчик нажатия на кнопку 'преобразовать в объектный код'

**public** **void** onAssemblClick(View view) {

String message = "";

**if** (file\_path.equals(""))

message = getApplicationContext().getResources().getString(

R.string.***notchosen\_error***);

**else** {

Assembler assembler = **new** Assembler(file\_path);

String mes = assembler.toAssembl();

// вытасикаем сообщение о результате ассемблирования из ресурсов

**switch** (mes) {

**case** "ext\_err":

**case** "read\_err":

**case** "close\_err":

message = getApplicationContext().getResources().getString(

R.string.***extention\_error***);

**break**;

**case** "lis\_err":

message = getApplicationContext().getResources().getString(

R.string.***listing\_error***);

**break**;

**case** "obj\_err":

message = getApplicationContext().getResources().getString(

R.string.***object\_error***);

**break**;

**case** "404\_err":

message = getApplicationContext().getResources().getString(

R.string.***notfound\_error***);

**break**;

**case** "suc":

message = getApplicationContext().getResources().getString(

R.string.***success\_asm***);

asm\_res = **true**;

**break**;

}

}

Toast.*makeText*(getApplicationContext(), message, Toast.***LENGTH\_SHORT***)

.show();

}

// обработчик нажатия на кнопку 'показать объектный код'

**public** **void** onShowObjectCodeClick(View view) {

String message = "";

// записываем путь к файлу объектного кода

message = path + "object\_code.w2f";

// инициализируем будущий переход между активити

asembler\_result\_activity = **new** Intent(MainActivity.**this**,

AssemblerResultActivity.**class**);

// передаём значение переменной в другое активити

asembler\_result\_activity.putExtra("result\_type", message);

// запускаем активити

startActivity(asembler\_result\_activity);

}

// обработчик нажатия на кнопку 'показать листинг'

**public** **void** onShowListingClick(View view) {

String message = "";

// записываем путь к файлу листинга

message = path + "listing.w2f";

// инициализируем будущий переход между активити

asembler\_result\_activity = **new** Intent(MainActivity.**this**,

AssemblerResultActivity.**class**);

// передаём значение переменной в другое активити

asembler\_result\_activity.putExtra("result\_type", message);

// запускаем активити

startActivity(asembler\_result\_activity);

}

// обработчик нажатия на кнопку 'выполнить'

**public** **void** onRunClick(View view) {

String message = "";

**if** (asm\_res == **false**) {

message = getApplicationContext().getResources().getString(

R.string.***notasm\_error***);

Toast.*makeText*(getApplicationContext(), message, Toast.***LENGTH\_SHORT***)

.show();

} **else** {

// открываем диалог для выбора режима выполнения

openRunDialog();

}

}

// окошко для выбора режима выполнения

**protected** **void** openRunDialog() {

**final** String[] m\_сhoose = getResources().getStringArray(

R.array.***mode\_arr***);

AlertDialog.Builder run\_dialog = **new** AlertDialog.Builder(**this**);

run\_dialog.setTitle(R.string.***mode***).setCancelable(**false**);

// добавляем переключатели

run\_dialog.setSingleChoiceItems(m\_сhoose, 0,

**new** DialogInterface.OnClickListener() {

@Override

**public** **void** onClick(DialogInterface dialog, **int** item) {

mode\_id = item;

}

});

run\_dialog.setPositiveButton(android.R.string.***yes***,

**new** DialogInterface.OnClickListener() {

@Override

**public** **void** onClick(DialogInterface dialog, **int** which) {

// **TODO** Auto-generated method stub

// инициализируем будущий переход между активити

run\_activity = **new** Intent(MainActivity.**this**,

RunActivity.**class**);

// передаём значение переменных в другое активити

run\_activity.putExtra("mem\_size", mem\_size);

run\_activity.putExtra("mode\_id", mode\_id);

run\_activity.putExtra("path", path);

// запускаем активити

startActivity(run\_activity);

}

});

// добавляем одну кнопку для закрытия диалога

run\_dialog.setNegativeButton(android.R.string.***cancel***,

**new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

dialog.cancel();

}

});

run\_dialog.show();

}

}

Класс Assembler:

**package** com.worm2fed.kursach\_prog;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** Assembler {

**private** String path, name, mes, listing\_name, object\_code\_name;

**public** Assembler(String file\_path) {

mes = "";

path = "";

// переменная хранит разбитую строку

String[] nm = file\_path.split("/");

// записываем имя файла

name = nm[nm.length - 1];

// формируем путь к файлу (без его имени)

**for** (**int** i = 0; i < (nm.length - 1); i++)

path = path + nm[i] + "/";

// имя для файла листинга

listing\_name = "listing.w2f";

// имя для файла с объектным кодом

object\_code\_name = "object\_code.w2f";

}

// функция проверки расширения

**private** **boolean** checkName(String name) {

// перменная хранит расширение файла

String[] check = name.split("\\.");

// если файл имеет не подходяещее разрешение - то "давай досвидания" ;D

**if** (check[check.length - 1].equals("txt")

|| check[check.length - 1].equals("w2f"))

**return** **true**;

**else**

**return** **false**;

}

// функция для преобразования операции из мнемокода в двоичный код

**private** String mnemoCodeToBinaryCode(String mnemo) {

String binary\_comand;

// поиск соответствия

**switch** (mnemo) {

**case** "LOAD":

binary\_comand = "000";

**break**;

**case** "STORE":

binary\_comand = "001";

**break**;

**case** "ADD":

binary\_comand = "010";

**break**;

**case** "SUB":

binary\_comand = "011";

**break**;

**case** "AND":

binary\_comand = "100";

**break**;

**case** "JO":

binary\_comand = "101";

**break**;

**case** "REV":

binary\_comand = "110";

**break**;

**case** "DUP":

binary\_comand = "111";

**break**;

**default**:

binary\_comand = "err";

}

**return** binary\_comand;

}

// функция проверки корректнсти операндов

**private** **boolean** checkAreOperandsCorect(String operand[], **int** num,

String mnemo) {

**byte** data = 0;

// если один операнд

**if** (num == 1) {

**if** (operand[0].equals("reg1") == **false**

&& operand[0].equals("reg2") == **false**)

**return** **false**;

**else** **if** (!mnemo.equals("STORE") && !mnemo.equals("LOAD"))

**return** **false**;

**else**

**return** **true**;

}

// если два операнда

**else** **if** (num == 2) {

**if** (!operand[0].equals("reg0"))

**return** **false**;

**else** **if** (!mnemo.equals("STORE"))

**return** **false**;

**else** {

// преобразуем второй операнд в число

**try** {

data = Byte.*parseByte*(operand[1], 16);

} **catch** (Exception e) {

**return** **false**;

}

// проверим диапазон

**if** (data > 127 || data < -127)

**return** **false**;

**else**

**return** **true**;

}

}

// если не 1 и не 2

**else**

**return** **false**;

}

// функция подготовки данных для файла листинга

**private** String prepareDataForListingFile(String data) {

String mnemo, listing\_line;

String[] operands = **new** String[2];

**int** operands\_num = 0;

// разбиваем строку по пробелам

String[] data\_split = data.split(" ");

// выделяем первое слово, которое должно содержать команду

mnemo = data\_split[0];

// переменная хранит двоичный код команды

String binary\_comand = mnemoCodeToBinaryCode(mnemo);

// проверка команды на корректность

**if** (binary\_comand.equals("err"))

listing\_line = data + ";" + " > UNKNOWN COMAND";

**else** {

// проверяем наличие операндов, и если они есть - выделяем их

**if** (data\_split.length == 1)

listing\_line = binary\_comand + " " + mnemo;

**else** {

**for** (**int** i = 1; i < data\_split.length; i++) {

operands\_num = i;

**if** (i >= 3)

listing\_line = data + ";" + " > INCORECT OPERANDS NUM";

**else**

operands[i - 1] = data\_split[i];

}

**if** (checkAreOperandsCorect(operands, operands\_num, mnemo) == **false**)

listing\_line = data + ";" + " > UNKNOWN OPERANDS";

**else** {

**if** (operands\_num == 1)

listing\_line = binary\_comand + "\_" + operands[0] + " "

+ mnemo;

**else**

listing\_line = binary\_comand + "\_" + operands[0] + "\_"

+ operands[1] + " " + mnemo;

}

}

}

**return** listing\_line;

}

// функция создания файла листинга

**private** **boolean** makeListingFile(String data[]) {

String listing\_line = **null**;

BufferedWriter bf\_writer = **null**;

**try** {

// инициализируем запись файла

FileWriter file\_writer = **new** FileWriter(path + listing\_name);

bf\_writer = **new** BufferedWriter(file\_writer);

**for** (**int** i = 0; data[i] != **null**; i++) {

// подготавливаем строку для записи

listing\_line = prepareDataForListingFile(data[i]);

bf\_writer.write(listing\_line);

bf\_writer.newLine();

**if** (listing\_line.contains(">"))

**break**;

}

} **catch** (IOException e) {

**return** **false**;

}

closeFile(bf\_writer);

// если строка содержит характерный символ ошбики - возращаем ошибку

**if** (listing\_line.contains(">"))

**return** **false**;

**else**

**return** **true**;

}

// функция подготовки данных для файла с объектным кодом

**private** String prepareDataForObjectCodeFile(String data) {

String mnemo, object\_code\_line;

String operands[] = **new** String[2];

String op = **null**;

**int** operands\_num = 0;

// разбиваем строку по пробелам

String data\_split[] = data.split(" ");

// выделяем первое слово, которое должно содержать команду

mnemo = data\_split[0];

// переменная хранит двоичный код команды

String binary\_comand = mnemoCodeToBinaryCode(mnemo);

// проверяем наличие операндов, и если они есть - выделяем их

**if** (data\_split.length == 1)

object\_code\_line = "00000" + binary\_comand;

**else** {

**for** (**int** i = 1; i < data\_split.length; i++) {

operands\_num = i;

operands[i - 1] = data\_split[i];

}

**if** (operands\_num == 1) {

**if** (operands[0].equals("reg1"))

op = "01";

**else** **if** (operands[0].equals("reg2"))

op = "10";

**else**

op = "11";

object\_code\_line = "00" + op + "0" + binary\_comand;

} **else** {

// получаем наши данные в двоичном виде

op = Integer.*toBinaryString*(Integer.*parseInt*(operands[1], 16));

// дополняем нулями недостающие разряды

**while** (op.length() != 8)

op = "0" + op;

object\_code\_line = "00" + op + "001" + binary\_comand;

}

}

**return** object\_code\_line;

}

// функция преобразования исходного файла в объектный код

**private** **boolean** makeObjectCodeFile(String[] data) {

String object\_code\_line = **null**;

BufferedWriter bf\_writer = **null**;

**try** {

FileWriter file\_writer = **new** FileWriter(path + object\_code\_name);

bf\_writer = **new** BufferedWriter(file\_writer);

**for** (**int** i = 0; data[i] != **null**; i++) {

object\_code\_line = prepareDataForObjectCodeFile(data[i]);

bf\_writer.write(object\_code\_line);

bf\_writer.newLine();

}

} **catch** (IOException e) {

**return** **false**;

}

closeFile(bf\_writer);

**return** **true**;

}

// функция для закрытия потока записи

**private** **boolean** closeFile(BufferedWriter bf\_writer) {

**try** {

**if** (bf\_writer != **null**)

bf\_writer.close();

} **catch** (IOException e) {

**return** **false**;

}

**return** **true**;

}

// функция для закрытия потока чтения

**private** **boolean** closeFile(BufferedReader bf\_reader) {

**try** {

**if** (bf\_reader != **null**)

bf\_reader.close();

} **catch** (IOException e) {

**return** **false**;

}

**return** **true**;

}

// основная функция класса, непосредственно выполняющая ассемблирование

**public** String toAssembl() {

String line = **null**;

**int** i = 0;

String[] data\_for\_file = **new** String[255];

BufferedReader bf\_reader = **null**;

// проверка расширения файла

**if** (checkName(name) == **false**)

mes = "ext\_err";

**else** {

**try** {

// открываем поток для чтения

FileReader file\_reader = **new** FileReader(path + name);

bf\_reader = **new** BufferedReader(file\_reader);

// читаем содержимое

**while** ((line = bf\_reader.readLine()) != **null**) {

// убираем комментарии

**if** (!line.contains("") && !line.contains(";")) {

mes = "lis\_err";

**return** mes;

} **else** {

String line\_done[] = line.split(";");

// если строка пуста - пропускаем её

**if** (line\_done[0].equals(""))

**continue**;

**else**

// тогда зановсим в массив наши данные попутно

// удаляя

// лишние пробелы

data\_for\_file[i] = line\_done[0].replaceAll(

"[\\s]{2,}", " ");

i++;

}

}

**if** (makeListingFile(data\_for\_file) == **false**)

mes = "lis\_err";

**else** {

**if** (makeObjectCodeFile(data\_for\_file) == **false**)

mes = "obj\_err";

**else**

mes = "suc";

}

} **catch** (FileNotFoundException e) {

mes = "404\_err";

} **catch** (IOException e) {

mes = "read\_err";

} **finally** {

**if** (closeFile(bf\_reader) == **false**)

mes = "close\_err";

}

}

**return** mes;

}

}

Класс AssemblerResultActivity:

**package** com.worm2fed.kursach\_prog;

**import** java.io.BufferedReader;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.widget.TextView;

**import** android.widget.Toast;

**public** **class** AssemblerResultActivity **extends** Activity {

**protected** **void** onCreate(Bundle savedInstanceState) {

// создаём активити

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***activity\_assembler\_result***);

String file = getIntent().getExtras().getString("result\_type");

String line = **null**, mes = **null**, text = "";

BufferedReader bf\_reader = **null**;

**try** {

// открываем поток для чтения

FileReader file\_reader = **new** FileReader(file);

bf\_reader = **new** BufferedReader(file\_reader);

// читаем содержимое

**while** ((line = bf\_reader.readLine()) != **null**) {

text = text + line + "\n";

}

} **catch** (FileNotFoundException e) {

mes = "File " + file + " not found!";

Toast.*makeText*(getApplicationContext(), mes, Toast.***LENGTH\_SHORT***)

.show();

} **catch** (IOException e) {

mes = "Error due reading file " + file + "!";

Toast.*makeText*(getApplicationContext(), mes, Toast.***LENGTH\_SHORT***)

.show();

} **finally** {

**if** (closeFile(bf\_reader) == **false**) {

mes = "Error due closing file " + file + "!";

Toast.*makeText*(getApplicationContext(), mes, Toast.***LENGTH\_SHORT***)

.show();

}

}

TextView result = (TextView) findViewById(R.id.***asm\_result***);

result.setText(text);

}

// функция для закрытия потока чтения

**private** **boolean** closeFile(BufferedReader bf\_reader) {

**try** {

**if** (bf\_reader != **null**)

bf\_reader.close();

} **catch** (IOException e) {

**return** **false**;

}

**return** **true**;

}

}

Класс RunActivity:

**package** com.worm2fed.kursach\_prog;

**import** android.app.Activity;

**import** android.app.AlertDialog;

**import** android.content.DialogInterface;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.ArrayAdapter;

**import** android.widget.ListView;

**import** android.widget.TextView;

**import** android.widget.Toast;

**public** **class** RunActivity **extends** Activity {

**private** String mes = **null**, path;

**private** **int** mem\_size, mode\_id;

**private** ALU alu;

TextView reg1, reg2, reg\_flag, mem\_left, last\_ten\_data;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***activity\_run***);

// получаем данные из основного активити

mem\_size = getIntent().getExtras().getInt("mem\_size");

mode\_id = getIntent().getExtras().getInt("mode\_id");

path = getIntent().getExtras().getString("path");

// инициализируем наши текстовые поля

reg1 = (TextView) findViewById(R.id.***reg1***);

reg2 = (TextView) findViewById(R.id.***reg2***);

reg\_flag = (TextView) findViewById(R.id.***reg\_flag***);

mem\_left = (TextView) findViewById(R.id.***mem\_left***);

last\_ten\_data = (TextView) findViewById(R.id.***last\_ten\_data***);

// создаём АЛУ

alu = **new** ALU(mem\_size, path);

}

**public** String[] getLeftData() {

**byte** left\_data[] = alu.returnMemory().showLeftData(

alu.returnMemorySizeValue());

String list\_arr[] = **new** String[left\_data.length];

**for** (**int** i = 0; i < left\_data.length; i++)

list\_arr[i] = i + ": " + left\_data[i];

**return** list\_arr;

}

// обработчик нажатия на кнопку 'показать оставшиеся данные'

**public** **void** onShowMemLeftClick(View view) {

AlertDialog.Builder mem\_dialog = **new** AlertDialog.Builder(**this**);

mem\_dialog.setTitle(R.string.***mem\_left***).setCancelable(**false**);

// добавляем список с данными

ListView mem\_list = **new** ListView(**this**);

String[] list\_arr = getLeftData();

ArrayAdapter<String> list\_adap = **new** ArrayAdapter<String>(**this**,

android.R.layout.***simple\_list\_item\_1***, list\_arr);

mem\_list.setAdapter(list\_adap);

mem\_dialog.setView(mem\_list);

// добавляем одну кнопку для закрытия диалога

mem\_dialog.setNegativeButton(android.R.string.***cancel***,

**new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

dialog.cancel();

}

});

mem\_dialog.show();

}

**public** **void** onRunAluClick(View view) {

// непосрдетсвенно выполнение

**switch** (mode\_id) {

// автоматический режим

**case** 0:

mes = alu.runAutoMode();

**break**;

// ручной режим

**case** 1:

mes = alu.runManualMode();

**break**;

// ошибка

**default**:

mes = getApplicationContext().getResources().getString(

R.string.***mode\_error***);

**break**;

}

// выводим соответствующие сообщения

**switch** (mes) {

**case** "prepare\_err":

mes = getApplicationContext().getResources().getString(

R.string.***prepare\_error***);

**break**;

**case** "jo":

mes = getApplicationContext().getResources().getString(

R.string.***jo\_err***);

**break**;

**case** "ex\_err":

mes = getApplicationContext().getResources().getString(

R.string.***ex\_error***);

**break**;

**case** "suc":

mes = getApplicationContext().getResources().getString(

R.string.***success\_run***);

**break**;

**default**:

mes = **null**;

**break**;

}

**if** (mes != **null**)

Toast.*makeText*(getApplicationContext(), mes, Toast.***LENGTH\_SHORT***)

.show();

reg1.setText(Byte.*toString*(alu.returnReg1Value()));

reg2.setText(Byte.*toString*(alu.returnReg2Value()));

**switch** (alu.returnRegFlagValue()) {

// overflow

**case** 0x01:

reg\_flag.setText("overflow");

**break**;

// zero

**case** 0x08:

reg\_flag.setText("zero");

**break**;

// negative

**case** 0x04:

reg\_flag.setText("negative");

**break**;

// positive

**case** 0x02:

reg\_flag.setText("positive");

**break**;

**default**:

reg\_flag.setText(Integer.*toHexString*(alu.returnRegFlagValue()));

**break**;

}

mem\_left.setText(Integer.*toString*(alu.returnMemorySizeValue()));

String[] data = getLeftData();

String text = "";

**if** (data.length > 10)

**for** (**int** i = 0; i < 10; i++)

text = text + data[i] + "\n";

**else**

**for** (**int** i = 0; i < data.length; i++)

text = text + data[i] + "\n";

last\_ten\_data.setText(text);

}

}

Класс Alu:

**package** com.worm2fed.kursach\_prog;

**import** java.io.BufferedReader;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.IOException;

**public** **class** ALU {

// путь к объектному файлу

**private** String path;

// создаём эмулируемую память

**private** Stack memory;

// создаём регистры общего назначения и регистр флагов

**private** **byte** reg1, reg2, reg\_flag;

// массив, хранящий строки объектного кода

**private** String[] data\_for\_run;

// переменная для ручного режима

**private** **int** count = 0;

**public** ALU(**int** mem\_size, String file) {

// создаём эмулируемую память, размером mem\_size байт

memory = **new** Stack(mem\_size);

path = file + "object\_code.w2f";

}

// функция для закрытия потока чтения

**private** **boolean** closeFile(BufferedReader bf\_reader) {

**try** {

**if** (bf\_reader != **null**)

bf\_reader.close();

} **catch** (IOException e) {

**return** **false**;

}

**return** **true**;

}

// подгатавливаем данные для выполнения

**private** **boolean** prepareDataFormFile() {

String line = **null**;

**int** i = 0;

data\_for\_run = **new** String[255];

BufferedReader bf\_reader = **null**;

**try** {

// открываем поток для чтения

FileReader file\_reader = **new** FileReader(path);

bf\_reader = **new** BufferedReader(file\_reader);

// читаем содержимое

**while** ((line = bf\_reader.readLine()) != **null**) {

// если строка пуста - пропускаем её

**if** (line.equals(""))

**continue**;

// тогда зановсим в массив наши данные

**else**

data\_for\_run[i] = line;

i++;

}

} **catch** (FileNotFoundException e) {

**return** **false**;

} **catch** (IOException e) {

**return** **false**;

} **finally** {

**if** (closeFile(bf\_reader) != **false**)

**return** **true**;

}

**return** **false**;

}

**private** **short** getComandFromString(String string) {

**short** com;

**try** {

com = Short.*parseShort*(string, 2);

// выделяем три бита, отвечающие за команду, остальные обнуляем

com = (**short**) (com & 0x0007);

} **catch** (NumberFormatException e) {

com = -1;

}

**return** com;

}

// функция для установки флагов

**private** **void** setFlags(**short** data) {

/\*

\* XXXX ZNPO X - doesn't matter Z - zero N - negative P - positive O -

\* overflow

\*/

// overflow

**if** (data > 127 || data < -127)

reg\_flag = 0x01;

// zero

**else** **if** (data == 0)

reg\_flag = 0x08;

// negative

**else** **if** (data < 0)

reg\_flag = 0x04;

// positive

**else**

reg\_flag = 0x02;

}

// функция для команды

**private** **boolean** comandRead(String raw\_comand) {

**byte** reg\_num;

// извлекаем данные из памяти

**byte** data = memory.pop();

// проверяем извлеклись ли данные

**if** (data == -128)

**return** **false**;

**else** {

/\*

\* XXRR XXXX X - doesn't matter RR - register code 0x20 - reg1 0x10

\* - reg2 0x30 - err

\*/

**try** {

reg\_num = Byte.*parseByte*(raw\_comand, 2);

} **catch** (NumberFormatException e) {

**return** **false**;

}

reg\_num = (**byte**) (reg\_num & 0x30);

// проверяем в какой регистр нужно записать данные

**if** (reg\_num == 0x10)

reg1 = data;

**else** **if** (reg\_num == 0x20)

reg2 = data;

**else**

**return** **false**;

**return** **true**;

}

}

// функция для команды

**private** **boolean** comandWrite(String raw\_comand) {

// если длина команды 8 символов

**if** (raw\_comand.length() == 8) {

**byte** com = Byte.*parseByte*(raw\_comand, 2);

// получаем бит t

**byte** t = (**byte**) (com & 0x08);

// если бит t не равен 0 - ошибка

**if** (t != 0)

**return** **false**;

**else** {

// получаем номер регистра

**byte** reg\_num = (**byte**) (com & 0x30);

// записываем данные

**if** (reg\_num == 0x10) {

// проверка на успешность записи в память

**if** (!memory.push(reg1))

**return** **false**;

reg1 = 0;

} **else** **if** (reg\_num == 0x20) {

// проверка на успешность записи в память

**if** (!memory.push(reg2))

**return** **false**;

reg2 = 0;

} **else**

**return** **false**;

}

}

// если длина команды 16 символов

**else** **if** (raw\_comand.length() == 16) {

**short** com = Short.*parseShort*(raw\_comand, 2);

// получаем бит t

**short** t = (**short**) (com & 0x0008);

// если бит t не равен 1 - ошибка

**if** (t != 8)

**return** **false**;

**else** {

// получаем номер регистра

**short** reg\_num = (**short**) (com & 0x0030);

**if** (reg\_num != 0)

**return** **false**;

**else** {

**byte** data = (**byte**) (com >> 6);

// проверка на успешность записи в память

**if** (!memory.push(data))

**return** **false**;

}

}

}

// если не 8 и не 16 - ошибка

**else**

**return** **false**;

**return** **true**;

}

// функция для команды

**private** **boolean** comandAdd() {

// сначала записываем результат в переменную типа short для проверки на

// переполнение

**short** res = (**short**) (reg1 + reg2);

// записываем результат в регистр1

reg1 = (**byte**) res;

reg2 = 0;

// устанавливаем флаги

setFlags(res);

**return** **true**;

}

// функция для команды

**private** **boolean** comandSup() {

// сначала записываем результат в переменную типа short для проверки на

// переполнение

**short** res = (**short**) (reg1 - reg2);

// записываем результат в регистр1

reg1 = (**byte**) res;

reg2 = 0;

// устанавливаем флаги

setFlags(res);

**return** **true**;

}

// функция для команды

**private** **boolean** comandAnd() {

// сначала записываем результат в переменную типа short для проверки на

// переполнение

**short** res = (**short**) (reg1 & reg2);

// записываем результат в регистр1

reg1 = (**byte**) res;

reg2 = 0;

// устанавливаем флаги

setFlags(res);

**return** **true**;

}

// функция для команды

**private** **boolean** comandJump() {

// если переподнение - возвращаем неудачное выполнение

**if** (reg\_flag == 0x01)

**return** **false**;

**else**

**return** **true**;

}

// функция для команды REV

**private** **boolean** comandRev() {

**byte** data1 = memory.pop();

**byte** data2 = memory.pop();

// проверяем извлеклись ли данные

**if** (data1 == -128)

**return** **false**;

**else** **if** (data2 == -128) {

// если не извлеклось второе значение - возвращаем первое наместо

memory.push(data1);

**return** **false**;

}

// записываем дублированное значение

**else** **if** (!memory.push(data1))

**return** **false**;

**else** **if** (!memory.push(data2))

**return** **false**;

**else**

**return** **true**;

}

// функция для команды DUP

**private** **boolean** comandDup() {

// получаем данные из памяти

**byte** data = memory.pop();

// если чтение вернуло ошибку, то выход

**if** (data == -128)

**return** **false**;

// возвращаем обратно в память

**else** **if** (memory.push(data) == **false**)

**return** **false**;

// записываем дублированное значение

**else** **if** (memory.push(data) == **false**)

**return** **false**;

**else**

**return** **true**;

}

**private** **boolean** executeComand(**int** index) {

**byte** comand = (**byte**) getComandFromString(data\_for\_run[index]);

**boolean** status = **false**;

// проверяем успешность выполнения получения команды

**if** (comand == -1)

status = **false**;

**else** {

// в зависимости от полученного кода выполняем соответствующую

// операцию

**switch** (comand) {

// чтение

**case** 0x00:

status = comandRead(data\_for\_run[index]);

**break**;

// запись

**case** 0x01:

status = comandWrite(data\_for\_run[index]);

**break**;

// сложение

**case** 0x02:

status = comandAdd();

**break**;

// вычитание

**case** 0x03:

status = comandSup();

**break**;

// логическое И

**case** 0x04:

status = comandAnd();

**break**;

// переход

**case** 0x05:

status = comandJump();

**break**;

// реверсировать

**case** 0x06:

status = comandRev();

**break**;

// дублировать

**case** 0x07:

status = comandDup();

**break**;

// если соответсвий нет - ошибка

**default**:

status = **false**;

}

}

**return** status;

}

**public** String runAutoMode() {

// **TODO** Auto-generated method stub

String status = "suc";

**if** (prepareDataFormFile() == **false**)

status = "prepare\_err";

**else** {

**for** (**int** i = 0; data\_for\_run[i] != **null**; i++)

**if** (!executeComand(i)) {

**if** (reg\_flag == 0x01)

status = "jo";

**else**

status = "ex\_err";

**break**;

}

}

**return** status;

}

**public** String runManualMode() {

// **TODO** Auto-generated method stub

String status = "";

**if** (prepareDataFormFile() == **false**)

status = "prepare\_err";

**else** **if** (data\_for\_run[count] == **null**)

status = "suc";

**else** {

**if** (!executeComand(count)) {

**if** (reg\_flag == 0x01)

status = "jo";

**else**

status = "ex\_err";

}

count++;

}

**return** status;

}

**public** **byte** returnReg1Value() {

**return** reg1;

}

**public** **byte** returnReg2Value() {

**return** reg2;

}

**public** **byte** returnRegFlagValue() {

**return** reg\_flag;

}

**public** **int** returnMemorySizeValue() {

**return** memory.returnStackSizeValue();

}

**public** Stack returnMemory() {

**return** memory;

}

}

Класс Stack:

**package** com.worm2fed.kursach\_prog;

**public** **class** Stack {

// переменная для будущего стека

**private** **byte**[] stack;

// переменная для размера стека

**private** **int** size = 0;

// стек по умолчанию

**public** Stack() {

// создаём стек размером в 16 ячеек.

**this**(16);

}

**public** Stack(**int** n) {

// создаём стек размером в n ячеек.

stack = **new** **byte**[n];

}

// функция проверки стека для записи (false - полный (записывать некуда),

// true - место есть)

**private** **boolean** checkPush(**byte**[] st) {

**if** (size == st.length)

**return** **false**;

**else**

**return** **true**;

}

// функция для записи данных в стек

**public** **boolean** push(**byte** data) {

**if** (!checkPush(stack))

**return** **false**;

**else** {

**if** (size != 0)

**for** (**int** i = (size - 1); i >= 0; i--)

stack[i + 1] = stack[i];

stack[0] = data;

size++;

**return** **true**;

}

}

// функция проверки стека для извлечения (false - пустой (извлекать нечего),

// true - данные есть)

**private** **boolean** checkPop(**byte**[] st) {

**if** (size == 0)

**return** **false**;

**else**

**return** **true**;

}

// функция для извлечения данных из стека

**public** **byte** pop() {

**if** (!checkPop(stack))

**return** (**byte**) -128;

**else** {

**byte** data = stack[0];

**for** (**int** i = 0; i < (size - 1); i++)

stack[i] = stack[i + 1];

size--;

**return** data;

}

}

**public** **int** returnStackSizeValue() {

**return** size;

}

**public** **byte**[] showLeftData(**int** size) {

// **TODO** Auto-generated method stub

**byte**[] left\_mem = **new** **byte**[size];

**for** (**int** i = 0; i < size; i++)

left\_mem[i] = stack[i];

**return** left\_mem;

}

}

# ПРИЛОЖЕНИЕ 2

Файл строковых ресурсов strings.xml:

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<resources>

<string name=*"app\_name"*>Эмулятор ассемблера</string>

<string name=*"assembler\_result"*>Просмотр</string>

<string name=*"title\_run"*>Выполнение</string>

<string name=*"choose\_file"*>Выбрать файл</string>

<string name=*"assembl"*>Преобразовать в объектный код</string>

<string name=*"run"*>Выполнить</string>

<string name=*"object\_code\_com"*>Файл с объектным кодом и листинг появятся в директории с исходным файлом, с именами, соответственно, object\_code.w2f и listing.w2f</string>

<string name=*"show\_object\_code"*>Показать объектный код</string>

<string name=*"show\_listing"*>Показать листинг</string>

<string name=*"success\_asm"*>Файл успешно ассемблирован!</string>

<string name=*"success\_run"*>Программа успешно выполнена!</string>

<string name=*"mode"*>Режим:</string>

<string name=*"delay"*>Задержка:</string>

<string name=*"reg1"*>Регистр 1:</string>

<string name=*"reg2"*>Регистр 2:</string>

<string name=*"reg\_flag"*>Регистр флагов:</string>

<string name=*"mem\_left"*>Данные в памяти:</string>

<string name=*"show\_mem\_left"*>Показать оставшиеся данные</string>

<string-array name=*"mode\_arr"*>

<item>Автоматический</item>

<item>Ручной</item>

</string-array>

<!-- ERRORS -->

<string name=*"sdcard\_error"*>SD-карта не доступна!</string>

<string name=*"access\_error"*>Ошибка доступа!</string>

<string name=*"notchosen\_error"*>Файл не выбран!</string>

<string name=*"extention\_error"*>Неподдерживаемое расширение!</string>

<string name=*"listing\_error"*>Ошибка при создании листинга!</string>

<string name=*"object\_error"*>Ошибка при создании объектного кода!</string>

<string name=*"notfound\_error"*>Файл не найден!</string>

<string name=*"reading\_error"*>Ошибка при чтении файла!</string>

<string name=*"closing\_error"*>Ошибка при закрытии файла!</string>

<string name=*"notasm\_error"*>Сначала необходимо получить объектный код!</string>

<string name=*"mode\_error"*>Не удаётся распознать выбранный режим!</string>

<string name=*"prepare\_error"*>Ошибка при подготовке команды!</string>

<string name=*"jo\_err"*>Произошло переполнение!</string>

<string name=*"ex\_error"*>Произошла ошибка неизвестная ошибка при выполнении программы!</string>

</resources>

Файл AndroidManifest.xml:

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"com.worm2fed.kursach\_prog"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<uses-sdk

android:minSdkVersion=*"15"*

android:targetSdkVersion=*"21"* />

<uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"* />

<application

android:allowBackup=*"true"*

android:icon=*"@drawable/ic\_launcher"*

android:label=*"@string/app\_name"*

android:theme=*"@style/AppTheme"* >

<activity

android:name=*".MainActivity"*

android:label=*"@string/app\_name"* >

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

<activity

android:name=*".AssemblerResultActivity"*

android:label=*"@string/assembler\_result"* >

</activity>

<activity

android:name=*".RunActivity"*

android:label=*"@string/title\_run"* >

</activity>

</application>

</manifest>

Файл разметки основной активити activity\_main.xml для портретной ориентации:

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:paddingBottom=*"@dimen/activity\_vertical\_margin"*

android:paddingLeft=*"@dimen/activity\_horizontal\_margin"*

android:paddingRight=*"@dimen/activity\_horizontal\_margin"*

android:paddingTop=*"@dimen/activity\_vertical\_margin"*

tools:context=*"com.worm2fed.kursach\_prog.MainActivity"* >

<Button

android:id=*"@+id/open\_file"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_alignParentStart=*"true"*

android:layout\_alignParentTop=*"true"*

android:onClick=*"onOpenFileClick"*

android:text=*"@string/choose\_file"* />

<TextView

android:id=*"@+id/file\_path"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignEnd=*"@+id/open\_file"*

android:layout\_alignLeft=*"@+id/open\_file"*

android:layout\_alignRight=*"@+id/open\_file"*

android:layout\_alignStart=*"@+id/open\_file"*

android:layout\_below=*"@+id/open\_file"* />

<Button

android:id=*"@+id/assembl"*

style=*"?android:attr/buttonStyleSmall"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignEnd=*"@+id/file\_path"*

android:layout\_alignLeft=*"@+id/file\_path"*

android:layout\_alignRight=*"@+id/file\_path"*

android:layout\_alignStart=*"@+id/file\_path"*

android:layout\_below=*"@+id/file\_path"*

android:onClick=*"onAssemblClick"*

android:text=*"@string/assembl"* />

<TextView

android:id=*"@+id/object\_code\_com"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignEnd=*"@+id/assembl"*

android:layout\_alignLeft=*"@+id/assembl"*

android:layout\_alignRight=*"@+id/assembl"*

android:layout\_alignStart=*"@+id/assembl"*

android:layout\_below=*"@+id/assembl"*

android:text=*"@string/object\_code\_com"* />

<Button

android:id=*"@+id/show\_object\_code"*

style=*"?android:attr/buttonStyleSmall"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignEnd=*"@+id/object\_code\_com"*

android:layout\_alignLeft=*"@+id/object\_code\_com"*

android:layout\_alignRight=*"@+id/object\_code\_com"*

android:layout\_alignStart=*"@+id/object\_code\_com"*

android:layout\_below=*"@+id/object\_code\_com"*

android:onClick=*"onShowObjectCodeClick"*

android:text=*"@string/show\_object\_code"* />

<Button

android:id=*"@+id/show\_listing"*

style=*"?android:attr/buttonStyleSmall"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignEnd=*"@+id/show\_object\_code"*

android:layout\_alignLeft=*"@+id/show\_object\_code"*

android:layout\_alignRight=*"@+id/show\_object\_code"*

android:layout\_alignStart=*"@+id/show\_object\_code"*

android:layout\_below=*"@+id/show\_object\_code"*

android:onClick=*"onShowListingClick"*

android:text=*"@string/show\_listing"* />

<Button

android:id=*"@+id/run"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignEnd=*"@+id/show\_listing"*

android:layout\_alignLeft=*"@+id/show\_listing"*

android:layout\_alignRight=*"@+id/show\_listing"*

android:layout\_alignStart=*"@+id/show\_listing"*

android:layout\_below=*"@+id/show\_listing"*

android:onClick=*"onRunClick"*

android:text=*"@string/run"* />

</RelativeLayout>

Файл разметки основной активити activity\_main.xml для ландшафтной ориентации:

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:paddingBottom=*"@dimen/activity\_vertical\_margin"*

android:paddingLeft=*"@dimen/activity\_horizontal\_margin"*

android:paddingRight=*"@dimen/activity\_horizontal\_margin"*

android:paddingTop=*"@dimen/activity\_vertical\_margin"*

tools:context=*"com.worm2fed.kursach\_prog.MainActivity"* >

<Button

android:id=*"@+id/open\_file"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_alignParentTop=*"true"*

android:onClick=*"onOpenFileClick"*

android:text=*"@string/choose\_file"* />

<TextView

android:id=*"@+id/file\_path"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignBaseline=*"@+id/open\_file"*

android:layout\_alignBottom=*"@+id/open\_file"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentStart=*"true"*

android:layout\_toLeftOf=*"@+id/open\_file"*

android:layout\_toStartOf=*"@+id/open\_file"* />

<Button

android:id=*"@+id/assembl"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/file\_path"*

android:layout\_alignStart=*"@+id/file\_path"*

android:layout\_below=*"@+id/open\_file"*

android:onClick=*"onAssemblClick"*

android:text=*"@string/assembl"* />

<Button

android:id=*"@+id/run"*

style=*"?android:attr/buttonStyle"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignBaseline=*"@+id/assembl"*

android:layout\_alignBottom=*"@+id/assembl"*

android:layout\_alignEnd=*"@+id/open\_file"*

android:layout\_alignLeft=*"@+id/open\_file"*

android:layout\_alignRight=*"@+id/open\_file"*

android:layout\_alignStart=*"@+id/open\_file"*

android:onClick=*"onRunClick"*

android:text=*"@string/run"* />

<TextView

android:id=*"@+id/object\_code\_com"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/assembl"*

android:layout\_alignStart=*"@+id/assembl"*

android:layout\_centerVertical=*"true"*

android:text=*"@string/object\_code\_com"* />

<Button

android:id=*"@+id/show\_object\_code"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/object\_code\_com"*

android:layout\_alignStart=*"@+id/object\_code\_com"*

android:layout\_below=*"@+id/object\_code\_com"*

android:onClick=*"onShowObjectCodeClick"*

android:text=*"@string/show\_object\_code"* />

<Button

android:id=*"@+id/show\_listing"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignEnd=*"@+id/object\_code\_com"*

android:layout\_alignRight=*"@+id/object\_code\_com"*

android:layout\_below=*"@+id/object\_code\_com"*

android:onClick=*"onShowListingClick"*

android:text=*"@string/show\_listing"* />

</RelativeLayout>

Файл разметки активити c просмотром результата ассемблирования activity\_assembler\_result.xml:

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"* >

<ScrollView

android:layout\_width=*"match\_parent"*

android:layout\_height=*"fill\_parent"*

android:paddingLeft=*"10dp"*

android:paddingRight=*"10dp"*

android:paddingTop=*"5dp"* >

<TextView

android:id=*"@+id/asm\_result"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:scrollbars=*"vertical"* />

</ScrollView>

</RelativeLayout>

Файл разметки активити с выполнением программы activity\_run.xml:

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:paddingBottom=*"@dimen/activity\_vertical\_margin"*

android:paddingLeft=*"@dimen/activity\_horizontal\_margin"*

android:paddingRight=*"@dimen/activity\_horizontal\_margin"*

android:paddingTop=*"@dimen/activity\_vertical\_margin"*

tools:context=*"com.worm2fed.kursach\_prog.RunActivity"* >

<TextView

android:id=*"@+id/reg1\_text"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentStart=*"true"*

android:layout\_alignParentTop=*"true"*

android:paddingBottom=*"5dp"*

android:paddingTop=*"5dp"*

android:text=*"@string/reg1"* />

<TextView

android:id=*"@+id/reg2\_text"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/reg1\_text"*

android:layout\_alignStart=*"@+id/reg1\_text"*

android:layout\_below=*"@+id/reg1\_text"*

android:paddingBottom=*"5dp"*

android:paddingTop=*"5dp"*

android:text=*"@string/reg2"* />

<TextView

android:id=*"@+id/reg1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/reg2"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_alignParentTop=*"true"*

android:layout\_alignStart=*"@+id/reg2"*

android:paddingBottom=*"5dp"*

android:paddingLeft=*"20dp"*

android:paddingRight=*"20dp"*

android:paddingStart=*"20dp"*

android:paddingTop=*"5dp"* />

<TextView

android:id=*"@+id/reg2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_alignTop=*"@+id/reg2\_text"*

android:layout\_toEndOf=*"@+id/reg2\_text"*

android:layout\_toRightOf=*"@+id/reg2\_text"*

android:paddingBottom=*"5dp"*

android:paddingLeft=*"20dp"*

android:paddingRight=*"20dp"*

android:paddingStart=*"20dp"*

android:paddingTop=*"5dp"* />

<View

android:id=*"@+id/separator1"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"1dp"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_below=*"@+id/reg2\_text"*

android:background=*"@android:color/darker\_gray"*

android:visibility=*"visible"* />

<View

android:id=*"@+id/separator2"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"1dp"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_below=*"@+id/flags\_text"*

android:background=*"@android:color/darker\_gray"*

android:visibility=*"visible"* />

<TextView

android:id=*"@+id/mem\_left\_text"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentStart=*"true"*

android:layout\_below=*"@+id/separator2"*

android:paddingBottom=*"5dp"*

android:paddingTop=*"5dp"*

android:text=*"@string/mem\_left"* />

<TextView

android:id=*"@+id/mem\_left"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignBaseline=*"@+id/mem\_left\_text"*

android:layout\_alignBottom=*"@+id/mem\_left\_text"*

android:layout\_alignEnd=*"@+id/separator2"*

android:layout\_alignRight=*"@+id/separator2"*

android:layout\_toEndOf=*"@+id/mem\_left\_text"*

android:layout\_toRightOf=*"@+id/mem\_left\_text"*

android:paddingBottom=*"5dp"*

android:paddingLeft=*"20dp"*

android:paddingRight=*"20dp"*

android:paddingTop=*"5dp"* />

<Button

android:id=*"@+id/show\_mem\_left"*

style=*"?android:attr/buttonStyleSmall"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_alignParentStart=*"true"*

android:layout\_below=*"@+id/mem\_left\_text"*

android:onClick=*"onShowMemLeftClick"*

android:text=*"@string/show\_mem\_left"* />

<View

android:id=*"@+id/separator3"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"1dp"*

android:layout\_alignBottom=*"@+id/show\_mem\_left"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentRight=*"true"*

android:background=*"@android:color/darker\_gray"*

android:visibility=*"visible"* />

<TextView

android:id=*"@+id/flags\_text"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/separator2"*

android:layout\_alignStart=*"@+id/separator1"*

android:layout\_below=*"@+id/separator1"*

android:paddingBottom=*"5dp"*

android:paddingTop=*"5dp"*

android:text=*"@string/reg\_flag"* />

<TextView

android:id=*"@+id/reg\_flag"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignBaseline=*"@+id/flags\_text"*

android:layout\_alignBottom=*"@+id/flags\_text"*

android:layout\_alignEnd=*"@+id/separator1"*

android:layout\_alignRight=*"@+id/separator1"*

android:layout\_toEndOf=*"@+id/flags\_text"*

android:layout\_toRightOf=*"@+id/flags\_text"*

android:paddingBottom=*"5dp"*

android:paddingLeft=*"20dp"*

android:paddingRight=*"20dp"*

android:paddingTop=*"5dp"* />

<Button

android:id=*"@+id/run\_alu"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_alignParentStart=*"true"*

android:layout\_below=*"@+id/show\_mem\_left"*

android:onClick=*"onRunAluClick"*

android:text=*"@string/run"* />

<View

android:id=*"@+id/separator4"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"1dp"*

android:layout\_alignBottom=*"@+id/run\_alu"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentRight=*"true"*

android:background=*"@android:color/darker\_gray"*

android:visibility=*"visible"* />

<TextView

android:id=*"@+id/last\_ten\_data"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentBottom=*"true"*

android:layout\_alignParentEnd=*"true"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_alignParentStart=*"true"*

android:layout\_below=*"@+id/run\_alu"*

android:textSize=*"20sp"* />

</RelativeLayout>

# ПРИЛОЖЕНИЕ 3

# ПРИЛОЖЕНИЕ 4