_____
(повне найменування вищого навчального закладу)
_____
(повна назва кафедри, циклової комісії)

# КУРСОВОЙ ПРОЕКТ

з _____
(назва дисципліни)
на тему:_____
_____

Студента (ки) _____ курсу _____ групи
напряму підготовки_____
спеціальності_____
_____
(прізвище та ініціали)
Керівник _____
_____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: _____Оцінка: ECTS _____

Члени комісії _____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали

**м. _____ - 20 __рік**

СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Одной из задач системного программирования является создание программных продуктов, предназначенных для отладки и тестирования программ и называемых отладчиками. Отладчики позволяют управлять процессом выполнения отлаживаемой программы, получать информацию об ее текущем состоянии; также, многие отладчики позволяют модифицировать содержимое регистров, памяти, а также код отлаживаемой программы.

Отладчики разделяют на 2 группы: универсальные и специализированные. Универсальные предназначены для отладки и просмотра процесса выполнения программы для программ, созданных на любых языках и в любых средах. Обычно такие отладчики ориентированы на управление процессом выполнения программы на уровне Ассемблера. Специализированные отладчики предназначены для управления процессом выполнения программы в определенной среде разработки или на определенном языке.[1]

Целью данной курсовой работы является приобретение навыков в разработке универсальных отладчиков, выявление основных трудностей и проблем возникающих при этом, а также путей их устранения.

# 1 ПОСТАНОВКА ЗАДАЧИ

Необходимо азработать среду отладчика-эмулятора, в которой предусмотреть следующие возможности:

- загрузка объектного кода в память;
- вывод значений регистров и заданной области памяти;
- вывод дизассемблированного текста программы;
- пошаговое выполнение программы.

Отладчик предназначен для эмуляции заданного подмножества инструкций процессора Intel 8086/8088 и облегчения процесса отладки программ, сгенерированных с помощью транслятора с учебного ассемблера. Ниже приведен список эмулируемых инструкций, которые должен выполнять отладчик:

1 MOV РОН,РОН
2 MOV РОН,НО
3 MOV РОН, ОП(баз)
4 MOV ОП(баз),РОН
5 XCHG РОН,РОН
6 XCHG РОН, ОП(баз)
7 XCHG ОП(баз),РОН
8 IMUL P
9 LOOP ссылка
10 INT 20H

Где РОН – регистр общего назначения, ОП – оперативная память, ссылка - адрес в памяти (символьное имя), баз – базовая адресация.

*Исходными данными* для отладчика являются объектный код и значения сегментных регистров. *Результатом работы программы* является вывод на

экран состояния регистров, регистра флагов, заданной области памяти и реассемблированного текста программы на экран.

На экране 4 окна:

1) Окно состояния регистров, которое показывает имя и значение регистров.
2) Окно вывода сегмента ОП: массив содержащий значения 16 байт и адрес
3) Окно вывода сегмента кода, содержит адрес команды, коды команд и текст программы на языке Assembler
4) Окно вывода регистра флагов, содержит поля: имя флагов и значения.

*Объектный файл должен иметь следующий формат:*

H <запись-заголовок> T <запись-тело>… E <запись-конец>

где H, T, E — признаки соответствующих записей.

*1. <Запись-заголовок>* должна быть самой первой записью в объектном файле и иметь следующий формат:

| <Длина имени> | <Имя сегмента> | <Длина сегмента> |
|---|---|---|

где *<Длина имени>* — целое число (байт), указывающее длину поля <Имя сегмента>;

*<Имя сегмента>* — строковая константа размером <Длина имени> байт, указывающая имя сегмента кода;

*<Длина сегмента>* — целое число (слово), указывающее длину сегмента кода.

Не допускается наличие нескольких H-записей в одном файле.

*2. <Запись-тело>* имеет следующий формат:

| <Смещение кода> | <Длина кода> | <Код> |
|---|---|---|

где *<Смещение кода>* — 16-битное смещение <Кода> относительно начала

сегмента;

<*Длина кода*> — целое число (слово), указывающее длину поля <Код> данной записи;

<*Код*> — массив размером в <Длину кода> байт, содержащий собственно объектный код.

*3.* <*Запись конец*> имеет следующий формат:

| <Точка входа> |
|---|

где <*Точка входа*> - адрес первой выполняемой команды.

Информацию, отображаемую на экране необходимо продублировать в файле диагностики.

К особым ситуациям, которые могут возникнуть при работе отладчика, относятся:

- системные ошибки (недостаток системных ресурсов, отсутствие дискеты в дисководе и т. п.);
- ошибки во входных данных.

Системные ошибки не должны приводить к аварийному останову отладчика (если ошибка не является критической для операционной системы).
Ошибки во входных данных и реакция на них представлены в таблице 1

Таблица 1. Возможные ошибки во входных данных и реакция отладчика на них

| **Ошибка** | **Реакция** |
|---|---|
| Входной файл имеет неверный формат | Отказ от загрузки файла |
| Входной файл имеет слишком большой размер для загрузки по указанному адресу | |
| Неверный код команды | Выдача сообщения об ошибке и отказ от выполнения неверной инструкции |
| Выход за пределы программы при выборке инструкций или данных из оперативной памяти | |

# 2 ОПРЕДЕЛЕНИЕ ФОРМАТОВ КОМАНД

Форматы кодирования команд соответствуют форматам команд микропроцессора Intel 8086/8088. Длина команд данной курсовой работы составляет от 2 до 4 байт. Код команды занимает первый байт, способ адресации задается во втором байте, в остальных байтах указываются данные, либо адрес памяти, где они находятся.

Структура адресного байта имеет вид, представленный на рисунке 2.1

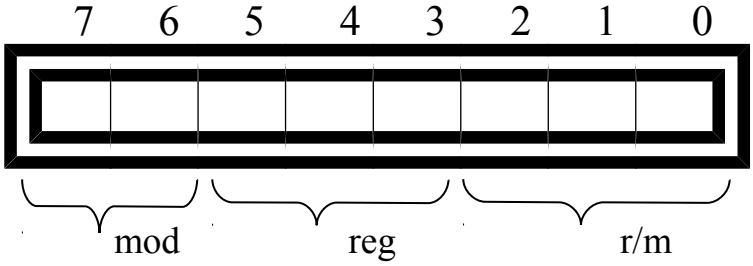


Рисунок 2.1 – Структура адресного байта

Таблица соответствий кодов и мнемонических имен регистров, представлена в таблице 2.1

Таблица 2.1 – Таблица соответствий кодов и мнемонических имен регистров

| reg или r/m | | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|---|
| Регистр | w=0 | AL | CL | DL | BL | AH | CH | DH | BH |
| | w=1 | AX | CX | DX | BX | SP | BP | SI | DI |

Значения битов mod задают различные способы адресации и имеют следующую интерпретацию:

- mod=00 – биты r/m задают абсолютный адрес, байт смещения отсутствует
- mod=01 – биты r/m задают абсолютный адрес памяти и имеется один байт смещения
- mod=10 – биты r/m задают абсолютный адрес памяти и имеется два байта смещения
- mod=11 – биты r/m и биты reg вместе с битом w определяют конкретные регистры

Кодирование типов адресаций приведено в таблице 2.2.

Таблица 2.2 – Таблица кодов адресации

| r/m | mod=00 | mod=01 или mod=10 |
|---|---|---|
| 000 | [BX]+[SI] | [BX]+[SI] +смещение |
| 001 | [BX]+[DI] | [BX]+[DI] +смещение |
| 010 | [BP]+[SI] | [BP]+[SI] +смещение |
| 011 | [BP]+[DI] | [BP]+[DI] +смещение |
| 100 | [SI] | [SI] +смещение |
| 101 | [DI] | [DI] +смещение |
| 110 | Прямая | [BP]+смещение |
| 111 | [BX] | [BX] +смещение |

Т.к. в разрабатываемой программе должна быть реализована только базово-индексная адресация, то из таблицы кодов адресации для данной курсовой работы характерны лишь 4 первые строчки.

Таблица форматов команд приведена в таблице 2.3.

В таблице используются следующие сокращения:

- w – бит определяющий длину операндов команды. При w=0 – размер операндов – байт, иначе – слово
- Data– байты с данными
- Disp– смещение в ОП
- d – бит, указывающий адресацию операндов
- КОП – код операции
- Рег8,Рег16 – однобайтный и двухбайтнй регистры соответственно
- ОП – оперативная память
- НО – непосредственный операнд

Таблица 2.3 - Таблица форматов команд [3]

| Команда: | Оп1: (x) | Оп2: (y) | 1й байт КОП: | | | 2й байт адресный | | | 3й байт | 4й байт |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | КОП | d | w | mod | reg | r/m | | |
| MOV | Per8 | Per8 | 100010 | 1 | 0 | 11 | xxx | yyy | | |
| MOV | Per16 | Per16 | 100010 | 1 | 1 | 11 | xxx | yyy | | |
| MOV | Per8 | Per8 | 100010 | 0 | 0 | 11 | yyy | xxx | | |
| MOV | Per16 | Per16 | 100010 | 0 | 1 | 11 | yyy | xxx | | |
| MOV | Per8 | ОП | 100010 | 1 | 0 | 10 | xxx | 11z | Disp16 | |
| MOV | Per16 | ОП | 100010 | 1 | 1 | 10 | xxx | 11z | Disp16 | |
| MOV | ОП | Per8 | 100010 | 0 | 0 | 10 | yyy | 11z | Disp16 | |
| MOV | ОП | Per16 | 100010 | 0 | 1 | 10 | yyy | 11z | Disp16 | |
| MOV | Per8 | НО | 110001 | 1 | 0 | 11 | 000 | xxx | Data8 | |
| MOV | Per16 | НО | 110001 | 1 | 1 | 11 | 000 | xxx | LL Data16 | HH |
| MOV | Per | НО | 10110xxx | | | Data8 | | | | |
| MOV | Per | НО | 10111xxx | | | LL Data16 | | | HH | |
| XCHG | Per8 | Per8 | 10010xxx | | | | | | | |
| XCHG | Per8 | Per8 | 100001 | 1 | 0 | 11 | xxx | yyy | | |
| XCHG | Per16 | Per16 | 100001 | 1 | 1 | 11 | xxx | yyy | | |
| XCHG | Per8 | ОП | 100001 | 1 | 0 | 10 | xxx | 11z | Disp16 | |
| XCHG | ОП | Per8 | 100001 | 1 | 0 | 10 | yyy | 11z | Disp16 | |
| XCHG | Per16 | ОП | 100001 | 1 | 1 | 10 | xxx | 11z | Disp16 | |
| XCHG | ОП | Per16 | 100001 | 1 | 1 | 10 | yyy | 11z | Disp16 | |
| IMUL | Per8 | - | 111101 | 1 | 0 | 11 | 101 | xxx | | |
| IMUL | Per16 | - | 111101 | 1 | 1 | 11 | 101 | xxx | | |
| LOOP | ссылка | - | 11100010 | | | Disp8 | | | | |
| INT | 20H | - | 11001101 | | | 20H | | | | |

# 3 РАЗРАБОТКА МЕТОДА РЕШЕНИЯ

В данной курсовой работе должен быть разработан отладчик эмулирующего типа, который имитирует работу процессора. В рассматриваемом случае исходным является микропроцессор Intel 8086/8088.

Программная модель процессора – это функциональная модель, используемая программистом при разработке программ в кодах ЭВМ или на языке Ассемблера. В такой модели игнорируются многие аппаратные особенности в работе процессора. Микропроцессор Intel 8086/8088 содержит регистры, каждый из которых имеет уникальную природу и предоставляет определенные возможности, не поддерживаемые другими регистрами или ячейками памяти. Регистры разбиваются на пять категории: регистры общего назначения (РОН), индексные регистры и регистры-указатели, регистр флагов, указатель команд и сегментные регистры. Все регистры 16-разрядные.

Разрабатываемый отладчик является связующим звеном между данными, находящимися в оперативной памяти и регистрами микропроцессора. Также отладчик должен управлять процессом отладки: загружать объектный код в ОП, задавать значения регистров, выполнять программу в автоматическом или пошаговом режимах.

На рисунке 3.1. представлена Функциональная схема отладчика

Функциональная схема отладчика включает в себя четыре основных этапа – загрузка объектного кода в оперативную память, реассемблирование объектного кода, эмуляция команд и вывод информации на экран.

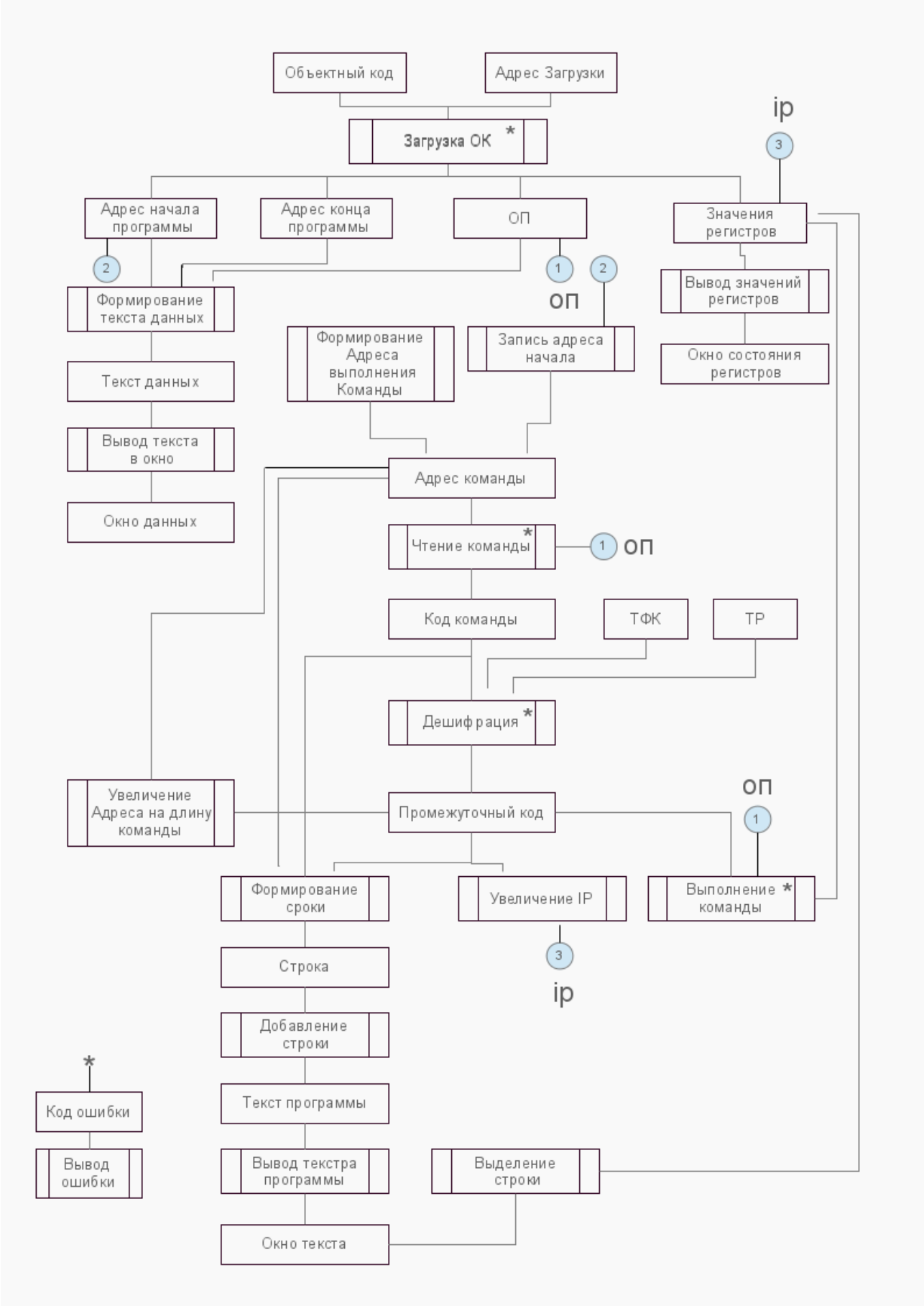


Рисунок 3.1 - Функциональная схема отладчика

# 4 ВЫБОР И ОБОСНОВАНИЕ СТРУКТУР ДАННЫХ

Функциональная схема работы отладчика, представленная на рис. 3.1, позволяет выделить наиболее значимые структуры данных.

Таблица форматов команд.

Таблица форматов команд необходима для дешифрации команд из объектного кода, что определяет присутствующие в ней поля.

Таблица форматов команд имеет следующие поля:

- строковое поле МНК – в нем находятся мнемокоды команд;
- символьное поле Тип первого операнда — указывает тип операнда
- символьное поле Тип второго операнда — указывает тип операнда
- байтовое поле длина – указывает длину команды в байтах;
- байтовое поле код – содержит код команды; бит d и w
- байтовое поле mod – содержит 2 бита xx, где xx-биты mod команды; это поле будет применяться вместе с байтом кода для поиска в таблице;
- булевское поле d — указывает на адресацию операндов
- числовое поле Номер формата — указывает номер группы формата команады

Таблица регистров

Содержит информацию о кодировании, имени регистров, их размере.

| Имя рег. | AL | CL | DL | BL | AH | CH | DH | BH |
|---|---|---|---|---|---|---|---|---|
| Код | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| размер | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Имя рег. | AX | CX | DX | BX | SP | BP | SI | DI |
| Код | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| размер | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

Массив, имитирующий оперативную память.

Каждый элемент имеет размер 1 байт, всего элементов 65536, что соответствует 64кБ оперативной памяти.

Массив значений регистров

Каждый элемент имеет размер 2 байт, всего элементов 12 что соответствует 4-м регистрам общего назначения, 4-м индексным регистрам и 4-м сегментным регистрам процессора Intel 8086/8088.

Регистр флагов

Регистр флагов удобно представлять в виде массива из 16 булевских элементов .

Промежуточный код команды

Используется как вспомогательный элемент. Содержит информацию о команде.

Поля 7,8 содержат информацию о размере операнда (1-2 байта) и способ задания операнда. Возможные значения:

R – двухбайтный регистр

r – однобайтный регистр

M – операнд задаётся ссылкой в ОП, длина операнда 2 байта

m – операнд ссылкой в ОП, длина операнда 1 байт

N – непосредственный операнд

Поля 5,6 содержат имя регистра (если операнд регистр), число в шестнадцатеричном формате (если операнд непосредственный) или BX+смещение (если операнд – ссылка в ОП) в зависимости от способа задания операнда

Поля 9,10 содержат код регистра (если операнд регистр), значение (если операнд непосредственный), ссылка(если операнд из ОП)

Список полей промежуточного кода команды

1. числовое поле код команды
2. числовое поле внутренний код команды
3. байтовое поле длина команды
4. строковое поле мнемоника операции
5. строковое поле мнемокод операнда1
6. строковое поле мнемокод операнда2
7. символьное поле тип операнда1
8. символьное поле тип операнда2
9. числовое поле значение операнда1
10. числовое поле значение операнда2
11. числовое поле номер группы формата команды
12. булевское поле адресации операндов

## 5 АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ

Поскольку отладчик разрабатывается как приложение, разумно разделить программу на блоки загрузи, эмуляции и вывода данных. Но при этом пользователь должен знать, что эмуляцию программы можно произвести только после загрузки объектного кода в модель ОП. Вывод данных в окна осуществляется после: загрузки кода в ОП; выполнения каждой команды в пошаговом режиме;

### Общий алгоритм

В алгоритме используется переменная состояния системы (ПСС) может принимать значения: -1 – объектный код не загружен , 0 – успешно загружен объектный код в ОП, 1 – обработка данных завершена

Алгоритм:

1 Вывод интерфейса на экран. Установка переменной состояния системы (ПСС) в -1.

2 Ожидание выбора действия.

3 Если выбрано действие «Загрузить»

3.1 Запустить загрузчик с определёнными параметрами (адрес начала загрузки программы в ОП, начальное состояние регистров).

3.2 Если код выполнения не равен 0 (неуспех), то переход к п.6. Иначе присвоить ПСС значение 0.

3.3 Реассемблирование, формирование и вывод значений регистров, значений ячеек ОП. Переход к п.2.

4 Если выбрано действие «Следующий шаг» или «Автоматически»

4.1 Если ПСС≠0, то переход к п.6

4.2 Выполнить эмуляцию. Если код выполнения >0, то установка ПСС=1. Если код выполнения равен 1, то переход к п.4.4, иначе переход к п. 6.

4.3 Если выбрано действие «Автоматически» , переход к п 4.1

4.4 Реассемблирование, формирование и вывод значений регистров, значений ячеек ОП. Переход к п.2.

5 Если выбрано действие «Закрыть», то закрыть программу и к п.7

6 Формирование текста сообщения об ошибке. Вывод сообщения на экран.

7 Конец.

**Алгоритм эмуляции**

Код результата выполнения эмуляции может принимать следующие значения:

-1 — идет обработка

0 – эмуляция завершилась успешно

1 – эмуляция завершилась успешно, остановка вычислений

3 – ошибка: неправильный формат команды

4- ошибка: обращение к адресу за пределами программы при считывании информации

5- ошибка: обращение к адресу за пределами программы при записи информации

Алгоритм:

1 Чтение из ОП 2 байта (код команды и способ адресации) по IP. Ели IP выходит за пределы выделенной памяти, то код ошибки=14, переход к п.7

2 Ваполнение алгоритма формирования промежуточного кода, если не успешно код выполнения=3, переход к п.7

3 Увеличение IP на длину команды.

4 Определение по промежуточному коду код команды, тип, значения операндов, устанавливаемые и влияющие флаги.

5 Считывание значений операндов, если не успешно код выполнения =4, переход к п.7

6 Выполнение (эмуляция) команды:

6.1 Если команда INT 20H, код выполнения эмуляции устанавливается в 1. Переход к п. 7

6.2 Выполнение команды в соответствии с ее кодом и установка изменяемых данной командой параметров системы.

6.3 Установка флагов (если нужно), запись результата в регистры или ОП, если не успешно, код выполнения =5, переход к п.7

6.4 Установка кода выполнения при эмуляции в 0.

7 Вернуть значение кода выполнения эмуляции.

8 Конец.

**Алгоритм реассемблирования**

1 Адрес команды для реассемблирования:=адрес начала программы

2 Если адрес команды для реассемблирования не выходит за пределы программы, иначе переход к п. 3

2.1 Чтение команды

2.2 Формирование промежуточного кода.

2.3 Определение по промежуточному коду мнемоники операции, мнемокоды операндов, длину команды.

2.4 Формирование строки кода программы

2.5 Добавление строки в текст. Увеличение адреса команды на длину команды. Переход к п. 2

3 Возврат полученного текста.

4 Конец.

**Алгоритм формирования промежуточного кода**

Код результата выполнения дешифрации команды может принимать следующие значения:

0 – дешифрация завершилась успешно

3 – ошибка: неправильный формат команды

Алгоритм:

1. Поиск по ТФК кода команды, совпадающей с прочитанным кодом

1.1 Определить ключ при поиске в ТФК: маска 0xFFC0 прочитанных 2х байт.

1.2 Поиск в ТФК. Если неуспешен, то КД:=3, переход к п.3

2. Определение длины команды, длины и места хранения операндов, байтов d, w, мнемоники операции, номер формата по ТФК и формирование по этим данным промежуточного кода. КД:=0. Переход к п.4

3. Определение мнемоники операции, мнемокодов операндов и формирование промежуточных кодов.

4. Вернуть значение КД.

5. Конец

# 6 ОПИСАНИЕ ПРОГРАММЫ

## 6.1. Назначение программы

Отладчик предназначен для облегчения процесса отладки и тестирования программ. Отладчик позволяет управлять процессом исполнения отлаживаемой программы и получать информацию о ее текущем состоянии и о содержимом памяти, приостанавливать и вновь возобновлять исполнение программы. Отлаживаемая программа выполняется виртуально, то есть каждая инструкция программы эмулируется отладчиком.

## 6.2. Требования к программному и техническому обеспечению

При написании программного назначения использовался кроссплатформенный язык программирования Java. Все требования к программному и техническому обеспечению сводятся к нему.

Для корректного функционирования написанного программного обеспечения, требуется установленный программный продукт Java Runtime Environment (JRE), процессор Pentium с тактовой частотой от 166 МГц и выше, должно быть как минимум 125 МБ свободного дискового пространства и не менее 32 МБ оперативной памяти.

## 6.3. Логическая структура программы

Язык Java объектно-ориентированный, по-этому разумно будет разбить программу на классы и методы.

Программа состои из следующих классов:

- class Core – класс в котором объявлены основные переменные и вспомогательные методы.
- Class Loader – наследует класс Core, выполняет функции загрузчка, загружает обектный код в ОП и задает адрес начала программы.
- Class Emulator – наследует класс Core, выполняет считывание команд из ОП и эмуляцию.
- Class MemoryTable – реализация модели таблицы для графического

интерфейса
- Class GUI – класс в котором реализован графический пользовательский интерфейс
- class TFK_row – реализует строку таблицы форматов команд
- class TFK_table – реализует таблицу форматов команд, массив элементов TFK_row
- class pCode – служит для формирования промежуточного кода
- class Reassembling — наследуе класс Core,реализазия реассемблирования

## 6.4 Используемые переменные и типы

В следующих таблицах описываются глобальные переменные, либо поля объектов классов программы.

Таблица 6.4.1 - Глабльные переменные класса Core

| Имя переменной | Тип | Назначение |
|---|---|---|
| Memory | Byte[65535] | Оперативная память |
| Reg | Short[8] | Базовые и Индексные регистры |
| Ip | Short | Регистр IP |
| Rf | Short | Регистр флагов |
| Start | Int | Адрес начала сегмента |
| Stop | Int | Адрес конца сегмента |
| Table | TFK_table | Таблица форматов команд |
| TFKresult | TFK_row | Результат поиска в таблтце форматов команд |
| CodeGet | Int | Результат считывания операнда |

Таблица 6.4.2 - Глабльные переменные класса Loader

| Имя переменной | Тип | Назначение |
|---|---|---|
| codeLength | int | длина программы |
| StateLoader | int | Код Результата работы загрузчика |
| inputFile | File | объектный файл |

**Таблица 6.4.3 - Глабльные переменные класса Emulator**

| Имя переменной | Тип | Назначение |
|---|---|---|
| stateEmulation | int | Код Результата работы эмулятора |

Таблица 6.4.4 - **Глабльные переменные класса GUI**

| Имя переменной | Тип | Назначение |
|---|---|---|
| listModel | DefaultListModel | Модель окна текста программы |
| Load | Loader | Объект класс Loader |
| Core | Core | Объект класса Core |
| Emulation | Emulator | Объект класса Emulator |
| res | Reassembling | Объект класса Reassembling |

Таблица 6.4.5 - Глабльные переменные класса TFK_row

| Имя переменной | Тип | Назначение |
|---|---|---|
| Mnk | String | Мнемокод операции |
| Type1 | Char | Тим первого операнда |
| Type2 | Char | Тим второго операнда |
| Length | Byte | Длина команды |
| Code | Byte | Код операции |
| Mode | Byte | Поле |
| GroupId | Int | Номер формата |
| D | Boolean | Указатель адресации операндов |

Таблица 6.4.1 - Глабльные переменные класса pCode

| Имя переменной | Тип | Назначение |
|---|---|---|
| MkOp | String | Мнемокод операции |
| TypeOper1 | Char | Тим первого операнда |
| TypeOper2 | Char | Тим второго операнда |

| | | |
|---|---|---|
| Length | Byte | Длина команды |
| Code | Byte | Код операции |
| MkOper1 | String | Мнемокод первого операнда |
| MkOper2 | String | Мнемокод второго операнда |
| ValueOper1 | Short | Значение первого операнда |
| ValueOper1 | Short | Значение второго операнда |
| InCode | Byte | Внутренний код команды |
| GroupId | Int | Номер группы формата команды |
| D | Boolean | Указатель адресации операндов |
| RegTableHmk | String[] | Массив имен двухбайных регистров |
| RegTableLmk | String[] | Массив имен однобайных регистров |
| InCodeTable | String[] | Массив мнемокодов команд |

Таблица 6.4.1 - Глабльные переменные класса pCode

| Имя переменной | Тип | Назначение |
|---|---|---|
| ReasList | List<String> | Список реассемблированных строк |

## 6.5 СПЕЦИФИКАЦИЯ ПРОЦЕДУР И ФУНКЦИЙ

Назначение, входные и выходные данные методов программы.

**Класс Core**

**Название процедуры:** void writeMemory(int i, byte k)

**Назначение:** Записывает 1 байт в указанную ячейку массива

**Входные данные:** i — числовой индекс ячейки, k — значение типа байт.

**Выходные данные:** отсутствуют

**Название процедуры:** short getShort(byte b1, byte b2)

**Назначение:**Возвращает конкатенацию двух байтов

**Входные данные:** b1- первый байт, b2 - второй байт

**Выходные данные:** результат конкатенации

**Название процедуры:** int getReg(byte i) и int getRm(byte i)

**Назначение:**Возвращает код регистра из указаеля Reg и Rm второго байта операции

**Входные данные:** i — второй байт операции. Тип byte.

**Выходные данные:** Код регистра, тип int

**Название процедуры:** byte[] getBytesFromInt(int sh)

**Назначение:** Разбивает 16 младших битов int на массив байтов

**Входные данные:** sh - Значение. Тип int

**Выходные данные:** Массив из двух байт

**Название процедуры:** int searchTFK(byte b1, byte b2)

**Назначение:** Выполняет поиск в таблице форматов команд

**Входные данные:** b1 — первый байт операции. b2- второй байт

**Выходные данные:** Код выполнения. 0 — успешно. 3 — неверный формат комады

**Название процедуры:** int setValue(short pointer,int value,char type)

**Назначение:** Процедура записи значения по типу операнда

**Входные данные:** pointer — Указатель типа short. Value- числовое значение типа int. Type — символьный тип операнда

**Выходные данные:** Код выполнения. 0 — успешно. 5 — выход за пределы памяти при записи

**Название процедуры:** int setValueReg(short pointer,int value,boolean part)

**Назначение:** Процедура записи в регистр по коду и размеру

**Входные данные:** pointer — Указатель типа short.Value - числовое значение типа int. Part -булевская переменая указывающая на размер (1 или 2 байта)

**Выходные данные:** Код выполнения. 0 — успешно

**Название процедуры:** int setValueMem(byte value, short address)

**Назначение:** Процедура записи байта в ОП по адресу

**Входные данные:** Value -значение, тип byte, Address — тип short

**Выходные данные:** Код выполнения. 0 — успешно 5 — выход за пределы памяти при записи

**Название процедуры:** int setValueMem(int value, short adress)

**Назначение:** Процедура записи двухбайтов в ОП по адресу

**Входные данные:** Value -значение, тип byte, Address — тип short

**Выходные данные:** Код выполнения. 0 — успешно 5 — выход за пределы памяти при записи

**Название процедуры:** int getValue(short pointer,char type)

**Назначение:** Процедура чтения значения, по указателю, типу операнда

**Входные данные:** pointer — Указатель типа short. Type — символьный тип операнда

**Выходные данные:** Код выполнения. 0 — успешно 4 — выход за пределы памяти при записи

**Название процедуры:** int getValueReg(byte code,boolean part)

**Назначение:** Процедура чтения значения из регистра, по коду и размеру

**Входные данные:** code - код регистра, part - булевская переменная указывающая на размер

**Выходные данные:** Код выполнения. 0 — успешно

**Название процедуры:** int getValueMem(short address,boolean getShort)

**Назначение:** Вызывает процедуру чтения значения из памяти, по адрессу и размеру

**Входные данные:** Address — указатель тип short. GetShort - булевская переменная указывающая на размер

**Выходные данные:** Код выполнения. 0 — успешно 4 — выход за пределы памяти при записи

**Название процедуры:** void reset()

**Назначение:** Сбрасывает ip, массив памяти, регистров, флагов и адреса начала и конца программы в начальное состояние.

**Class Loader**

**Название процедуры:** void doLoad()

**Назначение:** Запуск процедуры загрузки

**Входные данные:** Файл объектного кода.

**Выходные данные:** Код выполнения загрузки. 0 — успешно, остальные — ошибка. (Расшифрока в п.6.6)

**Название процедуры:** cartT(java.io.RandomAccessFile in) и cartE(java.io.RandomAccessFile in)

**Назначение:** Обработка карты T и E

**Входные данные:** Файл объектного кода.

**Выходные данные**: Код результата обработки

**Class Emulator**

**Название процедуры:** void doEmul()

**Назначение:** Запуск процедуры эмуляции

**Входные данные:** Адресс команды в оперативной памяти, тип int

**Выходные данные:** Код выполнения эмуляции (расшифровка в п.6.6.)

**Название процедуры:** int emulateType1(pCode pc), emulateType2, emulateType3, emulateType4, emulateType5, emulateType6

**Назначение:** Выполняет эмуляцию операции с заданной группой формата

**Входные данные:** Объект промежуточного кода

**Выходные данные:** Код выполнения 0 — успешно. 4- ошибка при записи результата, 5 — ошибка при считывании результата

**Название процедуры:** int getStateEmulation()

**Назначение:** Возвращает код состояния эмуляции

**Использует данные:** код состояния

**Выходные данные:** Код выполнения 0 — успешно. 4- ошибка при записи результата, 5 — ошибка при считывании результата

**Название процедуры:** int doIMUL(short m1,short m2) и int doIMUL(byte m1,byte m2)

**Назначение:** Выполняет умножение со знаком, двух операндов типа byte или short

**Входные данные:** m1 — первый операнд (тип byte или short). m2 — второй операнд (тип byte или short).

**Выходные данные:** Результат умножения тип int

**Название процедуры:** int setIMUL(int rez,boolean isShort)

**Назначение:** Записывает результат множения в регистры и устанавливает флаги

**Входные данные:** int rez — результат умножения. Boolean isShort — размер результата

**Выходные данные:** Код выполнения 0- успешно. 5- ошибка при записи.

**Class pCode**

**Название процедуры:** int doPCode(byte b1,byte b2,int adr)

**Назначение:** Выполняет формирование промежуточного кода

**Входные данные:** 2 байта операции и адресс текущей команды

**Выходные данные:** Код выполнения 0 — успешно. 3- неверный формат команды

**Class Reassembling**

**Название процедуры:** void doReassembling()

**Назначение:** Выполняет реассемблирование текста программы

**Входные данные:** отсутствуют

**Выходные данные**: отсутствуют

**Class GUI**

**Название процедуры:** void refresh()

**Назначение:** Обновляет данные в граф интерфейсе

**Использует данные:** массив регистров, флагов, оперативной памяти и текст программы

**Выходные данные:** отсутствуют

**Название процедуры:** void reset()

**Назначение:** Сбрасывает программу в начальное состояние

**Использует данные:** Файлы объектов Core, Emulation, Loader,File

**Выходные данные:** отсутствуют

## 6.6. СООБЩЕНИЯ, ВЫДАВАЕМЫЕ ПРОГРАММОЙ

В процессе работы программы, на экран интерфейса могут выводиться сообщения, характеризующие состояния программы.

Сообщения, выдаваемые программой представим в таблице 6.6.1

Таблица 6.6.1 – Сообщения, выдаваемые программой

| Сообщение | Описание сообщения |
|---|---|
| Прерывание. Возврат в операционную систему. | Успешное выполнение программы, встретилась команда INT 20h |
| Не правильный формат команды | Команда не найдена в таблице ТФК, либо имеет не правильный формат |
| Обращение к адресу за пределами программы при считывании информации | Невозможно считать значение за пределами памяти |
| Обращение к адресу за пределами программы при записи информации | Невозможно записать значение за пределами памяти |
| Отсутствует карта H | В объектном коде отсутствует карта H |
| Отсутствует карта T | В объектном коде отсутствует карта T |
| Отсутствует карта E | В объектном коде отсутствует карта E |
| Выход за пределы ОП при загрузке | Программа не помещается в память |
| Неизвестные данные в файле | Не удалось опознать карту |
| Неизвестная ошибка | Произошла непредвиденная ошибка |
| Файл не выбран | Файл с объектным кодом не выбран |

## 6.7. Описание интерфейса

Интерфейс представлен на рисунке 6.1. Окно программы можно разделить на четыре панели.

Панель «Управление». Содержит кнопки управления программой «Выбрать файл»,«Загрузить», «Шаг», «Сброс»,«Завершить».

Панель «Информация». Содержит текущее состояни регистров общего назначения, индексных и сегментных регистров, флагов, адреса загрузки и регистра IP, в шестнадцатеричном формате.

Панель «Текст программы». Содержит реассемблированый текст программы, который отобораżается после загрузки объектного файла. Строка следующей выполняемой операции выделяется цветом. Если произошла

ошибка при реассемблировании команды, то выводится только адрес и содержимое команды. Окно содержит адрес операции, мнемонику операции, мнемоники операндов и байты операции в шестнадцатеричном формате.

Панель «ОП». Таблица отображающая оперативную память. Сегмент текущей программы выделен цветом. Адресс и данные в ячейках представленны в шестнадцатеричном формате.

### 6.8. Вызов и загрузка программы

Программа вызывается запуском файла sp.jar. После запуска программы появляется окно (приведенное на рисунке 6.1). С помощью кнопки «Выбрать файл», следует вызвать диалоговое окно, в котором, нужно выбрать объектный файл для загрузки. Перед загрузкой программы можно установить адрес загрузки и значения регистров в соответствующих полях. Чтобы загрузить объектный код в ОП (одновременно с дизассемблированием программы) необходимо нажать кнопку «загрузить». Для выполнения одной команды, необходимо нажать на кнопку «Шаг». Сбросить программу в начальное состояние можно кнопкой «Сброс». Чтобы завершить работу отладчика следует нажать кнопку «Завершить»

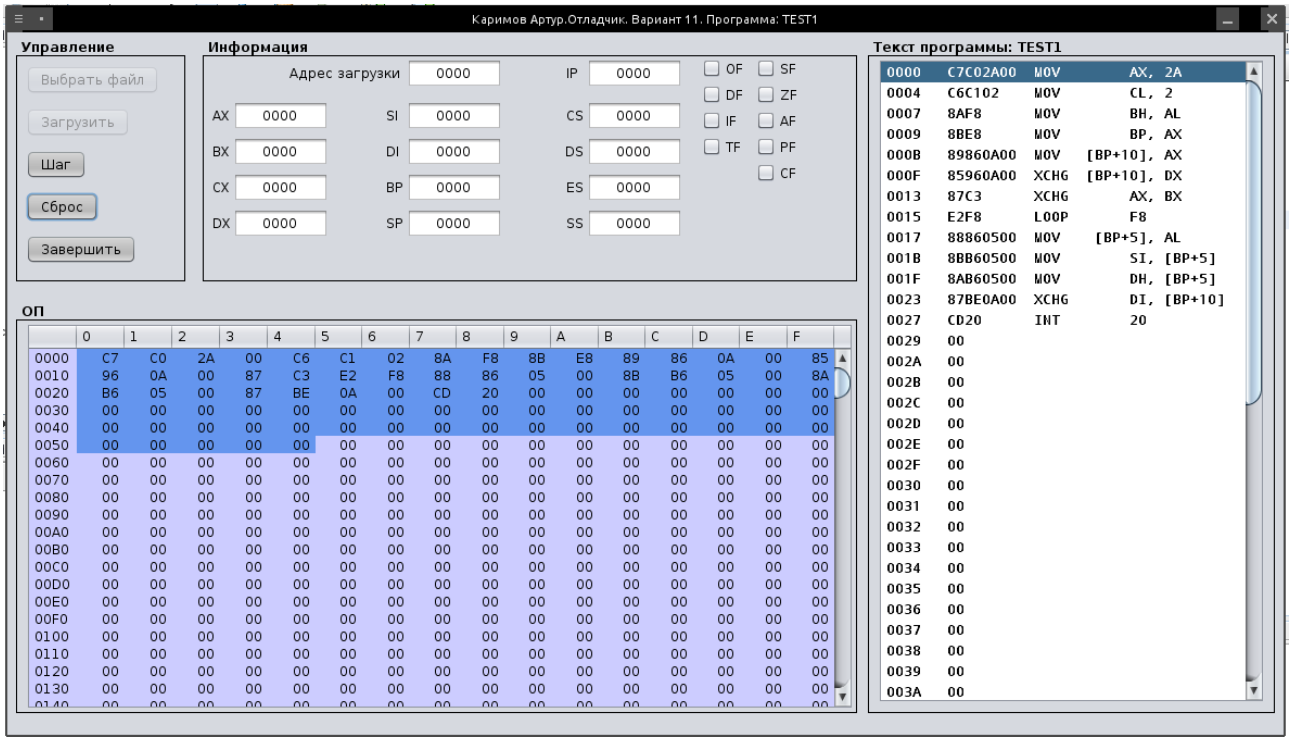


Рисунок 6.1. - Интерфейс отладчика

# 7 ТЕКСТ ПРОГРАММЫ

При написании программы использовалась среда NetBeans IDE 7.4. и все ее стандартные библиотеки и инструменты. В том числе и при проектировании графического пользовательского интерфейса.
Текст программы  приведен в приложении А.

Перечень файлов программы:

Core.java

Emulator.java

GUI.java

Loader.java,

MemoryTable.java

Reassembling.java

TFK_row.java

TFK_table.java

pCode.java

# 8 ТЕСТИРОВАНИЕ ПРОГРАММЫ

## 8.1. Разработка тестовых примеров

Случай 1. Правильная работа программы

Исходный текст программы:

```
TEST1 segment
    MOV         AX, 2A
    MOV         CL, 2
    MOV         BH, AL
    MOV         BP, AX
    MOV    [BP+10], AX
    XCHG   [BP+10], DX
    XCHG        AX, BX
    LOOP        F8
    MOV     [BP+5], AL
    MOV         SI, [BP+5]
    MOV         DH, [BP+5]
    XCHG        DI, [BP+10]
    INT         20

ends
end
```

Случай 2. Правильная работа программы

Исходный текст программы:

```
TEST1 segment
    MOV         CX, 5E5A
    MOV         AL, CL
    IMUL        CX,
    IMUL        CL,
    MOV         CL, 1
    IMUL        CL,
    MOV         BL, 1A
    XCHG        AL, [BX+7]
    XCHG        SP, [BX+7]
    XCHG        BL, CL
    INT         20


ends
ens
```

Случай 3. Ошибка при загрузке. Отсутствует карта H

Случай 4. Ошибка при записи

8.2. Результаты тестирования

Случай 1. Зададим перед загрузкой регистру DX значение 0001, Адрес загрузки 000A. Программа отработала без ошибок. Результат выполнения на рисунке 8.1.

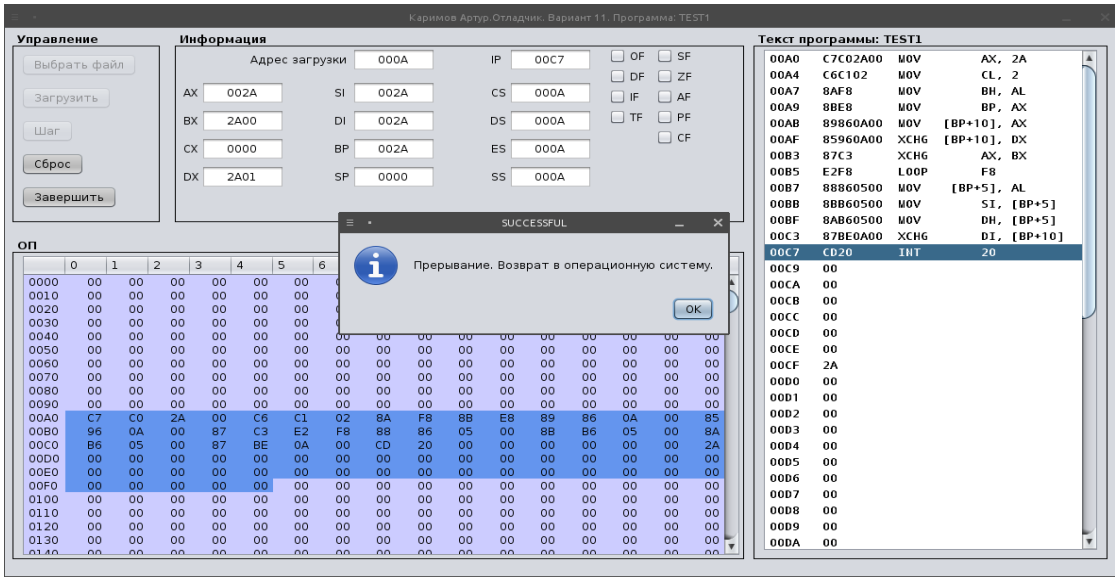


Рисунок 8.1. - Результат работы программы в Случае 1

Случай 2. Адрес загрузки 0010. Программа отработала без ошибок. Результат выполнения на рисунке 8.2

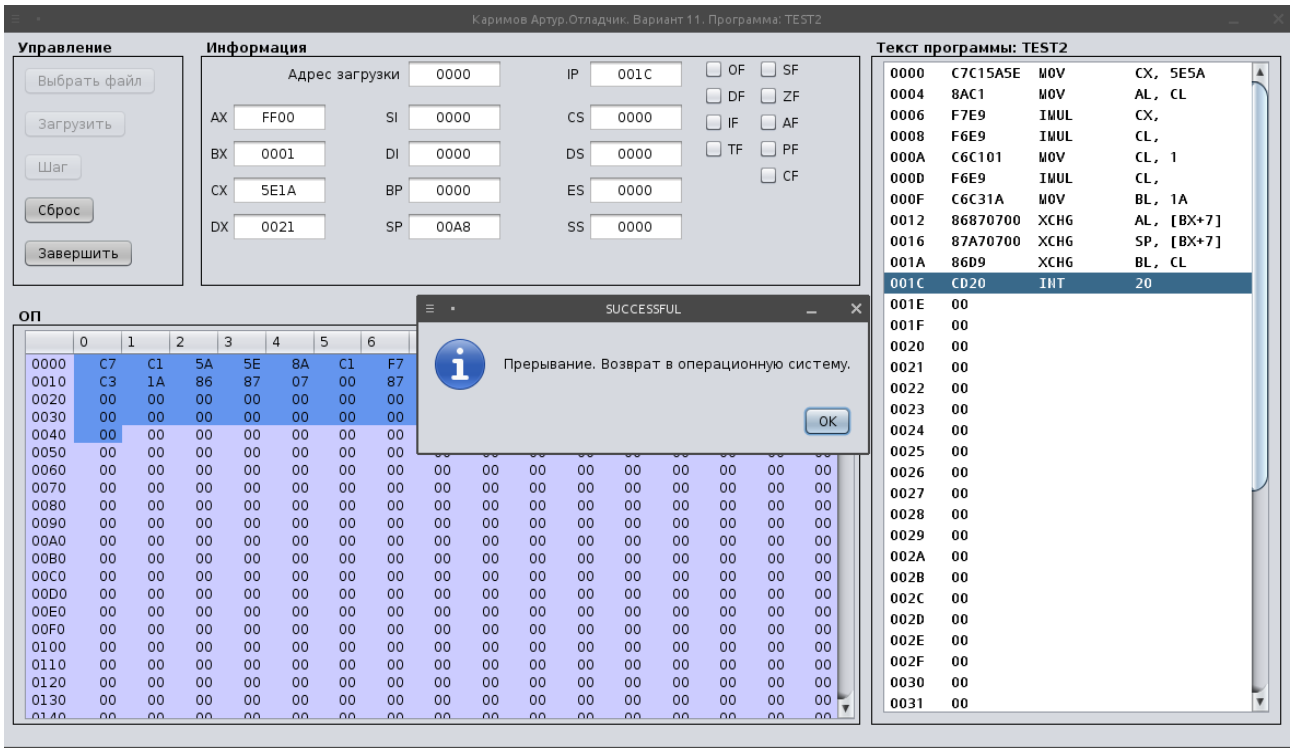


Рисунок 8.2. - Результат работы программы в Случае 2

Случай 3. Ошибка при загрузе, отсутствует карта H. Результат на Рисунке 8.3.

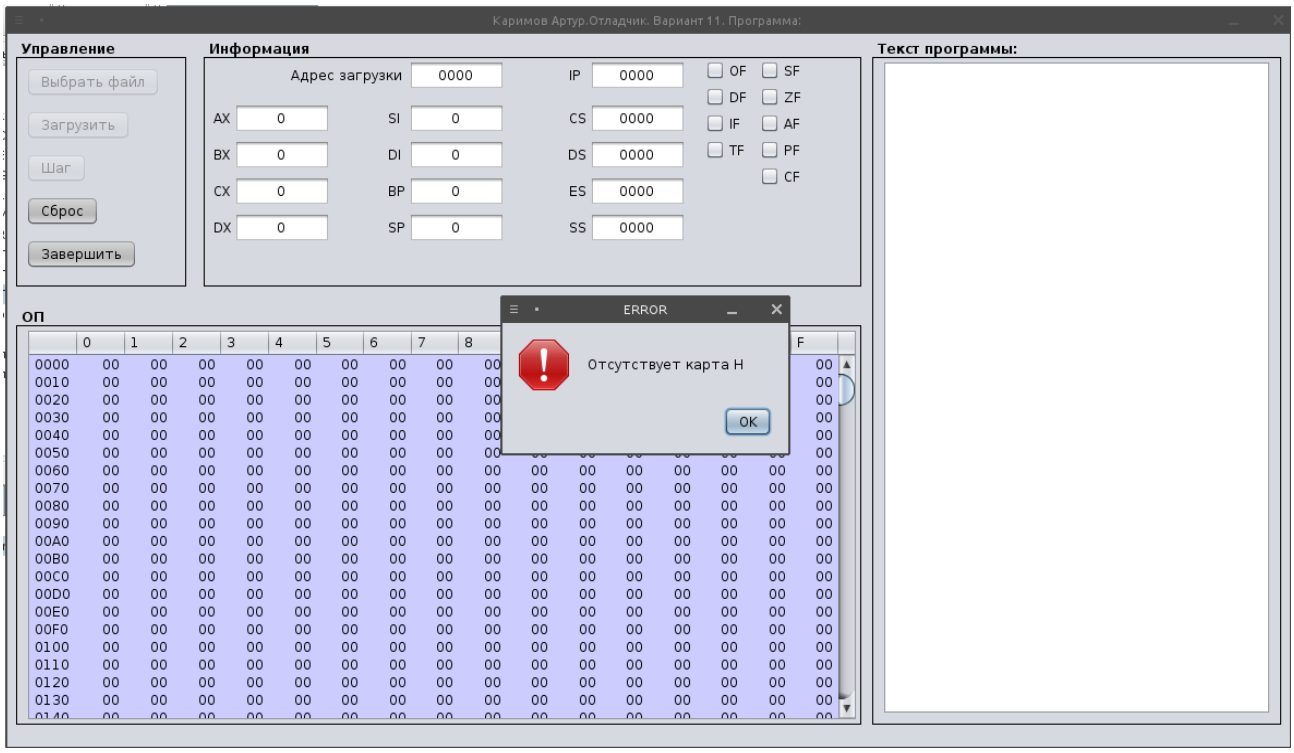


Рисунок 8.3. - Результат работы программы в Случае 3

Случай 4. Ошибка при загрузе, отсутствует карта H. Результат на Рисунке 8.4.

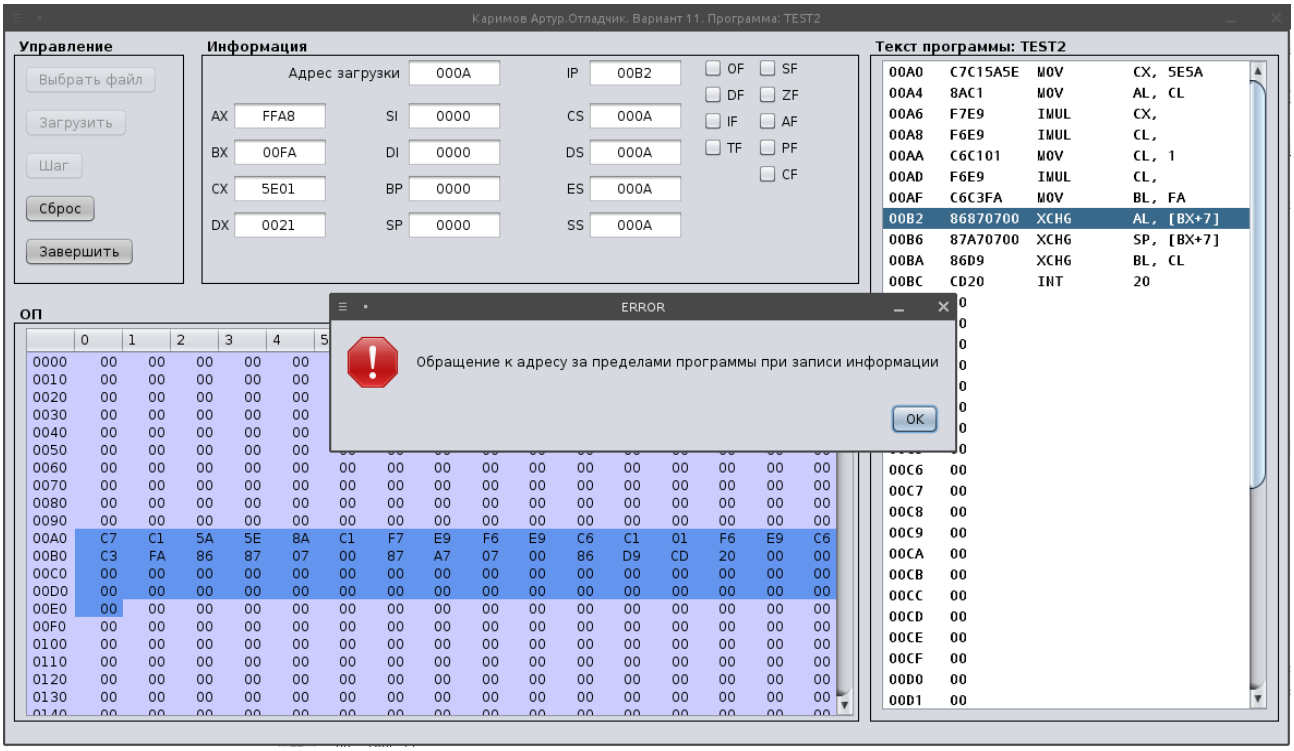


Рисунок 8.4. - Результат работы программы в Случае 4

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы были изучены форматы команд микропроцессора i8088. Разработана функциональная схема работы отладчика. Были получены практические навыки в написании программ эмуляции для языков низкого уровня. Программа написана на языке программирования Java в среде NetBeans IDE 7.4

Результаты испытаний показали, что на тестовых примерах программа работает корректно.

Библиографический список


1. Тертычный А.И. Методические указания к выполнению курсовой работы «Разработка транслятора для языка Ассемблер и универсального отладчика-эмулятора» по дисциплине «Системное программирование» для студентов специальности 7.091501 дневной и заочной форм обучения/ А.И. Тертычный, Е.М. Шалимова – Севастополь: издательство СевГТУ, 2001.- 20с.

2. Дао Л. Программирование микропроцессора 8088: Пер. с англ. – М.: Мир, 1988. – 357 с.
   Издательство: Москва «Мир» 1988.

3 Абель П. Язык Ассемблера для IBM PC и программирования: Пер. с англ. – М.: Высшая школа, 1992.-477 с.

4. Донован Дж. Системное программирование / Дж. Донован. –М.: Мир,1975.– 540с.

5. Баррон Д. Ассемблеры и загрузчики / Д. Баррон – М.: Мир, 1974. – 74с.

# ПРИЛОЖЕНИЕ А

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package sp_kurs;

import java.awt.Color;
import java.io.File;
import javax.swing.DefaultListModel;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JTable;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.TableCellRenderer;

/**
 *
 * @author grimfri
 */
public class GUI extends javax.swing.JFrame {

    /**
     * Creates new form MainFrame
     */
    public GUI() {
        initComponents();
        refresh();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        control = new javax.swing.JPanel();
        loadProg = new javax.swing.JButton();
        step = new javax.swing.JButton();
        terminate = new javax.swing.JButton();
        choose = new javax.swing.JButton();
        terminate1 = new javax.swing.JButton();
        oper = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        mem = new JTable(){

            public TableCellRenderer getCellRenderer(int row, int column) {
                int start=core.getStart();
                int stop = core.getStop();
                DefaultTableCellRenderer renderer = new DefaultTableCellRenderer();
                renderer.setHorizontalAlignment(JLabel.RIGHT);
                if ((column!=0)&&(row >= start / 16) && ((row < stop / 16)|((row == stop / 16)&&(column <= stop %
16 )))&&(load!=null)) {

                    renderer.setBackground(new Color(100,149,238));
                    return renderer;
```

```
            }
            return super.getCellRenderer(row, column);
        }
    };
    info = new javax.swing.JPanel();
    adr_label = new javax.swing.JLabel();
    adr_value = new javax.swing.JTextField();
    AX1 = new javax.swing.JLabel();
    BX1 = new javax.swing.JLabel();
    CX1 = new javax.swing.JLabel();
    DX1 = new javax.swing.JLabel();
    AX = new javax.swing.JTextField();
    BX = new javax.swing.JTextField();
    CX = new javax.swing.JTextField();
    DX = new javax.swing.JTextField();
    SI1 = new javax.swing.JLabel();
    SI = new javax.swing.JTextField();
    DI = new javax.swing.JTextField();
    DI1 = new javax.swing.JLabel();
    BP1 = new javax.swing.JLabel();
    BP = new javax.swing.JTextField();
    SP = new javax.swing.JTextField();
    SP1 = new javax.swing.JLabel();
    IP1 = new javax.swing.JLabel();
    IP = new javax.swing.JTextField();
    CS1 = new javax.swing.JLabel();
    CS = new javax.swing.JTextField();
    DS = new javax.swing.JTextField();
    DS1 = new javax.swing.JLabel();
    ES1 = new javax.swing.JLabel();
    ES = new javax.swing.JTextField();
    SS = new javax.swing.JTextField();
    SS1 = new javax.swing.JLabel();
    jPanel3 = new javax.swing.JPanel();
    OF = new javax.swing.JCheckBox();
    DF = new javax.swing.JCheckBox();
    IF = new javax.swing.JCheckBox();
    TF = new javax.swing.JCheckBox();
    CF = new javax.swing.JCheckBox();
    ZF = new javax.swing.JCheckBox();
    SF = new javax.swing.JCheckBox();
    PF = new javax.swing.JCheckBox();
    AF = new javax.swing.JCheckBox();
    Text = new javax.swing.JPanel();
    jScrollPane2 = new javax.swing.JScrollPane();
    listModel = new DefaultListModel();
    Text_Program = new javax.swing.JList(listModel);

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Каримов Артур.Отладчик. Вариант 11.");
    setResizable(false);


control.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)), "Управление", javax.swing.border.TitledBorder.LEFT,
javax.swing.border.TitledBorder.DEFAULT_POSITION));
    control.setName(""); // NOI18N

    loadProg.setBackground(new java.awt.Color(153, 153, 153));
    loadProg.setText("Загрузить");
    loadProg.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        loadProgActionPerformed(evt);
    }
});

step.setBackground(new java.awt.Color(153, 153, 153));
step.setText("Шаг");
step.setEnabled(false);
step.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        stepActionPerformed(evt);
    }
});

terminate.setBackground(new java.awt.Color(153, 153, 153));
terminate.setText("Завершить");
terminate.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        terminateActionPerformed(evt);
    }
});

choose.setBackground(new java.awt.Color(153, 153, 153));
choose.setText("Выбрать файл");
choose.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        chooseActionPerformed(evt);
    }
});

terminate1.setBackground(new java.awt.Color(153, 153, 153));
terminate1.setText("Сброс");
terminate1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        terminate1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout controlLayout = new javax.swing.GroupLayout(control);
control.setLayout(controlLayout);
controlLayout.setHorizontalGroup(
    controlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, controlLayout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(controlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(loadProg)
            .addComponent(step)
            .addComponent(terminate)
            .addComponent(choose))
        .addGap(21, 21, 21))
    .addGroup(controlLayout.createSequentialGroup()
        .addContainerGap()
        .addComponent(terminate1)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
controlLayout.setVerticalGroup(
    controlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, controlLayout.createSequentialGroup()
        .addContainerGap()
        .addComponent(choose)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(loadProg)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
            .addComponent(step)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(terminate1)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(terminate)
            .addContainerGap(13, Short.MAX_VALUE))
        );

        oper.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)), "ОП", javax.swing.border.TitledBorder.LEFT,
javax.swing.border.TitledBorder.DEFAULT_POSITION));

        jScrollPane1.setName(""); // NOI18N

        mem.setAutoCreateRowSorter(false);
        mem.setModel(
           new MemoryTable()
        );
        mem.setBackground(new java.awt.Color(204, 204, 255));

        mem.setDoubleBuffered(false);

        mem.setGridColor(new java.awt.Color(102, 102, 102));

        mem.setRowSelectionAllowed(false);

        mem.setSelectionForeground(new java.awt.Color(204, 204, 255));

        //mem.setDefaultRenderer(CegmentRender.class, new CegmentRender());

        mem.getTableHeader().setReorderingAllowed(false);
        mem.getColumnModel().getColumn(0).setResizable(false);
        mem.setUpdateSelectionOnSort(false);
        jScrollPane1.setViewportView(mem);

        javax.swing.GroupLayout operLayout = new javax.swing.GroupLayout(oper);
        oper.setLayout(operLayout);
        operLayout.setHorizontalGroup(
           operLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
           .addGroup(operLayout.createSequentialGroup()
              .addContainerGap()
              .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 763,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );
        operLayout.setVerticalGroup(
           operLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
           .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 356,
javax.swing.GroupLayout.PREFERRED_SIZE)
        );

        info.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)), "Информация"));

        adr_label.setText("Адрес загрузки");

        adr_value.setBackground(new java.awt.Color(255, 255, 255));
        adr_value.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        adr_value.setText("0000");

        AX1.setText("AX");

        BX1.setText("BX");
```

```
CX1.setText("CX");

DX1.setText("DX");

AX.setHorizontalAlignment(javax.swing.JTextField.CENTER);
AX.setText("0000");
AX.setToolTipText("");

BX.setHorizontalAlignment(javax.swing.JTextField.CENTER);
BX.setText("0000");

CX.setHorizontalAlignment(javax.swing.JTextField.CENTER);
CX.setText("0000");

DX.setHorizontalAlignment(javax.swing.JTextField.CENTER);
DX.setText("0000");

SI1.setText("SI");

SI.setHorizontalAlignment(javax.swing.JTextField.CENTER);
SI.setText("0000");

DI.setHorizontalAlignment(javax.swing.JTextField.CENTER);
DI.setText("0000");

DI1.setText("DI");

BP1.setText("BP");

BP.setHorizontalAlignment(javax.swing.JTextField.CENTER);
BP.setText("0000");

SP.setHorizontalAlignment(javax.swing.JTextField.CENTER);
SP.setText("0000");

SP1.setText("SP");

IP1.setText("IP");

IP.setEditable(false);
IP.setHorizontalAlignment(javax.swing.JTextField.CENTER);
IP.setText("0000");

CS1.setText("CS");

CS.setEditable(false);
CS.setBackground(new java.awt.Color(255, 255, 255));
CS.setHorizontalAlignment(javax.swing.JTextField.CENTER);
CS.setText("0000");

DS.setEditable(false);
DS.setBackground(new java.awt.Color(255, 255, 255));
DS.setHorizontalAlignment(javax.swing.JTextField.CENTER);
DS.setText("0000");

DS1.setText("DS");

ES1.setText("ES");

ES.setEditable(false);
ES.setBackground(new java.awt.Color(255, 255, 255));
```

```
        ES.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        ES.setText("0000");

        SS.setEditable(false);
        SS.setBackground(new java.awt.Color(255, 255, 255));
        SS.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        SS.setText("0000");

        SS1.setText("SS");

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 0, Short.MAX_VALUE)
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 0, Short.MAX_VALUE)
        );

        OF.setText("OF");

        DF.setText("DF");

        IF.setText("IF");

        TF.setText("TF");

        CF.setText("CF");

        ZF.setText("ZF");

        SF.setText("SF");

        PF.setText("PF");

        AF.setText("AF");

        javax.swing.GroupLayout infoLayout = new javax.swing.GroupLayout(info);
        info.setLayout(infoLayout);
        infoLayout.setHorizontalGroup(
            infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(infoLayout.createSequentialGroup()
                .addContainerGap()
                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(infoLayout.createSequentialGroup()
                        .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addGroup(infoLayout.createSequentialGroup()
                                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(DX1)
                                    .addComponent(BX1)
                                    .addComponent(CX1))
                                .addGap(3, 3, 3)
                                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false)
                                    .addComponent(CX, javax.swing.GroupLayout.DEFAULT_SIZE, 88, Short.MAX_VALUE)
                                    .addComponent(BX)
                                    .addComponent(DX)))
                            .addGroup(infoLayout.createSequentialGroup()
                                .addComponent(AX1)
                                .addGap(4, 4, 4)
```

```
                    .addComponent(AX, javax.swing.GroupLayout.DEFAULT_SIZE, 88, Short.MAX_VALUE)))
                .addGap(53, 53, 53)
                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(DI1)
                    .addComponent(BP1)
                    .addComponent(SP1)
                    .addComponent(SI1))
                .addGap(3, 3, 3))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, infoLayout.createSequentialGroup()
                .addComponent(adr_label)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)))
        .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addGroup(infoLayout.createSequentialGroup()
                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(DI, javax.swing.GroupLayout.DEFAULT_SIZE, 88, Short.MAX_VALUE)
                    .addComponent(BP)
                    .addComponent(SP))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, infoLayout.createSequentialGroup()
                        .addComponent(DS1)
                        .addGap(3, 3, 3)
                        .addComponent(DS, javax.swing.GroupLayout.PREFERRED_SIZE, 88,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, infoLayout.createSequentialGroup()
                        .addComponent(ES1)
                        .addGap(3, 3, 3)
                        .addComponent(ES, javax.swing.GroupLayout.PREFERRED_SIZE, 88,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, infoLayout.createSequentialGroup()
                        .addComponent(SS1)
                        .addGap(3, 3, 3)
                        .addComponent(SS, javax.swing.GroupLayout.PREFERRED_SIZE, 88,
javax.swing.GroupLayout.PREFERRED_SIZE))))
            .addGroup(infoLayout.createSequentialGroup()
                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                    .addComponent(adr_value)
                    .addComponent(SI, javax.swing.GroupLayout.DEFAULT_SIZE, 88, Short.MAX_VALUE))
                .addGap(59, 59, 59)
                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(infoLayout.createSequentialGroup()
                        .addComponent(CS1)
                        .addGap(3, 3, 3)
                        .addComponent(CS, javax.swing.GroupLayout.PREFERRED_SIZE, 88,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(infoLayout.createSequentialGroup()
                        .addComponent(IP1)
                        .addGap(8, 8, 8)
                        .addComponent(IP)))))
        .addGap(18, 18, 18)
        .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(OF)
            .addComponent(IF)
            .addComponent(TF)
            .addComponent(DF))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(SF)
            .addComponent(ZF)
            .addComponent(AF)
            .addComponent(PF)
```

```
                .addComponent(CF))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addContainerGap())
        );
        infoLayout.setVerticalGroup(
            infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(infoLayout.createSequentialGroup()
                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(infoLayout.createSequentialGroup()
                        .addComponent(OF)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addGroup(infoLayout.createSequentialGroup()
                                .addComponent(DF)
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(IF)
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(TF)
                                .addGap(0, 0, Short.MAX_VALUE))))
                    .addGroup(infoLayout.createSequentialGroup()
                        .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(infoLayout.createSequentialGroup()
                                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(adr_label)
                                    .addComponent(adr_value, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(IP1)
                                    .addComponent(IP, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(infoLayout.createSequentialGroup()

.addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                            .addComponent(AX1)
                                            .addComponent(AX, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                            .addComponent(BX1)
                                            .addComponent(BX, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                            .addComponent(CX1)
                                            .addComponent(CX, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                            .addComponent(DX1)
                                            .addComponent(DX, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
                                        .addGroup(infoLayout.createSequentialGroup()
```

```
.addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(SI1)
                                .addComponent(SI, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(CS1)
                                .addComponent(CS, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(DI1)
                                .addComponent(DI, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(DS1)
                                .addComponent(DS, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(BP1)
                                .addComponent(BP, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(ES1)
                                .addComponent(ES, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(infoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(SP1)
                                .addComponent(SP, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(SS1)
                                .addComponent(SS, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))))
                    .addGroup(infoLayout.createSequentialGroup()
                        .addComponent(SF)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(ZF)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(AF)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(PF)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(CF)))
                .addGap(0, 0, Short.MAX_VALUE)))
            .addContainerGap())
    );

    Text.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)), "Текст программы"));

    Text_Program.setFont(new java.awt.Font("Monospaced", 1, 12)); // NOI18N
    jScrollPane2.setViewportView(Text_Program);

    javax.swing.GroupLayout TextLayout = new javax.swing.GroupLayout(Text);
    Text.setLayout(TextLayout);
    TextLayout.setHorizontalGroup(
        TextLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(TextLayout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 357,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        TextLayout.setVerticalGroup(
            TextLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(TextLayout.createSequentialGroup()
                .addComponent(jScrollPane2)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(control, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(info, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addComponent(oper, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(Text, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(Text, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(control, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(info, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                        .addGap(18, 18, 18)
                        .addComponent(oper, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                .addGap(18, 18, 18))
        );

        control.getAccessibleContext().setAccessibleName("");
        info.getAccessibleContext().setAccessibleDescription("");

        pack();
    }// </editor-fold>

    private void loadProgActionPerformed(java.awt.event.ActionEvent evt) {
        startLoad();
    }

    private void stepActionPerformed(java.awt.event.ActionEvent evt) {
       step();
    }
```

```
    private void terminateActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void chooseActionPerformed(java.awt.event.ActionEvent evt) {

            JFileChooser fileopen = new JFileChooser();
            FileNameExtensionFilter binfilter=new FileNameExtensionFilter("BIN files","bin");
            fileopen.setFileFilter(binfilter);

            int ret = fileopen.showDialog(null, "Открыть файл");
            if (ret == JFileChooser.APPROVE_OPTION) {
                file = fileopen.getSelectedFile();
            }
    }

    private void terminate1ActionPerformed(java.awt.event.ActionEvent evt) {
        reset();
    }
    public void startLoad(){
    if (load == null) {
            if (file != null) { int start;
                try { start = Integer.parseInt(adr_value.getText(),16);} catch (Exception e) {start = 0;}
                inputReg();
                adr_value.setText(String.format("%04X",(start & 0xFFFF)));
                core.setSegmRegisters(start);
                //adr_value.setEditable(false);
                load = new Loader(start, file);
                load.doLoad();
                setTitle("Каримов Артур.Отладчик. Вариант 11. Программа:"+load.segmentName);
                choose.setEnabled(false);
                loadProg.setEnabled(false);
                step.setEnabled(true);

Text.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)), "Текст программы:"+load.segmentName));
                if(load.stateLoader==0){
                    refresh();
                }else{
                    codeError(load.stateLoader);
                }
            }else{codeError(99);}

        }
    }
    public void step(){
         emulation.doEmul();
        int state=emulation.getStateEmulation();
        if(state>0){
            if (state == 1) {
                step.setEnabled(false);
                javax.swing.JOptionPane.showMessageDialog(this,
                                            "Прерывание. Возврат в операционную систему.",
                                            "SUCCESSFUL",
                                            javax.swing.JOptionPane.INFORMATION_MESSAGE);
                refresh();
            }else{
                codeError(state);
            }
        }else{refresh();}
    }
```

```
public void reset(){
    core.reset();
    adr_value.setText(String.format("%04X",0));
    load=null;
    core=null;
    file=null;
    emulation=null;
    res=null;
    System.gc();
    choose.setEnabled(true);
    loadProg.setEnabled(true);
    step.setEnabled(false);
    core = new Core();
    emulation=new Emulator();
    res=new Reassembling();
    refresh();
}
public void refresh()
{
        for (int i = 0; i < 65536; i++) {
           mem.setValueAt(core.memory[i], i / 16, i % 16);
        }
    IP.setText(String.format("%04X", (core.ip & 0xFFFF)));
    AX.setText(String.format("%04X", (core.reg[0] & 0xFFFF)));
    BX.setText(String.format("%04X", (core.reg[3] & 0xFFFF)));
        CX.setText(String.format("%04X", (core.reg[1] & 0xFFFF)));
        DX.setText(String.format("%04X", (core.reg[2] & 0xFFFF)));
    SP.setText(String.format("%04X", (core.reg[4] & 0xFFFF)));
    BP.setText(String.format("%04X", (core.reg[5] & 0xFFFF)));
    SI.setText(String.format("%04X", (core.reg[6] & 0xFFFF)));
    DI.setText(String.format("%04X", (core.reg[7] & 0xFFFF)));
    CS.setText(String.format("%04X", (core.reg[8] & 0xFFFF)));
    DS.setText(String.format("%04X", (core.reg[9] & 0xFFFF)));
    ES.setText(String.format("%04X", (core.reg[10] & 0xFFFF)));
    SS.setText(String.format("%04X", (core.reg[11] & 0xFFFF)));


    mem.repaint();
    OF.setSelected(core.rf[4]);
    DF.setSelected(core.rf[5]);
    IF.setSelected(core.rf[6]);
    TF.setSelected(core.rf[7]);
    SF.setSelected(core.rf[8]);
    ZF.setSelected(core.rf[9]);
    AF.setSelected(core.rf[11]);
    PF.setSelected(core.rf[13]);
    CF.setSelected(core.rf[15]);

    res.doReassembling();
    listModel.removeAllElements();
    for (int i = 0; i < res.ReasList.size(); i++) {
       listModel.addElement(res.ReasList.get(i));
    }
    int point=0;
    for (int i = 0; i < res.ReasList.size(); i++) {
      if(res.ReasList.get(i).substring(0,4).equals(String.format("%04X",(core.ip&0xFFFF)))){point=i;break;};
    }
    Text_Program.repaint();
    res.ReasList.clear();
    Text_Program.setSelectedIndex(point);
}
public void inputReg()
```

```
{
   try {load.reg[0] = Integer.parseInt(AX.getText(),16);} catch (Exception e) {load.reg[0] = 0;}
        AX.setText("" + load.reg[0]);

        try {load.reg[1] = Integer.parseInt(CX.getText(),16);} catch (Exception e) {load.reg[1] = 0;}
        CX.setText("" + load.reg[1]);

        try {load.reg[2] = Integer.parseInt(DX.getText(),16);} catch (Exception e) {load.reg[2] = 0;}
        DX.setText("" + load.reg[2]);

        try {load.reg[3] = Integer.parseInt(BX.getText(),16);} catch (Exception e) {load.reg[3] = 0;}
   BX.setText("" + load.reg[3]);

        try {
           load.reg[4] = Integer.parseInt(SP.getText(),16);
        } catch (Exception e) {
           load.reg[4] = 0;
        }
   SP.setText("" + load.reg[4]);

   try {
           load.reg[5] = Integer.parseInt(BP.getText(),16);
        } catch (Exception e) {
           load.reg[5] = 0;
        }
   BP.setText("" + load.reg[4]);

   try {
           load.reg[6] = Integer.parseInt(SI.getText(),16);
        } catch (Exception e) {
           load.reg[6] = 0;
        }
   SI.setText("" + load.reg[6]);

   try {
           load.reg[7] = Integer.parseInt(DI.getText(),16);
        } catch (Exception e) {
           load.reg[7] = 0;
        }
   DI.setText("" + load.reg[7]);

}
 public void codeError(int sost)
{
        switch (sost) {

        case 3:showError("Не правильный формат команды"); break;
        case 4:showError("Обращение к адресу за пределами программы при считывании информации"); break;
        case 5:showError("Обращение к адресу за пределами программы при записи информации"); break;
    //loaderErrors
        case 11: showError("Отсутствует карта H"); break;
        case 12: showError("Отсутствует карта T"); break;
        case 13: showError("Отсутствует карта E"); break;
   case 14: showError("Выход за пределы ОП при загрузке"); break;
   case 15: showError("Неизвестные данные в файле "); break;
   case 99: showError("Файл не загружен"); break;
        default: showError("Неизвестная ошибка"); break;
        }
}

public void showError(String s)
{
```

```
        step.setEnabled(false);
        javax.swing.JOptionPane.showMessageDialog(this, s, "ERROR",
                                                  javax.swing.JOptionPane.ERROR_MESSAGE);
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new GUI().setVisible(true);
        }
    });
}
File file=null;
DefaultListModel listModel;
Loader load;
Core core = new Core();
Emulator emulation=new Emulator();
Reassembling res=new Reassembling();
// Variables declaration - do not modify
private javax.swing.JCheckBox AF;
private javax.swing.JTextField AX;
private javax.swing.JLabel AX1;
private javax.swing.JTextField BP;
private javax.swing.JLabel BP1;
private javax.swing.JTextField BX;
private javax.swing.JLabel BX1;
private javax.swing.JCheckBox CF;
private javax.swing.JTextField CS;
private javax.swing.JLabel CS1;
private javax.swing.JTextField CX;
private javax.swing.JLabel CX1;
private javax.swing.JCheckBox DF;
private javax.swing.JTextField DI;
private javax.swing.JLabel DI1;
private javax.swing.JTextField DS;
private javax.swing.JLabel DS1;
```

```
    private javax.swing.JTextField DX;
    private javax.swing.JLabel DX1;
    private javax.swing.JTextField ES;
    private javax.swing.JLabel ES1;
    private javax.swing.JCheckBox IF;
    private javax.swing.JTextField IP;
    private javax.swing.JLabel IP1;
    private javax.swing.JCheckBox OF;
    private javax.swing.JCheckBox PF;
    private javax.swing.JCheckBox SF;
    private javax.swing.JTextField SI;
    private javax.swing.JLabel SI1;
    private javax.swing.JTextField SP;
    private javax.swing.JLabel SP1;
    private javax.swing.JTextField SS;
    private javax.swing.JLabel SS1;
    private javax.swing.JCheckBox TF;
    private javax.swing.JPanel Text;
    private javax.swing.JList Text_Program;
    private javax.swing.JCheckBox ZF;
    private javax.swing.JLabel adr_label;
    private javax.swing.JTextField adr_value;
    private javax.swing.JButton choose;
    private javax.swing.JPanel control;
    private javax.swing.JPanel info;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JButton loadProg;
    private javax.swing.JTable mem;
    private javax.swing.JPanel oper;
    private javax.swing.JButton step;
    private javax.swing.JButton terminate;
    private javax.swing.JButton terminate1;
    // End of variables declaration
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package sp_kurs;

import java.util.Formatter;
import java.util.LinkedList;
import java.util.List;

/**
 *
 * @author grimfri
 */
public class Reassembling extends Core {
    List<String> ReasList=new LinkedList();
    Reassembling(){

    }

    void doReassembling(){
        String mkCode;
        int ReasIP=start;
```

```
        int ReasStop=stop;
        Formatter fmt;
        pCode RPCode=new pCode();
        while(ReasIP<ReasStop){
            int PCresult=RPCode.doPCode(memory[ReasIP],memory[ReasIP+1],ReasIP,true);
            mkCode="";
            for(int i=0;i<RPCode.length;i++){mkCode=mkCode+String.format("%02X",memory[ReasIP+i]&0xff);}
            fmt=new Formatter();
            if(PCresult==0){
                fmt.format("%04X%3S",(ReasIP&0xFFFF)," ");
                fmt.format("%-10S",mkCode);
                fmt.format("%-5S",RPCode.mkOp);
                if(RPCode.mkOper2.length()>0){fmt.format("%8S,",RPCode.mkOper1);}
                else{fmt.format("%8S",RPCode.mkOper1);}
                fmt.format(" %-8S",RPCode.mkOper2);

            }else{
                fmt.format("%04X%3S",(ReasIP&0xFFFF)," ");
                if(mkCode.length()>1){fmt.format("%-10S",mkCode);}
                else{fmt.format("0%-10S",mkCode);}
            }
            ReasList.add(fmt+"");
            fmt.close();
            ReasIP=ReasIP+RPCode.length;
        }
    }
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package sp_kurs;

/**
 *
 * @author grimfri
 */
public class Core {
    //int state;
    static byte[] memory = new byte[65536];//оперативная память
    static int[] reg = new int[12];//регистры
    static short ip;
    static boolean rf[]=new boolean[16]; //|-|-|-|-|OF|DF|IF|TF|SF|ZF|-|AF|-|PF|-|CF|
    static int start;
    static int stop;
    TFK_table table=new TFK_table();
    TFK_row TFKresult=new TFK_row();
    static int codeGet=0;

    public int getStart(){
        return start;
    }
    public int getStop(){
        return stop;
    }
    public static void writeMemory(int i, byte k)
    {
            if (i < 0) {
                i += 65536;
            }
```

```
        i += start;
        memory[i] = k;
}
public static short getShort(byte b1, byte b2)
{
        short i;
        if (b2 < 0) {
           b2 *= -1;
           i = b2;
           i = (short) (i << 8);
           i *= -1;
        } else {
           i = b2;
           i = (short) (i << 8);
        }
        if (b1 < 0) {
           i += b1 + 256;
        } else {
           i += b1;
        }
        return i;

}
public static int getReg(byte i)
{
        return ((i >> 3) & 7);
}

public static int getRm(byte i)
{
        return (i & 7);
}
public byte[] getBytesFromInt(int sh){
   byte[] regBytes=new byte[2];
   regBytes[0] = (byte)(sh & 0xff);
   regBytes[1] = (byte)((sh >> 8) & 0xff);

   return regBytes;
}
public short[] getShortesFromInt(int sh){
   short[] shortes=new short[2];
   shortes[0]=(short)(sh & 0xffff);
   shortes[1]=(short)((sh >>16) & 0xffff);
   return shortes;
}
int searchTFK(byte b1, byte b2){
   if((b1==(byte)0b11100010)||(b1==(byte)0b11001101)){
      //System.out.println("in2b1 "+b1);
      for(int i=0;i<table.TFK.length;i++){
         if(b1==table.TFK[i].code){
            TFKresult=table.TFK[i];
            return 0; // поиск успешен
         }
      }
   }
   else{
      b2=(byte)(b2 & 0xC0);
      b2=(byte)(b2 >> 1);
      b2=(byte)(b2 & 0x7F);
      b2=(byte)(b2 >> 5);
      //System.out.println("b1 "+b1);
      //System.out.println("b1 "+b2);
```

```
        for(int i=0;i<table.TFK.length;i++){
            if((b1==table.TFK[i].code)&&(b2==table.TFK[i].mode)){
                TFKresult=table.TFK[i];
                return 0; // поиск успешен
            }
        }
    }
        return 3; // поиск неуспешен
}


int setValue(short pointer,int value,char type){
    int rez=0;
    switch(type){
        case 'r':rez=setValueReg((byte)pointer,value,true);break;
        case 'R':rez=setValueReg((byte)pointer,value,false);break;
        case 'm':rez=setValueMem((byte)value,pointer);break;
        case 'M':rez=setValueMem(value,pointer);break;
    }
    return rez;
}
int setValueReg(short code,int value,boolean part){
    if(part){
        System.out.println("part="+part);
        System.out.println("value="+value);
        short codereg;
        boolean low;
        int old;
        if(code<4){codereg=code;low=true;}
        else{codereg=(byte)(code-4);low=false;}
        if(low){
          old=(reg[codereg]&0xff00);
          reg[codereg]=(value|old);
        }
        else{
          old=(reg[codereg]&0xff);
          value=((byte)value<<8);
          reg[codereg]=((short)value|old);
        }
    }
    else reg[code]=value;
    return 0;
}
int setValueMem(byte value, short adress){
    if((adress>=start)&(adress<stop)){
        memory[adress]=value;
        return 0;
    }
    else{return 5;} //выход за пределы при записи
}
int setValueMem(int value, short address){
    if((address>=start)&(address<stop)){
        byte bytes[]=getBytesFromInt(value);
        memory[address]=bytes[0];
        memory[address+1]=bytes[1];
        return 0;
    }
    else{return 5;} //выход за пределы при записи
}

int getValue(short pointer,char type){
    int value=0;
```

```
        switch(type){
            case 'r':value=getValueReg((byte)pointer,true); break;
            case 'R':value=getValueReg((byte)pointer,false); break;
            case 'm':value=getValueMem(pointer,false); break;
            case 'M':value=getValueMem(pointer,true); break;
            case 'N':case 'n':value=pointer; break;
        }
          return value;
     }

    int getValueReg(byte code,boolean part){
        if(part){
            byte codereg;
            boolean low;
            int result;
            if(code<4){codereg=code;low=true;}
            else{codereg=(byte)(code-4);low=false;}
            if(low){
                result=(reg[codereg]&0x00ff);
            }
            else{
                result=(reg[codereg]>>8);
            }
            return result;
        }
        else return reg[code];
    }

    int getValueMem(short address,boolean getShort){
        int result;
        if((address>=start)&(address<stop)){
            if(getShort){
                result=getShort(memory[address],memory[address+1]);
            }else{
                result=memory[address];
            }
        }else{codeGet=4;result=-66666;}
        return result;
    }
    void setSegmRegisters(int address){
        reg[8]=reg[9]=reg[10]=reg[11]=address;
    }
    void reset(){
        start=0;
        stop=0;
        ip=0;
        for(int i=0;i<memory.length;i++){
            if(i<12){reg[i]=0;}
            if(i<16){rf[i]=false;}
            memory[i]=0;
        }
    }
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package sp_kurs;

/**
```

```
 *
 * @author grimfri
 */
public class pCode extends Core{
    byte code;   //код операции
    int inCode;
    short length; //длина команды
    String mkOp; //мнемоника операции
    String mkOper1; //мнемокод операнда 1 (имя регистра, значение НО, BX+смещение)
    String mkOper2; // --||--||--||--
    char typeOper1; // тип операнда (R,r,M,m,N)
    char typeOper2; // --||--||--||--
    short valueOper1; //значение операнда (код регистра либо значение НО, либо ссылка)
    short valueOper2;
    int groupId;
    boolean d;

    String RegTableHmk[]={"AX","CX","DX","BX","SP","BP","SI","DI"};
    String RegTableLmk[]={"AL","CL","DL","BL","AH","CH","DH","BH"};

    pCode(){}

    int doPCode(byte b1,byte b2,int adr,boolean Reasb){
        int codeResult=0;
        codeResult=searchTFK(b1, b2);
        if (codeResult==0){
            code=TFKresult.code;
            length=TFKresult.length;
            mkOp=TFKresult.mnk;
            inCode=TFKresult.inCode;
            groupId=TFKresult.groupId;
            d=TFKresult.d;
            typeOper1=TFKresult.type1;
            typeOper2=TFKresult.type2;

            switch(groupId){
                case 1: valueOper1=(short)getReg(b2);
                        valueOper2=(short)getRm(b2);
                        mkOper1=getRegName(valueOper1,typeOper1);
                        mkOper2=getRegName(valueOper2,typeOper2);
                        break;
                case 2:
                        int regAdr=getRm(b2);
                        String regMnem="";
                            if(regAdr==6){regAdr=5;regMnem="BP";}
                            else{
                                if(regAdr==7){regAdr=3;regMnem="BX";}
                                else{
                                    codeResult=3;
                                }
                            }
                        if(d){
                            valueOper1=(short)getReg(b2);
                            if(Reasb){valueOper2=(short)(getShort(memory[adr+2],memory[adr+3]));}
                            else{valueOper2=(short)(reg[9]*16+reg[regAdr]+getShort(memory[adr+2],memory[adr+3]));}
                            if(valueOper2<0){valueOper2=(short)(65535-valueOper2);}
                            mkOper1=getRegName(valueOper1,typeOper1);
                            mkOper2="["+regMnem+"+"+Short.toString(valueOper2)+"]";
                        }else{
                            if(Reasb){valueOper1=(short)(getShort(memory[adr+2],memory[adr+3]));}
                            else{valueOper1=(short)(reg[9]*16+reg[regAdr]+getShort(memory[adr+2],memory[adr+3]));}
                            if(valueOper1<0){valueOper1=(short)(65535-valueOper1);}
```

```
                    valueOper2=(short)getReg(b2);
                    mkOper2=getRegName(valueOper2,typeOper2);
                    mkOper1="["+regMnem+"+"+Short.toString(valueOper1)+"]";
                }
                break;
            case 3: valueOper1=(short)getRm(b2);
                if(typeOper1=='R'){
                    valueOper2=getShort(memory[adr+2],memory[adr+3]);
                    if(valueOper2<0){valueOper2+= 65535;}
                }
                else{valueOper2=memory[adr+2];
                    if(valueOper2<0){valueOper2+= 256;}
                }
                mkOper1=getRegName(valueOper1,typeOper1);
                mkOper2=Integer.toHexString((valueOper2 & 0xFFFF));
                if((short)getReg(b2)!=0){codeResult=3;}
                break;

            case 4:
                if((short)getReg(b2)!=5){codeResult=3;}
                else{
                    valueOper1=(short)getRm(b2);
                    valueOper2=0;
                    mkOper1=getRegName(valueOper1,typeOper1);
                    mkOper2=" ";
                }
                break;
            case 5: valueOper1=b2;  //LOOP s
                mkOper1=String.format("%02X",b2);
                mkOper2="";
                break;
            case 6: valueOper1=b2;
                if(valueOper1!=0x20){codeResult=3;}
                mkOper1=Integer.toHexString(b2);
                mkOper2="";
                break;
        }
        return codeResult;
    }else{
        // Определение мнемоники операции, мнемокодов операндов для реассамблирования
        length=1;
        return codeResult;
    }
  }

  String getRegName(int codeReg, char type){
    if (type=='R'){
        return RegTableHmk[codeReg];
    }else{
        return RegTableLmk[codeReg];
    }
  }
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package sp_kurs;

/**
```

```
 *
 * @author grimfri
 */
public class TFK_table{
   TFK_row TFK[];

   public TFK_table() {
      this.TFK = new TFK_row[]{new TFK_row(0,"MOV",'r','r',(byte)2,(byte)0b10001010,(byte)0b11,true,1),//add
inCODE
                        new TFK_row(0,"MOV",'R','R',(byte)2,(byte)0b10001011,(byte)0b11,true,1),
                        new TFK_row(0,"MOV",'r','m',(byte)4,(byte)0b10001010,(byte)0b10,true,2),
                        new TFK_row(0,"MOV",'R','M',(byte)4,(byte)0b10001011,(byte)0b10,true,2),
                        new TFK_row(0,"MOV",'m','r',(byte)4,(byte)0b10001000,(byte)0b10,false,2),
                        new TFK_row(0,"MOV",'M','R',(byte)4,(byte)0b10001001,(byte)0b10,false,2),
                        new TFK_row(0,"MOV",'r','n',(byte)3,(byte)0b11000110,(byte)0b11,true,3),
                        new TFK_row(0,"MOV",'R','N',(byte)4,(byte)0b11000111,(byte)0b11,true,3),
                        new TFK_row(1,"XCHG",'r','r',(byte)2,(byte)0b10000110,(byte)0b11,true,1),
                        new TFK_row(1,"XCHG",'R','R',(byte)2,(byte)0b10000111,(byte)0b11,true,1),
                        new TFK_row(1,"XCHG",'r','m',(byte)4,(byte)0b10000110,(byte)0b10,true,2),
                        new TFK_row(1,"XCHG",'m','r',(byte)4,(byte)0b10000100,(byte)0b10,false,2),
                        new TFK_row(1,"XCHG",'R','M',(byte)4,(byte)0b10000111,(byte)0b10,true,2),
                        new TFK_row(1,"XCHG",'M','R',(byte)4,(byte)0b10000101,(byte)0b10,false,2),
                        new TFK_row(2,"IMUL",'r','r',(byte)2,(byte)0b11110110,(byte)0b11,true,4),
                        new TFK_row(2,"IMUL",'R','R',(byte)2,(byte)0b11110111,(byte)0b11,true,4),
                        new TFK_row(3,"LOOP",'l','-',(byte)2,(byte)0b11100010,(byte)0b00,true,5),
                        new TFK_row(4,"INT",'h','-',(byte)2,(byte)0b11001101,(byte)0b00,false,6)
                      };
   }

}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package sp_kurs;

/**
 *
 * @author grimfri
 */
public class TFK_row {
   int inCode;
   String mnk;
   char type1;
   char type2;
   byte length;
   byte code;
   byte mode;
   boolean d;
   int groupId; // 1- reg+reg; 2- reg+op; 3-reg+n; 4 - R; 5-loop;6-int20H
   TFK_row(){

   }
   TFK_row(int inCode, String mnk, char type1, char type2, byte length, byte code, byte mode,boolean d, int groupId)
{
       this.inCode=inCode;
       this.mnk=mnk;
       this.type1=type1;
       this.type2=type2;
       this.length=length;
```

```
        this.code=code;
        this.mode=mode;
        this.d=d;
        this.groupId=groupId;
    }
}
 /*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package sp_kurs;

import java.io.IOException;

/**
 *
 * @author grimfri
 */
public class Loader extends Core {
    java.io.File inputFile;
    int codeLength;
    int stateLoader;
    String segmentName = " ";

    Loader(int start, java.io.File in)
    {
            this.start = getValue((byte)0b1000,'R')*16;
            this.inputFile=in;
    }
     public void doLoad()
    {
        try {
            stateLoader = -1;
            java.io.RandomAccessFile in = new java.io.RandomAccessFile(inputFile, "r");
            int nameLength=0;
            stateLoader=0;
            if (in.readByte() == 72) {
                nameLength=in.readByte();
                for (int i = 0; i < nameLength; i++) {
                            segmentName += (char) in.readByte();
                        }
                System.out.println(segmentName);
                codeLength = getShort(in.readByte(), in.readByte());
                        stop = start + codeLength;
                if (stop < memory.length) {
                    int inputByte;
                    boolean ifCartT=false;
                    while((inputByte=in.readByte())==84){
                        ifCartT=true;
                        cartT(in);
                    }
                    if (ifCartT){
                        if (inputByte == 69) {
                          cartE(in);
                        }else{
                            stateLoader=13;
                        }
                    }else{stateLoader=12;}
                }else{
                    stateLoader=14;
```

```
            }
          }
          else{ stateLoader=11;}
       }catch (IOException e) {
          stateLoader=15;
          System.out.println(e);
            }
    }
     void cartT(java.io.RandomAccessFile in){
       try {
          short id = getShort(in.readByte(), in.readByte());
          int len1 = in.readByte();
          int len2 = in.readByte();
          if (len1 < 0) {len1 += 256;}
          if (len2 < 0) {len2 += 256;}
          short len=getShort((byte)len1,(byte)len2);
          for (int i = 0; i < len; i++) {
             byte t;
             writeMemory(id, in.readByte());
             id++;
          }
       }catch (IOException e) {
          stateLoader=15;
          System.out.println(e);
            }
    }
     void cartE(java.io.RandomAccessFile in){
       try {
          ip = (short) (getShort(in.readByte(), in.readByte())
                                                + start);
       }catch (IOException e) {
          stateLoader=15;
          System.out.println(e);
            }
     }
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package sp_kurs;

/**
 *
 * @author grimfri
 */
public class Emulator extends Core {
    int stateEmulation;
    public void doEmul(){
        stateEmulation=-1;
        while(stateEmulation==-1){
              byte b1=(byte)getValueMem(ip,false);
              byte b2=(byte)getValueMem((short)(ip+1),false);
              pCode p=new pCode();
              if(codeGet==0){
                 stateEmulation=p.doPCode(b1, b2,ip,false);
                 if (stateEmulation==0){
                     System.out.println("groupId "+p.groupId);
                   ip=(short) (ip+p.length);
                    switch(p.groupId){
```

```
                    case 1: stateEmulation=emulateType1(p);break;
                    case 2: stateEmulation=emulateType2(p);break;
                    case 3: stateEmulation=emulateType3(p);break;
                    case 4: stateEmulation=emulateType4(p);break;
                    case 5: stateEmulation=emulateType5(p);break;
                    case 6: stateEmulation=emulateType6(p);break;
                }
            }else{return;}
        }else{stateEmulation=codeGet;}
    }
}

public int emulateType1(pCode pc){
    int returnCode=0;
    if(pc.inCode==0){
        System.out.println("DO: "+pc.mkOp+" "+pc.mkOper1+" "+pc.mkOper2);
        returnCode=setValue((byte)pc.valueOper1,getValue(pc.valueOper2,pc.typeOper2),pc.typeOper1);
    }
    if(pc.inCode==1){
       System.out.println("DO: "+pc.mkOp+" "+pc.mkOper1+" "+pc.mkOper2);
       int t=getValue((byte)pc.valueOper1,pc.typeOper2);
       returnCode=setValue((byte)pc.valueOper1,getValue((byte)pc.valueOper2,pc.typeOper2),pc.typeOper1);
       returnCode=setValue((byte)pc.valueOper2,t,pc.typeOper2);
    }
    return returnCode;
}
public int emulateType2(pCode pc){
    int returnCode=0;
    System.out.println("DO: "+pc.mkOp+" "+pc.mkOper1+" "+pc.mkOper2);
    if(pc.inCode==0){
        int Oper2=getValue(pc.valueOper2,pc.typeOper2);
        if(codeGet==0){
            returnCode=setValue(pc.valueOper1,Oper2,pc.typeOper1);
        }else{returnCode=4;}
    } //MOV
    if(pc.inCode==1){//XCHG
        int temp=getValue(pc.valueOper1,pc.typeOper1);
        if (codeGet==0){
            returnCode=setValue(pc.valueOper1,getValue(pc.valueOper2,pc.typeOper2),pc.typeOper1);
            returnCode=setValue(pc.valueOper2,temp,pc.typeOper2);
        }else{returnCode=4;}
    }
    return returnCode;
}
public int emulateType3(pCode pc){
    System.out.println("DO: "+pc.mkOp+" "+pc.mkOper1+" "+pc.mkOper2+" NO: "+pc.valueOper2);
    setValue(pc.valueOper1,getValue(pc.valueOper2,pc.typeOper2),pc.typeOper1);
    return 0;
}

public int emulateType4(pCode pc){
    int returnCode=0;
    System.out.println("DO: "+pc.mkOp+" "+pc.mkOper1+" "+pc.mkOper2);
    int proiz;
    int mn1=getValue(pc.valueOper1,pc.typeOper1);
    int mn2=getValue(pc.valueOper2,pc.typeOper2);
    if(codeGet==0){
        if(pc.typeOper1=='R'){
            proiz=doIMUL((short)mn1,(short)mn2);
            returnCode=setIMUL(proiz,true);
        }
        else{
```

```
            proiz=doIMUL((byte)mn1,(byte)mn2);
            returnCode=setIMUL(proiz,false);
          }
      }else{returnCode=codeGet;}
      return returnCode;
   }
   public int emulateType5(pCode pc){
      System.out.println("DO: "+pc.mkOp+" "+pc.mkOper1);
      reg[1]=reg[1]-0b001;
      if(reg[1]!=0){ ip=(short)(ip+pc.valueOper1); }
      return 0;
   }
   public int emulateType6(pCode pc){
      System.out.println("DO: "+pc.mkOp+" "+pc.mkOper1);
      return 1;
   }

   int getStateEmulation(){
      return stateEmulation;
   }
   int doIMUL(short m1,short m2){
      int proiz=m1*m2;
      return proiz;
   }
   int doIMUL(byte m1,byte m2){
      int proiz=m1*m2;
      return proiz;
   }
   int setIMUL(int rez,boolean isShort){
      int returnCode;
      if(isShort){
         short p[]=getShortesFromInt(rez);
         returnCode=setValue((short)0b000,p[0],'R');
         returnCode=setValue((short)0b010,p[1],'R');
         if((rez>=-32768)&(rez<=32767)){rf[4]=false;rf[15]=false;}
         else{rf[4]=true;rf[15]=true;}
      }else{
         returnCode=setValue((short)0b000,rez,'r');
         if((rez>=-128)&(rez<=127)){rf[4]=false;rf[15]=false;}
         else{rf[4]=true;rf[15]=true;}
      }
      return returnCode;
   }
}
   /*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package sp_kurs;


import java.util.Formatter;
import javax.swing.table.AbstractTableModel;

/**
 *
 * @author grimfri
 */
public class MemoryTable extends AbstractTableModel
{
```

```
    String z[][] = new String[4096][16];
    String k[] = {" ", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A",
                          "B", "C", "D", "E", "F"};

    public int getRowCount()
    {
            return 4096;
    }

    public int getColumnCount()
    {
            return 17;
    }

    public Object getValueAt(int i, int j)
    {
            if (j == 0) {
           Formatter fmt = new Formatter();
           fmt.format("%04X", (i*16 & 0xFFFF));
                return fmt;
            }
            return z[i][j - 1];
    }

    public String getColumnName(int i)
    {
            return k[i];
    }

    public boolean isCellEditable(int i, int j)
    {
            return false;
    }

    public void setValueAt(Object h, int i, int j)
    {
       byte t=Byte.parseByte(h + "");
            z[i][j] = String.format("%02X", (t & 0xFF));

    }

    public Class getColumnClass(int i)
    {
            if (i == 0) {
               return java.lang.String.class;
            }
            return java.lang.Byte.class;
    }

    public boolean getCellResizable(int i, int j)
    {
            return false;
    }
}
```