Санкт-Петербургский государственный Политехнический университет

Институт Информационных Технологий и Управления

Кафедра Компьютерных Систем и Программных Технологий

Отчет по курсовому проекту на языке Java на тему:

«Симулятор компьютерной системы»

Выполнил студент группы 23501/4

Филиппов Виталий

Подпись:_____

Преподователь:

доцент Глухих М.И.

Санкт-Петербург 2014

Техническое задание

Постановка задачи

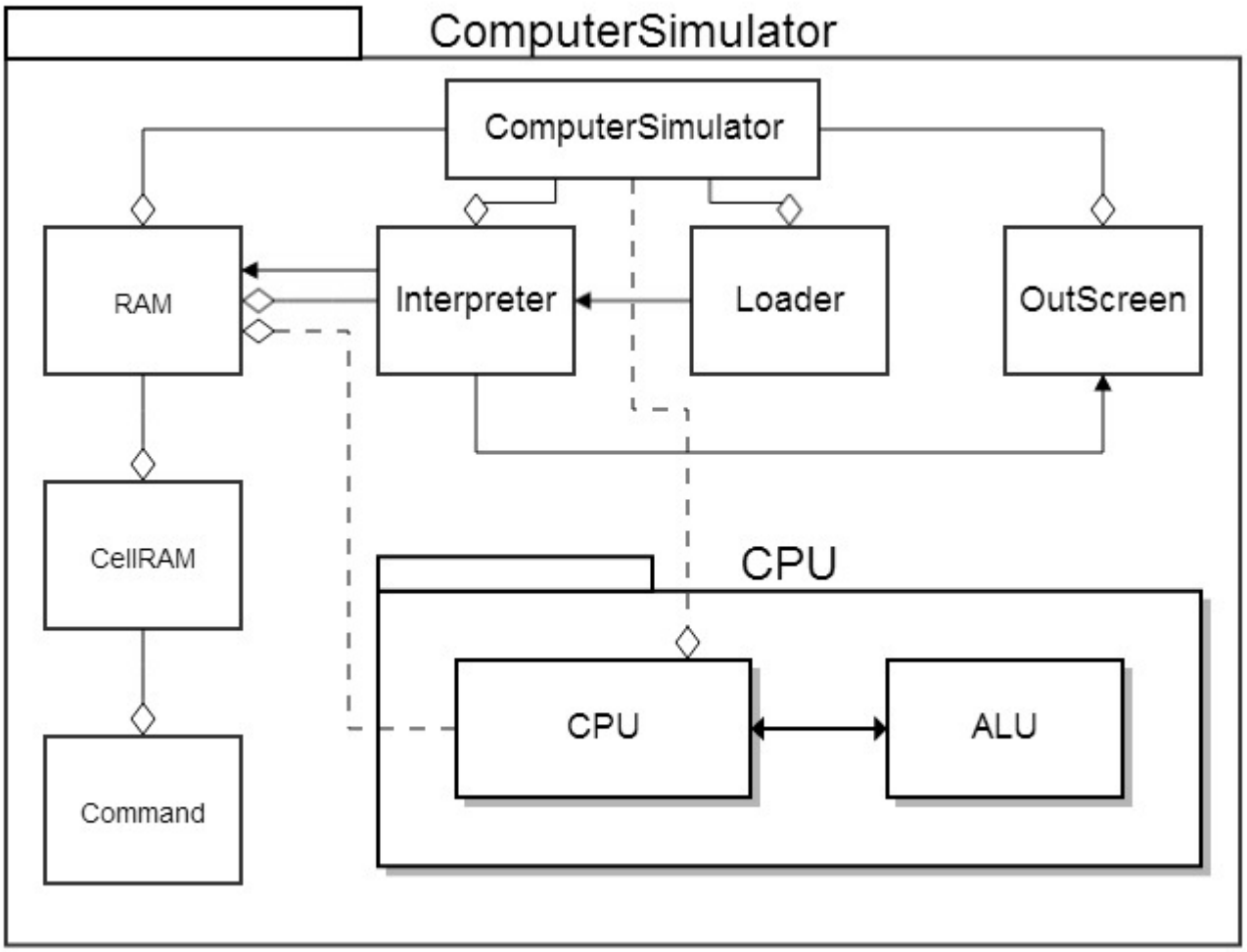
В компьютерную систему входит оперативная память заданного размера и процессор. Процессор включает в себя набор регистров и/или стек (конкретный вариант задается в конфигурации). Они предназначены для быстрого выполнения команд. Задаются команды, которые может выполнять процессор – необходимо придумать форму их записи. Для каждой команды задается код (номер). Коды команд, составляющих программу, записываются в начальные ячейки оперативной памяти. Программа должна обеспечить выполнение программы по шагам.

Визуализатор должен показывать состояние ячеек оперативной памяти и регистров процесса на каждом шаге выполнения программы.

Требования к программе:

- Осуществить возможность пользователю вводить команды в ОЗУ;
- Процессор должен реализовать последовательность команд, записанных в ОЗУ ;
- Должно присутствовать арифметико-логическое устройство;
- Программа должна отображать информацию о ходе выполнения программы на экран;
- Реализовать интерпретатор, который будет сопоставлять адресу ячейки имя переменной;
- Представление процессов, происходящих в системе в графическом интерфейсе;

Объектная модель проекта без GUI



Модель поведения.

Создается код (или выбирается ранее сохранённый). Этот код загружается в транслятор и преобразуется в ячейки оперативной памяти. Далее компьютер загружает себе одну (очередную) ячейку оперативной памяти и выполняет соответствующую команду. Это происходит до тех пор, пока он не обнаружит ячейку с командой о завершении выполнения программы. После трансляции, если пользователь ввел команды неправильно, информация об этом сообщается пользователю на экране вывода. Выполнение команд процессором при некорректных обращениях к памяти прекращается, и информация об этом выводится на экран вывода.

Команды, записываемые в ячейку ОЗУ.

Каждая ячейка ОЗУ состоит из двух частей *<команда>* и *<число>*. Либо *<№ команды>* и *<число>*. Пример «ReadA 1» или эквивалентная запись «01 1». Процессор знает, как реагировать на такие команды.

Список команд ОЗУ.

01 ReadA – считывание регистра А из текущей ячейки.

02 ReadB – то же, только для В.

03 ReadANum – считывание регистра А из адреса памяти, адрес храниться в текущей ячейке.

04 ReadBNum – то же, только для В.

05 WriteANum – запись регистра А в ячейку с адресом, адрес храниться в текущей ячейке.

06 SumAB – значение регистра А после этой команды примет значение суммиры регистров А и В.

07 InvB – значение регистра В после этой команды примет значение противоположного знака.

08 MulAB – значение регистра А после этой команды примет значение произведения регистров А и В.

09 DivAB – значение регистра А после этой команды примет значение частного деления регистра А на В.

10 JumpA0toN – если А>0, то счетчик увеличивается на число, записанное в этой ячейке.

11 data – означает, что эта ячейка выделена для хранения какого-то числа.

12 outA – выводит на экран вывода значение регистра А.

13 outB – выводит на экран вывода значение регистра В.

00 end – при этой команде счетчик перестает меняться – программа останавливается.

Команды интерпретатора.

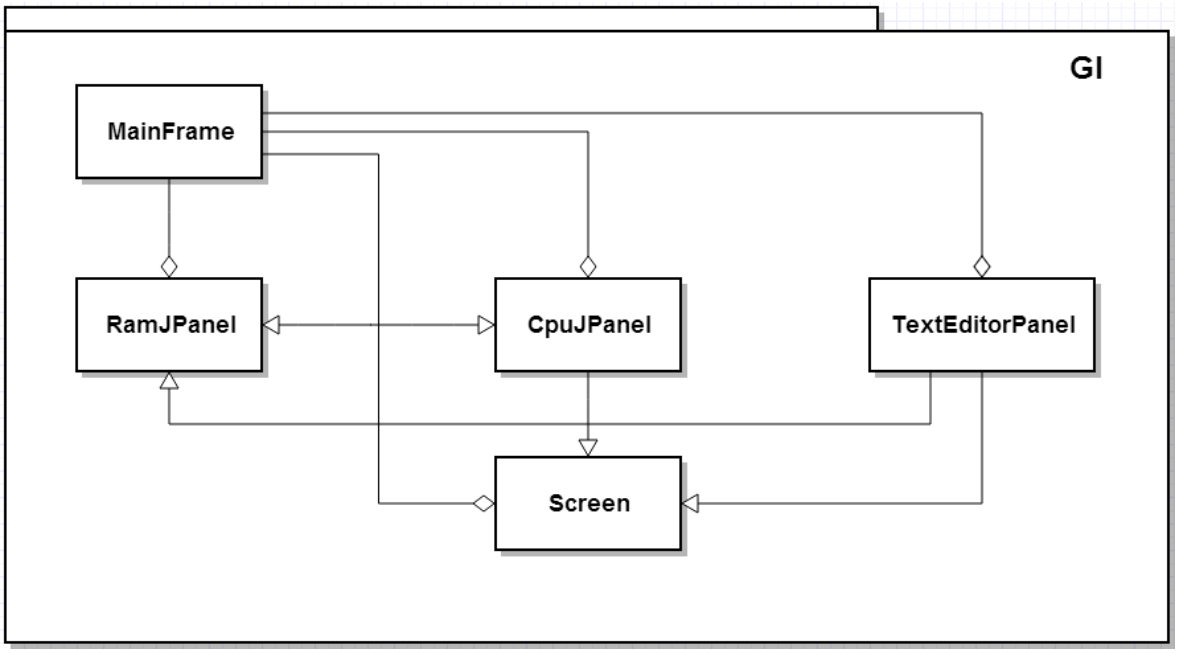
Команды, записанные в форме команд, записываемых в ячейки ОЗУ, интерпретатор оставляет без изменения. Он лишь расширяет список, приведенный выше. Команды интерпретатора, не вошедшие в список приведенный выше, записываются в форме *<команда>* и *<имя переменной>*, либо *<№ команды>* и *<имя переменной>*. Список таких команд приведен ниже.

WriteA – команда создания переменной с заданным именем. Команда сопоставляет ей первый свободный адрес в оперативной памяти. Но только если такая переменная не была создана ранее. Команда записи значения регистра А в переменную. При записи в ОЗУ команда заменяется на *WriteANum*, а число в этой ячейке будет равно адресу, выделенному на переменную.

ReadAVar – команда, считывания переменной по имени в регистр А. При записи в ОЗУ команда заменяется на *ReadANum* , а число в этой ячейке будет равно адресу, выделенному на переменную.

ReadAVar – то же, только для В.

Объектная модель GUI проекта



Класс RamJPanel

Класс, наследник JPanel, который хранит в себе объект класса RAM и представляет собой его графический вид. Массив ячеек объекта RAM отображается как List<JLabel>.

Класс CpuJPanel

Класс, наследник JPanel, который хранит в себе объект класса CPU и представляет собой его графический вид. Отображает текущее состояние регистров процессора и счетчиков.

Класс TextEditirPanel

Класс, наследник JPanel, имеет текстовое поле, с помощью которого вводятся код программы для симулятора. Имеет кнопки (объекты JButton), с помощью которых можно сохранять код в файл и загружать его из файла. Так же имеет кнопку для загрузки кода в ОЗУ. Эта кнопка активна, когда код написан правильно.

Класс MainFrame

Объект класса MainFrame, наследника JFrame, является главным и единственным окном приложения. Он хранит в себе панели (объекты классов приведенных выше) и управляет ими. В его конструкторе инициализируются все эти панели, а так же кнопке объекта класса TextEditorPanel добавляется слушатель (loadRamFromTextPanel), который загружает код в класс объект класса RamJPanel.

Так же класс MainFrame имеет в себе кнопки, управления процессом выполнения программы. Программу можно выполнять пошагово, или сразу всю.

Требования к реализации графического интерфейса программы.

Графический интерфейс должен визуализировать и показать пользователю всё, что описано в требованиях. Основная задача графического интерфейса – минимизировать действия пользователя. Предварительную модель графического интерфейса вы можете увидеть на рис. 1, а модель, реализованную в данном проекте на рис. 2.

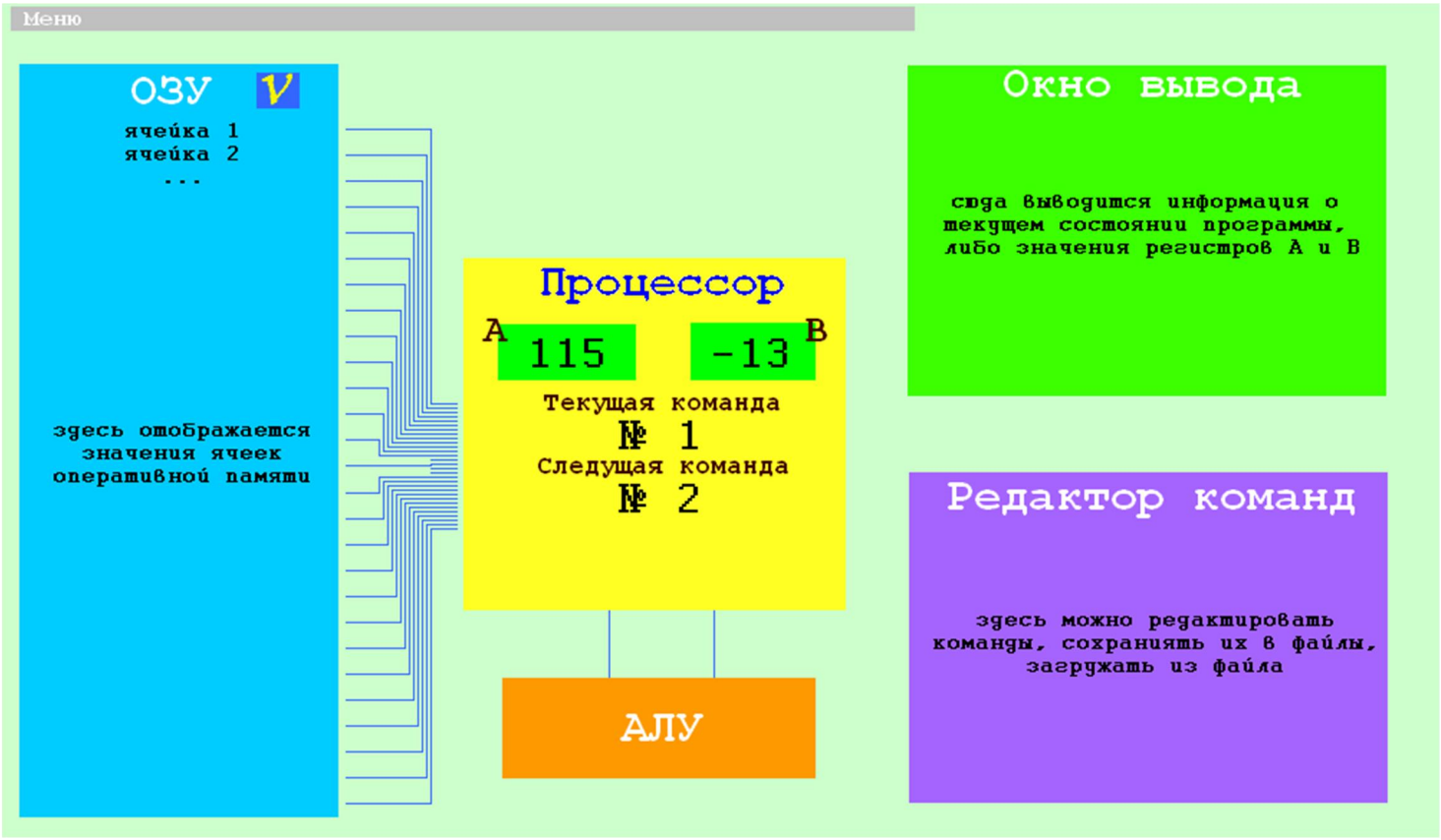


Рис.1 Предварительная модель графического интерфейса.

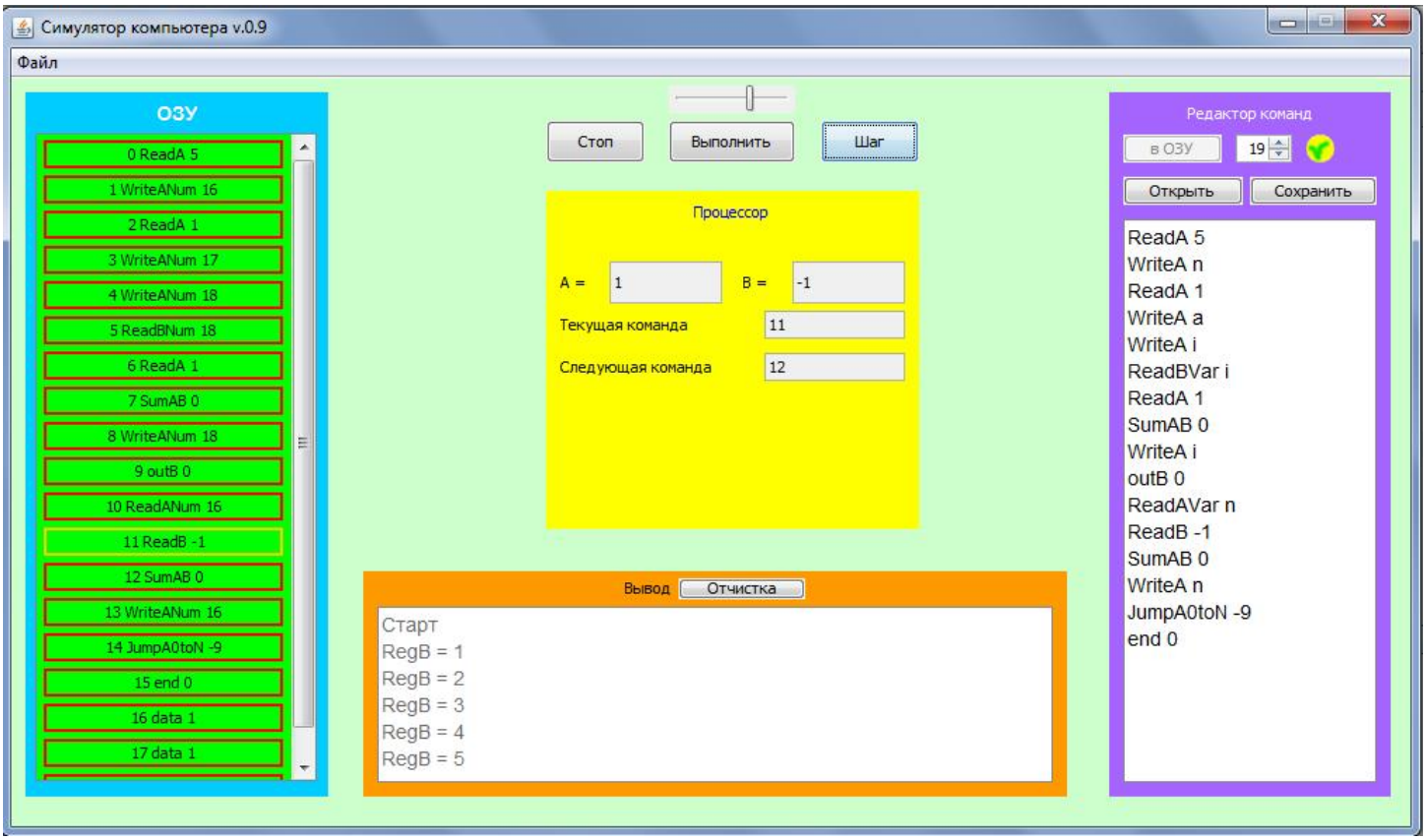


Рис. 2 Реализованный графический интерфейс приложения

На рис. 2 в программу загружен код для симулятора из файла «123…n», содержимое которого вы можете видеть в панели «Редактор команд», и процесс его выполнения. Данный пример иллюстрирует пример написания цикла со счетчиком, на языке симулятора.

Вывод

Как видно из рис. 1 и рис. 2, графический интерфейс был реализован так, как задумывалось, за исключением новых идей, которые возникли в процессе написания, такие как проверка правильности кода для симулятора, и установка скорости выполнения программы симулятором (маленький «ползунок» сверху).

По мере написания данного проекта, были приобретены навыки писания программ на языке Java, и использование графической библиотеки Swing.