Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

ПРОГРАММНЫЙ СИМУЛЯТОР PDP-11

Пояснительная записка

А.В.00001-01 81 01

листов 22

Исполнитель

_____ Тимощук А.А.

«___» _____ 20__ г.

Исполнитель

_____ Штырков В. С.

«___» _____ 20__ г.

Руководитель курсового проекта

_____ Рафиков А. Г.

«___» _____ 20__ г.

Заведующий кафедрой

_____ Басараб М. А.

«___» _____ 20__ г.

Москва, 2023

Аннотация

В данном программном документе приведена пояснительная записка к программе «Программный симулятор PDP-11».

В данном программном документе, в разделе «Введение» указано наименование программы и условное обозначение темы разработки.

В разделе «Назначение и область применения» указано назначение программы и краткая характеристика области применения программы.

В данном программном документе, в разделе «Технические характеристики» содержатся следующие подразделы:

- постановка задачи на разработку программы;
- описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи и возможные взаимодействия программы с другими программами;
- описание и обоснование выбора метода организации входных и выходных данных;
- описание и обоснование выбора состава технических и программных средств на основании проведенных расчетов и анализов.

В разделе «Ожидаемые технико-экономические показатели» указаны технико-экономические показатели, обосновывающие выбранного варианта технического решения, а также, ожидаемые оперативные показатели.

В данном программном документе, в разделе «Источники, использованные при разработке» указан перечень научно-технических публикаций, нормативно-технических документов и других научно-технических материалов, на которые есть ссылки в основном тексте.

Содержание

Основная часть

1 Введение

1.1 Наименование программы

Наименование программы – «Программный симулятор PDP-11».

1.2 Условное обозначение темы разработки

Наименование темы разработки – «Программный симулятор PDP-11».

2 Назначение и область применения

2.1 Назначение программы

Программа предназначена для изучения архитектуры и языка ассемблера ЭВМ PDP-11. Программа предоставляет возможность разрабатывать программы на языке ассемблера, ассемблировать их в машинный код и исполнять машинный код.

2.2 Область применения программы

Программа предназначена к применению в учебных целях для разработки программ на языке ассемблера ЭВМ PDP-11. Эмулятора позволяет разрабатывать, ассемблировать и исполнять программы.

3 Технические характеристики

3.1 Постановка задачи на разработку программы

Для подготовки квалифицированных кадров в области информационных технологий необходимо наличие учебных инструментов, позволяющих изучать работу вычислительных устройств не только современных, но и давно не использующихся. Такой подход помогает проследить развитие технологий и идей, которые лежали в более ранних моделях ЭВМ. В частности ЭВМ PDP-11, отличающейся от своих предшественников уникальным набором режимов адресации, универсальности команд с точки зрения использования операндов при данных режимах адресации, и общая шина, позволяющая расширять возможности ЭВМ путём подключения периферийных устройств.

Для предоставления возможности изучения устройства ЭВМ PDP-11, был разработан программный симулятор, позволяющий разрабатывать, ассемблировать и запускать программы, написанные на ассемблере PDP-11 с возможностью отображения состояния машины в каждый момент времени выполнения программы.

Проект симулятора логически разделён на три составных части, в число которых входит библиотека ассемблера PDP-11 (*Assembler*), библиотека исполнителя PDP-11 (*Executor*) и программа графического интерфейса (*GUI*) предоставляющая пользователю в удобном и интуитивно понятном формате возможность взаимодействовать с перечисленными выше библиотеками, которые в совокупности образуют симулятор ЭВМ PDP-11.

Все три части разработаны на языке C# на платформе .NET6, графический интерфейс разработан с помощью библиотеки Avalonia UI [1].

Программа симулятора может работать на компьютерах под управлением ОС Windows, Linux или MacOS.

3.2 Описание библиотеки ассемблера

3.2.1 Описание функционирования

Основной функцией библиотеки *Assembler* является трансляция программы, написанной на языке ассемблера PDP-11, в машинный код, который далее может быть прочитан и обработан какой-либо программой, поддерживающей формат, определённый в библиотеке.

Функционально библиотека *Assembler* решает те же задачи, что и программа ассемблера на реальном устройстве. В число таких задач входит:

- Трансляция инструкций в машинный код;
- Обработка всех режимов адресации и генерирование соответствующих им дополнительных слов в памяти;
- Расчёт относительных расстояний для инструкций ветвлений;
- Обработка стандартных псевдо-инструкций ассемблера.

3.2.2 Описание алгоритма

На вход библиотеке подается модель проекта IPoject (исходный код модели представлен в Тексте программы), который содержит информацию о путях к файлу с исходным кодом и к выходному объектному файлу.

Вся работа библиотеки *Assembler* разделена на три цикла ассемблирования:

1. Обработка файла с исходным кодом;
2. Обработка строчек кода;
3. Трансляция в машинный код;

На первом цикле ассемблирования файл с исходным кодом читается построчно. Каждая строчка разбивается на три компоненты: список меток, ассоциированных со строкой, исполняемая команда, список аргументов в команде. Эти три компоненты объединяются в класс *CommandLine*, который является абстракцией над одной строчкой кода, и сохраняются в список для последующей обработки.

На втором цикле ассемблирования список строк обрабатывается на отдельные токены. Токен является абстракцией над одним словом машинного кода. На этом цикле идет: расчет количества слов памяти, необходимых командам, расчет адресов меток, определение режимов адресации. Таким образом одна строчка кода может превратиться в множество токенов.

Все токены реализуют интерфейс *IToken*. Библиотека определят следующий токены (исходный код классов токенов и интерфейса представлен в Тексте программы А.В.00001-01 12 01):

1. *OperationToken* – токен, содержащий информацию о исполняемой инструкции и о символе;
2. *MarkRelatedToken* – токен, содержащий информацию об относительном расстоянии до метки;
3. *MarkRelocationToken* – токен, содержащий информацию об абсолютном адресе метки;
4. *ShiftOperationToken* – токен, содержащий информацию о сдвиге;
5. *RawToken* – токен, позволяющий записать любое слово.

Все токены сохраняются в список для последующей обработки.

На третьем цикле ассемблирования токены переводятся в машинный код и записываются в выходной объектный файл, формат которого представлен в листинге 1.

Листинг 1. Формат объектного файла

`000000`	Одно слово машинного кода в восьмеричной системе счисления
`010001;mov r0,r1`	Слово с соответствующим ему символом. Символ – строка исходного кода, соответствующая машинному коду.
`000000’`	Знаком апострофа обозначаются перемещаемые адреса – это адреса, которые зависят от начального адреса программы. Так как ассемблер не знает начальный адрес, то помечает слово апострофом.

3.2.3 Возможные взаимодействия с другими программами

Библиотека *Assembler* предназначена для использования другими программами и предоставляет класс *Compiler* с методом *Compile(IProject)* для ассемблирования исходных файлов на языке ассемблера PDP-11.

3.2.4 Описание и обоснование метода организации входных данных

Входными данными библиотеки являются объект *IProject*, который содержит информацию о путях к файлу с исходным кодом и к выходному объектному файлу. Формат данных обоснован функциональным назначением библиотеки – ассемблирование исходного кода в машинный.

3.2.5 Описание и обоснование метода организации выходных данных

Выходными данными библиотеки является объектный файл с машинным кодом. Формат данных обоснован функциональным назначением библиотеки – ассемблирование исходного кода в машинный.

Также выходными данными могут быть сообщения об ошибках:

- Синтаксические ошибки в исходном коде;
- Ошибки доступа к файлам.

3.3 Описание библиотеки исполнителя

3.3.1 Описание функционирования

Библиотека *Executor* реализует функцию исполнения программ для PDP-11. Для этого решаются следующие задачи:

1. Загрузка программы;
2. Декодирование команд;
3. Исполнение команд;
4. Работа с памятью;
5. Работа с внешними устройствами;
6. Установка флагов состояний в соответствии с логикой реального устройства PDP-11;
7. Обработка прерываний;
8. Обработка аппаратных ловушек.

Библиотека позволяет исполнять команды пошагово (одна команда за раз) или автоматически до достижения точки останова или остановки исполнения (команда HALT).

3.3.2 Описание алгоритма

Библиотека *Executor* предоставляет класс *Executor* с методами *Load*, *ExecuteAsync*, *ExecuteNextInstruction, AddBreakpoint(ushort), RemoveBreakpint(ushort)*. Также *Executor* предоставляет доступ на чтение к машинному коду программы, памяти, состоянию (регистры и слово состояния), списку внешних устройств.

Метод *Load* перезагружает программу из объектного файла и инициализирует все части исполнителя (память, регистры, слово состояния, внешние устройства).

Метод *ExecuteAsync* вызывает в цикле *ExecuteNextInstruction* до остановки исполнения (достижение точки останова или исполнение HALT).

Методы *AddBreakpoint* и *RemoveBreakpoint* устанавливают и удаляют точку останова с адреса соответственно.

Основная логика библиотеки содержится в методе *ExecuteNextInstruction*, блок-схема которого приведена на рисунке (см. рис. 1).

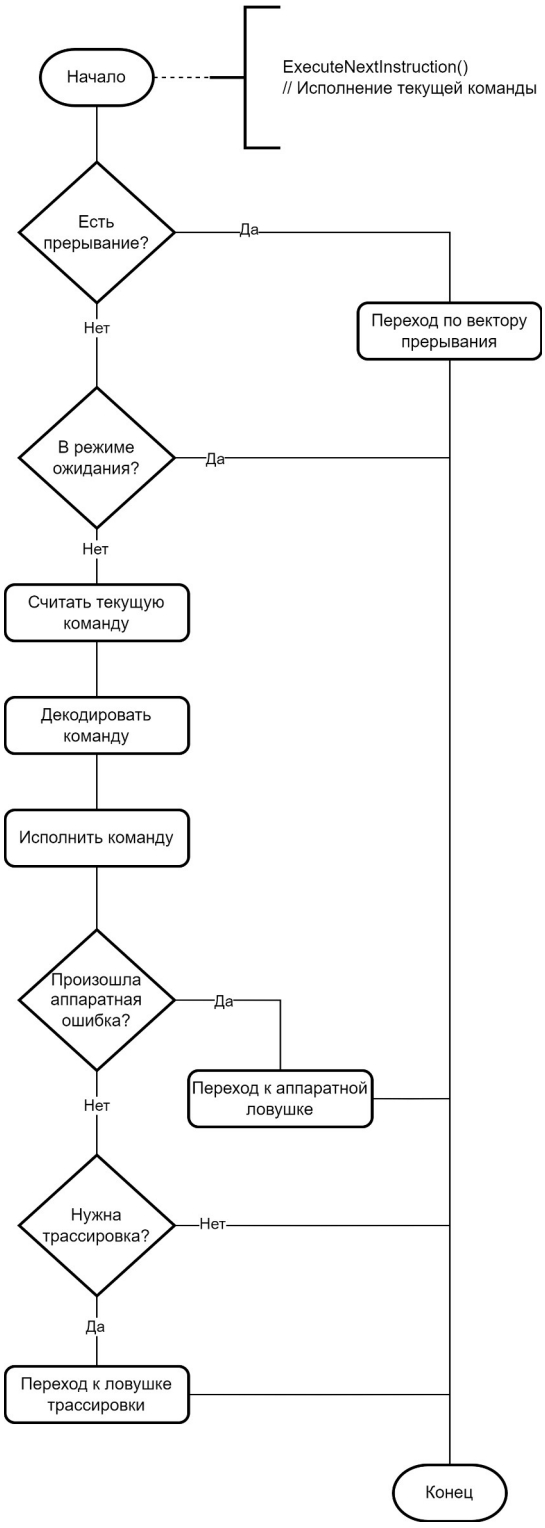

Рисунок 1. Блок-схема метода
исполнения команды

Считывание команды происходит из памяти по адресу, указанному в регистре счетчика команд PC. Далее команда декодируются. Доступны стандартный набор команд [2, стр. 253], расширенный набор команд (EIS) [3, стр. 75-78] и набор команд по работе с плавающей точкой (FIS) [3, 154-155].

Каждая команда реализует интерфейс *ICommand*, который предоставляет следующие методы:

- *OperationCode* – код команды;
- *GetArgumetns(ushort)* – определение аргументов из машинного кода команды;
- *Execute(IArguments[])* -- исполнение команды с данным списком аргументов.

Декодирование команды происходит в два шага: определение команды по ее коду, определение аргументов для команды.

Для уменьшения кода выделены абстракции для команд:

- *BranchOperation* – команды ветвления;
- *TrapInstruction* – команды ловушке;
- *InterruptReturn* – команды возврата из прерываний;
- *OneOperand* – команды с одним аргументом;
- *TwoOperand* – команды с двумя аргументами.

Все аргументы реализуют интерфейс *IArgument*, который содержит два метода: *GetValue()* и *SetValue(object)*. Реализованы следующие аргументы:

- *MarkArgument* – аргумент для инструкции *MARK*, содержит число;
- *SobArgument* – аргумент для инструкции *SOB*, содержит величину сдвига и номер регистра;
- *OffsetArgument* – аргумент для инструкций ветвления, содержит величину сдвига;
- *RegisterWordArgument* – аргумент, использующийся в большинстве команд, содержит номер регистра и режим адресации [2, стр. 61-84];
- *RegisterByteArgument* – аналогичен варианту выше, но работает с байтами, а не словами.

3.3.3 Возможные взаимодействия с другими программами

Исполнитель способен взаимодействовать с внешними устройствами через модуль внешних устройств (детальное описание модуля представлено в

пояснительной записке А.В.00001-01 81 02), которые с точки зрения симулятора являются динамическими библиотеками (DLL).

3.3.4 Описание и обоснование метода организации входных данных

Входными данными библиотеки является объект IProject, который содержит информацию о пути к объектному файлу, начальный адрес указателя стека, начальный адрес программы.

Формат данных обоснован функциональным назначением библиотеки – загрузка программы в память и ее исполнение.

3.3.5 Описание и обоснование метода организации выходных данных

Выходными данными библиотеки являются сообщения об ошибках:

- Нечетные адреса адресов стека и программы;
- Непредвиденная остановка (в случае двойной ошибки шины, например);
- Ошибки доступа к объектному файлу.

Также методы исполнения команд возвращают флаг, показывающий, была ли последняя команда последней (Истина – нет, Ложь – да).

3.4 Описание графического интерфейса

3.4.1 Описание функционирования

Программа графического интерфейса пользователя позволяет (ГИП) работать с библиотеками *Assembler* и *Executor* через удобный интерфейс. Также ГИП позволяет работать с исходными файлами программ и настраивать параметры проекта: начальные адреса программы и стека, список внешних устройств.

Программа решает следующие задачи:

- Работа с файлами исходных кодов;
  - Отображение;
  - Изменение;
  - Сохранение;
  - Удаление;

- Работа с параметрами проекта;
- Работа со списком внешних устройств;
- Работа с библиотекой *Assembler*;
- Работа с библиотекой *Executor*:
  - Отображение регистров и слова состояния;
  - Отображение памяти;
  - Отображение внешних устройств;
  - Настройка точек останова;
  - Отображение команд;
  - Исполнение в пошаговом и автоматическом режимах;
  - Перезагрузка исполнителя.

3.4.2 Описание алгоритма

При запуске ГИП появляется главное окно редактора и окно с запросом создания или открытия проекта (см. рис. 2). При выборе O*pen* запрашивается файл проекта с расширением *pdp11proj*. При выборе C*reate* запрашивается имя проекта (см. рис. 3) и папка, в которой проект будет создан. При выборе C*ancel* программа закрывается.

После открытия/создания проекта открывается главное окно (см. рис. 4), в центре которого расположено поле ввода текста, выше которого список вкладок открытых файлов, выше которого главное меню ГИП.

В разделе *File* содержатся кнопки создания, открытия, сохранения, удаления файлов.

В разделе *Project* находятся кнопки создания и открытия проекта.

Кнопка *Build* передает проект в библиотеку *Assemlber* для ассемблирования.

Кнопка *Settings* открывает окно настроек (см. рис. 5), в котором находятся поля настроек текста (шрифт и размер) и список внешних устройств. Список имеет контекстное меню (открывается на правую кнопку

мыши (ПКМ)) с кнопками добавить, удалить, проверить (проверят внешнее устройство на корректность).

В разделе *Help* находятся кнопки *Tutorial* – открывает окно со справкой, содержащей список команд и информацию о режимах адресации (см. рис. 6) – и *Architecture* – открывает окно со схемой ЭВМ (см. рис. 7).

Кнопка *Run* открывает окно исполнителя (см. рис. 8), в верхней части которого расположены кнопки:

- *Run* – запуск в автоматическом режиме;
- *Step* – исполнение одной команды;
- *Pause* – остановить автоматическое исполнение;
- *Reset* – перезапускает исполнитель.

В центральной части окна расположен код текущей программы, крайний левый столбец позволяет расставлять точки останова.

В правой части расположен блок состояний исполнителя, стоящий из трех вкладок (селектор расположен в верхней части блока):

1. Регистры и слово состояния;
2. Карта памяти (см. рис. 9), которое имеет контекстное меню, позволяющее выбирать вид карты (слова или байты);
3. Список внешних устройств (см. рис. 10).

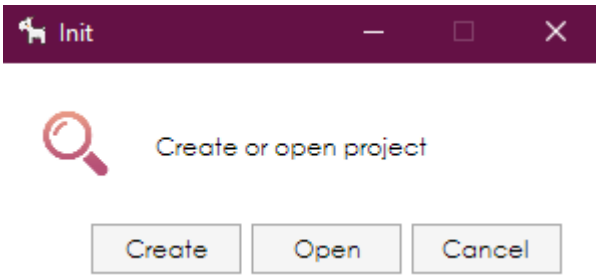

Рисунок 2. Запрос создания/открытия проекта

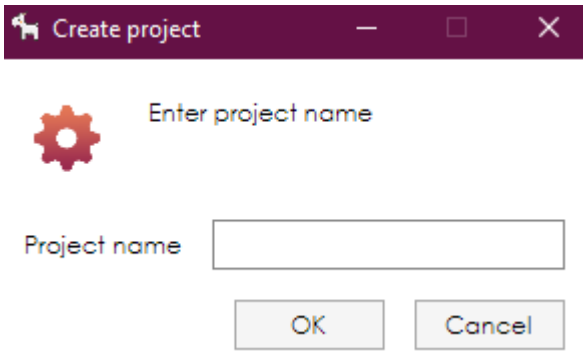

Рисунок 3. Запрос имени проекта

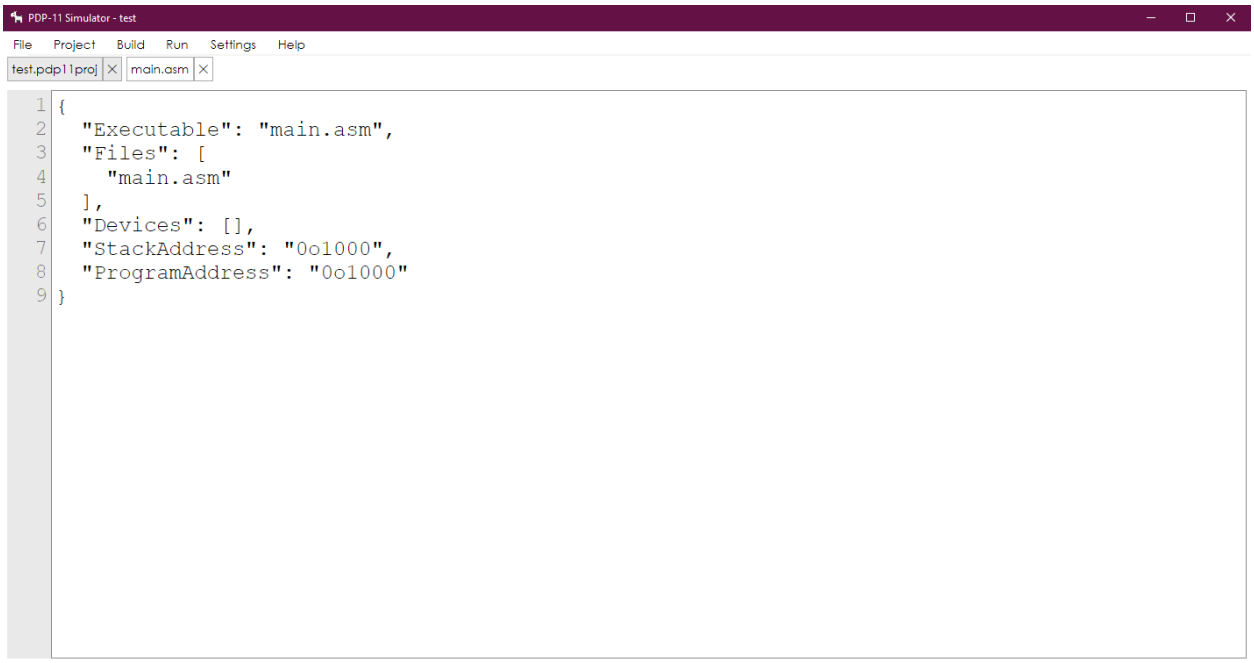

Рисунок 4. Главное окно

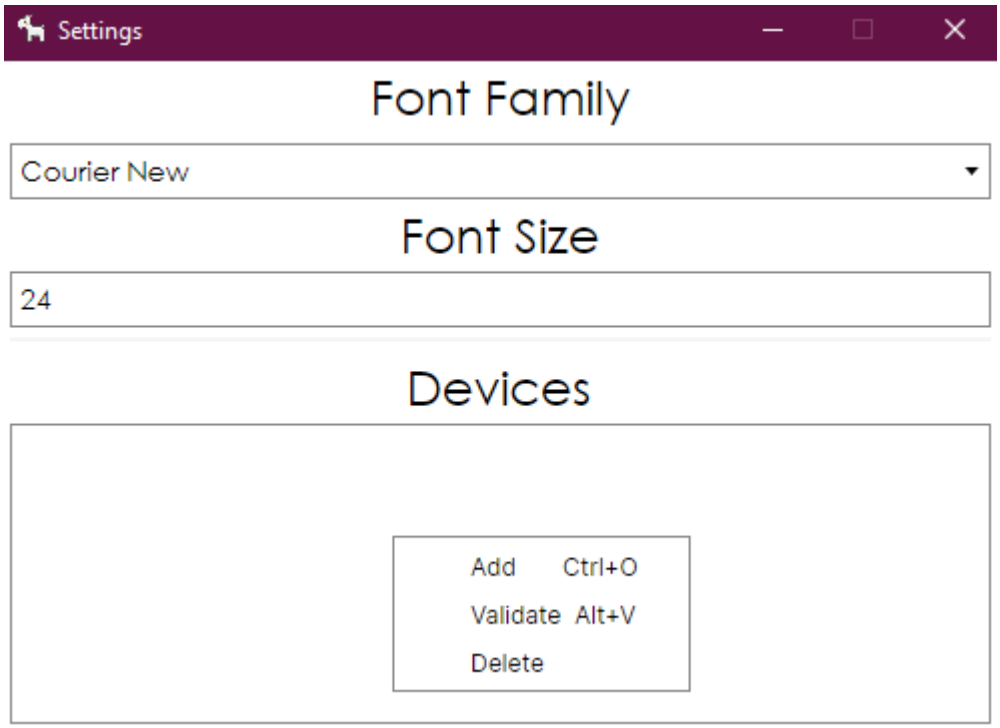

Рисунок 5. Окно настроек

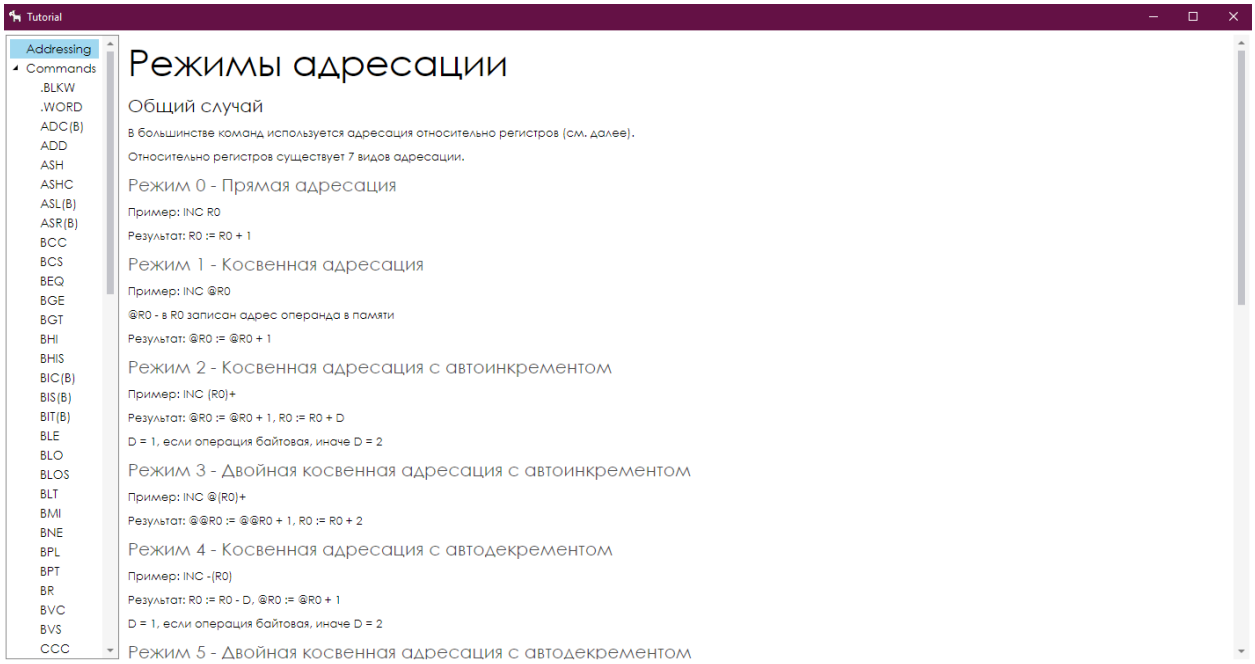

Рисунок 6. Окно справки

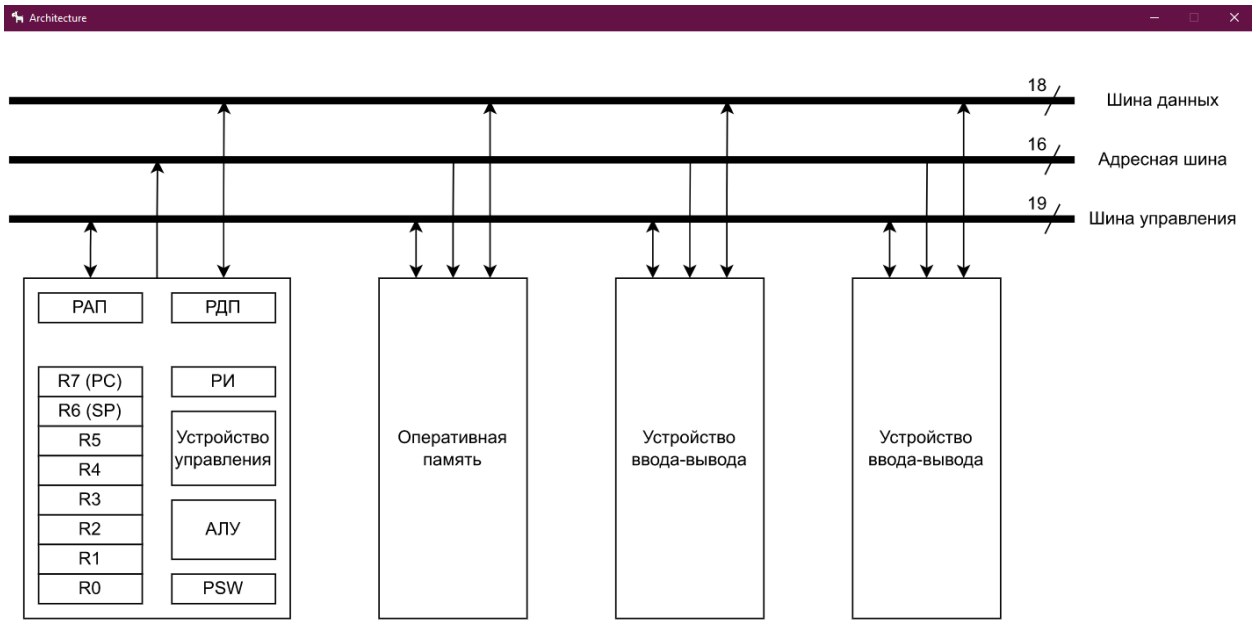

Рисунок 7. Окно со схемой ЭВМ

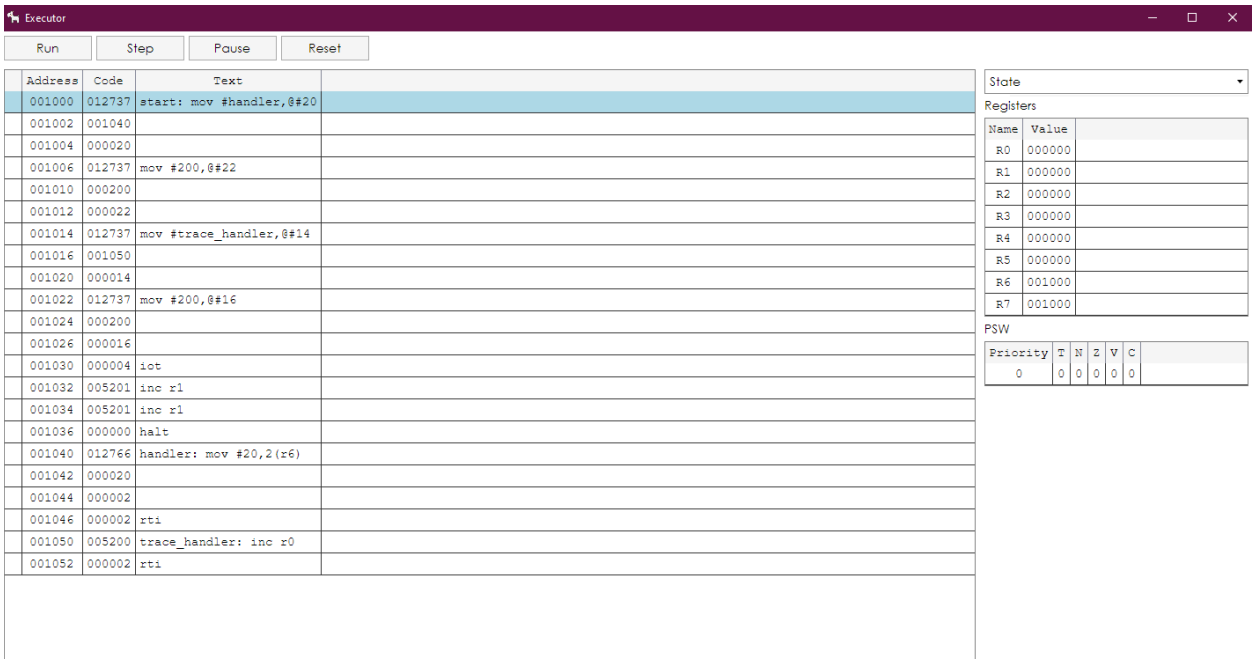

Рисунок 8. Окно исполнителя

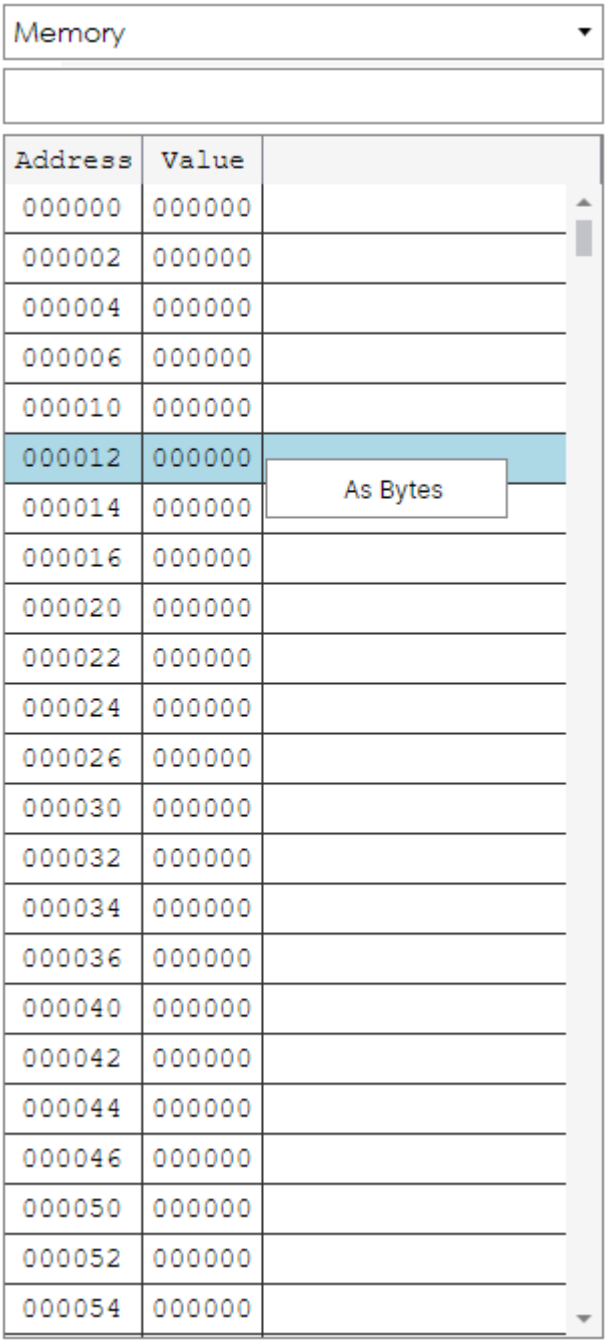

Рисунок 9. Карта памяти окна исполнителя

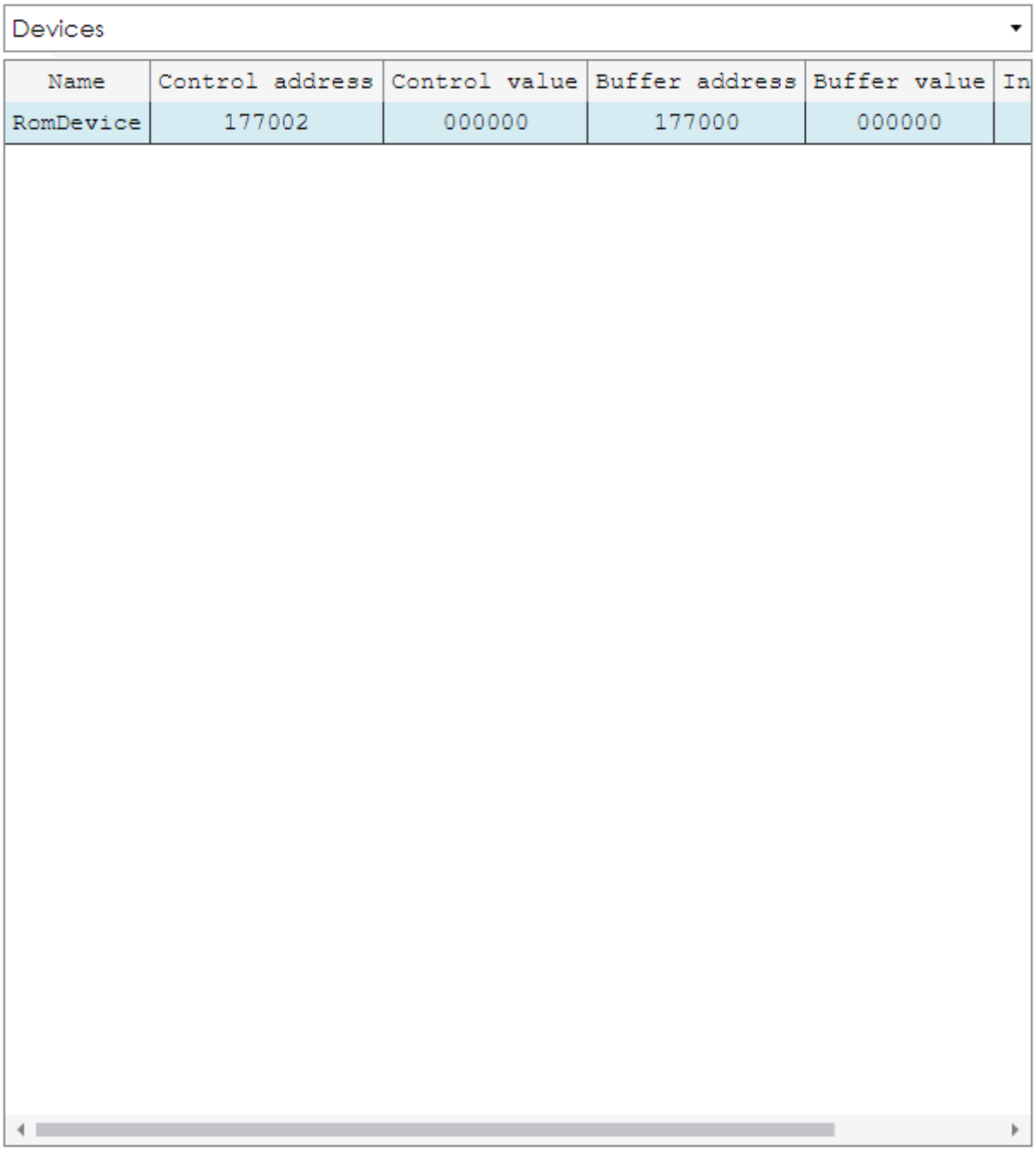

Рисунок 10. Список устройств исполнителя

3.4.3 Возможные взаимодействия с другими программами

ГИП взаимодействует с библиотеками *Assembler* для ассимилирования программ и *Executor* для исполнения программ.

3.4.4 Описание и обоснование метода организации входных данных

Входными данными ГИП являются файл *appsettings.json* с параметрами редактора, файл проекта с расширением *pdp11proj* и файлы с исходными кодами.

Формат данных обоснован назначением программы – отображение содержимого файлов, ассемблирование и исполнение программ.

Также входными данными являются данные, введенные пользователем в процессе работы, а также данные запрошенные у пользователя (например, имена файлов, проектов, пути к файлам).

3.4.5 Описание и обоснование метода организации выходных данных

Выходными данными ГИП являются файл *appsettings.json* с параметрами редактора (при изменении настроек), файл проекта с расширением *pdp11proj* и файлы с исходными кодами при сохранении файлов.

Формат данных обоснован назначением программы – сохранение содержимого файлов.

Также выходными данными являются информационные сообщения об ошибках.

4 Ожидаемые технико-экономические показатели

Программный симулятор PDP-11 позволяет более детально ознакомится с архитектурой ЭВМ PDP-11 путём написания и отлаживания программ под него с возможностью просмотра содержимого всей карты памяти и регистров общего назначений на каждом этапе выполнения программы. Наличие такого инструментария в свободном доступе для учащихся дает хорошее подспорье для повышения общей квалификации специалистов, что в свою очередь ведёт к большему технологическому прогрессу.

5 Источники, используемые при разработке

1 Avalonia UI [Электронный ресурс] // Avalonia UI – URL: https://avaloniaui.net/ (дата обращения: 21.12.2023)

2 Лин В. PDP-11 и VAX-11 Архитектура ЭВМ и программирование на языке ассемблера. М.: «Радио и связь», 1989. – 321 с.

3 PDP-11/40 Processor Handbook. M: Digital Equipment Corporation, 1972. – 212 с.

Лист регистрации изменений

Номера листов (страниц)					Всего листов	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
изм	измененных	измененных	новых	аннулированных					