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

Руководство программиста

А.В.00001-01 33 01

листов 15

Исполнитель, студент группы ИУ8-71

_____ Тимощук А.А.

«___» _____ 20__ г.

Исполнитель, студент группы ИУ8-71

_____ Шаповалов М. Е,

«___» _____ 20__ г.

Исполнитель, студент группы ИУ8-71

_____ Штырков В. С.

«___» _____ 20__ г.

Руководитель курсового проекта,
преподаватель кафедры ИУ8

_____ Рафиков А. Г.

«___» _____ 20__ г.

Заведующий кафедрой ИУ8

_____ Басараб М. А.

«___» _____ 20__ г.

Москва, 2023

Аннотация

В данном программном документе приведено руководство программиста по настройке и использованию программы «Программный симулятор PDP-11».

Содержание

Основная часть

1 Назначение и условия применения программы

1.1 Назначение программы

Программа предназначена для разработки, ассемблирования и исполнения программ ЭВМ PDP-11.

1.2 Функции, выполняемые программой

Программа позволяет:

- Редактировать файлы с исходными кодами программ (создание, изменение, удаление);
- Подключать внешние устройства;
- Ассемблировать программы;
- Исполнять программы;
  - Поддерживается автоматическое и пошаговое исполнение;
  - Поддерживаются точки останова;
- Просматривать состояние симулятора – память, регистры, подключенные устройства, текущая исполняемая команда.

1.3 Условия, необходимые для выполнения программы

1.3.1 Объем оперативной памяти

Устройство, на котором предполагается выполнение программы, должно иметь не менее 2 ГБ ОЗУ.

1.3.2 Требования к составу периферийных устройств

Особых требований не предъявляется.

1.3.3 Требования к параметрам периферийных устройств

Особых требований не предъявляется.

1.3.4 Требования к программному обеспечению

Устройство, на котором предполагается выполнение программы, должно работать под управлением ОС Windows, Linux или MacOS с поддержкой графического интерфейса.

Детальные требования к версиям ОС и установленным библиотекам описаны на следующей странице: https://github.com/dotnet/core/blob/main/release-notes/6.0/supported-os.md

2 Характеристика программы

2.1 Структура программы

Программа состоит из нескольких функциональных блоков:

- Ассемблер – ассемблирует написанные программы;
- Исполнитель – исполняет написанные программы;
- Графический интерфейс – позволяет писать программы, изменять подключаемые устройства, управлять ассемблером и исполнителем.

2.2 Режим работы программы

Особых условий работы не заявлены. Программа запускается и завершается по запросу пользователя.

2.3 Контроль правильности выполнения программы

Работоспособность программы проверяется следующими способами:

1. При вызове исполняемого файла открывается главное окно программы;
2. При запуске программа запрашивает создать или открыть проект;
   a) При создании проекта в файловой системе создается файл проекта и пустой файл с исходным кодом;
   b) При открытии проекта открывается файл проекта и все указанные в нем файлы исходного кода;
3. При работе с файлами происходит корректная работа с файловой системой: файлы создаются, читаются, сохраняются;
4. При изменении настроек шрифта: изменятся шрифт редактора, изменятся файл *appsettings.json*;

5. При добавлении внешних устройств изменятся список внешних устройств, изменяется файл проекта;
6. При сборке проекта появляется окно с результатом;
7. При исполнении проекта открывается окно исполнителя с заполненной программой, состоянием регистров, памяти и внешних устройств;
8. Исполнение программы соответствует поведению реальной ЭВМ PDP-11;

3 Обращение к программе

3.1 Загрузка и установка программы

Для скачивания программы необходимо перейти на следующую страницу: https://github.com/mrypdm/asvt_sem7_kursach/releases?q=simulator . И выбрать версию в соответствие с операционной системой, на которой планируется работа программы.

Для установки нужно извлечь файлы программы по удобному для пользователя пути.

3.2 Запуск программы

Для запуска программы в окно командной строки/терминала необходимо ввести имя исполняемого файла: “pdp11simulator”

3.3 Завершение программы

Для завершения программы необходимо нажать кнопку «Закрыть» – на большинстве ОС представлена иконкой крестика в правом верхнем углу окна.

3.4 Разработка программ

3.4.1 Настройка проекта

После запуска программа запросит создание или открытие проекта. При выборе «создать» программа запросит имя проекта и директорию для создания файлов проекта – будут созданы файл проекта и пустой файл исходного кода. При выборе «открыть» программа запросит файл проекта –

будут открыты файл проекта и все файлы исходного кода, связанные с проектом.

3.4.2 Написание программ

Написание программ происходит в поле ввода в главной части окна. Для работы с файлами в главном меню программы предусмотрена секция «Файл», которая позволяет – создать новый файл, открыть файл, сохранить файл, удалить файл.

3.4.3 Изменение настроек редактора

Для изменения шрифта (имя и размер) необходимо открыть окно настроек (кнопка «Настройки» в главном меню).

3.4.4 Настройка внешних устройств

Для изменения списка подключенных внешних устройств не необходимо открыть окно настроек (кнопка «Настройки» в главном меню).

Список в нижней части окна содержит контекстное меню, которое открывается по нажатий правой кнопки мыши (ПКМ) и которое содержит кнопки «Добавить» – программа запросит файл внешнего устройства – «Удалить» – программа удалит из проекта выделенные устройства – «Проверить» – программа проверит, что внешние устройства корректны.

3.5 Сборка программ

Для сборки программы необходимо нажать кнопку «Собрать» в главном меню. В случае успеха появится информационное окно с сообщением о завершении сборки.

3.6 Исполнение программ

Для исполнения программы необходимо нажать кнопку «Исполнить» в главном меню. При нажатии откроется окно исполнителя, содержащее:

- блок кнопок:
  - Старт – исполнение а автоматическом режиме;
  - Шаг – исполнение в пошаговом режиме;
  - Стоп – остановить исполнение;

- Сброс – сбросить исполнитель;

- текущую программу:
  - первый столбец позволяет устанавливать точки останова;
  - второй содержит адрес команды;
  - третий – машинный код команды;
  - четвертый – соответствующую строчку исходного кода;
- раздел состояния:
  - меню выбора:
    - Регистры;
    - Память;
    - Устройства;
  - блок с информацией, соответствующей выбранной в меню:
    - Блок «Регистры» содержит состояние регистров и слово состояния процессора (PSW);
    - Блок «Память» содержит карту памяти – карта обладает контекстным меню (открывается на ПКМ), позволяющее менять вид карты: отображение по словам или по байтам;
    - Блок «Устройства» содержит информацию о подключенных внешних устройствах.

3.6.1 Настройка указателя стека и адреса начала программы

Для настройки начального значения указателя стека необходимо в файле проекта изменить параметр «StackAddress».

Для настройки начального адреса программы необходимо в файле проекта изменить параметр «ProgramAddress».

3.7 Разработка внешних устройств

3.7.1 Общие сведения

Одной из функций программы является возможность подключить внешние устройства.

Внешнее устройство (ВУ) представляет собой динамическую библиотеку (DLL) и является «драйвером», связывающим симулятор с реальным или виртуальным внешним устройством (далее под ВУ будем понимать именно библиотеку).

Для разработки ВУ разработан SDK на языке программирования C#, состоящий из одного интерфейса (см. листинг 1).

Так как любое внешнее устройство с точки зрения ЭВМ PDP-11 является парой регистров – буфера и управления – которые имеют адреса в том же пространстве, что и основная память, то интерфейс предоставляет эти регистры и их адреса. Также, так как ЭВМ PDP-11 поддерживает прерывания, то интерфейс определяет адрес вектора прерывания и флаг прерывания, который сообщает исполнителю, что внешнее устройство запросило прерывание.

Функция Init() вызывается при старте исполнения или при исполнении команды RESET.

Функция AcceptInterrupt() вызывается исполнителем, когда он принял прерывание в обработку.

Так как ВУ может ссылаться на неуправляемые данные и процессы, то есть на данные, которые находятся за пределами среды CLR, то интерфейс наследует от IDisposable, который реализует всего один метод Dispose(), который позволяет очищать подобные неуправляемые ресурсы.

Листинг 1. Интерфейс внешнего устройства

```
public interface IDevice : IDisposable
{
    /// <summary>
    /// Имя устройства, используется для отображения в окне исполнителя
    /// </summary>
    string Name { get; }

    /// <summary>
    /// Адрес регистра буфера
    /// </summary>
    ushort BufferRegisterAddress { get; }

    /// <summary>
    /// Адрес регистра управления
    /// </summary>
```

```
    ushort ControlRegisterAddress { get; }

    /// <summary>
    /// Адрес вектора прерывания
    /// </summary>
    ushort InterruptVectorAddress { get; }

    /// <summary>
    /// Флаг запроса прерывания
    /// </summary>
    bool HasInterrupt { get; }

    /// <summary>
    /// Значение регистра буфера
    /// </summary>
    ushort BufferRegisterValue { get; set; }

    /// <summary>
    /// Значение регистра управления
    /// </summary>
    ushort ControlRegisterValue { get; set; }

    /// <summary>
    /// Функция инициализации
    /// </summary>
    int Init();

    /// <summary>
    /// Функция обработки прерывания, вызывается исполнителем, чтобы сообщить ВУ,
что он принял прерывание в обработку
    /// </summary>
    void AcceptInterrupt();
}
```

При разработке ВУ рекомендуется:

- Реализовывать всю логику в отдельном потоке (thread) или асинхронной задаче (task), чтобы логика ВУ не исполнялась в потоке исполнителя;
- Работу по инициализации устройства следует проводить в методе Init(), а не в конструкторе;
- В случае, если ВУ ссылается на неуправляемые ресурсы (например, процессы, сокеты, дескрипторы и т. п.), необходимо реализовать очистку таких устройств в методе Dispose() и реализовать деструктор.

Пример ВУ с исходным кодом представлен на следующей странице: https://github.com/mrypdm/asvt_sem7_kursach/tree/master/sdk/ROM

3.7.2 Настройка окружения для разработки

Для разработки ВУ необходим .NET6 SDK и NuGet пакет с ВУ SDK. Последнюю версию SDK можно найти на следующей странице: https://github.com/mrypdm/asvt_sem7_kursach/releases?q=sdk

После настройки SDK создаем директорию, в которой будет вестись разработка (обозначим ее *$ProjectDir$*). Создаем в ней три файла в соответствии с листингами 2-4. Код в листинге 3 является примером реализации ВУ, его следует изменять в соответствии с требуемым от ВУ функционалом.

После разработки следует выполнить команду из листинга 5. В дочерней директории *publish* появятся несколько файлов, один из которых *Device.dll* – этот файл следует добавлять в список ВУ в графическом интерфейсе.

Листинг 2. Файл Device.csproj

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="DeviceSdk" Version="1.0.0" />
  </ItemGroup>

</Project>
```

Листинг 3. Файл Device.cs

```
using System;
using DeviceSdk;

namespace Device;

public sealed class DemoDevice : IDevice
{
    public string Name => GetType().Name;

     public ushort BufferRegisterAddress => Convert.ToUInt16("177562",
8);

    public ushort ControlRegisterAddress => Convert.ToUInt16("177560",
```

```
8);

    public ushort InterruptVectorAddress => Convert.ToUInt16("60", 8);

    public bool HasInterrupt => false;

    public ushort BufferRegisterValue { get; set; }

    public ushort ControlRegisterValue { get; set; }

    public int Init()
    {
        BufferRegisterValue = 0;
        ControlRegisterValue = 0;
        return 0;
    }

    public void AcceptInterrupt()
    {
    }

    public void Dispose()
    {
    }
}
```

Листинг 4. Файл NuGet.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <packageSources>
    <add key="Current Project Local Repository" value="$ProjectDir$"/>
  </packageSources>
</configuration>
```

Листинг 5. Сборка ВУ

```
dotnet publish -o ./publish
```

4 Входные и выходные данные

4.1 Сведения о входных данных

Входными данными программы являются:

- файл *appsettings.json*, содержащий настройки шрифта окна редактора; данный файл расположен рядом с исполняемым файлом;
- файлы проектов с расширением *pdp11proj* – при открытии, сборке, исполнении проекта;

- фалы с исходным кодом на языке ассемблера – при открытии файлов, сборке проекта;
- объектные файлы программ с расширением *pdp11bin* – при исполнении программы.

4.2 Сведения о выходных данных

Выходными данными программы являются:

- файл *appsettings.json*, содержащий настройки шрифта окна редактора – при изменении настроек шрифта;
- файлы проектов с расширением *pdp11proj* – при создании проекта, изменении проекта;
- файлы с исходным кодом – при создании, изменении файлов;
- объектные файлы программ с расширением *pdp11bin* – при ассимилировании проекта.

5 Сообщения

Программа может выдавать сообщения нескольких видов:

- Информационные:
  - Об успехе ассемблирования;
  - При завершении исполнения программы;
- Предупреждающие:
  - При открытии нового проекта;
  - При закрытии программы и наличии не сохраненных файлов;
  - При открытии ранее открытого файла;
- Об ошибках:
  - При попытке закрыть файл проекта;
  - При добавлении некорректного внешнего устройства;
  - При ошибках исполнения программы;
  - При ошибках ассемблирования программы.

Лист регистрации изменений

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