**САНКТ–ПЕТЕРБУРГСКОЕ ГОСУДАРСТВЕННОЕ**

**БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»**

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

УП01

Специальность 09.02.07

«Информационные системы и программирование»

Квалификация «Программист»

Руководитель учебной практики:

Смирнова И. П.

Другие преподаватели:

Матысик И. А.

Полякова А. Н.

Фомин А. В.

Выполнил студент группа 493:

сидоров антон дмитриевич

Санкт–Петербург, 2023

**СОДЕРЖАНИЕ**

[1 УП01.01. Разработка программных модулей 3](#_Toc129203152)

[1.1 Техническое задание 3](#_Toc129203153)

[1.2 Общая часть 6](#_Toc129203154)

[1.2.1 Цель разработки 6](#_Toc129203155)

[1.2.2 Описание предметной области 6](#_Toc129203156)

[1.2.3 Анализ средств и технологий 7](#_Toc129203157)

[1.3 Проектная часть 8](#_Toc129203158)

[1.3.1 Постановка задачи 8](#_Toc129203159)

[1.3.2 Функционально-логическая структура проекта 13](#_Toc129203160)

[1.3.2.1 Use-case диаграмма 13](#_Toc129203161)

[1.3.2.2 Диаграмма деятельности прецедента «добавление товара» 14](#_Toc129203162)

[2 УП01.02. Поддержка и тестирование программных модулей 15](#_Toc129203163)

[2.1 Неделя 1 15](#_Toc129203164)

[2.1.1 Задание расчёт сырья 15](#_Toc129203165)

[2.1.2 Unit–tests 16](#_Toc129203166)

[2.1.3 Test-Cases 27](#_Toc129203167)

[2.2 Неделя 2 28](#_Toc129203168)

[2.2.1 Предметная область 28](#_Toc129203169)

[2.2.2 Описание программного продукта 28](#_Toc129203170)

[3 УП01.03. Разработка мобильных приложений 29](#_Toc129203171)

[3.1 Цель работы 29](#_Toc129203172)

[3.2 Вариант задания 29](#_Toc129203173)

[3.3 Макеты окон 29](#_Toc129203174)

[3.4 Диаграмма базы данных 32](#_Toc129203175)

[3.5 Описание API 33](#_Toc129203176)

[3.6 Программный код 37](#_Toc129203177)

[4 УП 01.04. Системное программирование 39](#_Toc129203178)

[4.1 Лабораторная работа №1. Изучение архитектуры МП 39](#_Toc129203179)

[4.2 Лабораторная работа №2. Исследование команд прямой адресации 43](#_Toc129203180)

[4.3 Лабораторная работа №3. Исследование команд непосредственной адресации 45](#_Toc129203181)

[4.4 Лабораторная работа №4. Исследование команд косвенной адресации 48](#_Toc129203182)

[4.5 Лабораторная работа №5. Исследование команд стековой адресации 50](#_Toc129203183)

[4.6 Лабораторная работа №6. Пример программы для микропроцессора 52](#_Toc129203184)

[4.7 Лабораторная работа №7. Программа сложения двух однобайтных чисел X и У 56](#_Toc129203185)

[4.8 Лабораторная работа №8. Программа вычитания двух однобайтных чисел X и У 59](#_Toc129203186)

[4.9 Лабораторная работа №9. Сложение массива однобайтных чисел 61](#_Toc129203187)

[4.10 Лабораторная работа №10. Сложение двухбайтовых десятичных чисел 63](#_Toc129203188)

[4.11 Лабораторная работа №11. Вычитание одинаковых по длине чисел 66](#_Toc129203189)

[4.12 Лабораторная работа №12. Изучение алгоритма ветвления 68](#_Toc129203190)

[4.13 Лабораторная работа №13. изучение возможностей вывода информации на экран 72](#_Toc129203191)

[4.14 Лабораторная работа №14. изучение возможностей МП для умножения целых чисел без знака 76](#_Toc129203192)

[4.15 Лабораторная работа №15. изучение алгоритмов деление двух однобайтных чисел, составление и выполнение программы деления двух однобайтных чисел без знака. 79](#_Toc129203193)

[4.16 Лабораторная работа №16. Программирование арифметических задач на Ассемблере для микропроцессора К580 82](#_Toc129203194)

[4.16.1 Задание 1. Команды логического умножения 82](#_Toc129203195)

[4.16.2 Задание 2. Команды обнуления битов 84](#_Toc129203196)

[4.16.3 Задание 3. Команды логического сложения 86](#_Toc129203197)

[4.16.4 Задание 4. Команды восстановления битов 88](#_Toc129203198)

[4.16.5 Задание 5. Инвертирование чисел 90](#_Toc129203199)

[4.16.6 Задание 6. Инвертирование флага переноса 91](#_Toc129203200)

[4.16.7 Задание 7. Инвертирование заданных битов числа 93](#_Toc129203201)

[4.16.8 Задание 8. Сравнение чисел 95](#_Toc129203202)

# УП01.01. Разработка программных модулей

## Техническое задание

* + 1. **Общие сведения**

Название темы проекта – *Информационная система «Магазин музыкальных инструментов ООО “Ритм”».*

Данное предприятие занимается продажей музыкальных инструментов, музыкального оборудования, музыкального пособия и иных товаров, связанных с музыкой.

На данном предприятии работают Директор, системный администратор, менеджеры по заказам и по складу, операторы и продавцы.

Покупатели заказывают товары. Если этот процесс происходит по телефону, то оформлением заказа занимается *оператор*, а если на кассе, то *Менеджер по заказам*. Выдаёт заказ покупателю *продавец*. Товары хранятся на складе, и список товаров, вместе с категориями, производителями и поставщиками составляет *менеджер по складу*. Обслуживает оборудование на предприятии *Системный администратор*. Главный на предприятии – *Директор*. Директор, касаемо данной информационной системы, может нанимать работников и увольнять их, а также отправлять их в отпуск/командировку.

* + 1. **Назначение разработки**

Информационная система должна автоматизировать оформление заказов. Также, Информационная система, также, должна будет упростить процедуру оформления и выдачи заказов.

* + 1. **Требования к программе или программному изделию**
       1. **Требования к функциональным характеристикам**

Каждый из пользователей имеет возможность авторизироваться в системе, а покупатель, также зарегистрироваться в системе. Также, каждый пользователь может менять пароль от своей учётной записи. Каждый пользователь имеет одну или несколько ролей из ниже перечисленных (кроме, «*Гость*»).

*Гость* и авторизированный *покупатель* имеют, почти одинаковый функционал. *Гость* может просматривать список товаров и формировать заказ. В каталоге товаров *гость* может сортировать товары по категориям и скидкам, а также выполнять поиск товаров. Авторизированный *покупатель* имеет те же возможности, на также, может просматривать свои заказы. Обработка заказов далее переходит к *менеджеру по заказам*.

*Менеджер по складу* может просматривать список товаров, добавлять туда новые товары, удалять товары из каталога, а также, редактировать их количество.

*Оператор* может формировать и корректировать заказ. Обработка заказов далее переходит к *менеджеру по заказам*.

*Менеджер по заказам*, также, как и *оператор* может формировать и корректировать заказ. Также, он может удалять и редактировать заказы, в том числе, изменять их статус.

*Продавец* может просматривать список заказов, выполнять их поиск (в том числе, по номеру), а также, отмечать о выдаче заказа.

*Директор* может просматривать список пользователей в системе. Также, может добавлять и удалять пользователей, а также блокировать их.

*Системный администратор* имеет все возможности и функции выше перечисленных ролей, а также заходить под любой ролью.

* + - 1. **Требования к надёжности**

Пользователю должно быть позволено вводить, только корректные данные. Должны выводиться сообщения об ошибках. Некорректные данные не должны отправляться в систему.

Авторизация должна проходить успешно только после правильного ввода логина и пароля. При создании учётной записи без пароля, а также при удалении пароля без установки нового должно выводиться предупреждение об уязвимости учётной записи при отсутствии пароля. Также пользователь может входить в систему, если он имеет хотя бы одну роль, и иметь функционал, который прописан для данной роли. Пользователи, не имеющие ролей не могут входить в систему. Если пользователь сам регистрируется в системе, то он получает роль «*Покупатель*».

Пользователи должны использовать свой функционал из программы/приложения, не заходя в базу данных.

* + - 1. **Требования к информационной и программной совместимости**

Программа должна разрабатываться на языке *C#* в *Visual Studio 2017* или *Visual Studio 2019*, за исключением триггеров и хранимых процедур в базе данных.

База данных должна храниться на сервере и разрабатываться в среде *SQL Management Studio 18*. Язык для скриптов базы данных, а также для триггеров и хранимых процедур *Transact SQL*. При разработке базы данных допустимо использовать любые средства (в том числе визуальные) среды разработки, а также подключаться к базе данных через *Microsoft Access*.

* + 1. **Требования к программной документации**

Перечень программной документации:

* Руководство пользователя
* Тест-кейсы
  + 1. **Стадии и этапы разработки**

1. Формирование требований;
2. Проектирование;
3. Реализация;
4. Тестирование;
5. Внедрение;
6. Эксплуатация и сопровождение.

## Общая часть

## Цель разработки

Разработать информационную систему, автоматизирующую работу предприятия

## Описание предметной области

Название темы проекта – *Информационная система «Магазин музыкальных инструментов ООО “Ритм”».*

Данное предприятие занимается продажей музыкальных инструментов, музыкального оборудования, музыкального пособия и иных товаров, связанных с музыкой.

На данном предприятии работают Директор, системный администратор, менеджеры по заказам и по складу, операторы и продавцы.

Покупатели заказывают товары. Если этот процесс происходит по телефону, то оформлением заказа занимается оператор, а если на кассе, то Менеджер по заказам. Выдаёт заказ покупателю продавец. Товары хранятся на складе, и список товаров, вместе с категориями, производителями и поставщиками составляет менеджер по складу. Обслуживает оборудование на предприятии Системный администратор. Главный на предприятии – Директор. Директор, касаемо данной информационной системы, может нанимать работников и увольнять их, а также отправлять их в отпуск/командировку.

Исходя из выше описанного, в информационной системе имеются следующие роли:

1. Менеджер по складу – Редактирует каталог товаров и категории товаров в базе данных: просматривает, добавляет, редактирует и удаляет товары, поставщиков, производителей и категории в каталоге, а также изменяет количество товаром.

2. Менеджер по заказам – Работает с заказами: просматривает, добавляет, редактирует и удаляет заказы, а также изменяет статус заказа.

3. Оператор – Вносит заказ в базу данных, получив его по телефону.

4. Продавец – Выдаёт заказ покупателю

5. Покупатель – Единственная роль, с возможностью зарегистрироваться самостоятельно в системе. Может просматривать список товаров и сделанных заказов, а также создавать свой заказ.

6. Директор – Может работать с пользователями в системе: Добавлять, удалять, блокировать, редактировать, просматривать список.

7. Системный администратор – Имеет все права в системе, что и Директор, менеджер по складу и менеджер по заказам.

Информационная система имеет одну базу данных. Для доступа в информационную систему необходима авторизация с помощью логина и пароля, но в данной системе присутствует, также Гость, который может заходить без авторизации, и имеет все права, что и покупатель, кроме просмотра списка заказов (для этого необходима авторизация).

## Анализ средств и технологий

Для разработки информационной системы используется следующее программное обеспечение:

1. *Visual Studio 2019*
2. SQL Management Studio 18

## Проектная часть

## Постановка задачи

Требуется разработать информационную систему (ИС) ООО «Ритм» – магазин по продаже музыкальных инструментов. Для этого требуется разработать основные программные модули ИС для следующих пользователей:

* *неавторизованный клиент (гость)* и *авторизованный клиент (покупатель)* может просматривать товары и информацию о конкретном товаре, а также формировать заказ и просматривать свои заказы (для гостя в последних 2 действиях, необходима авторизация/регистрация);
* *Гость* (Является неавторизированным пользователям) может зарегистрироваться и получить аккаунт покупателя;
* *менеджер по заказам* может просматривать товары, формировать и редактировать заказы, работать с клиентами;
* *Менеджер по складу* может добавлять/редактировать/удалять товары, просматривать и редактировать заказы, создавать отчеты по работе магазина;
* *Продавец* (также, кассир) может формировать заказы в магазине и изменять статус заказа, сделанного в том же магазине;
* *Оператор* работает с клиентами по телефону/Email, формирует заказы;
* *Директор* работает с учётными записями сотрудников: добавляет, удаляет, блокирует и разблокирует;
* *Администратор* имеет все права выше перечисленных ролей (кроме гостя и покупателя) и входить под этими ролями, а также работать с пользователями (добавлять, редактировать, удалять, заблокировать и разблокировать).
  + - 1. **Модуль №1. Анализ и проектирование бизнес-процессов**

1. Создать диаграмму прецедентов (Use Case) для пользователей системы.
2. Разработайте диаграмму деятельности для прецедента «Добавление товара».

**1.3.1.2 Модуль №2. Проектирование реляционного хранилища данных**

1. Спроектировать ER-диаграмму для ИС. Обязательна 3 нормальная форма с обеспечением ссылочной целостности.
2. Создать словарь данных – набор информации, описывающий, какой тип данных хранится в БД, их формат, структуру и способы использования данных.

**1.3.1.3 Модуль №3. Разработка баз данных, объектов баз данных и импорт**

1. Создать БД ООО «Ритм» на сервере баз данных. Создайте таблицы основных сущностей, атрибуты, отношения и необходимые ограничения.
2. Подготовьте данные для импорта и загрузите в разработанную базу данных.

**1.3.1.4 Модуль №4. Разработка desktop-приложений**

**1.3.1.4.1 Авторизация**

При запуске приложения окно входа – первое, что видит пользователь. На ней пользователю предлагается ввести свой логин и пароль или есть возможность перейти на экран просмотра товаров в роли гостя. Только после удачной авторизации пользователь получает доступ к остальным модулям системы.

**1.3.1.4.2 Список товаров**

Необходимо реализовать вывод товаров, которые хранятся в БД, согласно предоставленному макету. При отсутствии изображения необходимо вывести картинку-заглушку из ресурсов (picture.png).

Вывод должен осуществляться списком. Каждая строка с товаром в списке должна совпадать с макетом, представленным на рисунке 1.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Макет строки с товаром.

По кнопке «Просмотреть информацию о товаре» осуществляется переход на окно с информацией о товаре:

* Наименование товара;
* Категория (с учётом подкатегорий);
* Производитель;
* Поставщик;
* Наличие на складе/в магазине (при наличии – количество на складе);
* Цена;
* Скидка;
* Цена со скидкой;
* Параметры товара;
* Описание товара.

В этом окне, также должна быть для покупателя и гостя должна быть кнопка добавления в заказ, а также возможность изменения количества в заказе (при количестве, равном нулю, товар из заказа удаляется). Добавление товара в заказ может проводиться из формы просмотра товаров.

Пользователь должен иметь возможность искать товары по названию, сортировать товары и по их стоимости (по возрастанию, убыванию и без сортировки).

Кроме этого, пользователь должен иметь возможность отфильтровать данные по размеру скидки (диапазоны скидок для фильтрации указываются в зависимости от исходных данных). Первым элементом в выпадающем списке должно быть значение “Все диапазоны”, при выборе которого настройки фильтра сбрасываются.

Кроме этого, пользователь должен иметь возможность отфильтровать данные по категории. Выбор категории должен быть из выпадающего списка. Первым элементом в выпадающем списке должен быть “Все категории”, при выборе которого настройки фильтра сбрасываются.

Все фильтры должны работать совместно и без дополнительных кнопок фильтрации.

В верхней части окна необходимо показывать количество выведенных данных и общее количество записей в базе.

* + - * 1. **Добавление/редактирование товаров**

Необходимо реализовать возможность добавления и редактирования исходных товаров. Редактировать данные может только *менеджер по складу* и *администратор*.

Редактирование и добавление товара выполняется в новом окне. Переходы на данное окно должны быть реализованы из формы списка: для редактирования – при нажатии на кнопку у конкретного элемента, для добавления – при нажатии кнопки “Добавить товар”.

ID товара при добавлении не отображается, автоматически вычисляется +1 к имеющемуся в БД, при редактировании ID доступно только для чтения.

Пользователь может добавить/заменить изображение товара.

В окне редактирования товара должна присутствовать кнопка “Удалить”, которая удаляет товар из базы данных. Товар, который присутствует в заказе, удалить нельзя.

После редактирования/добавления/удаления товаров данные в окне списка товаров должны быть обновлены.

* + - * 1. **Формирование заказа**

Реализуйте возможность формирования заказа из окна просмотра списка товаров для авторизованных пользователей и для гостя ООО «Ритм».

Гость перед подтверждением должен, сначала пройти процедуру авторизации или регистрации.

Из авторизированных пользователей заказ может формировать покупатель, оператор, менеджер по заказам, а в магазине, также – продавец.

При просмотре списка товаров пользователь может выделить товар, по нажатию правой клавиши мыши вызвать контекстное меню с командой «Добавить к заказу» или кнопкой у данного товара. При нажатии на команду происходит формирование заказа и добавление в заказ выбранной позиции в количестве 1 единицы.

Если хотя бы один товар добавлен к заказу, в интерфейсе появляется кнопка для просмотра заказа. При просмотре товаров все добавления производятся в один заказ.

При просмотре заказа отображается вся информация о товарах, в том числе изображение. Реализована возможность удалить товар, указав количество 0 или нажатием на элемент интерфейса. При формировании заказа:

• ведется подсчет суммы заказа, подсчет скидки заказа, если в нем есть товары со скидкой;

• заказу автоматически присваивается номер (+1 к существующему в БД);

• информация о заказе хранится в БД.

Формируется чек к заказу с возможностью сохранения в pdf и его дальнейшей печати. Чек содержит: состав заказа, сумму заказа, сумму скидки.

Если заказ формирует авторизованный клиент, в окне просмотра заказа отображается ФИО клиента.

**1.3.1.5 Модуль №5. Руководство пользователя**

Необходимо разработать руководство пользователя для приложения, которое описывает последовательность действий для выполнения всех функций вашей системы.

## Функционально-логическая структура проекта

## Use-case диаграмма

Диаграмма деятельности представлена на рисунке 2.

Изображение выглядит как карта

Автоматически созданное описание

Рисунок 2 – Диаграмма деятельности

## Диаграмма деятельности прецедента «добавление товара»

Диаграмма деятельности прецедента представлена на рисунке 2.

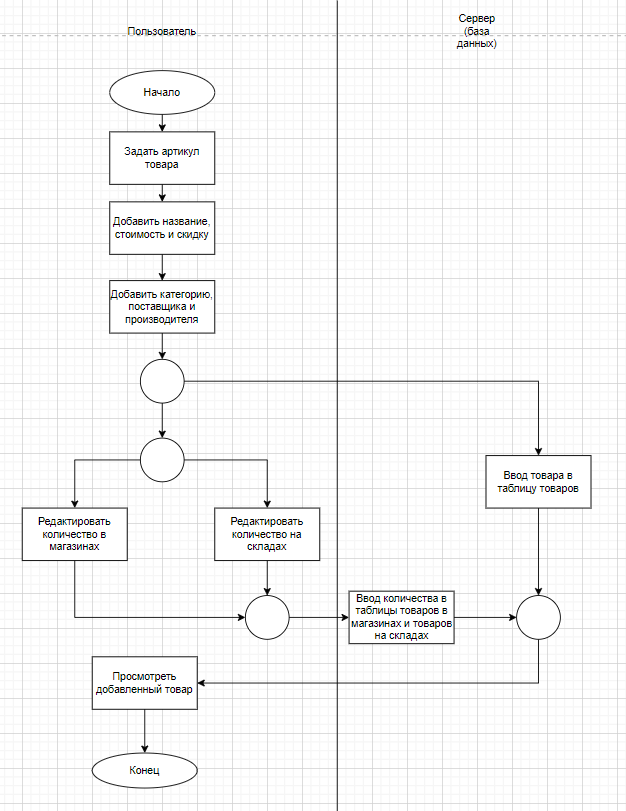


Рисунок 3 – Диаграмма деятельности прецедента

# УП01.02. Поддержка и тестирование программных модулей

## Неделя 1

## Задание расчёт сырья

2.1.1.1 Задание

Для того чтобы в производстве могли быстро и одинаково рассчитывать количество необходимого сырья для производства той или иной продукции, необходимо разработать библиотеку классов.

Чтобы система правильно интегрировалась вам необходимо обязательно следовать правилам именования библиотек, классов и методов в них. В случае ошибок в рамках именования ваша работа не может быть проверена и ваш результат не будет зачтен. Классы и методы должны содержать модификатор public (если это реализуемо в рамках платформы), чтобы внешние приложения могли получить к ним доступ.

В качестве названия для библиотеки необходимо использовать: WSUniversalLib. Вам необходимо загрузить исходный код проекта с библиотекой в отдельный репозиторий с названием, совпадающим с названием проекта.

2.1.1.2 Спецификация метода

Метод должен принимать идентификатор типа продукции, идентификатор типа материала, количество необходимой продукции для производства, ширину продукции и длину продукции, а возвращать целое число – количество необходимого сырья с учетом возможного брака сырья, как показано в таблице 1.

Таблица 1 – Спецификация матода

|  |  |
| --- | --- |
| Библиотека классов | WSUniversalLib.dll |
| Название класса | Calculation |
| Название метода | GetQuantityForProduct() |
| Входящие  обязательные  параметры | int productType,  int materialType,  int count,  float width,  float length |
| Возвращаемые  параметры | int |

2.1.1.3 Программный код

Программный код находится в репозитории на GitHub по адресу <https://github.com/AntonSidorov1/PracticeTesting_sidorov493>.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WSUniversalLib

{

public class Calculation

{

static List<Product> products = new List<Product>(new Product[] { new Product(1, 1.1f), new Product(2, 2.5f), new Product(3, 8.43f) });

static List<Material> materials = new List<Material>(new Material[] {new Material(1, 0.3f), new Material(2, 0.12f) });

public static int GetQuantityForProduct(float productType, float materialType, int count, float width, float length)

{

float materialCount = productType \* width \* length;

materialCount \*= count;

materialCount += (materialCount \* materialType);

float materialCount1 = (float)Math.Round(materialCount);

//materialCount = (float)Math.Round(materialCount, MidpointRounding.AwayFromZero);

materialCount = materialCount1 >= materialCount ? materialCount1 : materialCount1 + 1;

return (int)materialCount;

}

public static int GetQuantityForProduct(int productType, int materialType, int count, float width, float length)

{

Product product = products.Find(p => p.ProductType == productType);

Material material = materials.Find(p => p.MaterialType == materialType);

return GetQuantityForProduct(product.MaterialCount, material.Defect, count, width, length);

}

}

}

## Unit–tests

Первые 2 теста будут для метода *public static int GetQuantityForProduct(float productType, float materialType, int count, float width, float length)*, поскольку, как видно из кода, который приведён в предыдущем разделе, этот метод вызывается из того метода, который на самом деле нужно тестировать, а значит, от того, насколько правильно работает этот метод, зависит работа второго метода.

Первый unit-тест:

/// <summary>

/// Результат расчётов без использования метода совпадает с результатом рассчётов без использования метода

/// </summary>

[TestMethod]

public void TheResultCalculatedWithoutApplyingThisMethodCoincidesWithTheResultCalculatedUsingTheMethod()

{

float productType = 8.43f;

float materialType = 0.003f;

int count = 15;

float width = 20;

float length = 45;

float materialCount = productType \* width \* length;

materialCount \*= count;

materialCount += (materialCount \* materialType);

float materialCount4 = materialCount;

float materialCount1 = (float)Math.Round(materialCount);

materialCount = materialCount1 >= materialCount ? materialCount1 : materialCount1 + 1;

//materialCount = (float)Math.Round(materialCount, MidpointRounding.AwayFromZero);

int materialCount2 = (int)materialCount;

int materialCount3 = Calculation.GetQuantityForProduct(productType, materialType, count, width, length);

Assert.AreEqual(materialCount2, materialCount3, 0.001, materialCount2 + $"({materialCount4})" + "!=" + materialCount3);

}

Второй unit-тест:

/// <summary>

/// Для 15 единиц продукции шириной 20 и длиной 15, где количества материала на 1 единицу требуется 8,43 с процентом деффекта 0.03, требуется 114147 единиц материала

/// </summary>

[TestMethod]

public void For15ProductUnits20WidthAnd45LongProductType8And43MaterialType03TheMaterialQuantityIs114147()

{

float productType = 8.43f;

float materialType = 0.003f;

int count = 15;

float width = 20;

float length = 45;

int materialCount = Calculation.GetQuantityForProduct(productType, materialType, count, width, length);

Assert.AreEqual(114147, materialCount, 0.001, materialCount + "!=" + 114147);

}

Оба теста проводятся на правильность результата. Оба теста были успешными, как показано на рисунке 1.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Unit-тесты

Теперь, переходим ко второму методу (*GetQuantityForProduct(int productType, int materialType, int count, float width, float length)*), который вычисляет результат на основе данных, приведённых в коллекциях *products* (вид продукции: тип продукции и количество материала) и *materials* (материалы: тип материала и процент брака)

Первый unit-тест:

/// <summary>

/// для тестирования случая передачи несуществующего типа продукции.

/// </summary>

[TestMethod]

public void GetQuantityForProduct\_NonExistentProductType()

{

try

{

int materialCount = Calculation.GetQuantityForProduct(4, 2, 12, 14, 34);

Assert.AreEqual(114147, materialCount, 0.001, materialCount + "!=" + 114147);

}

catch (Exception e)

{

NullReferenceException ex = new NullReferenceException();

Assert.AreEqual(ex.Message, e.Message);

}

}

Второй unit-тест:

/// <summary>

/// Для 5 единиц продукции 1 типа шириной 4 и длиной 2 материала 2 типа требуется 45 единиц материала

/// </summary>

[TestMethod]

public void For5UnitsOfProductType1WithAWidthOf4AndALengthOf2MaterialsOfType2Is45UnitsOfMaterialAreRequired()

{

try

{

int materialCount = Calculation.GetQuantityForProduct(1, 2, 4, 2, 5);

Assert.AreEqual(45, materialCount, 0.001, materialCount + "!=" + 114147);

}

catch (Exception e)

{

NullReferenceException ex = new NullReferenceException();

Assert.AreEqual(ex.Message, e.Message);

}

}

Третий unit-тест:

/// <summary>

/// Для 50 единиц продукции 3 типа шириной 40 и длиной 80 материала 1 типа требуется 1352847 единиц материала

/// </summary>

[TestMethod]

public void For50UnitsOfProduct3TypesWithAWidthOf40AndALengthOf80MaterialOfType1Is1352847UnitsOfMaterialAreRequired()

{

try

{

int materialCount = Calculation.GetQuantityForProduct(3, 1, 40, 80, 50);

Assert.AreEqual(1352847, materialCount, 0.001, materialCount + "!=" + 1352847);

}

catch (Exception e)

{

NullReferenceException ex = new NullReferenceException();

Assert.AreEqual(ex.Message, e.Message);

}

}

Четвёртый unit-тест:

/// <summary>

/// Для 150 единиц продукции 2 типа шириной 25 и длиной 75 материала 2 типа требуется 703969 единиц материала

/// </summary>

[TestMethod]

public void For150UnitsOfProductType2WithAWidthOf25AndALengthOf75MaterialOfType2Is703969UnitsOfMaterialAreRequired()

{

try

{

int materialCount = Calculation.GetQuantityForProduct(2, 2, 25, 75, 150);

Assert.AreEqual(703969, materialCount, 0.001, materialCount + "!=" + 703969);

}

catch (Exception e)

{

NullReferenceException ex = new NullReferenceException();

Assert.AreEqual(ex.Message, e.Message);

}

}

Эти все тесты были пройдены успешно, как показано на рисунке 2.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Unit-тесты

Теперь, напишем Unit-тесты, которые не будут выполнены успешно.

/// <summary>

/// Получить количество материала при несуществующем типе материала

/// </summary>

[TestMethod]

public void GetQuantityOfMaterialWhenMaterialTypeDoesNotExist()

{

int materialCount = Calculation.GetQuantityForProduct(2, 12, 12, 14, 34);

Assert.AreEqual(114147, materialCount, 0.001, materialCount + "!=" + 114147);

}

/// <summary>

/// Получить количество материала, если тип продукции - текст

/// </summary>

[TestMethod]

public void GetMaterialQuantityIfproductTypeIsText()

{

int materialCount = Calculation.GetQuantityForProduct(Convert.ToInt32("abc"), 12, 12, 14, 34);

Assert.AreEqual(114147, materialCount, 0.001, materialCount + "!=" + 114147);

}

/// <summary>

/// Получить количество материала, если тип продукции - Не целое число

/// </summary>

[TestMethod]

public void GetMaterialQuantityIfProductTypeIsNotAnInteger()

{

int materialCount = Calculation.GetQuantityForProduct(Convert.ToInt32(5.12.ToString()), 12, 12, 14, 34);

Assert.AreEqual(114147, materialCount, 0.001, materialCount + "!=" + 114147);

}

Все 3 теста были провалены, как показано на рисунке 3.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 3 – Unit-тесты

Для того, чтобы тесты были успешными изменим код следующим образом:

/// <summary>

/// Получить количество материала при несуществующем типе материала

/// </summary>

[TestMethod]

public void GetQuantityOfMaterialWhenMaterialTypeDoesNotExist()

{

try

{

int materialCount = Calculation.GetQuantityForProduct(2, 12, 12, 14, 34);

Assert.AreEqual(114147, materialCount, 0.001, materialCount + "!=" + 114147);

}

catch (Exception e)

{

NullReferenceException ex = new NullReferenceException();

Assert.AreEqual(ex.Message, e.Message);

}

}

/// <summary>

/// Получить количество материала, если тип продукции - текст

/// </summary>

[TestMethod]

public void GetMaterialQuantityIfproductTypeIsText()

{

try

{

int materialCount = Calculation.GetQuantityForProduct(Convert.ToInt32("abc"), 12, 12, 14, 34);

Assert.AreEqual(114147, materialCount, 0.001, materialCount + "!=" + 114147);

}

catch (Exception e)

{

FormatException ex = new FormatException("Входная строка имела неверный формат.");

Assert.AreEqual(ex.Message, e.Message);

}

}

/// <summary>

/// Получить количество материала, если тип продукции - Не целое число

/// </summary>

[TestMethod]

public void GetMaterialQuantityIfProductTypeIsNotAnInteger()

{

try

{

int materialCount = Calculation.GetQuantityForProduct(Convert.ToInt32(5.12.ToString()), 12, 12, 14, 34);

Assert.AreEqual(114147, materialCount, 0.001, materialCount + "!=" + 114147);

}

catch (Exception e)

{

FormatException ex = new FormatException("Входная строка имела неверный формат.");

Assert.AreEqual(ex.Message, e.Message);

}

}

Теперь, все тесты успешны, как показано на рисунке 4.

Изображение выглядит как текст, стол

Автоматически созданное описание

Рисунок 4 – Unit-тесты

Ещё один тест будет на вложенный метод. Его нынешний код:

public static int GetQuantityForProduct(float productType, float materialType, int count, float width, float length)

{

float materialCount = productType \* width \* length;

materialCount \*= count;

materialCount += (materialCount \* materialType);

float materialCount1 = (float)Math.Round(materialCount);

//materialCount = (float)Math.Round(materialCount, MidpointRounding.AwayFromZero);

materialCount = materialCount1 >= materialCount ? materialCount1 : materialCount1 + 1;

return (int)materialCount;

}

Про результат данного метода, вопрос: Что будет при вводе отрицательного значения количества продукции, длины или ширины? Unit-тест:

/// <summary>

/// Получить количество материала при отрицательном значении количества продукции, длины или ширины

/// </summary>

[TestMethod]

public void GetMaterialQuantityWhenProductQuantityOrLengthOrWidthIsNegative()

{

ArgumentException ex = new ArgumentException();

try

{

int materialCount = Calculation.GetQuantityForProduct(1, 1, -12, -14, -34);

Assert.AreEqual(ex.Message, materialCount);

}

catch (Exception e)

{

//ArgumentException ex = new ArgumentException();

Assert.AreEqual(ex.Message, e.Message);

}

}

Данный тест не был успешным, как показано на рисунке 5.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 5 – Unit-тесты

Для этого изменим код метода следующим образом:

public static int GetQuantityForProduct(float productType, float materialType, int count, float width, float length)

{

if(productType < 0 || materialType <0 || count < 0 || width < 0 || length < 0)

{

throw new ArgumentException();

}

float materialCount = productType \* width \* length;

materialCount \*= count;

materialCount += (materialCount \* materialType);

float materialCount1 = (float)Math.Round(materialCount);

//materialCount = (float)Math.Round(materialCount, MidpointRounding.AwayFromZero);

materialCount = materialCount1 >= materialCount ? materialCount1 : materialCount1 + 1;

return (int)materialCount;

}

Теперь тест выполнен успешно, как показано на рисунке 6.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 6 – Unit-тесты

## Test-Cases

Тест-кейсы были созданы на основе Unit-тестов до их изменения.

Тестовый пример №1 представлен в таблице 2.

Таблица 2 – Тестовый пример №1.

|  |  |
| --- | --- |
| **ТТестовый пример №** | 1 |
| **ППриоритет тестирования** | Средний |
| **ЗЗаголовок/название теста** | Получить количество материала при отрицательном значении количества продукции, длины или ширины |
| **ККраткое изложение теста** | Отрицательные значения количества, длины или ширины являются недопустимыми |
| **ЭЭтапы теста** | Ввести длину  Ввести ширину  Ввести количество  Ввести тип продукции  Ввести тип материала |
| **ТТестовые данные** | Длина = -15  Ширина = -9  Количество = -2  Тип продукции = 1  Тип материала = 1 |
| **ООжидаемый результат** | Исключение типа ArgumentException |
| **ФФактический результат** | Рассчитан результат |
| **ССтатус** | Незачёт |
| **ППредварительное условие** | Наличие списка типов продукции и типов материала |
| **ППостусловие** |  |
| **ППримечания/комментарии** | После изменения кода метода, статус сменится на «*Зачёт*» |

Тестовый пример №2 представлен в таблице 3.

Таблица 3 – Тестовый пример №2.

|  |  |
| --- | --- |
| **ТТестовый пример №** | 2 |
| **ППриоритет тестирования** | Высокий |
| **ЗЗаголовок/название теста** | Для 15 единиц продукции шириной 20 и длиной 15, где количества материала на 1 единицу требуется 8,43 с процентом деффекта 0.03, требуется 114147 единиц материала |
| **ККраткое изложение теста** | Тестирование вложенного метода, который не зависит от списка типов продукции и материала. От работы данного метода зависит, полностью, работа программы |
| **ЭЭтапы теста** | Ввести длину  Ввести ширину  Ввести количество  Ввести количество материала на единицу продукции  Ввести процент деффекта |
| **ТТестовые данные** | Длина = 20  Ширина = 15  Количество = 15  Количество материала на единицу продукции = 8,43  Процент деффекта = 0,03 |
| **ООжидаемый результат** | 114147 |
| **ФФактический результат** | 114147 |
| **ССтатус** | Зачёт |
| **ППредварительное условие** | Наличие значений |
| **ППостусловие** |  |
| **ППримечания/комментарии** | Только при правильной работе данного метода возможна правильная работа метода, указанного в спецификации |

Тестовый пример №3 представлен в таблице 4.

Таблица 4 – Тестовый пример №3.

|  |  |
| --- | --- |
| **ТТестовый пример №** | 3 |
| **ППриоритет тестирования** | Средний |
| **ЗЗаголовок/название теста** | Для 150 единиц продукции 2 типа шириной 25 и длиной 75 материала 2 типа требуется 703969 единиц материала |
| **ККраткое изложение теста** | То же самое, что и предыдущий метод, только важно наличие списка типов продукции и материалов |
| **ЭЭтапы теста** | Ввести длину  Ввести ширину  Ввести количество  Ввести тип продукции  Ввести тип материала |
| **ТТестовые данные** | Длина = 75  Ширина = 25  Количество = 150  Тип продукции = 2  Тип материала = 2 |
| **ООжидаемый результат** | 703969 |
| **ФФактический результат** | 703969 |
| **ССтатус** | Зачёт |
| **ППредварительное условие** | Наличие списка типов продукции и типов материала |
| **ППостусловие** |  |
| **ППримечания/комментарии** |  |

Тестовый пример №4 представлен в таблице 5.

Таблица 5 – Тестовый пример №4.

|  |  |
| --- | --- |
| **ТТестовый пример №** | 4 |
| **ППриоритет тестирования** | Средний |
| **ЗЗаголовок/название теста** | для тестирования случая передачи несуществующего типа продукции/материала. |
| **ККраткое изложение теста** | При несуществующем типе продукции/материала должна быть ошибка |
| **ЭЭтапы теста** | Ввести длину  Ввести ширину  Ввести количество  Ввести тип продукции  Ввести тип материала |
| **ТТестовые данные** | Длина = 75  Ширина = 25  Количество = 150  Тип продукции = 6  Тип материала = 9 |
| **ООжидаемый результат** | Исключение типа NullReferenceException |
| **ФФактический результат** | Исключение типа NullReferenceException |
| **ССтатус** | Зачёт |
| **ППредварительное условие** | Наличие списка типов продукции и типов материала |
| **ППостусловие** |  |
| **ППримечания/комментарии** | Выдающееся исключение имеет текст «Ссылка на объект не указывает на экземпляр объект» |

Тестовый пример №5 представлен в таблице 6.

Таблица 6 – Тестовый пример №5.

|  |  |
| --- | --- |
| **ТТестовый пример №** | 5 |
| **ППриоритет тестирования** | Высокий |
| **ЗЗаголовок/название теста** | Получить количество материала, если тип продукции - Не целое число |
| **ККраткое изложение теста** | Тип продукции имеет тип «*int*», являющийся типом «целое число» |
| **ЭЭтапы теста** | Ввести длину  Ввести ширину  Ввести количество  Ввести тип продукции  Ввести тип материала |
| **ТТестовые данные** | Длина = 12,75  Ширина = 25  Количество = 45  Тип продукции = 6  Тип материала = 9 |
| **ООжидаемый результат** | Исключение типа FormatException |
| **ФФактический результат** | Исключение типа FormatException |
| **ССтатус** | Зачёт |
| **ППредварительное условие** |  |
| **ППостусловие** |  |
| **ППримечания/комментарии** | Выдающееся исключение имеет текст «Входная строка имела неверный формат» |

## Неделя 2

## Предметная область

Магазин музыкальных инструментов «ООО “Ритм”». Подробное описание предметной области представлено в [разделе 1.2.2](#_Описание_предметной_области).

## Описание программного продукта

Описание программного продукта представлено в [разделе 1.3.1](#_Постановка_задачи).

## Unit-тесты

# УП01.03. Разработка мобильных приложений

## Цель работы

Разработать приложение на основе предоставленного API, согласно индивидуальному варианту.

## Вариант задания

Разработка мобильного приложения на тему «чат»

## Макеты окон

3.1.1 Примечания

Окна в приложении могут отличаться от своих макетов.

3.1.2 Макеты окон

Макеты окон приложения показаны на рисунке 1.



Рисунок 1 – Макеты окон

Также, есть, ещё окна, макеты которых показаны на рисункуе 2.

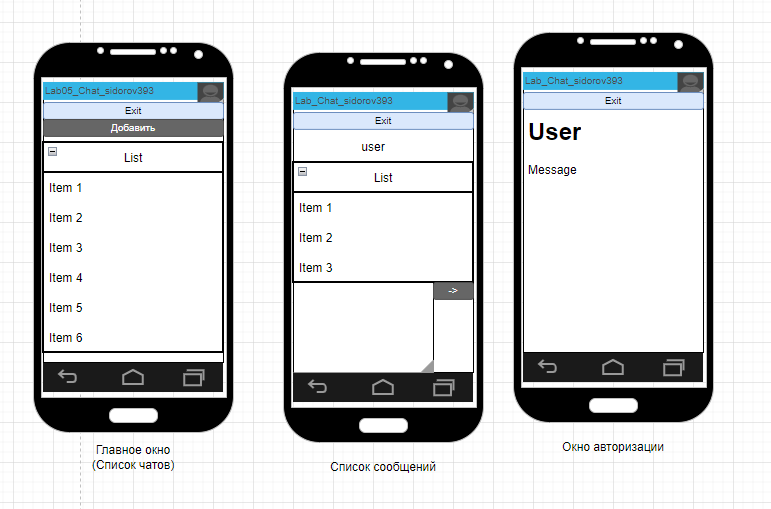


Рисунок 2 – Макеты окон

Окно главного и редактора url–ссылки меню показано на рисунке 3.

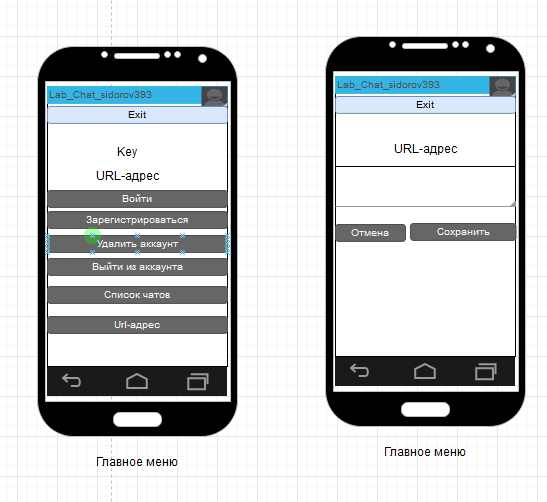


Рисунок 3 – Главное меню

3.1.3 Описание окон (функции)

Окно авторизации:

* Поле ввода логина
* Поле ввода пароля
* Флажок для показа/скрытия пароля
* Флажок для сохранения/несохранения данных на устройстве
* Кнопка отмены входа
* Кнопка входа в аккаунт

Окно регистрации:

* Поле ввода логина
* Поле ввода пароля
* Флажок для показа/скрытия пароля
* Флажок для сохранения/несохранения данных на устройстве
* Кнопка отмены входа
* Кнопка создания аккаунта и входа в него

Окно выхода из аккаунта:

* Предупреждающая надпись
* Кнопки подтверждения (да/нет)

Окно удаления аккаунта:

* Предупреждающая надпись
* Кнопки подтверждения (да/нет)

Окно списка чатов:

* Список чатов, с возможностью перейти к чату
* Кнопка добавления чата, перенаправляющая к списку аккаунтов

Окно списка аккаунтов:

* Список аккаунтов, с возможностью перейти к аккаунту и начать с ним чат

Окно редактора url–ссылки:

* Поле ввода url–адреса
* Кнопка сохранения url–адреса
* Кнопка отмены сохранения url–адреса

Главное меню (Первое открывающееся окно):

* Отображаемый url–адрес
* Отображаемый ключ сессии
* Кнопка авторизации, перенаправляющая на окно авторизации
* Кнопка регистрации, перенаправляющая на окно регистрации
* Кнопка выхода из аккаунта
* Кнопка удаления аккаунта
* Кнопка перехода к списку чатов
* Кнопка, перенаправляющая к редактору url–ссылок

Все окна, также, имеют кнопу выхода, перенаправляющую на предыдущее окно.

## Диаграмма базы данных

База данных предназначена для локального хранения данных об аккаунте и о url–ссылок.

Диаграмма базы данных представлена на рисунке 5.

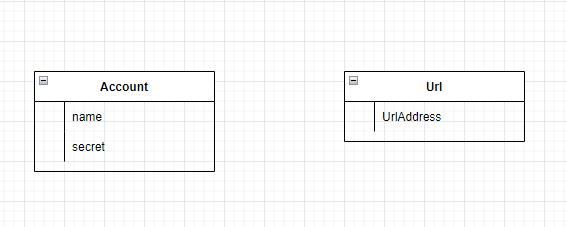


Рисунок 5 – Диаграмма базы данных

Таблица *Account* предназначена для хранения данных о пользователе: логин (*name*) и пароль (*secret*).

Таблица *UrlAddress* предназначена для хранения адреса сервера с API в поле *UrlAddress*.

## Описание API

Вся документация API находится по адресу <http://spbcoit.ru/lab/chat/docs/>..

Первая функция – *sign\_in* с адресом [Http://spbcoit.ru/lab/chat/api/rpc/sign\_in](http://spbcoit.ru/lab/chat/api/rpc/sign_in). Эта функция открывает сессию пользователя при входе в аккаунт. На вход подаются имя пользователя (*name*) и пароль (*password*). На выход выдаётся ключ сессии. Выполнение данной функции на Postman показано на рисунке 6.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – API на Postman

Обратное действие производит *sign\_out* с адресом <http://spbcoit.ru/lab/chat/api/rpc/sign_out>. Это, соответственно, закрытие сессии пользователя при выходе из аккаунта. На вход подаётся ключ сессии (*stoken*). На выход выдаётся значение, указывающее, удачно ли было закрытие сессии. Выполнение данной функции на Postman показано на рисунке 7.

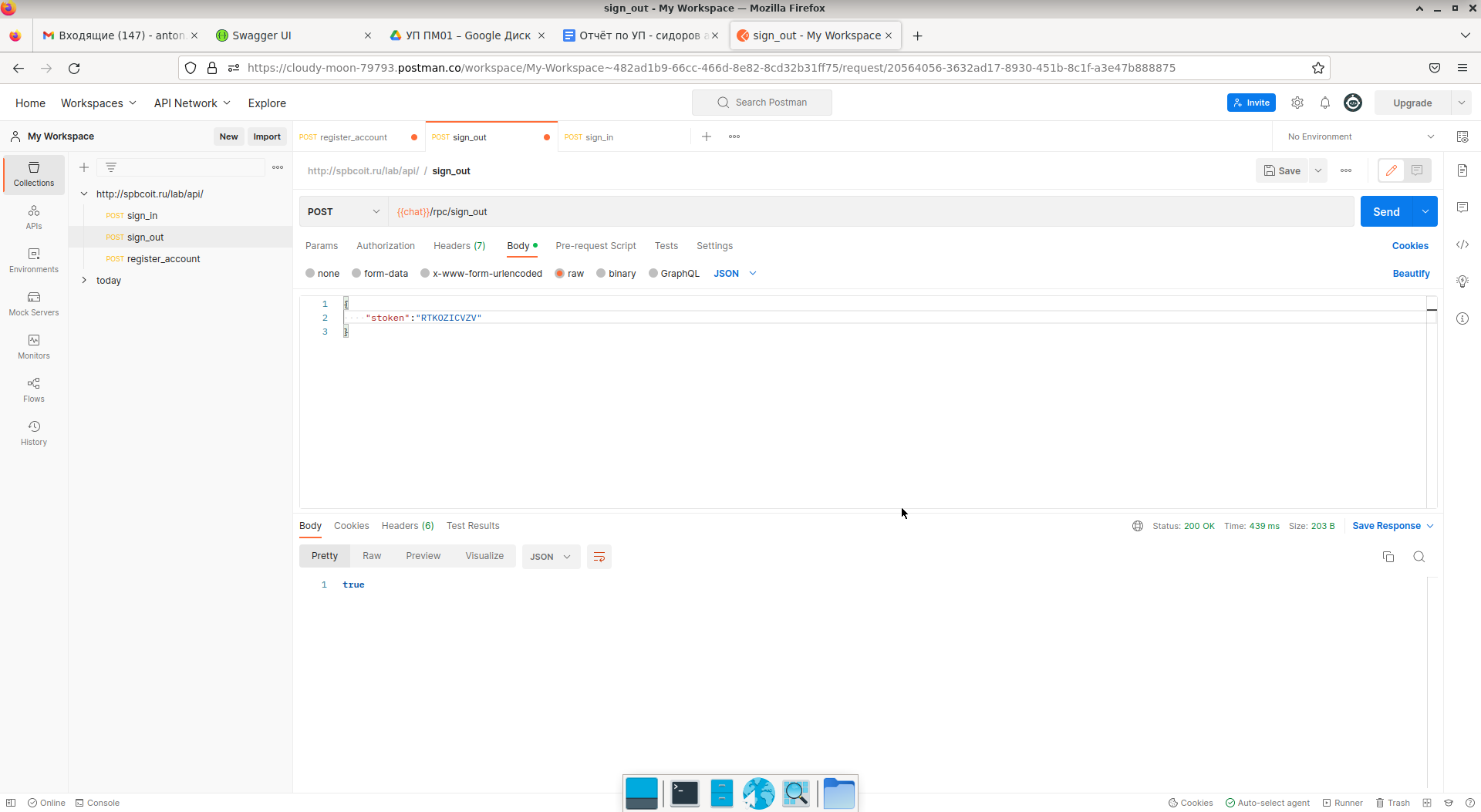


Рисунок 7 – API на Postman

Для создания аккаунта используется функция *register\_account*, расположенная по адресу <http://spbcoit.ru/lab/chat/api/rpc/register_account>. Эта функция открывает создаёт аккаунт пользователя при входе в аккаунт. На вход подаются имя пользователя (*name*) и пароль (*password*). На выход выдаётся сообщение об удачной регистрации. Выполнение данной функции на Postman показано на рисунке 8.

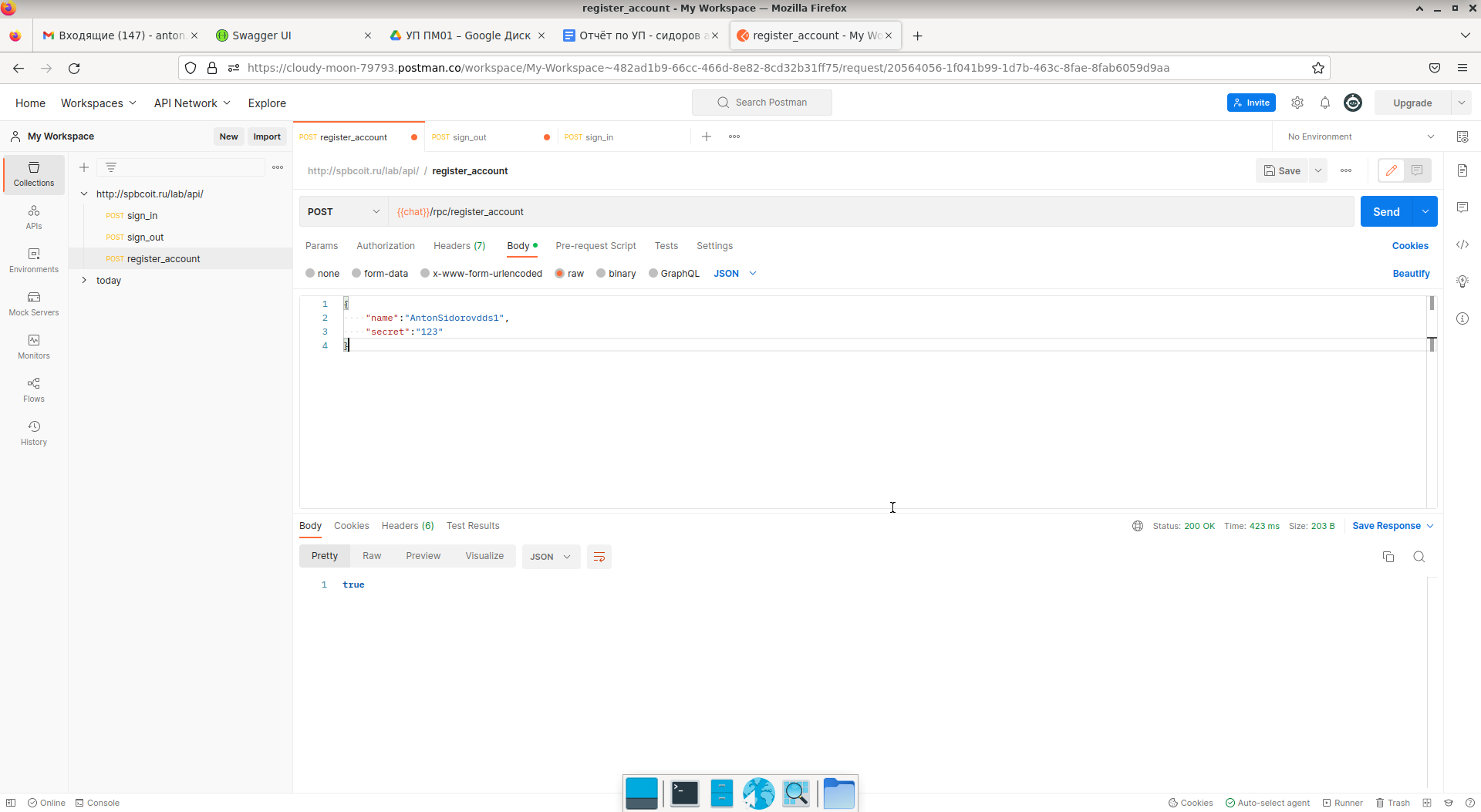


Рисунок 8 – API на Postman

Для удаления аккаунта используется функция *close\_account*, расположенная по адресу <http://spbcoit.ru/lab/chat/api/rpc/close_account>. На вход подаётся ключ сессии (*token*). На выход выдаётся значение, указывающее, удачно ли было закрытие сессии. Выполнение данной функции на Postman показано на рисунке 8.

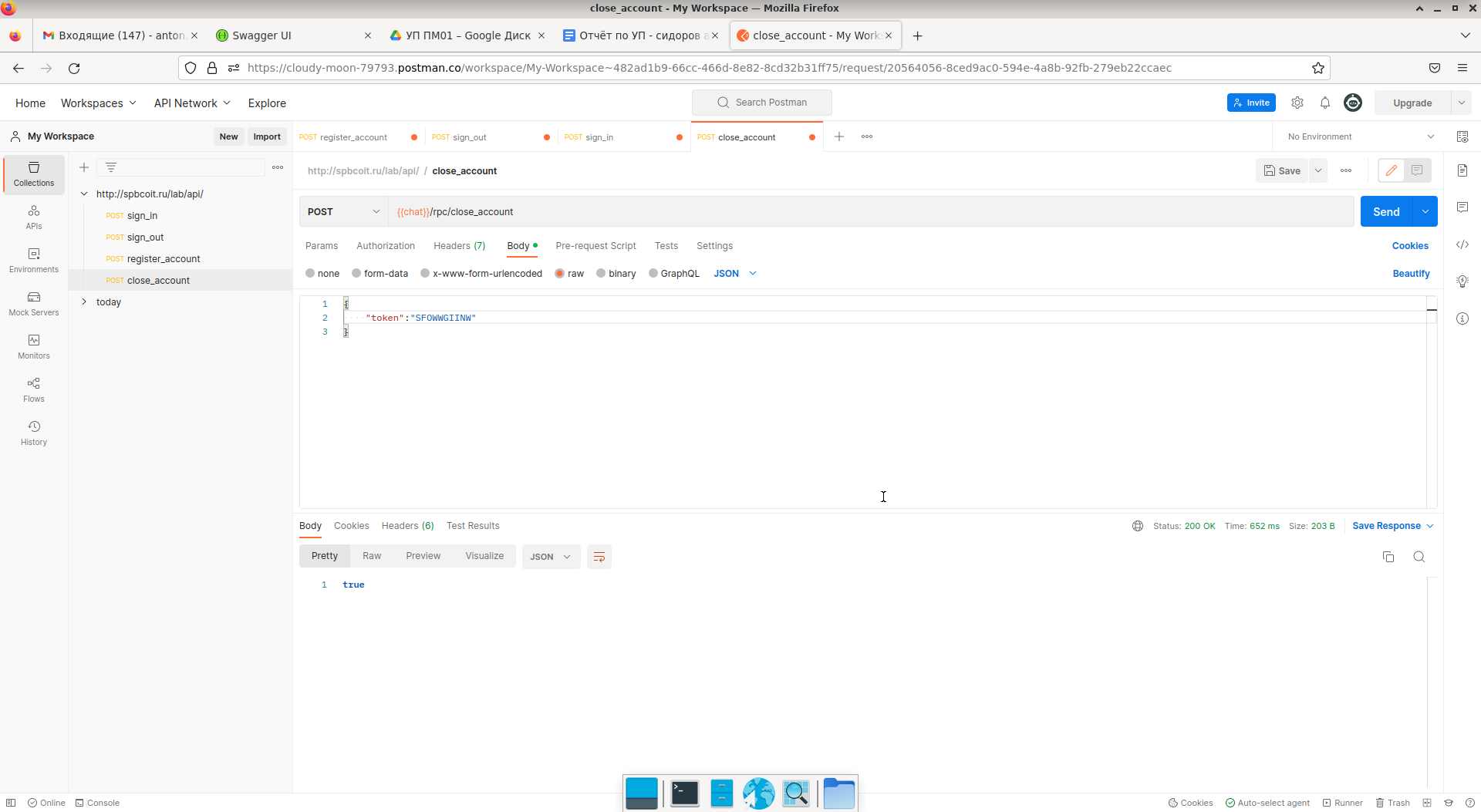


Рисунок 8 – API на Postman

Для просмотра списка аккаунтов используется *list\_accounts*, расположенная по адресу <http://spbcoit.ru/lab/chat/api/rpc/list_accounts>. На выход выдаётся список всех существующих аккаунтов. Выполнение данной функции на Postman показано на рисунке 9.

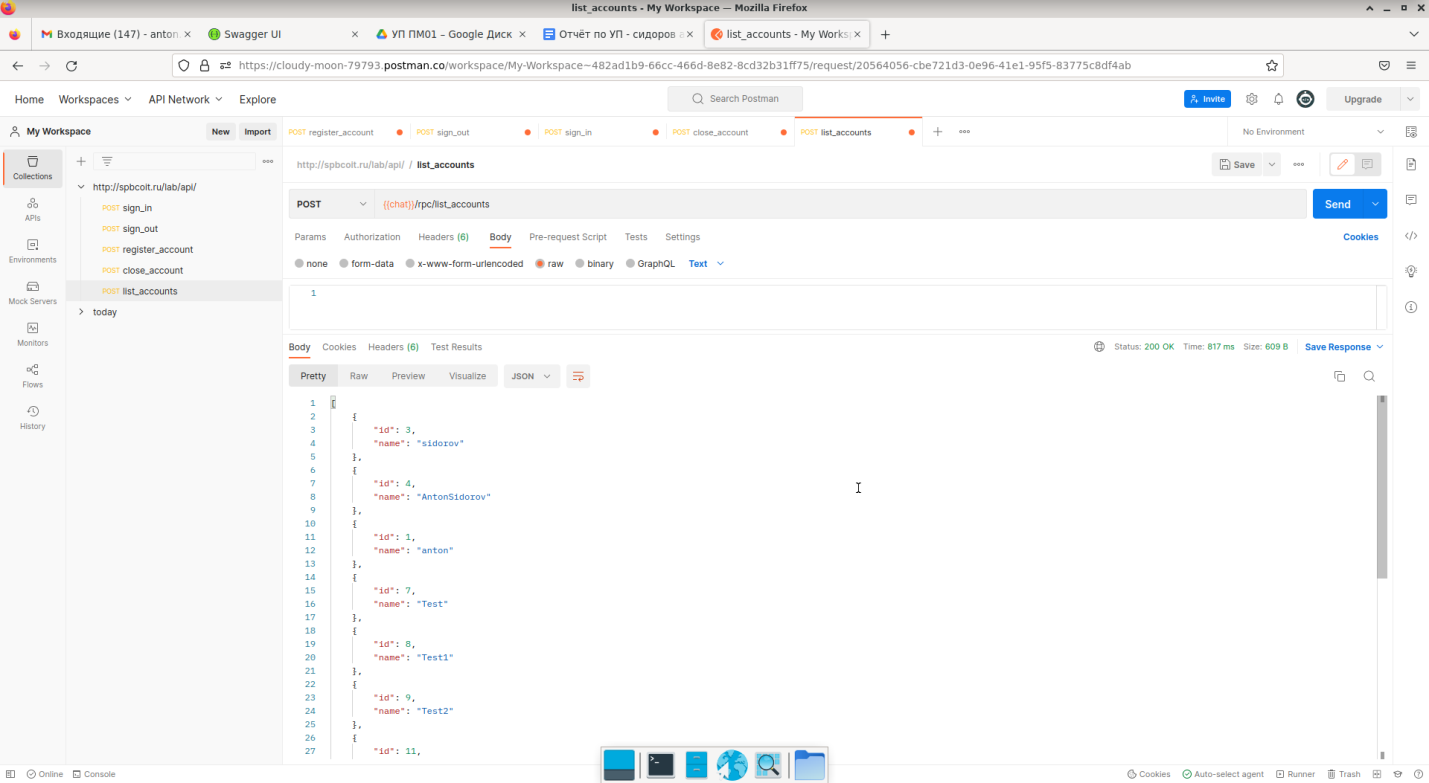


Рисунок 9 – API на Postman

Для отправки сообщения используется функция *post\_message*, находящаяся по адресу <http://spbcoit.ru/lab/chat/api/rpc/post_message>. На вход передаётся ключ сессии отправителя (*token*), id аккаунта-получателя (*to\_account*) и текст сообщения (*content*).На выход выдаётся значение, указывающее, удачно ли было отправлено сообщение. Выполнение данной функции на Postman показано на рисунке 10.

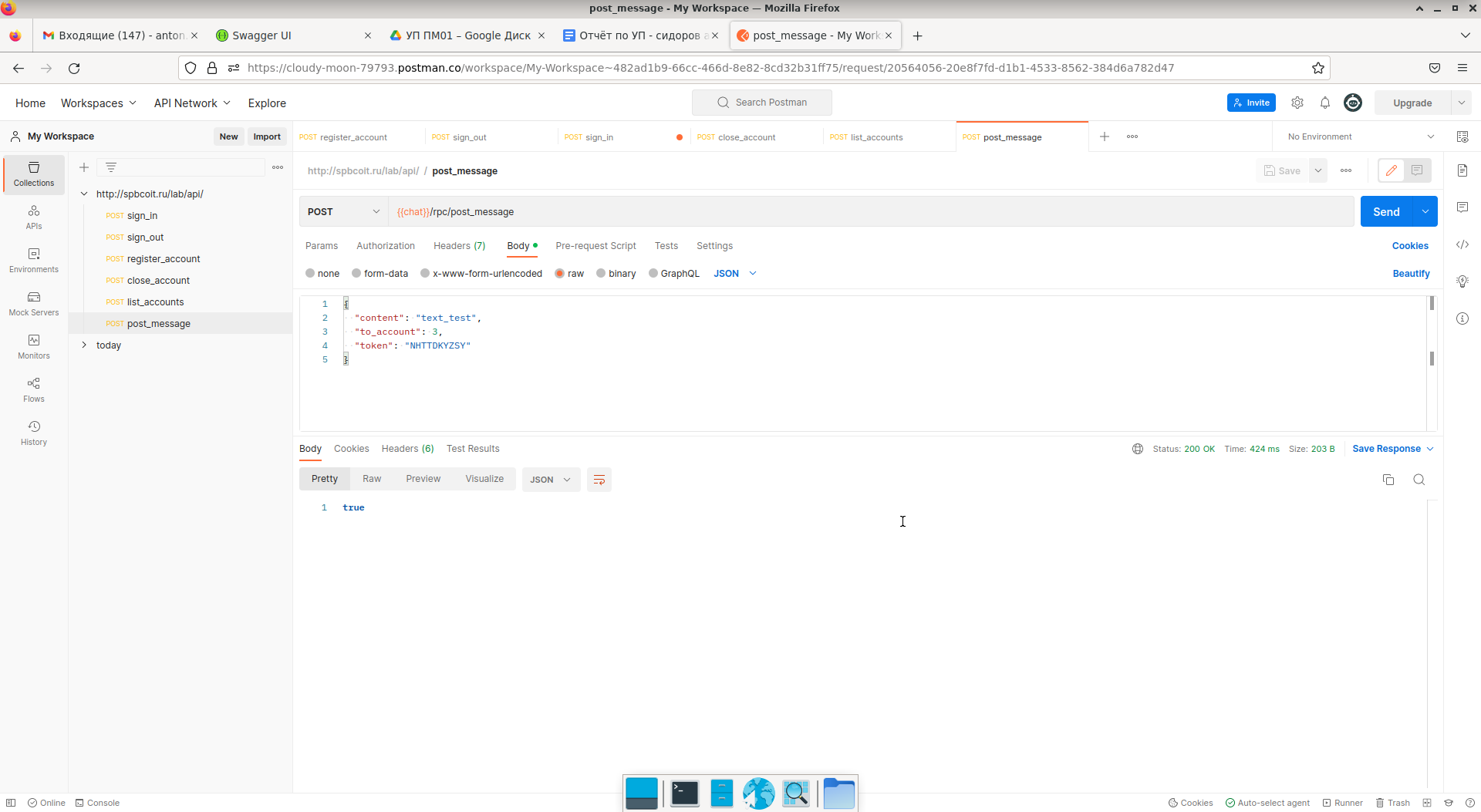


Рисунок 10 – API на Postman

Для просмотра списка чатов используется функция *get\_chats*, находящаяся по адресу <http://spbcoit.ru/lab/chat/api/rpc/get_chats>. На вход подаётся ключ сессии (*token*). На выход выдаётся список чатов (id отправителя и количество полученных сообщений). Выполнение данной функции на Postman показано на рисунке 11.

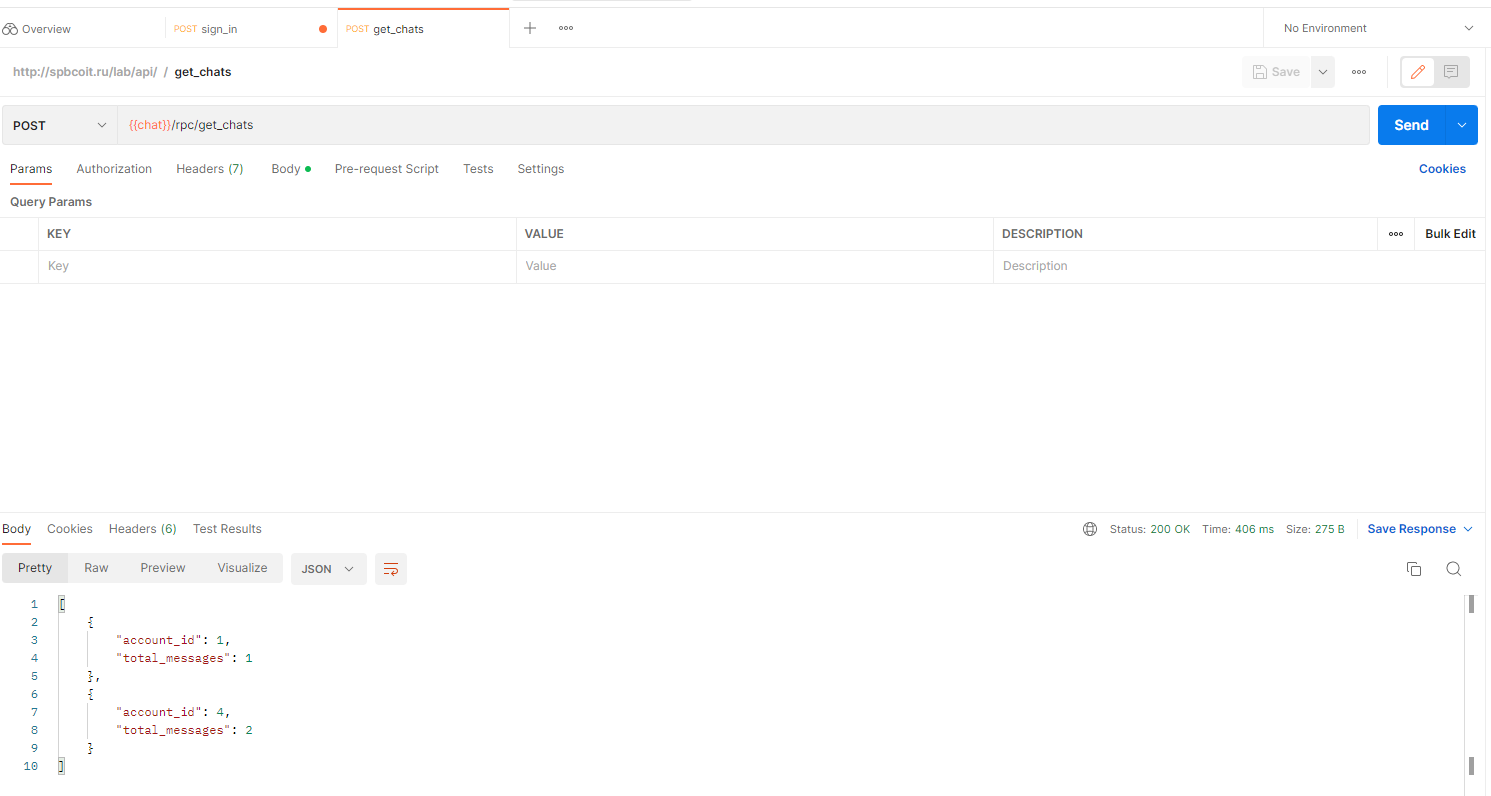


Рисунок 11 – API на Postman

Для просмотра списка полученных сообщений используется функция *get\_messages*, находящаяся по адресу <http://spbcoit.ru:80/lab/chat/api/rpc/get_messages>. На вход подаётся ключ сессии (*token*), минимальное id сообщения (*min\_msg\_id*) и id отправителя (*from\_account*). На выход выдаётся список сообщений. Выполнение данной функции на Postman показано на рисунке 12.

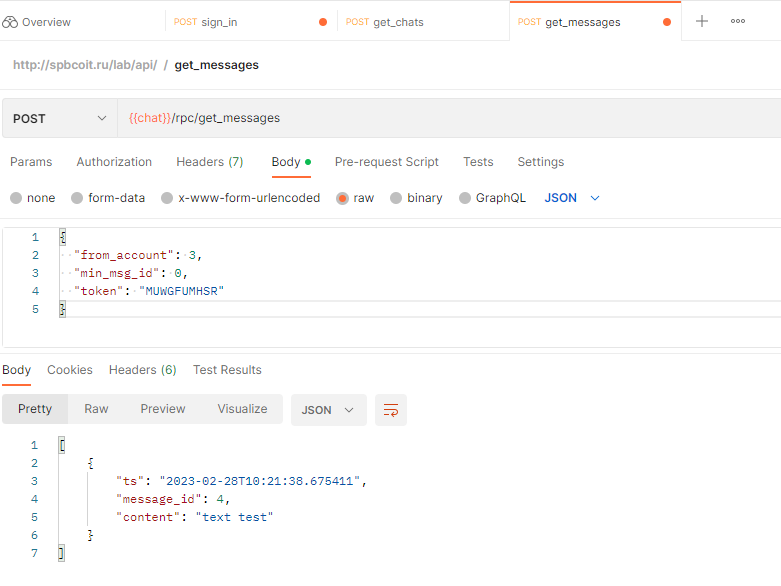


Рисунок 12 – API на Postman

## Программный код

# УП 01.04. Системное программирование

## Лабораторная работа №1. Изучение архитектуры МП

4.1.1 Цель работы

Изучение архитектуры микропроцессора КР580.

4.1.2 Cтруктура МП

Числовые обозначения структуры представлены на рисунке 1.

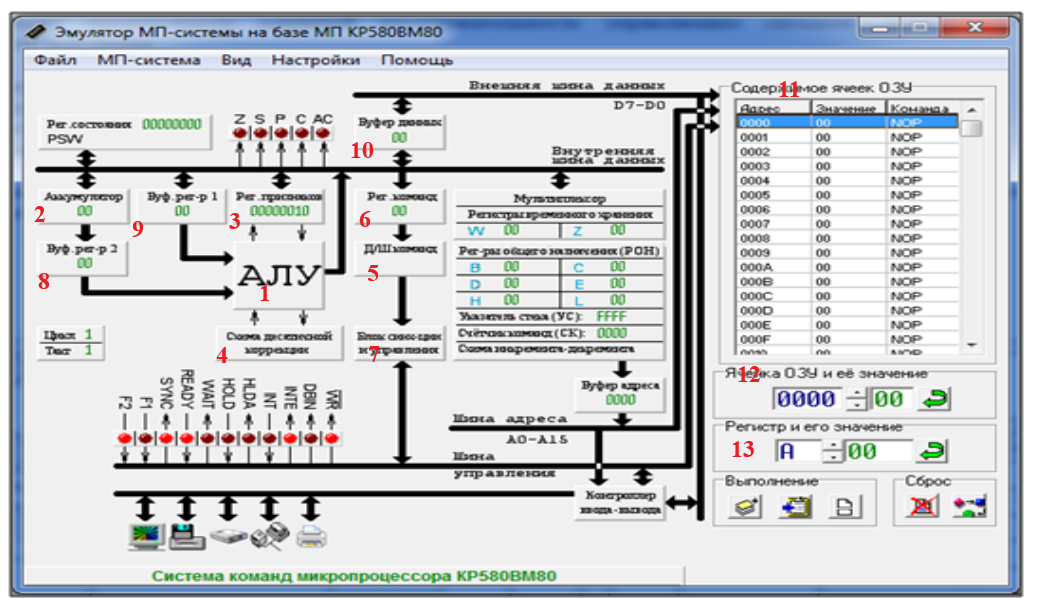


Рисунок 1 –Структура МП

Обозначения (Красным цветом на фотографии):

1. 8–разрядное арифметико–логическое устройство АЛУ (ALU)
2. аккумулятор (А)
3. регистр признаков RS, фиксирующий признаки, вырабатываемые АЛУ в процессе выполнения команды
4. десятичный корректор (DAA), выполняющий перевод информации из двоичной в двоично–десятичную форму
5. дешифратор команд (DCU)
6. регистр команд (IR), предназначенный для хранения первого байта команды, содержащего код операции
7. схема управления и синхронизации (CU), формирующая последовательности управляющих сигналов для работы ALU и регистров
8. однонаправленный 16–разрядный буферный регистр адреса (ВА)
9. двунаправленный 8–разрядный буферный регистр данных (BD)
10. регистр временного хранения операндов (RGb)
11. Пространство памяти и ввода–вывода
12. Значения ячеек памяти
13. Значения регистров

4.1.3 Таблица регистров, которые имеет МП

Таблица 1 – Назначение регистров

|  |  |  |
| --- | --- | --- |
| **Регистр** | **Назначение** | **Разрядность** |
| регистрами общего назначения (РОН) | 8–разрядные регистры F, A вместе с 16–разрядными регистрами HL, SP и PC образуют стандартный регистровый набор микропроцессора с аккумулятором. Этот набор расширен четырьмя 8–разрядными регистрами общего назначения (РОН): B, C, D, E, которые в некоторых командах объединяются в 16–разрядные парные регистры BC и DE. Младшими регистрами пары являются соответственно регистры C и E. Введение РОН позволило создать достаточно эффективный микропроцессор с широкими функциональными возможностями. | 8 бит |
| аккумулятор А | используется в подавляющем большинстве команд логической и арифметической обработки. Обычно он адресуется неявно и служит как источником операнда, так и приемником результата. Благодаря этому в командах ВМ80А явно указывается только один операнд. | 8 бит |
| регистр HL | как правило, служит адресным регистром. При косвенной регистровой адресации он хранит 16–разрядный адрес основной памяти. В этом случае к нему ссылаются с помощью мнемоники М (Memory), например:  MOV A, M; содержимое ячейки (HL) заносится в аккумулятор | 16 бит |
| Регистры PC и SP | выполняют свою обычную функцию счетчика команд и указателя стека. |  |
|  |  |  |
|  |  |  |

**4.1.4 блок–схема функционирования МП во время выполнения команды сложения содержимого аккумулятора и регистра В, имеющая мнемоническое обозначение ADD B.**

Данная блок–схема показана на рисунке 1.

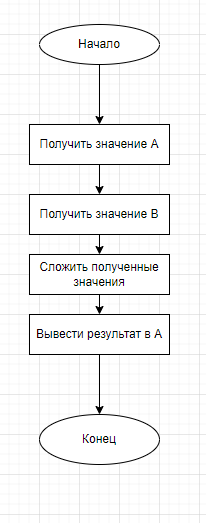


Рисунок 2 – Блок–схема

**4.1.5 Таблица флагов**

Таблица 2 – Назначение флагов

| **Обозначение флага** | **Признак флага** | **Условие установки флагов** |
| --- | --- | --- |
| C (Carry) | признак переноса | наличие переноса (при сложении) или заема (при вычитании) из старшего разряда аккумулятора, иначе сбрасывается |
| M (Minus) | признак отрицательного результата | устанавливается, если знаковый бит результата операции (седьмой разряд аккумулятора) равен 1, иначе сбрасывается |
| Z (Zero) | признак нуля | устанавливается, если результат операции в аккумуляторе равен нулю, иначе сбрасывается |
| P (Parity) | признак паритета/четности | устанавливается, если результат операции в аккумуляторе содержит четное число единиц, иначе сбрасывается |
| AC (Auxiliary Carry) | признак половинного переноса | устанавливается при наличии переноса из третьего разряда аккумулятора в четвертый, иначе сбрасывается |
|  |  |  |
|  |  |  |

## Лабораторная работа №2. Исследование команд прямой адресации

4.2.1 Цель лабораторной работы

Целью данной работы является ознакомление с командами микропроцессора КР580 для прямой адресации.

4.2.2 Программный код

Таблица 3 – Программный код в лабораторной работе

|  |  |  |  |
| --- | --- | --- | --- |
| 0000 | 3A | LDA adr | загрузить содержимое в аккумулятор |
| 0001 | 0A | LDAX B | из ячейки памяти с адресом Ah |
| 0002 | 00 | NOP |  |
| 0003 | 32 | STA adr | загрузить содержимое аккумулятора в |
| 0004 | 0B | DCX B | ячейку памяти с адресом Bh |
| 0005 | 00 | NOP |  |
| 0006 | 00 | NOP |  |
| 0007 | 76 | HLT | Остановка выполнения программы |
| 0008 | 00 | NOP |  |
| 0009 | 00 | NOP |  |
| 000A | 40 | MOV B, B | Значение в ячейке памяти |
| 000B | 00 | NOP |  |
| 000C | 00 | NOP |  |
| 000D | 00 | NOP |  |
| 000E | 00 | NOP |  |
| 000F | 00 | NOP |  |
| 0010 | 00 | NOP |  |

4.2.3 Ввод операндов при прямой адресации

Скрин ввода операндов и начала работы программы представлен на рисунке 3.

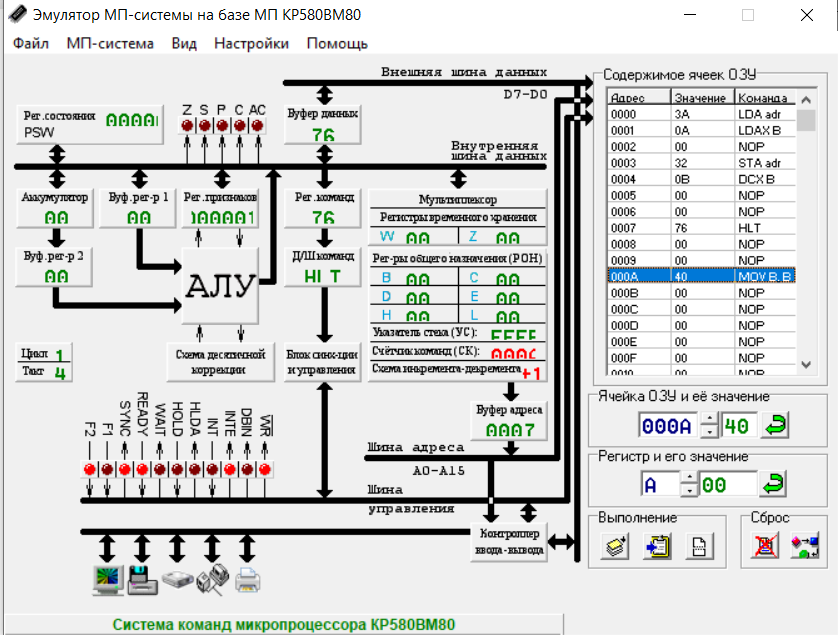


Рисунок 3 – Ввод операндов при прямой адресации

Операнд вводится в ячейку памяти *Ah*.

4.2.4 Результат работы модели при прямой адресации

Скрин результата работы программы представлен на рисунке 4.

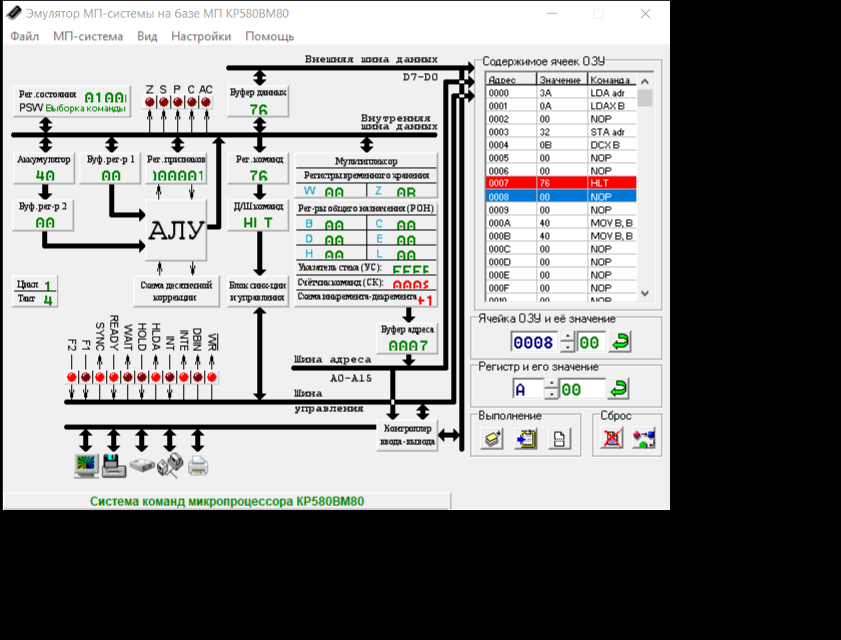


Рисунок 4 – Результат работы модели при прямой адресации

Результат выводится в ячейку памяти *Bh*.

4.2.5 Заключение

Освоено использование команд для прямой адресации.

## Лабораторная работа №3. Исследование команд непосредственной адресации

4.3.1 Цель лабораторной работы

Целью данной работы является ознакомление с командами микропроцессора КР580 для непосредственной адресации

4.3.2 Программный код

Таблица 4 – Программный код в лабораторной работе

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | Положить в регистр A (аккумулятор) |
| 0001 | 20 | – | число 20h |
| 0002 | 06 | MVI B, d8 | Положить в регистр B |
| 0003 | 05 | DCR B | число 5h |
| 0004 | 26 | MVI H, d8 | Положить в регистр H |
| 0005 | D1 | POP D | число D1h |
| 0006 | D7 | RST 2 | Перейти к команде по в ячейке с адресом 10h |
| 0007 | 00 | NOP |  |
| 0008 | 00 | NOP |  |
| 0009 | 00 | NOP |  |
| 000A | 00 | NOP |  |
| 000B | 00 | NOP |  |
| 000C | 00 | NOP |  |
| 000D | 00 | NOP |  |
| 000E | 00 | NOP |  |
| 000F | 00 | NOP |  |
| 0010 | 00 | NOP |  |
| 0011 | 00 | NOP |  |
| 0012 | 00 | NOP |  |
| 0013 | 00 | NOP |  |
| 0014 | 76 | HLT | Остановить работу программы |
| 0015 | 00 | NOP |  |

4.3.3 Ввод операндов при непосредственной адресации

Скрин ввода операндов и начала работы программы представлен на рисунке 5.

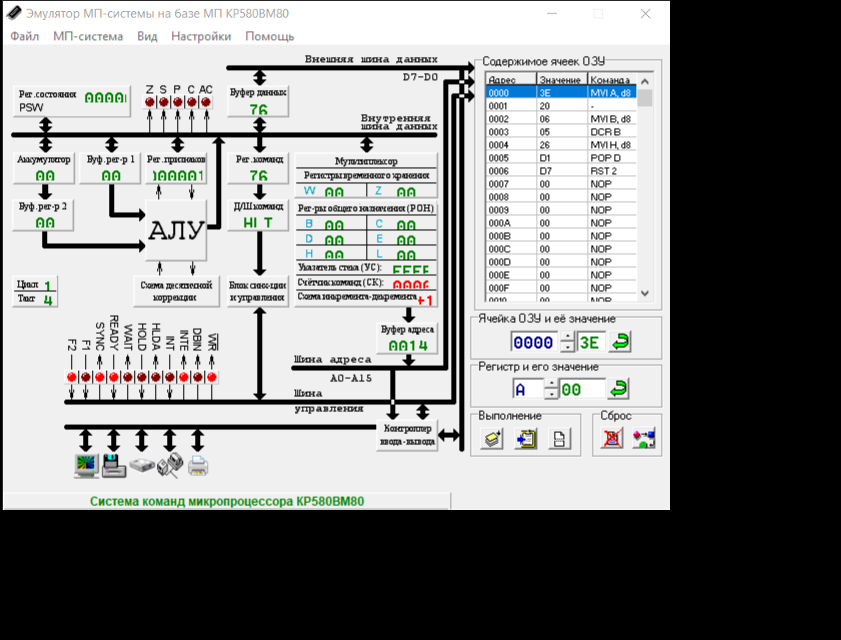


Рисунок 5 – Ввод операндов при непосредственной адресации

В регистр *A* будет вводиться число *20h*, В регистр *B* – *5h*, *H – D1h*.

4.3.4 Результат работы модели при непосредственной адресации

Скрин результата работы программы представлен на рисунке 6.

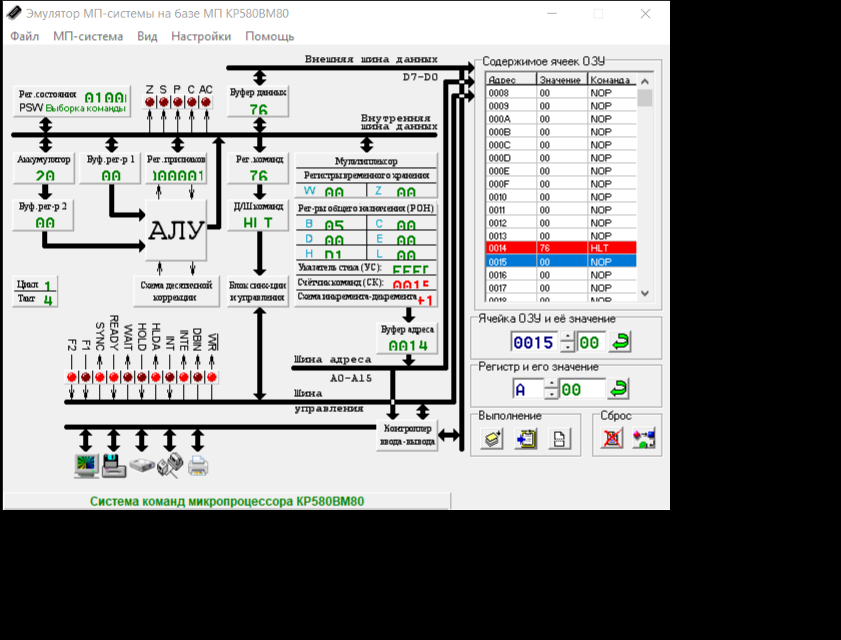


Рисунок 6 – Результат работы модели при непосредственной адресации

Значения регистров приведены в таблице ниже:

|  |  |  |  |
| --- | --- | --- | --- |
|  | A= | 20 |  |
| B= | 05 | C= | 00 |
| D= | 00 | E= | 00 |
| H= | D1 | L= | 00 |
| W= | 00 | Z= | 00 |

4.3.5 Заключение

Освоено использование команд для непосредственной адресации.

## Лабораторная работа №4. Исследование команд косвенной адресации

4.4.1 Цель лабораторной работы

Целью данной работы является ознакомление с командами микропроцессора КР580 для косвенной адресации

4.4.2 Программный код

Таблица 5 – Программный код в лабораторной работе

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 06 | MVI B, d8 | Переместить адрес ячекйки в регистр B |
| 0001 | 00 | NOP | Адрес ячейки – 00h |
| 0002 | 0E | MVI C, d8 | Переместить адрес ячейки в регистр C |
| 0003 | 0A | LDAX B | Адрес ячейки – 0Ah |
| 0004 | 0A | LDAX B | Перенести значение из ячейки с адресом в регистре B, в аккумулятор |
| 0005 | 00 | NOP |  |
| 0006 | 00 | NOP |  |
| 0007 | 76 | HLT |  |
| 0008 | 00 | NOP |  |
| 0009 | 00 | NOP |  |
| 000A | 50 | MOV D, B | Значение в ячейке – 50h |
| 000B | 00 | NOP |  |
| 000C | 00 | NOP |  |

4.4.3 Ввод операндов при косвенной адресации

Скрин ввода операндов и начала работы программы представлен на рисунке 7.

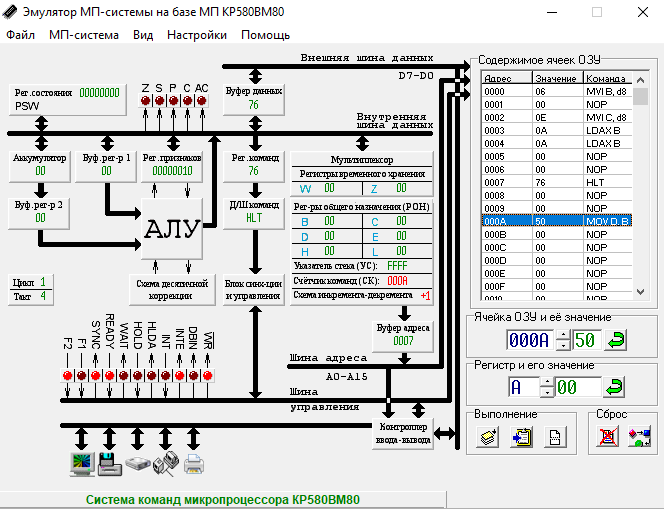


Рисунок 7 – Ввод операндов при косвенной адресации

Вводимое число – 50h.

4.4.4 Результат работы модели при косвенной адресации

Скрин результата работы программы представлен на рисунке 8.



Рисунок 8 – Результат работы модели при косвенной адресации

Значения регистров приведены в таблице 6:

Таблица 6 – Значения в регистрах

|  |  |  |  |
| --- | --- | --- | --- |
|  | A= | 50 |  |
| B= | 00 | C= | 0A |
| D= | 00 | E= | 00 |
| H= | 00 | L= | 00 |
| W= | 00 | Z= | 00 |

4.4.5 Заключение

Освоено использование команд для косвенной адресации.

## Лабораторная работа №5. Исследование команд стековой адресации

4.5.1 Цель лабораторной работы

Целью данной работы является ознакомление с простейшими арифметическими действиями на микропроцессоре КР580

4.5.2 Программный код

Таблица 7 – Программный код в лабораторной работе

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 06 | MVI B, d8 | Переместить адрес ячейки в регистр B |
| 0001 | 00 | NOP | Адрес ячейки – 00h |
| 0002 | 0E | MVI C, d8 | Переместить адрес ячейки в регистр C |
| 0003 | 0A | LDAX B | Адрес ячейки – 0Ah |
| 0004 | 0A | LDAX B | Перенести значение из ячейки с адресом в регистре B, в аккумулятор |
| 0005 | 00 | NOP |  |
| 0006 | 00 | NOP |  |
| 0007 | 76 | HLT |  |
| 0008 | 00 | NOP |  |
| 0009 | 00 | NOP |  |
| 000A | 50 | MOV D, B | Значение в ячейке – 50h |

4.5.3 Ввод операндов при стековой адресации

Скрин ввода операндов и начала работы программы представлен на рисунке 9.

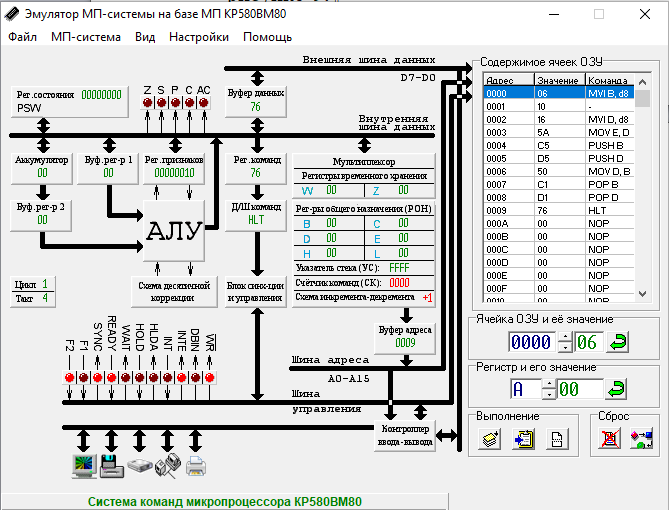


Рисунок 9 – Ввод операндов при стековой адресации

Данные помещаются в регистры В и D, после чего их содержимое посылается в стек. После этого производится обмен значениями между регистрами и значения регистров возвращаются из стека

4.5.4 Результат работы модели при стековой адресации

Скрин результата работы программы представлен на рисунке 10.

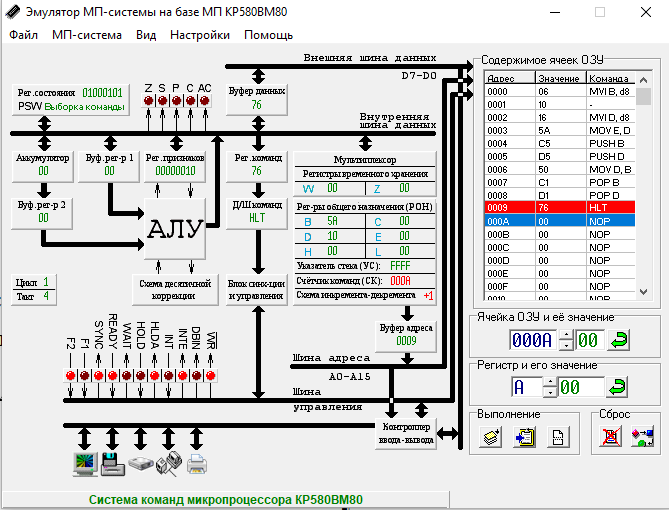


Рисунок 10 – Результат работы модели при стековой адресации

Значения регистров приведены в таблице 8:

Таблица 8 – Значения в регистрах

|  |  |  |  |
| --- | --- | --- | --- |
|  | A= | 00 |  |
| B= | 5A | C= | 00 |
| D= | 10 | E= | 00 |
| H= | 00 | L= | 00 |
| W= | 00 | Z= | 00 |
|  |  |  |  |
|  | SP= | FFFF |  |

4.5.5 Заключение

Освоено использование команд для стековой адресации.

## Лабораторная работа №6. Пример программы для микропроцессора

4.6.1 Цель лабораторной работы

Целью данной работы является ознакомление с простейшими арифметическими действиями на микропроцессоре КР580

4.6.2 Программный код

Программный код для сложения показан в таблице 9.

Таблица 9 – Программный код в лабораторной работе

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | AF | XRA A | Очистить аккумулятор |
| 0001 | 3E | MVI A, d8 | Записать в аккумулятор |
| 0002 | 38 | - | число 5610 = 38h16 |
| 0003 | 06 | MVI B, d8 | Записать в регистр В |
| 0004 | A3 | ANA E | число 16310 = a3h16 |
| 0005 | 80 | ADD B | Сложить 38h16 и a3h16 (5610 + 16310 = 21910 = db16) |
| 0006 | E7 | RST 4 | Прервать выполнение программы |
| 0007 | 00 | NOP |  |
| 0008 | 00 | NOP |  |
| 0009 | 00 | NOP |  |
| 000A | 00 | NOP |  |
| 000B | 00 | NOP |  |
| 000C | 00 | NOP |  |
| 000D | 00 | NOP |  |
| 000E | 00 | NOP |  |
| 000F | 00 | NOP |  |
| 0010 | 00 | NOP |  |
| 0011 | 00 | NOP |  |
| 0012 | 00 | NOP |  |
| 0013 | 00 | NOP |  |
| 0014 | 00 | NOP |  |
| 0015 | 00 | NOP |  |
| 0016 | 00 | NOP |  |
| 0017 | 00 | NOP |  |
| 0018 | 00 | NOP |  |
| 0019 | 00 | NOP |  |
| 001A | 00 | NOP |  |
| 001B | 00 | NOP |  |
| 001C | 00 | NOP |  |
| 001D | 00 | NOP |  |
| 001E | 00 | NOP |  |
| 001F | 00 | NOP |  |
| 0020 | 00 | NOP |  |
| 0021 | 76 | HLT | Остановить выполнение программы |
| 0022 | 00 | NOP |  |

Программный код для вычитания показан в таблице 10.

Таблица 10 – Программный код в лабораторной работе.

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | AF | XRA A | Очистить аккумулятор |
| 0001 | 3E | MVI A, d8 | Записать в аккумулятор |
| 0002 | E8 | - | число 24810 = E8h16 |
| 0003 | 06 | MVI B, d8 | Записать в регистр В |
| 0004 | A3 | ANA E | число 16310 = a3h16 |
| 0005 | 90 | SUB B | Сложить E8h16 и a3h16 (24810 – 16310 = 6510 = 4116) |
| 0006 | E7 | RST 4 | Прервать выполнение программы |
| 0007 | 00 | NOP |  |
| 0008 | 00 | NOP |  |
| 0009 | 00 | NOP |  |
| 000A | 00 | NOP |  |
| 000B | 00 | NOP |  |
| 000C | 00 | NOP |  |
| 000D | 00 | NOP |  |
| 000E | 00 | NOP |  |
| 000F | 00 | NOP |  |
| 0010 | 00 | NOP |  |
| 0011 | 00 | NOP |  |
| 0012 | 00 | NOP |  |
| 0013 | 00 | NOP |  |
| 0014 | 00 | NOP |  |
| 0015 | 00 | NOP |  |
| 0016 | 00 | NOP |  |
| 0017 | 00 | NOP |  |
| 0018 | 00 | NOP |  |
| 0019 | 00 | NOP |  |
| 001A | 00 | NOP |  |
| 001B | 00 | NOP |  |
| 001C | 00 | NOP |  |
| 001D | 00 | NOP |  |
| 001E | 00 | NOP |  |
| 001F | 00 | NOP |  |
| 0020 | 00 | NOP |  |
| 0021 | 76 | HLT | Остановить выполнение программы |
| 0022 | 00 | NOP |  |

4.6.3 Простое сложение двух однобайтных чисел

Скрин программы показан на рисунке 11.

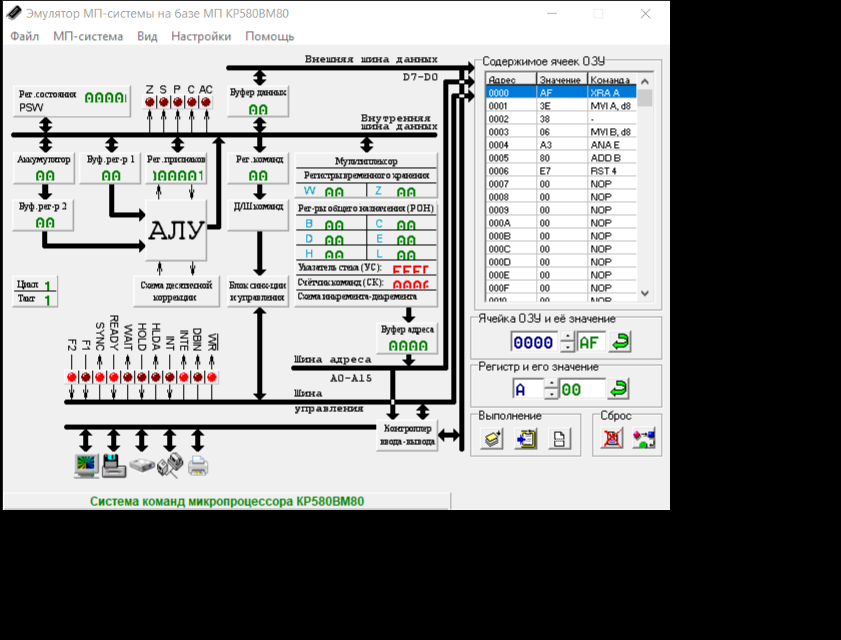


Рисунок 11 – Простое сложение двух однобайтных чисел

Значения регистров, после выполнения программы представлены в таблице 11.

Таблица 11 – Значения в регистрах

|  |  |  |  |
| --- | --- | --- | --- |
|  | A= | DB |  |
| B= | A3 | C= | 00 |
| D= | 00 | E= | 00 |
| H= | 00 | L= | 00 |
| W= | 00 | Z= | 00 |

4.6.4 Простое вычитание двух однобайтных чисел

Скрин программы показан на рисунке 12.

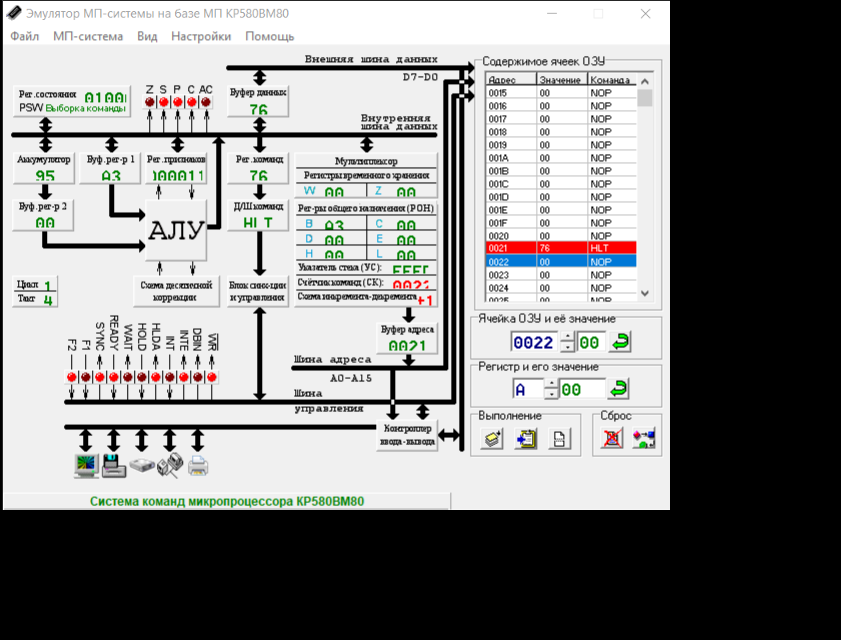


Рисунок 12 – Простое вычитание двух однобайтных чисел

Значения регистров, после выполнения программы представлены в таблице 12.

Таблица 12 – Значения в регистрах

|  |  |  |  |
| --- | --- | --- | --- |
|  | A= | 95 |  |
| B= | A3 | C= | 00 |
| D= | 00 | E= | 00 |
| H= | 00 | L= | 00 |
| W= | 00 | Z= | 00 |

4.6.5 Заключение

Освоено использование команд для простейших арифметических действий.

## Лабораторная работа №7. Программа сложения двух однобайтных чисел X и У

4.7.1 Цель лабораторной работы

Рассмотреть особенности выполнения простейших арифметических операций над целыми числами без знака на эмуляторе МП К580, познакомиться с программированием в машинных кодах и мнемокодах, научиться пользоваться средствами управления и кнопками эмулятора.

4.7.2 Вариант задания

X = 3910 = 2716 и Y = 19810 = C616. X+Y = 2716 + C616 = 3910 + 19810 = 23710 = ED16.

4.7.3 Программный код

Таблица 13 – Программный код в лабораторной работе.

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | AF | XRA A | Очистить аккумулятор |
| 0001 | 3E | MVI A, d8 | Записать в аккумулятор |
| 0002 | 27 | DAA | число X (3910 = 2716) |
| 0003 | 06 | MVI B, d8 | Записать в регистр В |
| 0004 | C6 | ADI d8 | число У (19810 = С616) |
| 0005 | 80 | ADD B | Сложить X и Y Сложить 2716 и С616 (3910 + 19810 = 23710 = ED16) |
| 0006 | 32 | STA adr | Записать содержимое аккумулятора в |
| 0007 | 0D | DCR C | ячейку 000D |
| 0008 | 00 | NOP |  |
| 0009 | 76 | HLT | Остановить программу |

4.7.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 13.



Рисунок 13 – Скрин программы

Скрин программы после её выполнения представлен на рисунке 14.

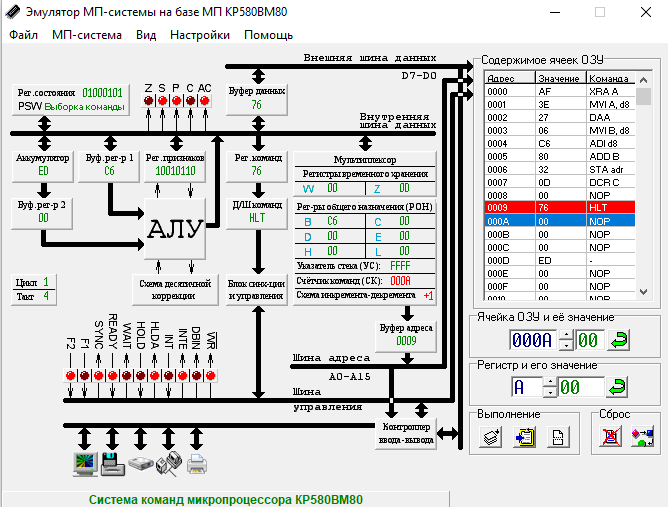


Рисунок 14 – Скрин программы

Результат вычислений был выведен в ячейку *000D*. Этот результат равен *ED16 (23710)*.

4.7.5 Заключение

Освоено выполнение сложения двух чисел.

## Лабораторная работа №8. Программа вычитания двух однобайтных чисел X и У

4.8.1 Цель лабораторной работы

Рассмотреть особенности выполнения простейших арифметических операций над целыми числами без знака на эмуляторе МП К580, познакомиться с программированием в машинных кодах и мнемокодах, научиться пользоваться средствами управления и кнопками эмулятора.

4.8.2 Вариант задания

X = 20010 = C816 и Y = 4510 = 2D16. X–Y = C816 – 2D16 = 20010 – 4510 = 15510 = 9B16.

4.8.3 Программный код

Таблица 14 – Программный код в лабораторной работе.

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | AF | XRA A | Очистить аккумулятор |
| 0001 | 3E | MVI A, d8 | Записать в аккумулятор |
| 0002 | C8 | RZ | число X (20010 = C816) |
| 0003 | 06 | MVI B, d8 | Записать в регистр В |
| 0004 | 2D | DCR L | число У (4510 = 2D16) |
| 0005 | 90 | SUB B | Вычесть X и Y Сложить C816 и 2D16 (20010 – 4510 = 14410 = 9B16) |
| 0006 | 32 | STA adr |  |
| 0007 | 0D | DCR C |  |
| 0008 | 00 | NOP |  |
| 0009 | 76 | HLT |  |
| 000A | 00 | NOP |  |

4.8.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 15.

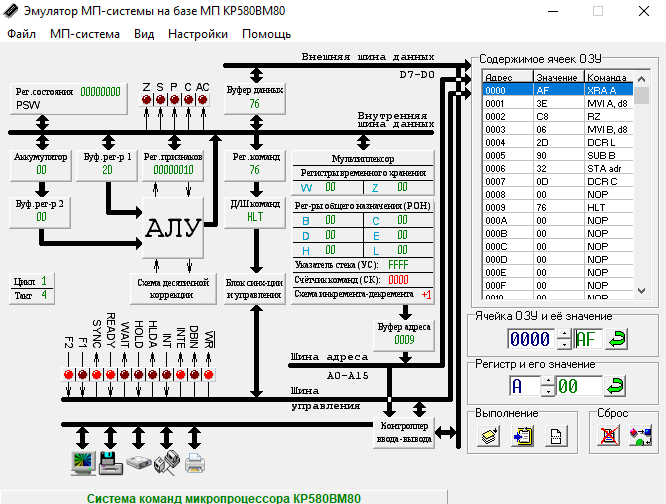


Рисунок 15 – Скрин программы

Скрин программы после её выполнения представлен на рисунке 16.

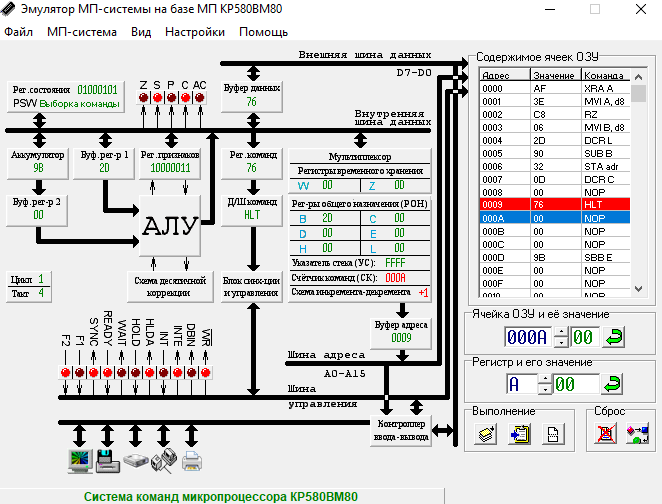


Рисунок 16 – Скрин программы

Результат вычислений был выведен в ячейку *000D*. Этот результат равен *9B16 (15510)*.

4.7.5 Заключение

Освоено выполнение вычитание двух чисел.

## Лабораторная работа №9. Сложение массива однобайтных чисел

4.9.1 Цель лабораторной работы

Рассмотреть особенности выполнения простейших арифметических операций над целыми числами без знака на эмуляторе МП К580, познакомиться с программированием в машинных кодах и мнемокодах, научиться пользоваться средствами управления и кнопками эмулятора.

4.9.2 Вариант задания

12310 + 5610+ 17810 + 8910 + 9010 + 9910 = 7B16 + 3816 + B216 + 5916 + 5A16 + 6316 = 63510 = 27B16.

4.9.3 Программный код

Таблица 15 – Программный код в лабараторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 21 | LXI H, d16 | Загрузить в регистры HL, адрес первого слагаемого |
| 0001 | 20 | - | Первое слагаемое дано в ячейке 0020 |
| 0002 | 00 | NOP |  |
| 0003 | 0E | MVI C, d8 | Загрузить в регистр С количество слагаемых |
| 0004 | 06 | MVI B, d8 | Количество слагаемых = 6 |
| 0005 | AF | XRA A | Очистить аккумулятор |
| 0006 | 47 | MOV B, A | Очистить регистр В |
| 0007 | 86 | ADD M | Прибавить к содержимому аккумулятора число из массива слагаемых |
| 0008 | D2 | JNC adr | Если переноса нет, то идти на М2 |
| 0009 | 0D | DCR C | M2=000D |
| 000A | 00 | NOP |  |
| 000B | 04 | INR B | Увеличить содержимое регистра В на 1 |
| 000C | B7 | ORA A | Очистить флаг переноса |
| 000D | 23 | INX H | Указать на следующий адрес слагаемого |
| 000E | 0D | DCR C | Уменьшить содержимое регистра С на 1 |
| 000F | C2 | JNZ adr | Если не все слагаемые, то идти на M1 |
| 0010 | 07 | RLC | M1=0007 |
| 0011 | 00 | NOP |  |
| 0012 | 32 | STA adr | Записать значение аккумулятора |
| 0013 | 2A | LHLD adr | в ячейку 002A |
| 0014 | 00 | NOP |  |
| 0015 | 78 | MOV A, B | Перенести значение из регистра B в регистр A |
| 0016 | 32 | STA adr | Записать значение аккумулятора |
| 0017 | 29 | DAD H | в ячейку 0029 |
| 0018 | 00 | NOP |  |
| 0019 | 76 | HLT | Остановить программу |
| 001A | 00 | NOP |  |
| 001B | 00 | NOP |  |
| 001C | 00 | NOP |  |
| 001D | 00 | NOP |  |
| 001E | 00 | NOP |  |
| 001F | 00 | NOP |  |
| 0020 | 7B | MOV A, E | Массив чисел |
| 0021 | 38 | - |
| 0022 | B2 | ORA D |
| 0023 | 59 | MOV E, C |
| 0024 | 5A | MOV E, D |
| 0025 | 63 | MOV H, E |
| 0026 | 00 | NOP |  |
| 0027 | 00 | NOP |  |
| 0028 | 00 | NOP |  |

4.9.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 17.

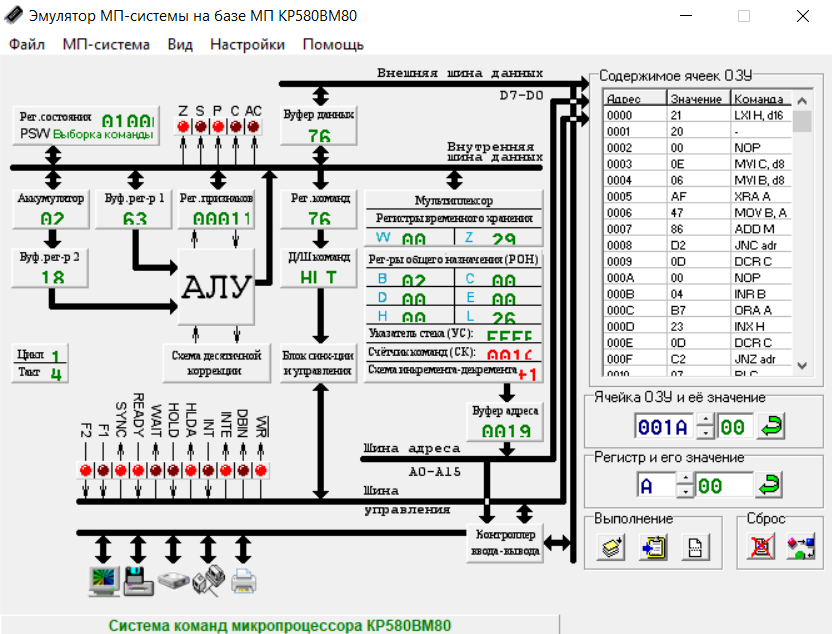


Рисунок 17 – Скрин программы

Скрин программы после её выполнения представлен на рисунке 18.

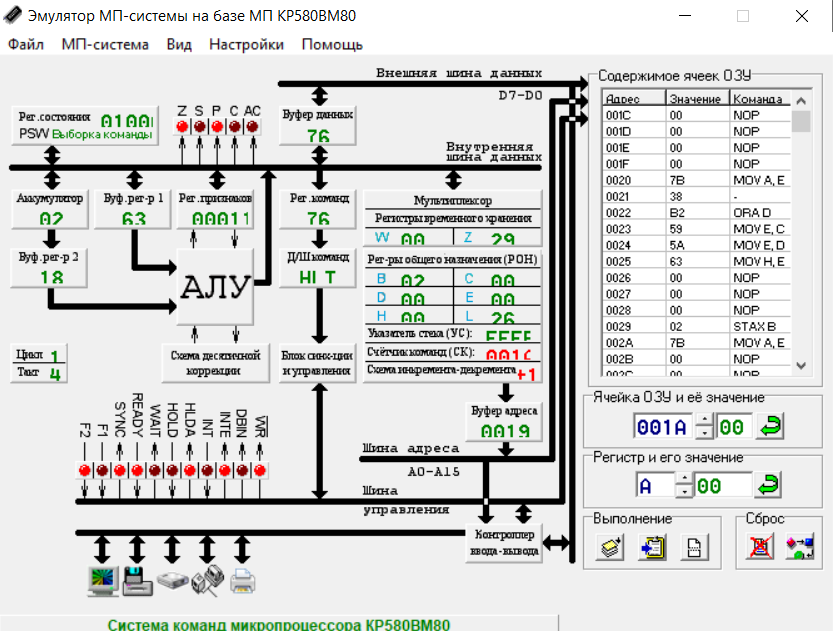


Рисунок 18 – Скрин программы

Результат вычислений был выведен в ячейки *0029* и *002A*. Этот результат равен *27B16 (63510)* – *02* выведено в ячейку *0029*, а *7B* – в *002A*.

4.7.5 Заключение

Освоена работа с массивами.

## Лабораторная работа №10. Сложение двухбайтовых десятичных чисел

4.10.1 Цель лабораторной работы

Выполнить операцию сложения двух двухбайтных чисел

4.10.2 Вариант задания

670110 + 467010 = 1A2D16 + 123E16 = 1137110 = 2C6B16.

4.10.3 Программный код

Таблица 16 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | Младший байт первого слагаемого заносится в аккумулятор 2D16 –> A |
| 0001 | 2D | DCR L |
| 0002 | 06 | MVI B, d8 | Младший байт второго слагаемого заносится в регистр B: 3E16 –> B |
| 0003 | 3E | MVI A, d8 |
| 0004 | 80 | ADD B | Сложить (2D16 + 3E16 = 6B16) |
| 0005 | 32 | STA adr | Записать значение аккумулятора |
| 0006 | 14 | INR D | В ячейку *0014* |
| 0007 | 00 | NOP |  |
| 0008 | 3E | MVI A, d8 | Старший байт первого слагаемого заносится в аккумулятор 1A16 –> A |
| 0009 | 1A | LDAX D |
| 000A | 06 | MVI B, d8 | Старший байт второго слагаемого заносится в регистр B: 1216 –> B |
| 000B | 12 | STAX D |
| 000C | 88 | ADC B | Сложить (1A16 + 1216 = 2C16) |
| 000D | 32 | STA adr | Записать значение аккумулятора |
| 000E | 15 | DCR D | В ячейку *0014* |
| 000F | 00 | NOP |  |
| 0010 | 76 | HLT |  |
| 0011 | 00 | NOP |  |

4.10.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 19.

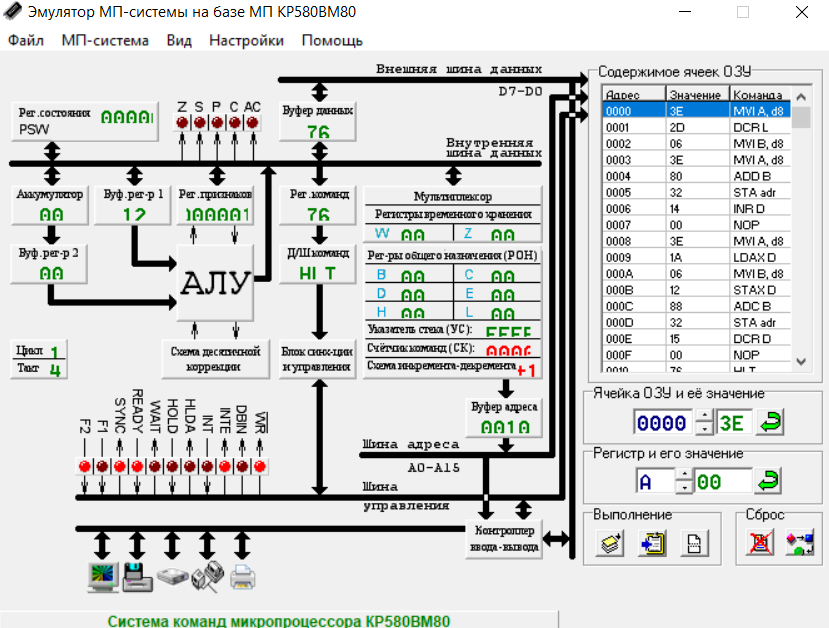


Рисунок 19 – Скрин программы

Скрин программы после её выполнения представлен на рисунке 20.

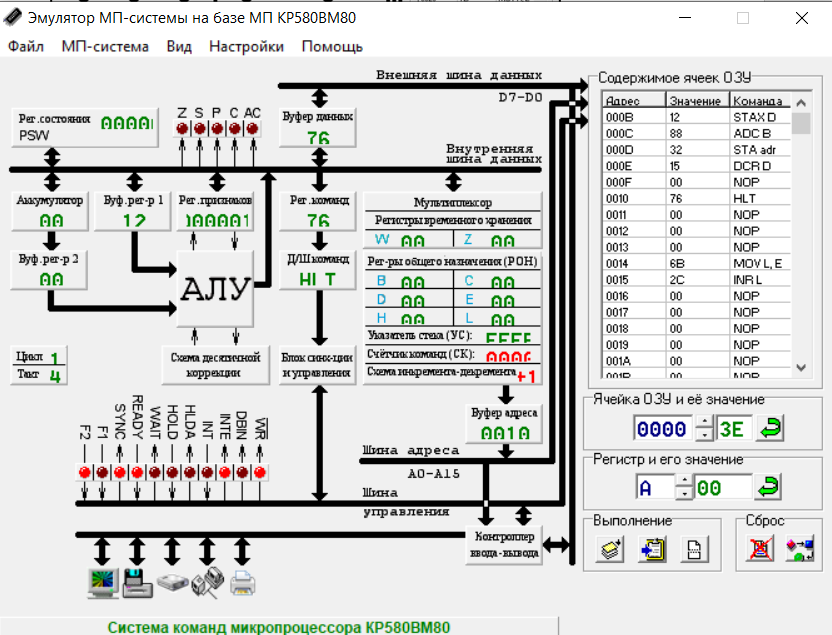


Рисунок 20 – Скрин программы

Результат вычислений был выведен в ячейки *0014* и *0015*. Этот результат равен *2C6B16 (1137110)* – *6B (Младший байт)* выведено в ячейку *0014*, а *2C (Старший байт)* – в *0015*.

4.7.5 Заключение

Освоено сложение двухбайтовых чисел.

## Лабораторная работа №11. Вычитание одинаковых по длине чисел

4.11.1 Цель лабораторной работы

Выполнить операцию вычитание двух двухбайтных чисел

4.11.2 Вариант задания

6367010 – 1795410 = F8B616 – 462216 = 4571610 = B29416.

4.11.3 Программный код

Таблица 17 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | Младший байт первого слагаемого заносится в аккумулятор B616 –> A |
| 0001 | B6 | ORA M |
| 0002 | D6 | SUI d8 | Вычесть (B616 + 2216 = 9416) |
| 0003 | 22 | SHLD adr |
| 0004 | 32 | STA adr | Записать значение аккумулятора в ячейку *0010* |
| 0005 | 10 | - |
| 0006 | 00 | NOP |  |
| 0007 | 3E | MVI A, d8 | Старший байт первого слагаемого заносится в аккумулятор F816 –> A |
| 0008 | F8 | RM |
| 0009 | DE | SBI d8 | Вычесть (F816 + 4616 = B216) |
| 000A | 46 | MOV B, M |
| 000B | 32 | STA adr | Записать значение аккумулятора в ячейку *0011* |
| 000C | 11 | LXI D, d16 |
| 000D | 00 | NOP |  |
| 000E | 76 | HLT | Остановить программу |
| 000F | 00 | NOP |  |

4.11.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 21.

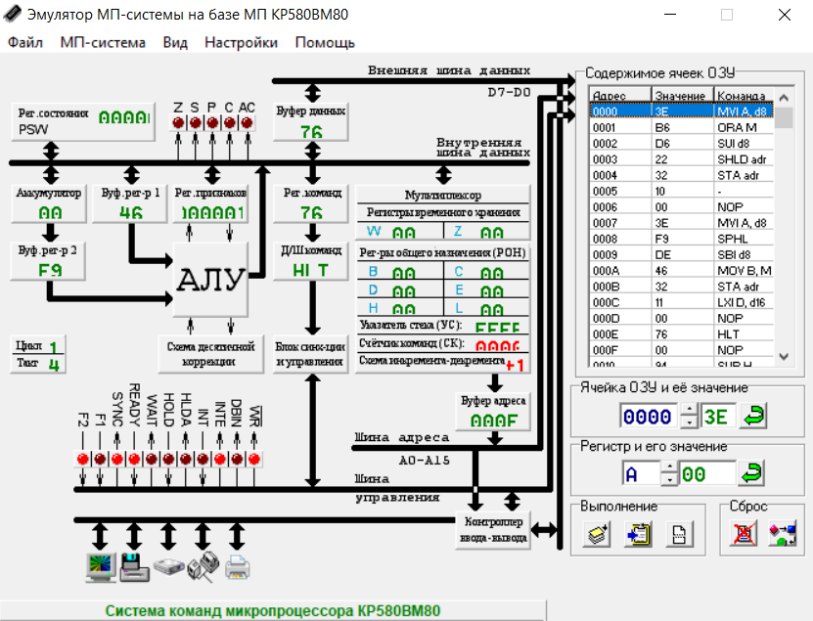


Рисунок 21 – Скрин программы

Скрин программы после её выполнения представлен на рисунке 22.

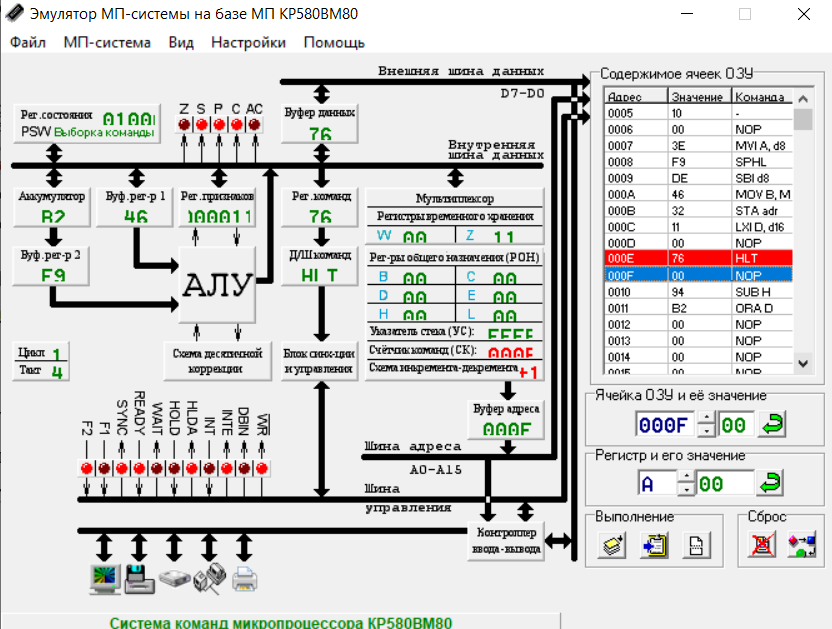


Рисунок 22 – Скрин программы

Результат вычислений был выведен в ячейки *0014* и *0015*. Этот результат равен *B29416 (15510)* – *94 (Младший байт)* выведено в ячейку *0010*, а *B2 (Старший байт)* – в *0010*.

4.11.5 Заключение

Освоено вычитание двухбайтовых чисел.

## Лабораторная работа №12. Изучение алгоритма ветвления

4.12.1 Цель лабораторной работы

Изучение алгоритма ветвления

4.12.2 Вариант задания

Вариант задания показан на рисунке 23.



Рисунок 23 – Вариант задания

4.12.3 Программный код

Таблица 18 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3A | LDA adr | Перенести значение из ячейки 0020 в аккумулятор |
| 0001 | 20 | - |
| 0002 | 00 | NOP |
| 0003 | FE | CPI d8 | Сравнить значение в аккумуляторе и 316, и если значение в аккумуляторе меньше, то перейти к команде по адресу 0011 |
| 0004 | 03 | INX B |
| 0005 | FA | JM adr |
| 0006 | 11 | LXI D, d16 |
| 0007 | 00 | NOP |
| 0008 | 47 | MOV B, A | Удвоить значение аккумулятора |
| 0009 | 80 | ADD B |
| 000A | 47 | MOV B, A | Увеличить значение аккумулятора на 1 |
| 000B | 3E | MVI A, d8 |
| 000C | 01 | LXI B, d16 |
| 000D | 00 | NOP |
| 000E | 80 | ADD B |
| 000F | C3 | JMP adr | Перейти к команде в ячейке 1416 |
| 0010 | 14 | INR D |
| 0011 | 00 | NOP |
| 0012 | 47 | MOV B, A | Удвоить значение аккумулятора |
| 0013 | 80 | ADD B |
| 0014 | 00 | NOP |  |
| 0015 | 32 | STA adr | Записать значение аккумулятора в ячейку памяти 001A |
| 0016 | 1A | LDAX D |
| 0017 | 00 | NOP |
| 0018 | 76 | HLT | Остановить программу |
| 0019 | 00 | NOP |  |

4.12.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 23.

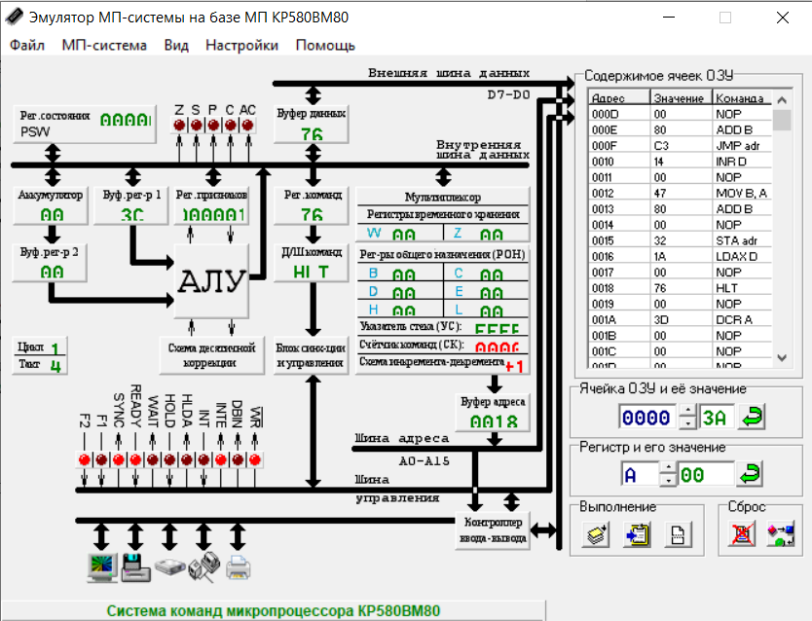


Рисунок 23 – Скрин программы

Результат выводится в ячейку 001A. Тестирование программы в таблице 19.

Таблица 19 – Тестирование программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ теста** | **Входное значение (x)** | **Ожидаемый результат** | **Фактический результат (программа, вроде, написана правильно)** | **Скрин программы** |
| 1 | 2016 (3210) | 4116 (6510) | 4116 (6510) |  |
| 2 | 316 (610) | 716 (710) | 616 (616) |  |
| 3 | 216 (210) | 416 (410) | 516 (510) |  |
| 4 | 416 (410) | 916 (910) | 816 (810) |  |
| 5 | 1B16 (2710) | 3716 (5510) | 3616 (5410) |  |
| 6 | 1A16 (2610) | 3516 (5310) | 3516 (5310) |  |

4.12.5 Заключение

Освоена работа с ветвлениями.

## Лабораторная работа №13. изучение возможностей вывода информации на экран

4.13.1 Цель лабораторной работы

Изучение возможностей вывода информации на экран

4.13.2 Программный код

Таблица 20 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | 71 => A, A=71 {цвет символа в Акк} A->PORT 00,PORT 00=71 {цвет символа в порт монитора} |
| 0001 | 71 | MOV M, C |
| 0002 | D3 | OUT N |
| 0003 | 00 | NOP |
| 0004 | 3E | MVI A, d8 | 47 => A, A=47 {номер символа в Акк, G->A} A =>PORT 00,PORT 00=47 {номер символа в порт монитора} |
| 0005 | 47 | MOV B, A |
| 0006 | D3 | OUT N |
| 0007 | 00 | NOP |
| 0008 | 3E | MVI A, d8 | 71 => A, A=71 {цвет символа в Акк} A->PORT 00,PORT 00=71 {цвет символа в порт монитора} |
| 0009 | 71 | MOV M, C |
| 000A | D3 | OUT N |
| 000B | 00 | NOP |
| 000C | 3E | MVI A, d8 | 34 => A, A=34 {номер символа в Акк, 4->A} A =>PORT 00,PORT 00=34 {номер символа в порт монитора} |
| 000D | 34 | INR M |
| 000E | D3 | OUT N |
| 000F | 00 | NOP |
| 0010 | 3E | MVI A, d8 | 71 => A, A=71 {цвет символа в Акк} A->PORT 00,PORT 00=71 {цвет символа в порт монитора} |
| 0011 | 71 | MOV M, C |
| 0012 | D3 | OUT N |
| 0013 | 00 | NOP |
| 0014 | 3E | MVI A, d8 | 34 => A, A=39 {номер символа в Акк, 4->A} A =>PORT 00,PORT 00=39 {номер символа в порт монитора} |
| 0015 | 39 | DAD SP |
| 0016 | D3 | OUT N |
| 0017 | 00 | NOP |
| 0018 | 3E | MVI A, d8 | 71 => A, A=71 {цвет символа в Акк} A->PORT 00,PORT 00=71 {цвет символа в порт монитора} |
| 0019 | 71 | MOV M, C |
| 001A | D3 | OUT N |
| 001B | 00 | NOP |
| 001C | 3E | MVI A, d8 | 34 => A, A=33 {номер символа в Акк, 4->A} A =>PORT 00,PORT 00=33 {номер символа в порт монитора} |
| 001D | 33 | INX SP |
| 001E | D3 | OUT N |
| 001F | 00 | NOP |
| 0020 | 3E | MVI A, d8 | 65 => A, A=65 {цвет символа в Акк} A->PORT 00,PORT 00=65 {цвет символа в порт монитора} |
| 0021 | 65 | MOV H, L |
| 0022 | D3 | OUT N |
| 0023 | 00 | NOP |
| 0024 | 3E | MVI A, d8 | 54 => A, A=53 {номер символа в Акк, S->A} A =>PORT 00,PORT 00=53 {номер символа в порт монитора} |
| 0025 | 53 | MOV D, E |
| 0026 | D3 | OUT N |
| 0027 | 00 | NOP |
| 0028 | 3E | MVI A, d8 | 65 => A, A=65 {цвет символа в Акк} A->PORT 00,PORT 00=65 {цвет символа в порт монитора} |
| 0029 | 65 | MOV H, L |
| 002A | D3 | OUT N |
| 002B | 00 | NOP |
| 002C | 3E | MVI A, d8 | 69 => A, A=69 {номер символа в Акк, i->A} A =>PORT 00,PORT 00=53 {номер символа в порт монитора} |
| 002D | 69 | MOV L, C |
| 002E | D3 | OUT N |
| 002F | 00 | NOP |
| 0030 | 3E | MVI A, d8 | 65 => A, A=65 {цвет символа в Акк} A->PORT 00,PORT 00=65 {цвет символа в порт монитора} |
| 0031 | 65 | MOV H, L |
| 0032 | D3 | OUT N |
| 0033 | 00 | NOP |
| 0034 | 3E | MVI A, d8 | 64 => A, A=64 {номер символа в Акк, d->A} A =>PORT 00,PORT 00=64 {номер символа в порт монитора} |
| 0035 | 64 | MOV H, H |
| 0036 | D3 | OUT N |
| 0037 | 00 | NOP |
| 0038 | 3E | MVI A, d8 | 65 => A, A=65 {цвет символа в Акк} A->PORT 00,PORT 00=65 {цвет символа в порт монитора} |
| 0039 | 65 | MOV H, L |
| 003A | D3 | OUT N |
| 003B | 00 | NOP |
| 003C | 3E | MVI A, d8 | 6F => A, A=6F {номер символа в Акк, o->A} A =>PORT 00,PORT 00=6F {номер символа в порт монитора} |
| 003D | 6F | MOV L, A |
| 003E | D3 | OUT N |
| 003F | 00 | NOP |
| 0040 | 3E | MVI A, d8 | 65 => A, A=65 {цвет символа в Акк} A->PORT 00,PORT 00=65 {цвет символа в порт монитора} |
| 0041 | 65 | MOV H, L |
| 0042 | D3 | OUT N |
| 0043 | 00 | NOP |
| 0044 | 3E | MVI A, d8 | 72 => A, A=72 {номер символа в Акк, r->A} A =>PORT 00,PORT 00=72 {номер символа в порт монитора} |
| 0045 | 72 | MOV M, D |
| 0046 | D3 | OUT N |
| 0047 | 00 | NOP |
| 0048 | 3E | MVI A, d8 | 65 => A, A=65 {цвет символа в Акк} A->PORT 00,PORT 00=65 {цвет символа в порт монитора} |
| 0049 | 65 | MOV H, L |
| 004A | D3 | OUT N |
| 004B | 00 | NOP |
| 004C | 3E | MVI A, d8 | 6F => A, A=6F {номер символа в Акк, o->A} A =>PORT 00,PORT 00=6F {номер символа в порт монитора} |
| 004D | 6F | MOV L, A |
| 004E | D3 | OUT N |
| 004F | 00 | NOP |
| 0050 | 3E | MVI A, d8 | 65 => A, A=65 {цвет символа в Акк} A->PORT 00,PORT 00=65 {цвет символа в порт монитора} |
| 0051 | 65 | MOV H, L |
| 0052 | D3 | OUT N |
| 0053 | 00 | NOP |
| 0054 | 3E | MVI A, d8 | 76 => A, A=76 {номер символа в Акк, v->A} A =>PORT 00,PORT 00=76 {номер символа в порт монитора} |
| 0055 | 76 | HLT |
| 0056 | D3 | OUT N |
| 0057 | 00 | NOP |
| 0058 | 3E | MVI A, d8 | 79 => A, A=79 {цвет символа в Акк} A->PORT 00,PORT 00=79 {цвет символа в порт монитора} |
| 0059 | 79 | MOV A, C |
| 005A | D3 | OUT N |
| 005B | 00 | NOP |
| 005C | 3E | MVI A, d8 | 76 => A, A=76 {номер символа в Акк, A->A} A =>PORT 00,PORT 00=76 {номер символа в порт монитора} |
| 005D | 41 | MOV B, C |
| 005E | D3 | OUT N |
| 005F | 00 | NOP |
| 0060 | 3E | MVI A, d8 | 79 => A, A=79 {цвет символа в Акк} A->PORT 00,PORT 00=79 {цвет символа в порт монитора} |
| 0061 | 79 | MOV A, C |
| 0062 | D3 | OUT N |
| 0063 | 00 | NOP |
| 0064 | 3E | MVI A, d8 | 6E => A, A=6E {номер символа в Акк, n->A} A =>PORT 00,PORT 00=6E {номер символа в порт монитора} |
| 0065 | 6E | MOV L, M |
| 0066 | D3 | OUT N |
| 0067 | 00 | NOP |
| 0068 | 3E | MVI A, d8 | 79 => A, A=79 {цвет символа в Акк} A->PORT 00,PORT 00=79 {цвет символа в порт монитора} |
| 0069 | 79 | MOV A, C |
| 006A | D3 | OUT N |
| 006B | 00 | NOP |
| 006C | 3E | MVI A, d8 | 74 => A, A=74 {номер символа в Акк, t->A} A =>PORT 00,PORT 00=74 {номер символа в порт монитора} |
| 006D | 74 | MOV M, H |
| 006E | D3 | OUT N |
| 006F | 00 | NOP |
| 0070 | 3E | MVI A, d8 | 79 => A, A=79 {цвет символа в Акк} A->PORT 00,PORT 00=79 {цвет символа в порт монитора} |
| 0071 | 79 | MOV A, C |
| 0072 | D3 | OUT N |
| 0073 | 00 | NOP |
| 0074 | 3E | MVI A, d8 | 6F => A, A=6F {номер символа в Акк, o->A} A =>PORT 00,PORT 00=6F {номер символа в порт монитора} |
| 0075 | 6F | MOV L, A |
| 0076 | D3 | OUT N |
| 0077 | 00 | NOP |
| 0078 | 3E | MVI A, d8 | 79 => A, A=79 {цвет символа в Акк} A->PORT 00,PORT 00=79 {цвет символа в порт монитора} |
| 0079 | 79 | MOV A, C |
| 007A | D3 | OUT N |
| 007B | 00 | NOP |
| 007C | 3E | MVI A, d8 | 6E => A, A=6E {номер символа в Акк, n->A} A =>PORT 00,PORT 00=6E {номер символа в порт монитора} |
| 007D | 6E | MOV L, M |
| 007E | D3 | OUT N |
| 007F | 00 | NOP |
| 0080 | 76 | HLT | Остановить программу |

4.13.3 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 24.

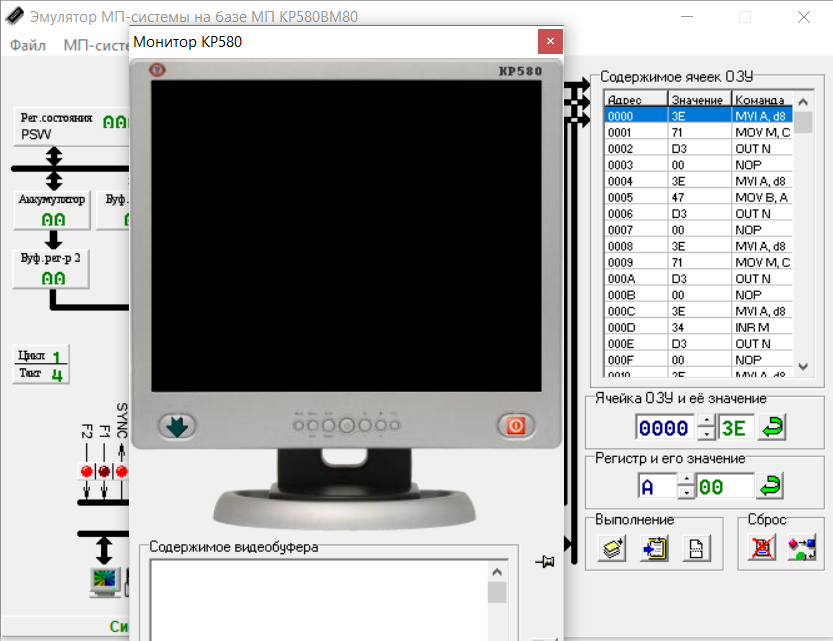


Рисунок 24 – Скрин программы

Скрин программы после её выполнения представлен на рисунке 25.

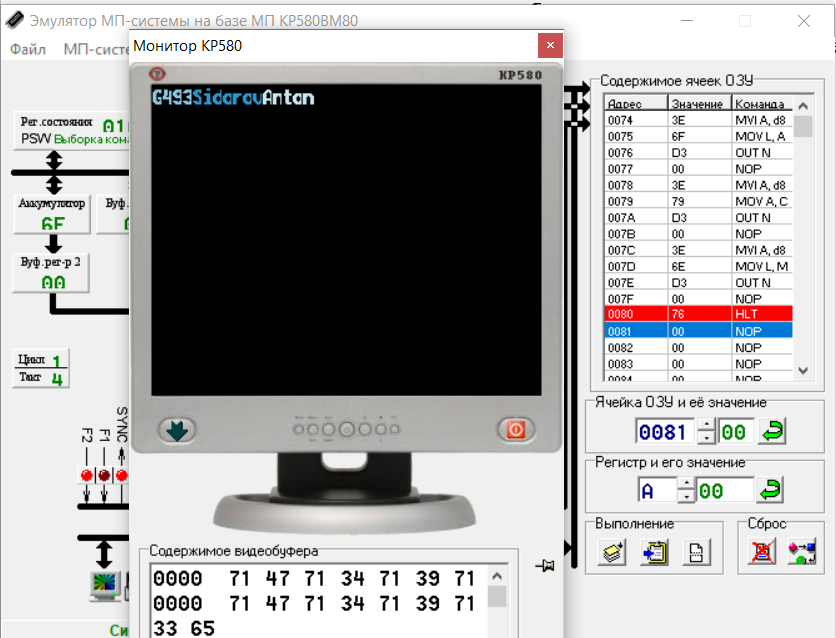


Рисунок 25 – Скрин программы

4.13.5 Заключение

Освоена работа с монитором.

## Лабораторная работа №14. изучение возможностей МП для умножения целых чисел без знака

4.14.1 Цель лабораторной работы

Изучение возможностей МП для умножения целых чисел без знака.

4.14.2 Вариант задания

2510 \* 1610 = 1916 \* 1016 = 40010 = 19016.

4.14.3 Программный код

Таблица 21 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 00 | NOP | Не используется |
| 0001 | 1E | MVI E, d8 | Записать в регистр E число 19h (2510) |
| 0002 | 19 | DAD D |
| 0003 | 0E | MVI C, d8 | Записать в регистр C число 10h (1010) |
| 0004 | 10 | - |
| 0005 | AF | XRA A | Очистить аккумулятор |
| 0006 | 67 | MOV H, A | Очистить регистр Н |
| 0007 | 6F | MOV L, A | Очистить регистр L |
| 0008 | 57 | MOV D, A | Очистить регистр D |
| 0009 | 83 | ADD E | A = A + E |
| 000A | C8 | RZ | Если А = 0, то выход |
| 000B | AF | XRA A | Очистить аккумулятор |
| 000C | 81 | ADD C | A – множитель |
| 000D | C8 | RZ | Если А = 0, то выход |
| 000E | 1F | RAR | Арифметический сдвиг аккумулятора вправо |
| 000F | D2 | JNC adr | Флаг переноса не установлен(JUMP IF NO CARRY) |
| 0010 | 13 | INX D |
| 0011 | 00 | NOP |
| 0012 | 19 | DAD D | Сдвиг множимого влево |
| 0013 | EB | XCHG | Обменять пару регистров DE с парой HL (EXCHANGE) |
| 0014 | 29 | DAD H | Прибавить к паре регистров HL содержимое пары регистров H |
| 0015 | EB | XCHG | Обменять пару регистров DE с парой HL (EXCHANGE) |
| 0016 | B7 | ORA A | Логическое сложение «ИЛИ» аккумулятора. В данном случае очистить флаг переноса |
| 0017 | C2 | JNZ adr | Флаг нуля не установлен(JUMP IF NOT ZERO) |
| 0018 | 0E | MVI C, d8 |
| 0019 | 00 | NOP |
| 001A | 7C | MOV A, H | Перенести значение из регистра H в аккумулятор |
| 001B | 32 | STA adr | Записать значение аккумулятора в ячейку памяти 0024 |
| 001C | 24 | INR H |
| 001D | 00 | NOP |
| 001E | 7D | MOV A, L | Перенести значение из регистра L в аккумулятор |
| 001F | 32 | STA adr | Записать значение аккумулятора в ячейку памяти 0025 |
| 0020 | 25 | DCR H |
| 0021 | 00 | NOP |
| 0022 | 76 | HLT | Остановить программу |

4.14.3 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 26.

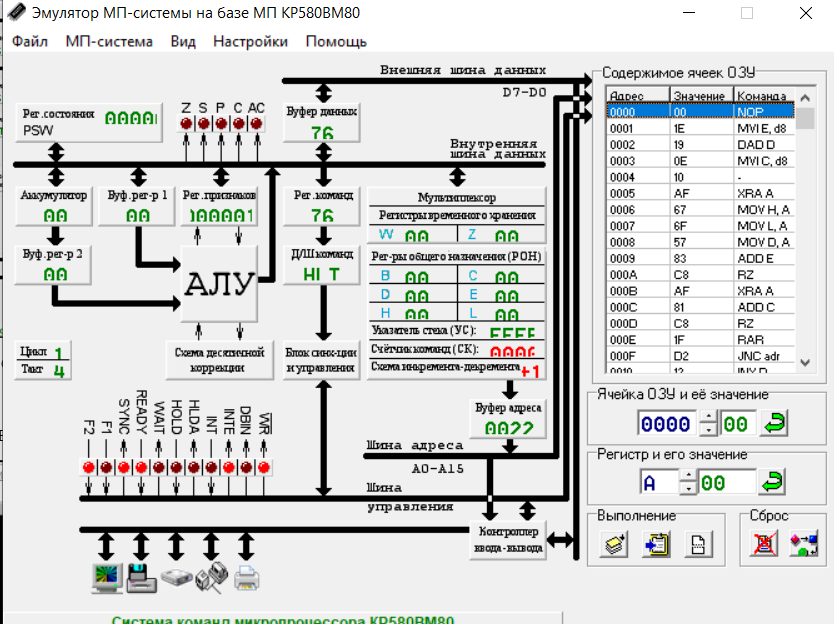


Рисунок 26 – Скрин программы

Полученное значение – *19016*. Результат выводится в ячейки памяти *0024* (*01h*) и *0025* (*90h*).

Скрин программы после её выполнения представлен на рисунке 27.

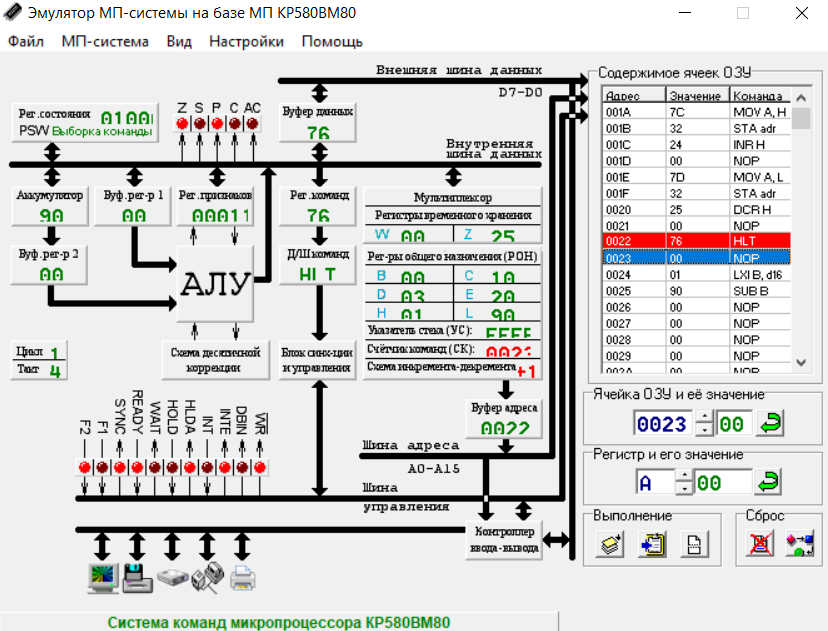


Рисунок 27 – Скрин программы

4.14.5 Заключение

Освоено умножение целых чисел без знака.

## Лабораторная работа №15. изучение алгоритмов деление двух однобайтных чисел, составление и выполнение программы деления двух однобайтных чисел без знака.

4.15.1 Цель лабораторной работы

Изучение алгоритмов деление двух однобайтных чисел, составление и выполнение программы деления двух однобайтных чисел без знака.

4.15.2 Вариант задания

20810 / 1310 = D016 \* D16 = 1610 = 1016.

4.15.3 Программный код

Таблица 22 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 11 | LXI D, d16 | Загрузка данных в пару DE: |
| 0001 | D0 | RNC | делимого в регистр E |
| 0002 | 0D | DCR C | делителя в регистр D |
| 0003 | 21 | LXI H, d16 | Загрузка данных в пару HL: |
| 0004 | 08 | - | Загрузка счётчика в регистр L |
| 0005 | 00 | NOP | Очистка регистра H |
| 0006 | 0E | MVI C, d8 | Очистка промежуточного остатка |
| 0007 | 00 | NOP |
| 0008 | 7B | MOV A, E | Загрузка делимого в аккумулятор |
| 0009 | 17 | RAL | Смещение делимого влево на один бит |
| 000A | 5F | MOV E, A | Пересылка делимого в регистр E |
| 000B | 79 | MOV A, C | Загрузка промежуточного остатка в A |
| 000C | 17 | RAL | Смещение промежуточного остатка влево |
| 000D | 92 | SUB D | Вычитание делителя от A |
| 000E | D2 | JNC adr | Если C = 0 - переход к команде в ячейку с адресом 0012 |
| 000F | 12 | STAX D |
| 0010 | 00 | NOP |
| 0011 | 82 | ADD D | Восстановление остатка |
| 0012 | 4F | MOV C, A | Пересылка промежуточного остатка в регистр C |
| 0013 | 3F | CMC | Инверсия признака переполнения |
| 0014 | 7C | MOV A, H | Запись доли в A |
| 0015 | 17 | RAL | Смещение доли |
| 0016 | 67 | MOV H, A | Сохранение доли |
| 0017 | 2D | DCR L | Уменьшение регистра L |
| 0018 | C2 | JNZ adr | Организация цикла |
| 0019 | 08 | - |
| 001A | 00 | NOP |
| 001B | 7C | MOV A, H | Пересылка младшего байта (частного) в A |
| 001C | 32 | STA adr | Сохранение сожержимого A в памяти по адресу 0030 |
| 001D | 25 | DCR H |
| 001E | 00 | NOP |
| 001F | 79 | MOV A, C | Пересылка старшего байта (остатка) в A |
| 0020 | 32 | STA adr | Сохранение сожержимого A в памяти по адресу 0031 |
| 0021 | 26 | MVI H, d8 |
| 0022 | 00 | NOP |
| 0023 | 76 | HLT | Остановка программы |
| 0024 | 00 | NOP |  |

4.15.3 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 28.

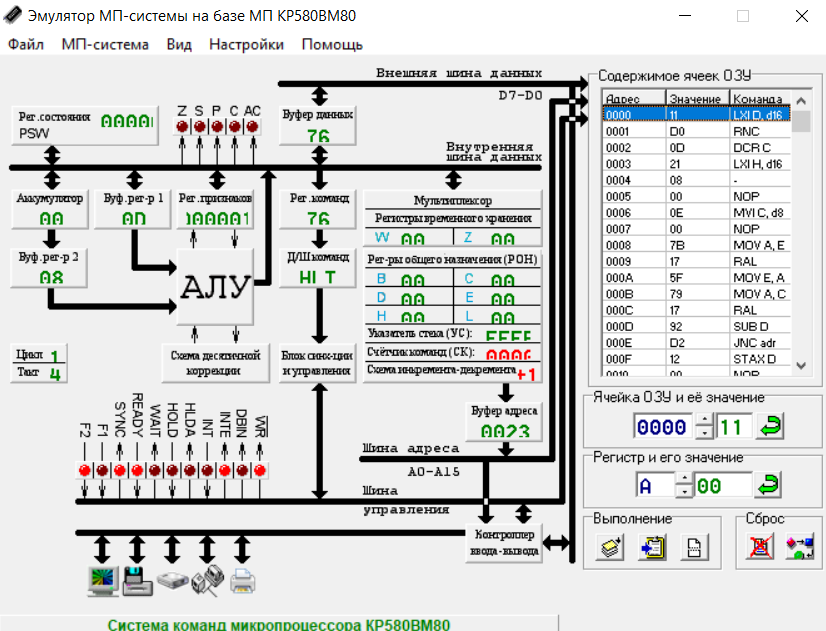


Рисунок 28 – Скрин программы

Частное (*10h*) выводится в ячейки памяти *0025*. Остаток (*00h*) – в *0026*.

Скрин программы после её выполнения представлен на рисунке 29.

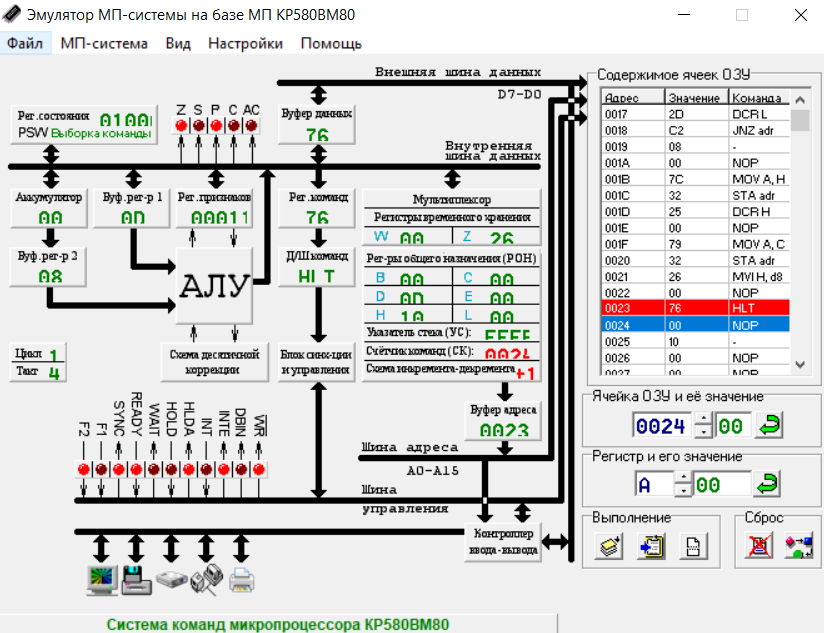


Рисунок 29 – Скрин программы

4.14.5 Заключение

Освоено деление целых чисел без знака.

## Лабораторная работа №16. Программирование арифметических задач на Ассемблере для микропроцессора К580

Цель лабораторной работы

Рассмотреть особенности выполнения простейших арифметических операций над целыми числами без знака на МП, познакомиться с программированием в машинных кодах и мнемокодах, научиться пользоваться средствами управления и клавиатурой устройств.

## Задание 1. Команды логического умножения

4.16.1.1 Цель

Исследование команд логического умножения.

4.16.1.2 Вариант задания

21110 & 3510 = D316 & 2316 = 110100112 & 001000112 = 000000112 = 310 = 316.

4.16.1.3 Программный код

Таблица 22 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | Загрузить в аккумулятор число D3h |
| 0001 | D3 | OUT N |
| 0002 | 06 | MVI B, d8 | Загрузить в регистр B число 23h |
| 0003 | 23 | INX H |
| 0004 | A0 | ANA B | D3h & 23 & = 211 & 35 = 3h = 3 (Результат остаётся в аккумуляторе) |
| 0005 | 32 | STA adr | Записать результат в память в ячейку 000A |
| 0006 | 0A | LDAX B |
| 0007 | 00 | NOP |
| 0008 | 76 | HLT | Остановить программу |

4.16.1.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 30.

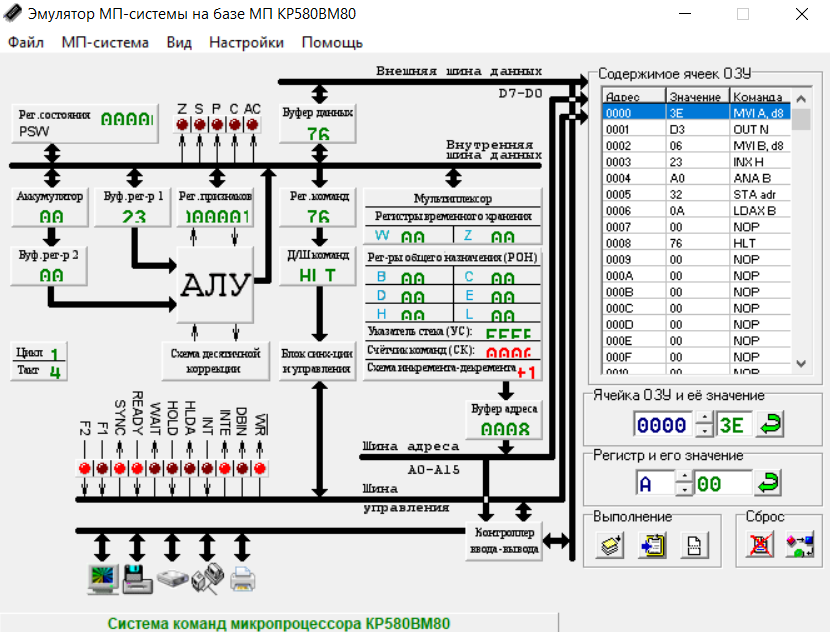


Рисунок 30 – Скрин программы

Результат (*3h*) выводится в ячейки памяти *000A*.

Скрин программы после её выполнения представлен на рисунке 31.

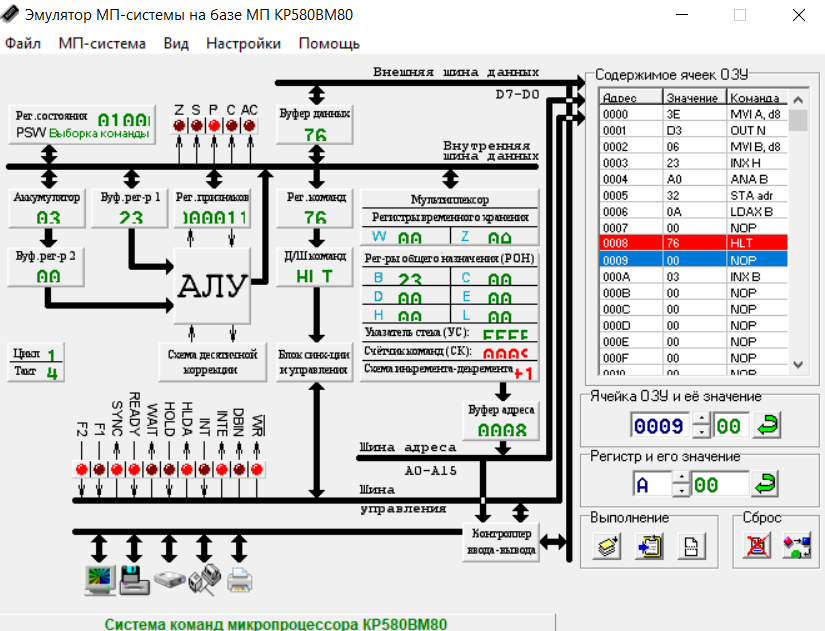


Рисунок 31 – Скрин программы

4.16.1.5 Заключение

Освоено выполнение логического умножения.

## Задание 2. Команды обнуления битов

4.16.2.1 Цель

Исследование команд маскирования.

В соответствии с таблицей кодировки символов ASCII найти код символа и при помощи маски обнулить пятый бит (в результате из кода строчной буквы получится код прописной буквы).

4.16.2.2 Вариант задания

c (63)

011000112 & 110100112 = 6316 & DF16 = 4316 = 010000112.

4.16.2.3 Программный код

Таблица 23 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | Загрузить в аккумулятор число 63h |
| 0001 | 63 | MOV H, E |
| 0002 | 06 | MVI B, d8 | Загрузить маску в регистр B - число DF |
| 0003 | DF | RST 3 |
| 0004 | A0 | ANA B | 63h & DFh = 01100011 & 11011111 = 43h = 04000011 |
| 0005 | 32 | STA adr | Загрузить результат в ячейку памяти 000C |
| 0006 | 0C | INR C |
| 0007 | 00 | NOP |
| 0008 | 76 | HLT | Остановить программу |

4.16.2.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 32.

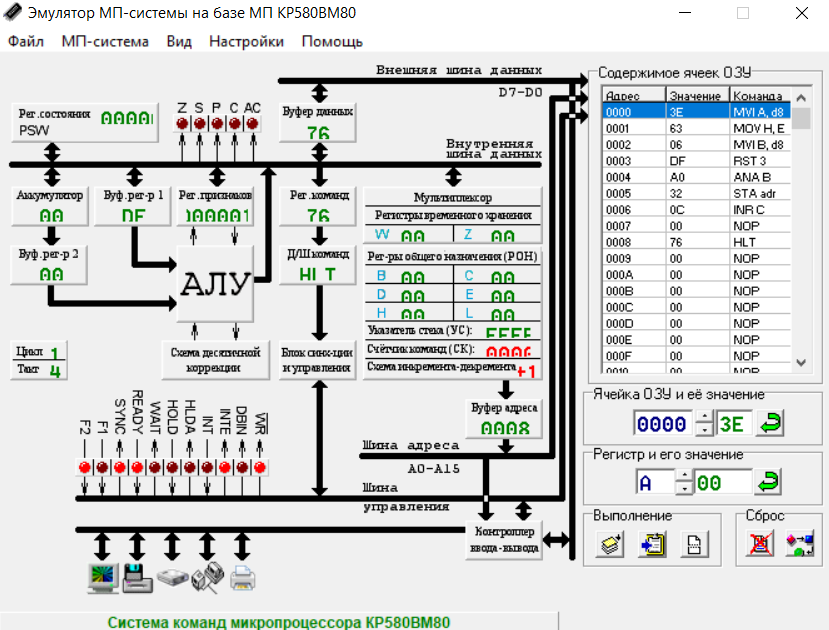


Рисунок 32 – Скрин программы

Результат (4*3h*) выводится в ячейки памяти *000C*.

Скрин программы после её выполнения представлен на рисунке 33.

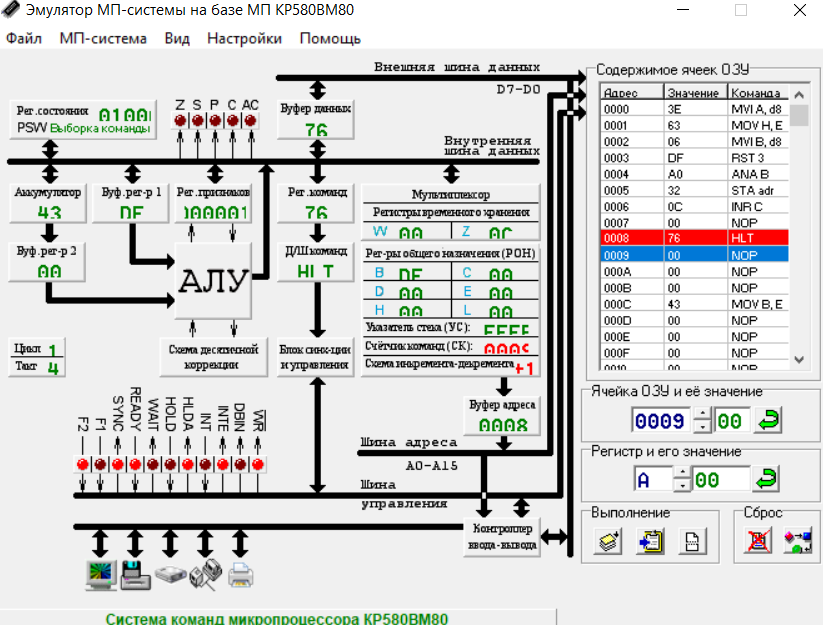


Рисунок 33 – Скрин программы

4.16.2.5 Заключение

Освоено выполнение маскирования.

## Задание 3. Команды логического сложения

4.16.3.1 Цель

Исследование команд логического сложения.

4.16.3.2 Вариант задания

18910 + 8110 = 110011012 + 010100012 = BD16 + 5116 = 25310 = FD16 = 111111012.

4.16.3.3 Программный код

Таблица 24 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | Загрузить в аккумулятор число BDh |
| 0001 | BD | CMP L |
| 0002 | 06 | MVI B, d8 | Загрузить в регистр B число 51h |
| 0003 | 51 | MOV D, C |
| 0004 | B0 | ORA B | BDh+51h=189+81=11001101+01010001  =FDh=253=11111101 |
| 0005 | 32 | STA adr | Загрузить результат в регистр 000A |
| 0006 | 0A | LDAX B |
| 0007 | 00 | NOP |
| 0008 | 76 | HLT | Остановить программу |

4.16.3.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 34.

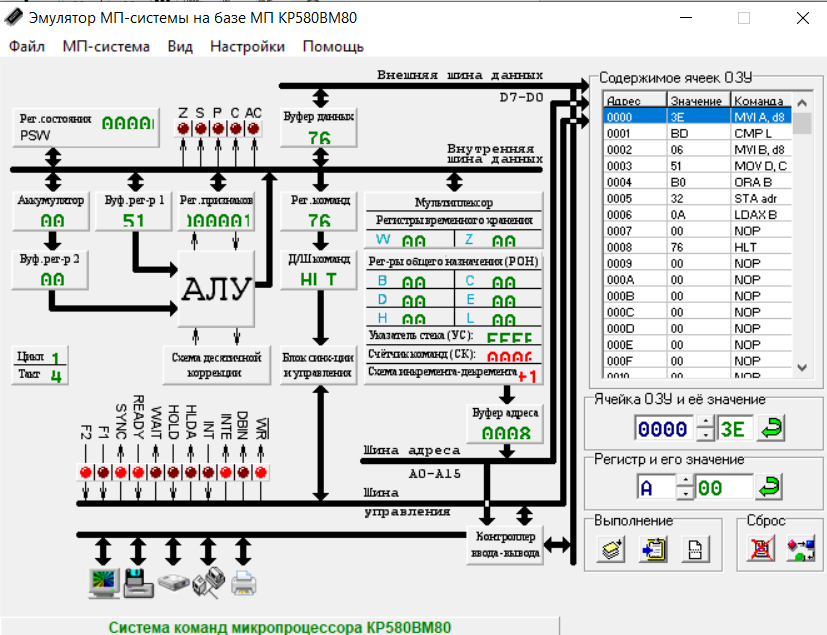


Рисунок 34 – Скрин программы

Результат (*FDh*) выводится в ячейки памяти *000A*.

Скрин программы после её выполнения представлен на рисунке 35.

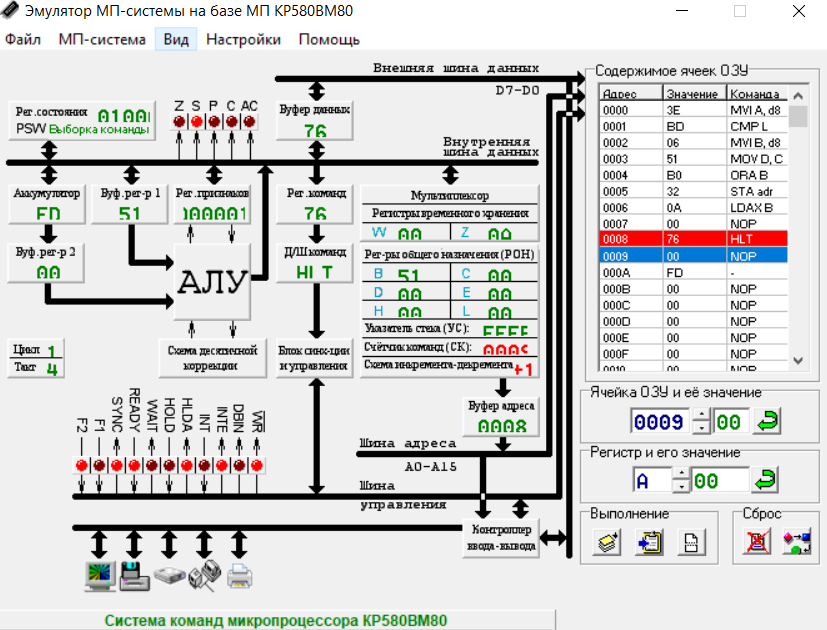


Рисунок 35 – Скрин программы

4.16.3.5 Заключение

Освоено выполнение логического сложения.

## Задание 4. Команды восстановления битов

4.16.4.1 Цель

Исследование команд логического сложения.

В соответствии с таблицей кодировки символов ASCII найти код символа и при помощи маски восстановить пятый бит (в результате из кода строчной буквы получится код прописной буквы).

4.16.4.2 Вариант задания

G (47)

010001112 + 001000002 = 4716 + 2016 = 6716 = 011001112.

4.16.4.3 Программный код

Таблица 25 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | Загрузить в аккумулятор число 47h |
| 0001 | 47 | MOV B, A |
| 0002 | 06 | MVI B, d8 | Загрузить маску (20h) в регистр B |
| 0003 | 20 | - |
| 0004 | B0 | ORA B | 47h+20h=01000111+00100000=67h=01100111 |
| 0005 | 32 | STA adr | Загрузить результат в память |
| 0006 | 0A | LDAX B |
| 0007 | 00 | NOP |
| 0008 | 76 | HLT | Остановить программу |

4.16.4.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 36.

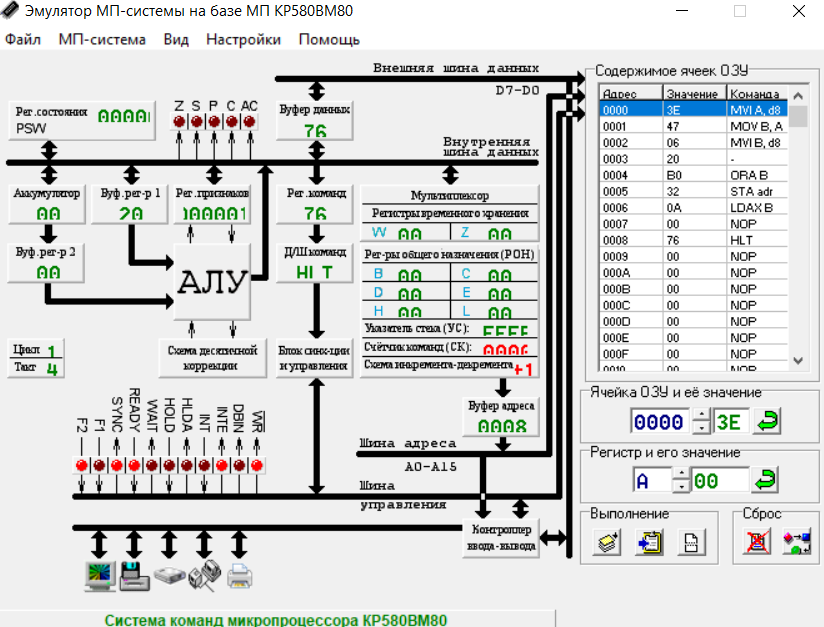


Рисунок 36 – Скрин программы

Результат (*67h*) выводится в ячейки памяти *000A*.

Скрин программы после её выполнения представлен на рисунке 37.

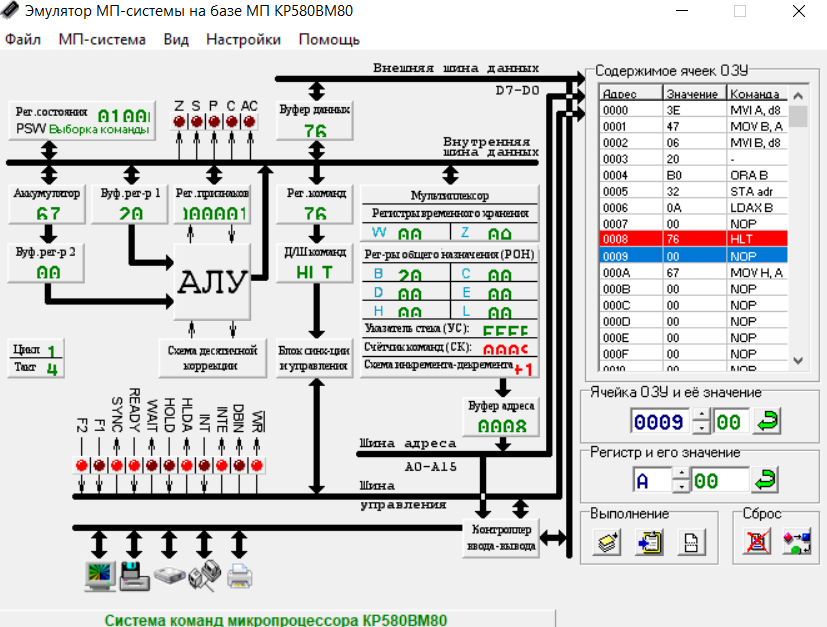


Рисунок 37 – Скрин программы

4.16.4.5 Заключение

Освоено выполнение логического сложения.

## Задание 5. Инвертирование чисел

4.16.5.1 Цель

Исследование команд инвертирования чисел.

4.16.5.2 Вариант задания

Не 16916 = не A916 = не 101010012 = 8616 = 5616 = 010101102

4.16.5.3 Программный код

Таблица 25 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | Загрузить в аккумулятор A9h |
| 0001 | A9 | XRA C |
| 0002 | 2F | CMA | не A9h = не 169 = не 10101001 = 56h = 86 = 01010110 |
| 0003 | 32 | STA adr | Загрузить результат в ячейку памяти 000A |
| 0004 | 0A | LDAX B |
| 0005 | 00 | NOP |
| 0006 | 76 | HLT | Остановить программу |

4.16.5.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 38.

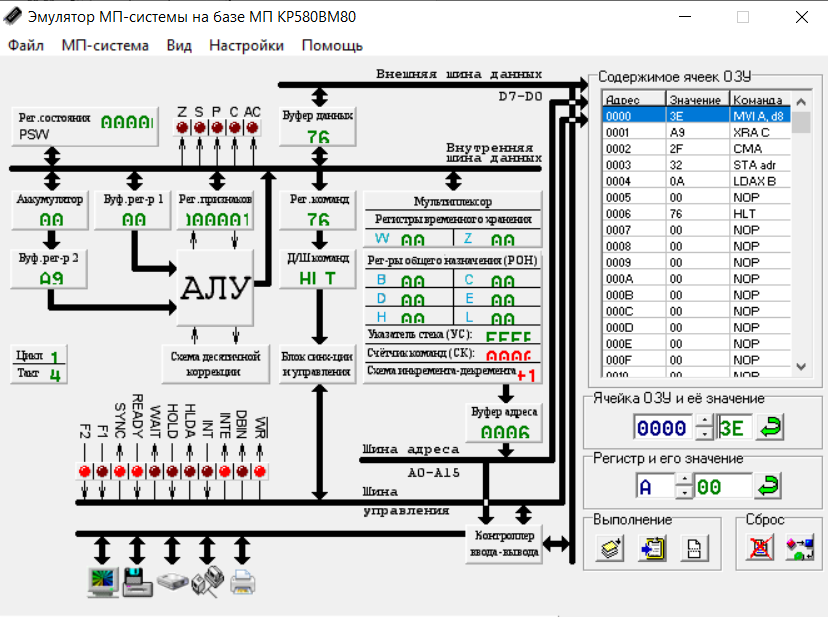


Рисунок 38 – Скрин программы

Результат (*56h*) выводится в ячейки памяти *000A*.

Скрин программы после её выполнения представлен на рисунке 39.

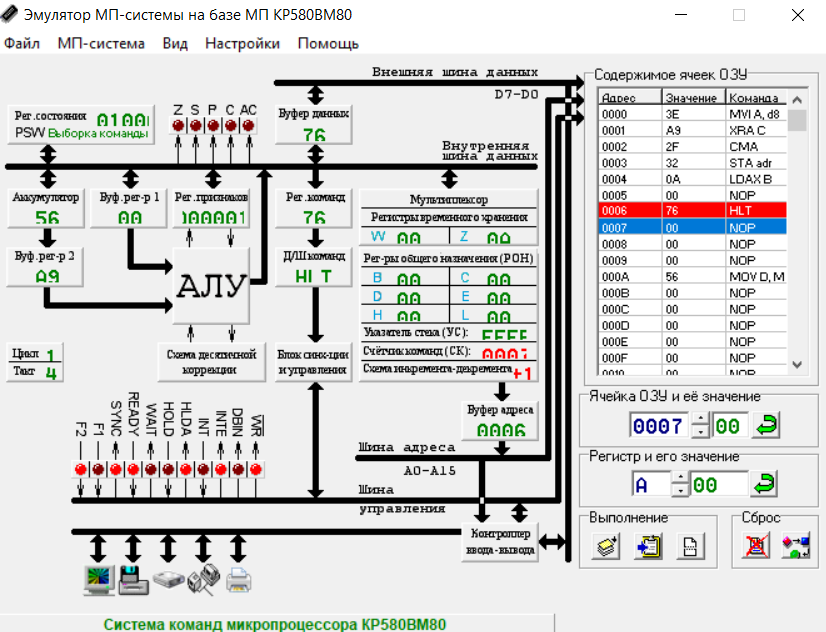


Рисунок 39 – Скрин программы

4.16.5.5 Заключение

Освоено выполнение логического сложения.

## Задание 6. Инвертирование флага переноса

4.16.6.1 Цель

Исследование команд инвертирования.

4.16.6.2 Программный код

Таблица 27 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3F | CMC | Инвертировать бит переноса |
| 0001 | 76 | HLT | Остановить программу |

4.16.6.3 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 40.

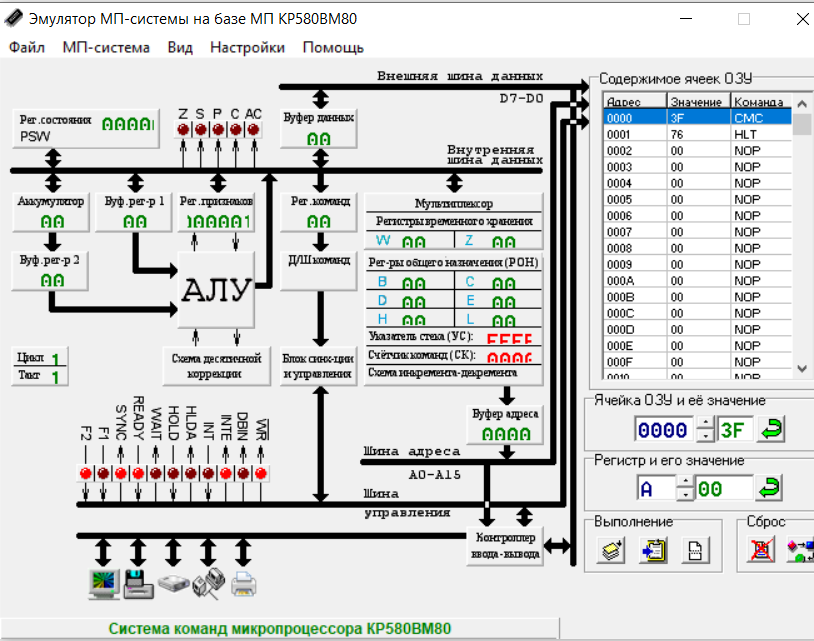


Рисунок 40 – Скрин программы

Скрин программы после её выполнения представлен на рисунке 41.

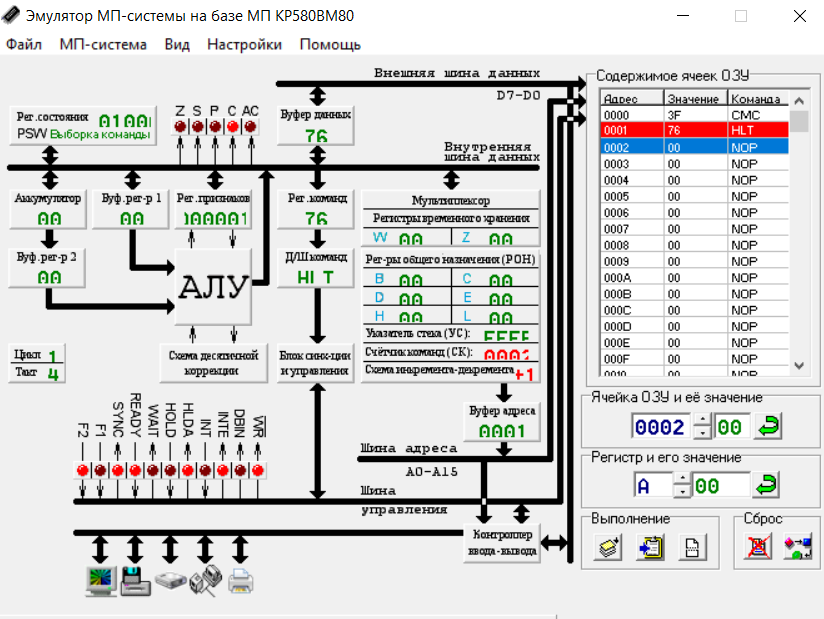


Рисунок 41 – Скрин программы

Был восстановлен бит переноса (поскольку, до выполнения команды, он был сброшен), как показано на рисунке 42.



Рисунок 42 – Скрин программы

4.16.6.4 Заключение

Освоено выполнение инвертирования бита переноса.

## Задание 7. Инвертирование заданных битов числа

4.16.7.1 Цель

Исследование команд инвертирования чисел. Инвертировать 2-й, 4-й и 5-й биты числа

4.16.7.2 Вариант задания

18910 не 5210 = BD16 не 3416 = 101111012 не 001101002 =13710 = 8916 =100010012.

4.16.7.3 Программный код

Таблица 28 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | Загрузить в аккумулятор число BDh |
| 0001 | BD | CMP L |
| 0002 | 0E | MVI C, d8 | Загрузить маску (число 34h) в регистр C |
| 0003 | 34 | INR M |
| 0004 | A9 | XRA C | BDh не 34h = 189 не 52 = 10110100 не 00110100 = 89h = 137 = 10001001 |
| 0005 | 32 | STA adr | Загрузить результат в ячейку памяти 000A |
| 0006 | 0A | LDAX B |
| 0007 | 00 | NOP |
| 0008 | 76 | HLT | Остановить программу |

4.16.7.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 43.

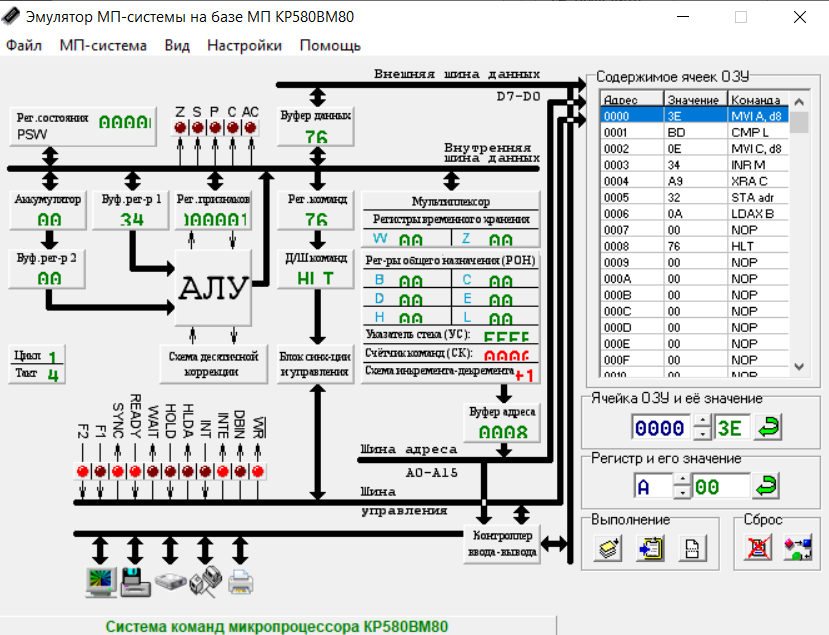


Рисунок 43 – Скрин программы

Результат (*89h*) выводится в ячейки памяти *000A*.

Скрин программы после её выполнения представлен на рисунке 44.

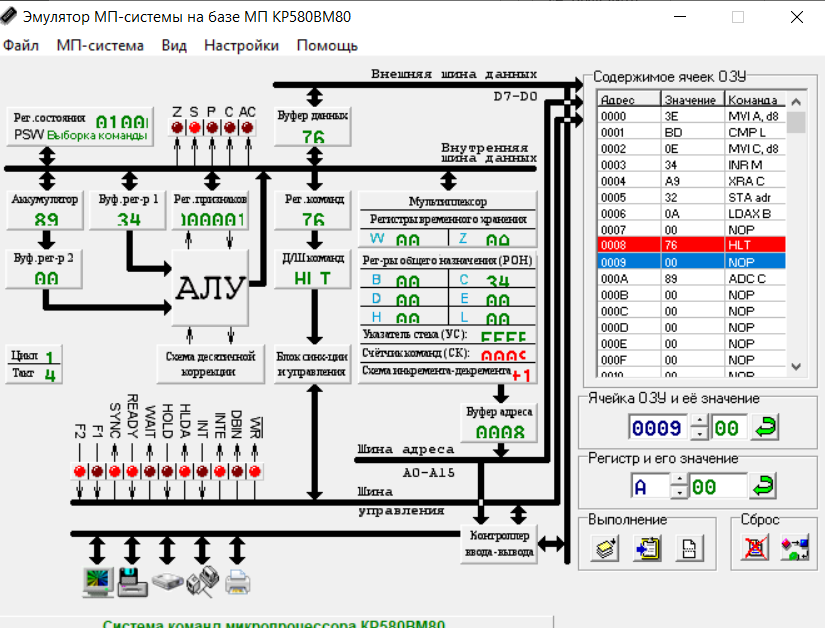


Рисунок 44 – Скрин программы

4.16.7.5 Заключение

Освоено выполнение инвертирования определённых битов числа.

## Задание 8. Сравнение чисел

4.16.8.1 Цель

Исследование команд поразрядного сравнения чисел (операция исключающая или).

4.16.8.2 Вариант задания

14110 и 19810 = 8D16 и C616 = 8D16 ⊕ C616 = 100011012 и 110001102 = 7510 = 4B16 = 010010112.

4.16.8.3 Программный код

Таблица 29 – Программный код в лабораторной работы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ Ячейки** | **Значение** | **Команда** | **Комментарий** |
| 0000 | 3E | MVI A, d8 | Записать в аккумулятор число 8Dh |
| 0001 | 8D | ADC L |
| 0002 | EE | XRI d8 | 8Dh и C6h = 141 и 189 = 10001101 и 11000110 = 4Bh = 75 = 01001011 |
| 0003 | C6 | ADI d8 |
| 0004 | 32 | STA adr | Записать результат в ячейку 000A |
| 0005 | 0A | LDAX B |
| 0006 | 00 | NOP |
| 0007 | 76 | HLT | Остановить программу |
| 0008 | 00 | NOP |  |

4.16.8.4 Выполнение программы

Скрин программы до её выполнения представлен на рисунке 45.

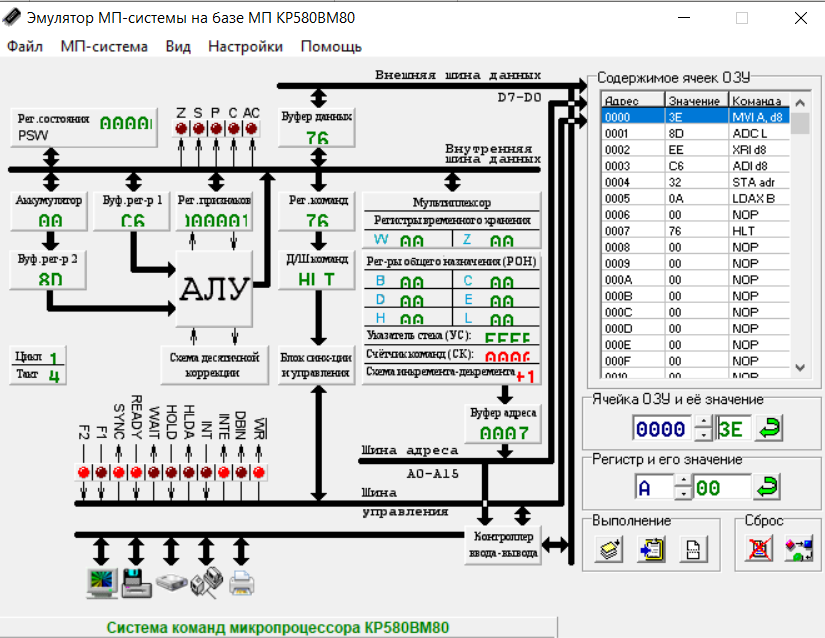


Рисунок 45 – Скрин программы

Результат (*4Bh*) выводится в ячейки памяти *000A*.

Скрин программы после её выполнения представлен на рисунке 46.

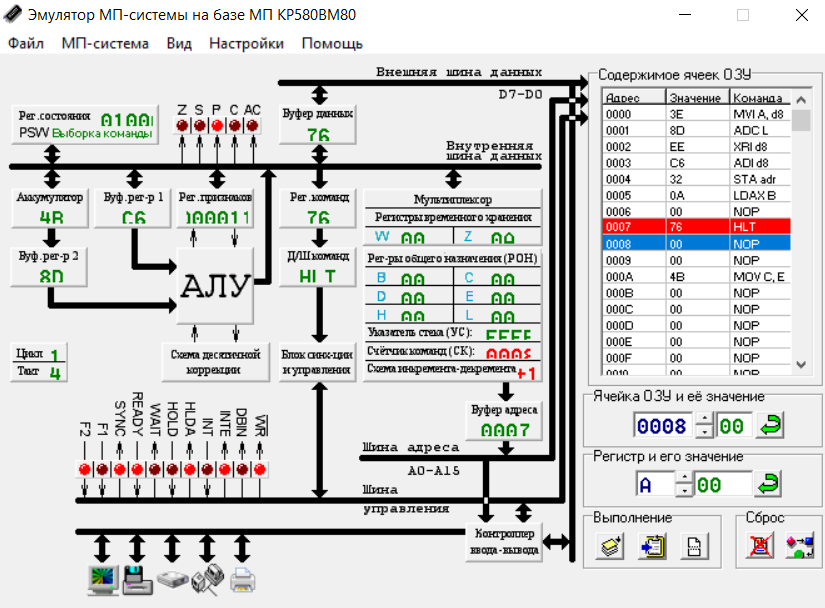


Рисунок 46 – Скрин программы

4.16.8.5 Заключение

Освоено выполнение инвертирования определённых битов числа.