Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет Программной инженерии и компьютерной техники*

**Лабораторная работа №5**

Асинхронный обмен данными с ВУ

Вариант №3107

Группа: P3131

Выполнил: Хайкин О. И.

Проверил:

Перцев Тимофей Сергеевич

Санкт-Петербург

2022г

Оглавление

[Задание 3](#_Toc100079290)

[Выполнение работы 3](#_Toc100079291)

[Составленная программа 3](#_Toc100079292)

[Составленная программа (ассемблер) 4](#_Toc100079293)

[Назначение программы и реализуемые ею функции 5](#_Toc100079294)

[Область представления исходных данных и результата 5](#_Toc100079295)

[Область допустимых значений исходных данных и результата 5](#_Toc100079296)

[Расположение в памяти ЭВМ программы, исходных данных и результата 5](#_Toc100079297)

[Адреса первой и последней выполняемых команд 5](#_Toc100079298)

[Таблица трассировки 6](#_Toc100079299)

[Заданное слово 8](#_Toc100079300)

[Вывод 8](#_Toc100079301)

# Задание

По выданному преподавателем варианту разработать программу асинхронного обмена данными с внешним устройством. При помощи программы осуществить ввод или вывод информации, используя в качестве подтверждения данных сигнал (кнопку) готовности ВУ.

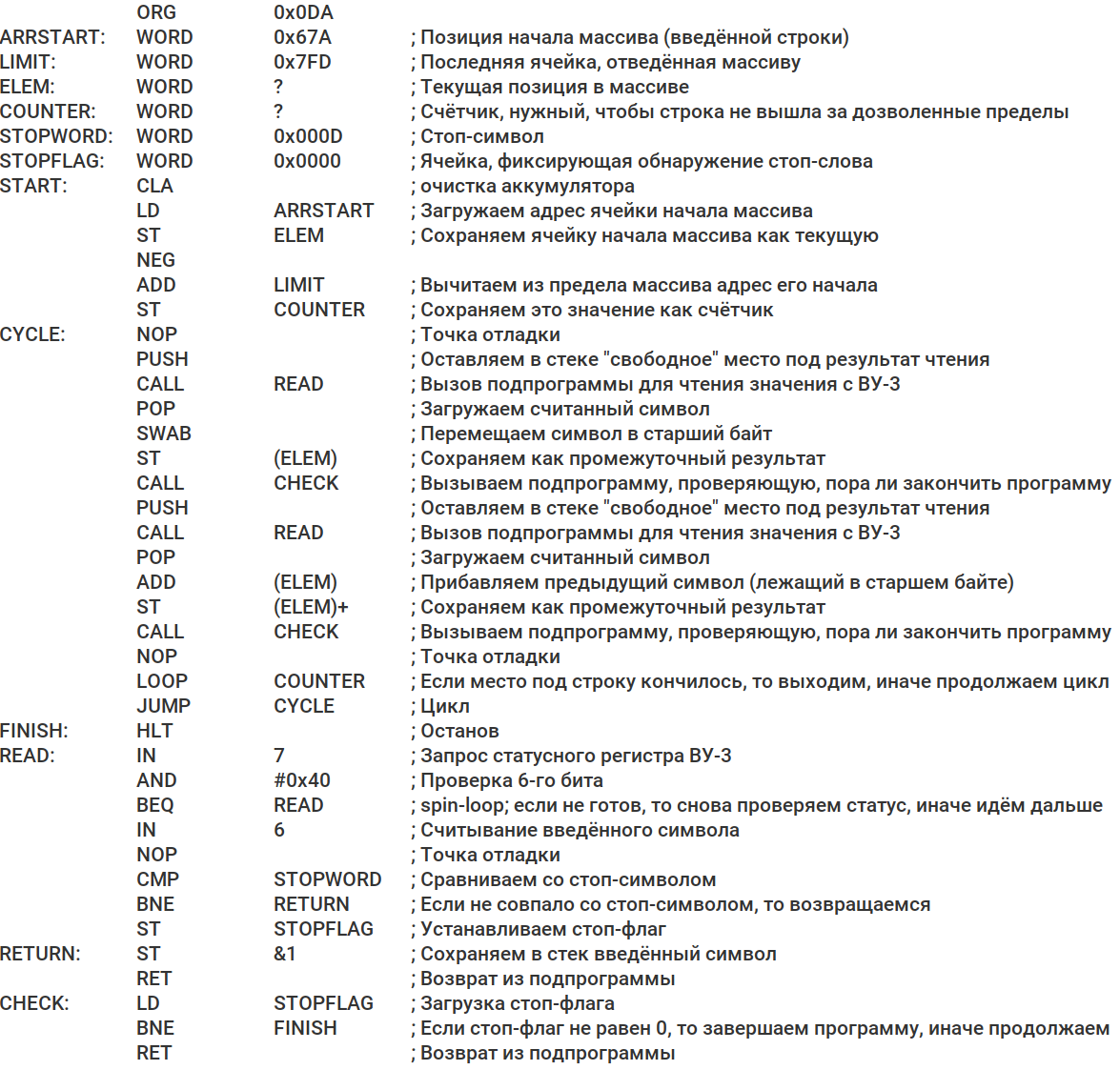
1. Программа осуществляет асинхронный ввод данных с ВУ-3
2. Программа начинается с адреса 0DA16. Размещаемая строка находится по адресу 61A16.
3. Строка должна быть представлена в кодировке ISO-8859-5.
4. Формат представления строки в памяти: АДР1: СИМВ1 СИМВ2 АДР2: СИМВ3 СИМВ4 ... СТОП\_СИМВ.
5. Ввод или вывод строки должен быть завершен по символу c кодом 0D (CR). Стоп символ является обычным символом строки и подчиняется тем же правилам расположения в памяти что и другие символы строки.

# Выполнение работы

## Составленная программа

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Код команды | Мнемоника | Комментарии |
| 0DA | 067A | число | Адрес начала массива (введённой строки) |
| 0DB | 07FD | число | Адрес последней ячейки, отведённой массиву |
| 0DC | 0000 | число | Ячейка текущей позиции в массиве |
| 0DD | 0000 | число | Счётчик ячеек массива |
| 0DE | 000D | число | Стоп-символ |
| 0DF | 0000 | число | Стоп-флаг |
| 0E0 | 0200 | CLA | Очистка аккумулятора |
| 0E1 | AEF8 | LD IP-8 | Загрузка адреса начала массива |
| 0E2 | EEF9 | ST IP-7 | Сохранение в текущую позицию |
| 0E3 | 0780 | NEG | Перевод аккумулятора в доп.код |
| 0E4 | 4EF6 | ADD IP-A | Сложение аккумулятора с адресом последней ячейки |
| 0E5 | EEF7 | ST IP-9 | Сохранение в счётчик |
| 0E6 | 0000 | NOP | Точка отладки |
| 0E7 | 0C00 | PUSH | Оставляем в стеке "свободное" место под результат чтения |
| 0E8 | DE0E | CALL IP+E | Вызов подпрограммы для чтения значения с ВУ-3 |
| 0E9 | 0800 | POP | Загружаем считанный символ |
| 0EA | 0680 | SWAB | Меняем местами старший и младший байт аккумулятора |
| 0EB | E8F0 | ST (IP-10) | Сохраняем символ в массив |
| 0EC | DE14 | CALL IP+14 | Вызов подпрограммы проверки |
| 0ED | 0C00 | PUSH | Оставляем в стеке "свободное" место под результат чтения |
| 0EE | DE08 | CALL IP+8 | Вызов подпрограммы для чтения значения с ВУ-3 |
| 0EF | 0800 | POP | Загружаем считанный символ |
| 0F0 | 48EB | ADD (IP-15) | Прибавляем предыдущий символ (лежащий в старшем байте) |
| 0F1 | EAEA | ST (IP-16)+ | Сохраняем 2 символа в массив и инкрементируем адрес |
| 0F2 | DE0E | CALL IP+E | Вызов подпрограммы проверки |
| 0F3 | 0000 | NOP | Точка отладки |
| 0F4 | 8EE8 | LOOP IP-18 | Декремент и пропуск по счётчику |
| 0F5 | CEF0 | JUMP IP-10 | Цикл |
| 0F6 | 0100 | HLT | Останов |
| 0F7 | 1207 | IN 7 | Запрос статусного регистра ВУ-3 |
| 0F8 | 2F40 | AND #0x40 | Маска для проверки 6-го бита |
| 0F9 | F0FD | BEQ IP-3 | spin-loop |
| 0FA | 1206 | IN 6 | Считывание введённого символа |
| 0FB | 0000 | NOP | Точка отладки |
| 0FC | 7EE1 | CMP IP-1F | Сравнение со стоп-символом |
| 0FD | F101 | BNE IP+1 | Пропускаем следующую команду, если символ не совпал со стоп-символом |
| 0FE | EEE0 | ST IP-20 | Устанавливаем стоп-флаг |
| 0FF | EC01 | ST SP+1 | Сохраняем в стек введённый символ |
| 100 | 0A00 | RET | Возврат из подпрограммы |
| 101 | AEDD | LD IP-23 | Загрузка стоп-флага |
| 102 | F1F3 | BNE IP-D | Идём на Останов, если стоп-флаг установлен |
| 103 | 0A00 | RET | Возврат из подпрограммы |

## Составленная программа (ассемблер)



## Назначение программы и реализуемые ею функции

Назначение: программа считывает данные, введённые с ВУ-3 в формате ISO-8859-5.

## Область представления исходных данных и результата

Исходные данные:

Адрес начала массива под строку (0DA): число, представляющее адрес ячейки БЭВМ (11-разрядное беззнаковое число)

Адрес последней ячейки, отведённой массиву (0DB): число, представляющее адрес ячейки БЭВМ (11-разрядное беззнаковое число)

Стоп-символ (0DE): 8-разрядное беззнаковое число, *желательно не совпадающее со значением какого-либо символа в кодировке* *ISO-8859-5.*

Результат (67A-…): массив ячеек, содержащий строку в кодировке ISO-8859-5. В каждой ячейке содержится 2 8-разрядных символа.

## Область допустимых значений исходных данных и результата

Адрес начала массива под строку (0DA): 104≤A≤7FD

Адрес последней ячейки, отведённой массиву (0DB): A≤B≤7FD

Стоп-символ (0DE): теоретически – 0≤S≤FF, но если брать только значения, не «занятые» символами, то 0≤S≤1F ˅ 7F≤S≤9F.

Результат (67A-…): (вводимая строка) строка из N символов, где 0≤N≤B-A

## Расположение в памяти ЭВМ программы, исходных данных и результата

**Исходные данные:**

0DA, 0DB, 0DE

**Промежуточные данные:**

0DC, 0DD, 0DF

**Программа:**

0E0 – 103

**Результат:**

67A – …

## Адреса первой и последней выполняемых команд

Первая – 0E0,

Последняя – 0F6

## Таблица трассировки

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выполняемая команда | | Содержимое регистров после выполнения команды | | | | | | | | Ячейка, содержимое которой изменилось после выполнения команды | |
| Адрес | Код | IP | CR | AR | DR | SP | BR | AC | NZVC | Адрес | Новый код |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

## Заданное слово

apple

A 41 - 0100 0001

P 50 - 0101 0000

P 50 - 0101 0000

L 4C - 0100 1100

E 45 - 0100 0101

STOP: 0D - 0000 1101

# Вывод

В ходе выполнения пятой лабораторной работы я изучил организацию асинхронного ввода-вывода в БЭВМ

ORG 0x0DA

ARRSTART: WORD 0x67A ; Позиция начала массива (введённой строки)

ELEM: WORD ? ; Текущая позиция в массиве

RES: WORD ? ; Ячейка под промежуточный результат считывания

STOPWORD: WORD 0x000D ; Стоп-символ

STOPFLAG: WORD 0x0000 ; Ячейка, фиксирующая обнаружение стоп-слова

ORG 0x0DF

START: CLA ; очистка аккумулятора

LD ARRSTART

ST ELEM ; Сохраняем ячейку начала массива как текущую

CYCLE: PUSH ; Оставляем в стеке "свободное" место под результат чтения

CALL READ ; Вызов подпрограммы для чтения значения с ВУ-3

POP ; Загружаем считанный символ

SWAB ; Перемещаем символ в старший байт

ST RES ; Сохраняем как промежуточный результат

CALL CHECK ; Вызываем подпрограмму, проверяющую, пора ли закончить программу

PUSH ; Оставляем в стеке "свободное" место под результат чтения

CALL READ ; Вызов подпрограммы для чтения значения с ВУ-3

POP ; Загружаем считанный символ

ADD RES ; Прибавляем предыдущий символ (лежащий в старшем байте)

ST RES ; Сохраняем как промежуточный результат

CALL CHECK ; Вызываем подпрограмму, проверяющую, пора ли закончить программу

LD RES ; Загружаем считанную пару символов

ST (ELEM)+ ; Сохраняем в массив и инкрементируем адрес сохранения

JUMP CYCLE ; Цикл

READ: IN 7 ; Запрос статусного регистра ВУ-3

AND #0x40 ; Проверка 6-го бита

BEQ READ ; spin-loop; если не готов, то снова проверяем статус, иначе идём дальше

IN 6 ; Считывание введённого символа

CMP STOPWORD ; Сравниваем со стоп-символом

BNE RETURN ; Если не совпало со стоп-символом, то возвращаемся

ST STOPFLAG ; Устанавливаем стоп-флаг

RETURN: ST &1 ; Сохраняем в стек введённый символ

RET ; Возврат из подпрограммы

CHECK: LD STOPFLAG ; Загрузка стоп-флага

BNE FINISH ; Если стоп-флаг не равен 0, то завершаем программу, иначе продолжаем

RET ; Возврат из подпрограммы

FINISH: LD RES ; Загрузка последнего результата

ST (ELEM) ; Сохранение последнего результата

HLT ; Останов