Отчет

**по лабораторной работе №5**

**«Асинхронный обмен данными с ВУ»**

по дисциплине «Основы профессиональной деятельности» вариант 10673

Выполнил: Караганов П.Э., группа P3110

Преподаватель: Блохина Е.Н.

Оглавление

[Текст задания 3](#_Toc196381481)

[Текст исходной программы 3](#_Toc196381482)

[Текст программы на ассемблере: 3](#_Toc196381483)

[Текст программы в таблице 4](#_Toc196381484)

[Описание программы 5](#_Toc196381485)

[Назначение программы и реализуемая ею функция 5](#_Toc196381486)

[Описание исходных данных 5](#_Toc196381487)

[Область представления 5](#_Toc196381488)

[Область допустимых значений 5](#_Toc196381489)

[Расположение в памяти БЭВМ 5](#_Toc196381490)

[Таблица трассировки 5](#_Toc196381491)

[Передаваемое сообщение: «Ноябрь?» 5](#_Toc196381492)

[Трассировка 5](#_Toc196381493)

[Вывод 6](#_Toc196381494)

# Текст задания

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 вывод данных на ВУ-3
2. Программа начинается с адреса 1F316. Размещаемая строка находится по адресу 5C316.
3. Строка должна быть представлена в кодировке Windows-1251.
4. Формат представления строки в памяти: АДР1: СИМВ2 СИМВ1 АДР2: СИМВ4 СИМВ3 ... СТОП\_СИМВ.
5. Ввод или вывод строки должен быть завершен по символу c кодом 0A (NL). Стоп символ является обычным символом строки и подчиняется тем же правилам расположения в памяти что и другие символы строки.

Реализовать ввод с клавиатуры до введения стоп-символа, после окончания ввода вывести все введенное на текстовый принтер в обратном порядке.

Все остальное согласно основному заданию.

# Текст исходной программы

## Текст программы на ассемблере:

ORG 0x1F3 ; Начинаем программу

res: WORD 0x5C3 ; Адрес результата

finish: WORD 0x0A ; Стоп-символ

bitmsk: WORD 0x00FF ; битмаска для обнуления старшего байта

START: CLA ;

s1: IN 7 ; Ожидание разрешения на ввод нечётного символа

AND #0x40 ; Зануляем все биты, кроме 6-ого

BEQ s1 ; Если AC=0, то ВУ-3 не готово к обмену

LD (res) ; Записываем в AC первые два байта слова

AND bitmsk ; Оставляем только маладший байт

OUT 6 ; Выводим на ВУ-3

chek1: CMP finish ; Проверяем на стоп-слово

BEQ exit ; Выходим если AC равен стоп-слову

s2: CLA ;

IN 7 ; Ожидание разрешения на ввод чётного символа

AND #0x40 ; Зануляем все биты, кроме 6-ого

BEQ s2 ; Если AC=0, то ВУ-3 не готово к обмену

LD (res)+ ; Вводим четный символ

SWAB ; Меняем местами старший и младший байты

AND bitmsk ; Оставляем только маладший байт

OUT 6 ; Выводим на ВУ-3

CMP finish ; Проверяем на стоп-слово

BEQ exit ; Выходим если AC равен стоп-слову

CLA ;

JUMP s1 ; Возвращаемся к вводу нового символа

exit: HLT ; Остановка

ORG 0x5C3

WORD 0xEECD ; Первые два байта

WORD 0xE1FF ; Вторые два байта

WORD 0xFCF0 ; Третьи два байта

WORD 0x0A3F ; 7 байт + стоп-слово

## Текст программы в таблице

|  |  |  |  |
| --- | --- | --- | --- |
| **Адрес** | **Код команды** | **Мнемоника** | **Комментарии** |
| 1F6 | 0200 | CLA | Очистка аккумулятора  0 → AC |
| 1F7 | 1207 | IN 7 | Ожидание разрешения на ввод нечётного символа |
| 1F8 | 2F40 | AND #40 | Зануляем все биты, кроме 6-ого |
| 1F9 | F0FD | BEQ IP-3 | Если AC=0, то ВУ-3 не готово к обмену |
| 1FA | A8F8 | LD IP-8 | Записываем в AC первые два байта слова |
| 1FB | 2EF9 | AND IP-7 | Оставляем только маладший байт c помощью битмаски |
| 1FC | 1306 | OUT 6 | Выводим на ВУ-3 |
| 1FD | 7EF7 | CMP IP-9 | Проверяем на стоп-слово |
| 1FE | F00C | BEQ IP+13 | Выходим если AC равен стоп-слову |
| **1FF** | **0200** | CLA | Очистка аккумулятора  0 → AC |
| 200 | **1207** | IN 7 | Ожидание разрешения на ввод чётного символа |
| **201** | **2F40** | AND #40 | Зануляем все биты, кроме 6-ого |
| **202** | **F0FC** | BEQ IP-4 | Если AC=0, то ВУ-3 не готово к обмену |
| **203** | **AAEF** | LD (IP-17)+ | Вводим четный символ |
| **204** | **0680** | SWAB | Меняем местами старший и младший байты |
| **205** | **2EEF** | AND IP-17 | Оставляем только маладший байт |
| **206** | **1306** | OUT 6 | Выводим на ВУ-3 |
| **207** | **7EEC** | CMP IP-20 | Проверяем на стоп-слово |
| **208** | **F002** | BEQ IP+2 | Выходим если AC равен стоп-слову |
| **209** | **0200** | CLA | Очистка аккумулятора  0 → AC |
| **20A** | **CEEC** | JUMP IP-20 | Возвращаемся к вводу нового символа |
| **20B** | **0100** | HLT | Остановка |

# Описание программы

## Назначение программы и реализуемая ею функция

Программа осуществляет посимвольный асинхронный ввод данных на ВУ-3 из памяти. Программа будет отправлять символы до тех пор, пока не встретит стоп-символ с кодировкой 0x0A, который она выведет на Ву-3 и завершит выполнение программы

## Описание исходных данных

* 1F3(res) – ссылка на результат
* 1F4(finish) – стоп-слово (0x0A)
* 1F5(bitmask) – маска для отбрасывания старшего байта
* [5C3;7FF] ∪ [000; 1F2] – строка, которую мы передаём

## Область представления

* res – 11-разрядная ячейка
* finish, bitmask – 16-разрядная константа
* [5C3;7FF] ∪ [000; 1F2] - 16-разрядные ячейки, хранящие в себе по два символа в кодировке Windows-1251

## Область допустимых значений

* res ∈ [00016; 1F216] ∪ [5C316; 7FF16]
* Введённый символ ∈ [0016; FF16]
* 5С316 ≤ STRING ≤ 5C316 + ⌊ (Длина строки) \ 2 ⌋, где (Длина строки) / 2 ≤ 7FF16 – 5C316, **иначе** STRING ∈ [00016; ⌊ (Длина строки) \ 2 ⌋ - 7FF16 – 5C316] ∪ [5C316;7FF16], где (Длина строки) ≤ 86016 (214410)
* Длина строки ∈ [0; 2144] c учётом стоп-слова (|[5C316; 7FF16] ∪ [00016; 1F216]| = 573 + 499 = 1072 ячейки памяти => макс 2144 символов с учётом стоп-слова)

## Расположение в памяти БЭВМ

* *Программа:* Ячейки (1F6) – (20B)
* *Данные для программы:* Ячейки (1F3) – (1F5)
* *Адрес первой выполняемой команды:* (1F6)
* *Адрес последней выполняемой команды:* (20B)

# Таблица трассировки

## **Передаваемое сообщение**: «Ноябрь?»

В кодировке Windows-1251: CD EE FF E1 F0 FC 3F

В кодировке UTF-8: D0 9D D0 BE D1 8F D0 B1 D1 80 D1 8C 3F

В кодировке UTF-16(LE): 1D 04 3E 04 4F 04 31 04 40 04 3C 04 3F 00

## Трассировка

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Выполненная Команда** | | **Содержание аккумуляторов процессора после выполнения команды** | | | | | | | | **Ячейка, содержимое которой изменилось после выполнения команды** | |
| Адрес | Код | IP | CR | AR | DR | SP | BR | AC | NZVC | Адрес | Новый код |
| 1F6 | 0200 | 1F7 | 0200 | 1F6 | 0200 | 000 | 01F6 | 0000 | 0100 |  |  |
| 1F7 | 1207 | 1F8 | 1207 | 1F7 | 1207 | 000 | 01F7 | 0040 | 0100 |  |  |
| 1F8 | 2F40 | 1F9 | 2F40 | 1F8 | 0040 | 000 | 0040 | 0040 | 0000 |  |  |
| 1F9 | F0FD | 1FA | F0FD | 1F9 | F0FD | 000 | 01F9 | 0040 | 0000 |  |  |
| 1FA | A8F8 | 1FB | A8F8 | 5C3 | EECD | 000 | FFF8 | EECD | 1000 |  |  |
| 1FB | 2EF9 | 1FC | 2EF9 | 1F5 | 00FF | 000 | FFF9 | 00CD | 0000 |  |  |
| 1FC | 1306 | 1FD | 1306 | 1FC | 1306 | 000 | 01FC | 00CD | 0000 |  |  |
| 1FD | 7EF6 | 1FE | 7EF6 | 1F4 | 000A | 000 | FFF6 | 00CD | 0001 |  |  |
| 1FE | F00C | 1FF | F00C | 1FE | F00C | 000 | 01FE | 00CD | 0001 |  |  |
| 1FF | 0200 | 200 | 0200 | 1FF | 0200 | 000 | 01FF | 0000 | 0101 |  |  |
| 200 | 1207 | 201 | 1207 | 200 | 1207 | 000 | 0200 | 0040 | 0101 |  |  |
| 201 | 2F40 | 202 | 2F40 | 201 | 0040 | 000 | 0040 | 0040 | 0001 |  |  |
| 202 | F0FC | 203 | F0FC | 202 | F0FC | 000 | 0202 | 0040 | 0001 |  |  |
| 203 | AAEF | 204 | AAEF | 5C3 | EECD | 000 | FFEF | EECD | 1001 | 1F3 | 05C4 |
| 204 | 0680 | 205 | 0680 | 204 | 0680 | 000 | 0204 | CDEE | 1001 |  |  |
| 205 | 2EEF | 206 | 2EEF | 1F5 | 00FF | 000 | FFEF | 00EE | 0001 |  |  |
| 206 | 1306 | 207 | 1306 | 206 | 1306 | 000 | 0206 | 00EE | 0001 |  |  |
| 207 | 7EEC | 208 | 7EEC | 1F4 | 000A | 000 | FFEC | 00EE | 0001 |  |  |
| 208 | F002 | 209 | F002 | 208 | F002 | 000 | 0208 | 00EE | 0001 |  |  |
| 209 | 0200 | 20A | 0200 | 209 | 0200 | 000 | 0209 | 0000 | 0101 |  |  |
| 20A | CEEC | 1F7 | CEEC | 20A | 01F7 | 000 | FFEC | 0000 | 0101 |  |  |

РАЗОБРАТЬСЯ КАК РАБОТЕТ:

* *IN*: DR→BR; IP→DP; BR→IP; ~0 + SP→SP, **AR**; DR→MEM(AR);

# Вывод

Во время выполнения лабораторной работы я закрепил знания по тому, как происходит асинхронный программно-управляемый обмен данными с ВУ, как в БЭВМ храниться и обрабатываются символьные строковые данные, как работают команды ввода-вывода. Я уверен мне это пригодиться в будущем

# ДОП.

## Задание

Реализовать ввод с клавиатуры до введения стоп-символа, после окончания ввода вывести все введенное на текстовый принтер в обратном порядке.

Все остальное согласно основному заданию.

## Решение

; ------------------ Ввод с клавиатуры (ВУ-8) ------------------

ORG 0x1F3 ; Начинаем программу

res: WORD 0x5C3 ; Адрес результата

finish: WORD 0x0A ; Стоп-символ

temp: WORD ? ; Временная ячейка для записи символов

cnt: WORD 0x0 ; Кол-во символов

bitmsk: WORD 0x00FF ; битмаска для обнуления старшего байта

START: CLA

in1: IN 0x19 ; Ожидание ввода нечётного символа

AND #0x40 ; Зануляем все биты, кроме 6-ого

BEQ in1 ; Если AC=0, то ВУ-3 не готово к обмену

LD (cnt)+ ; Увеличиваем кол-во числе на 1

IN 0x18 ; Ввод байта в AC

ST (res) ; Загружаем в результат значение нечётного символа

chek1: CMP finish ; Проверяем на стоп-слово

BEQ exit\_ ; Выходим если AC равен стоп-слову

in2: IN 0x19 ; Ожидание ввода чётного символа

AND #0x40 ; Зануляем все кроме 6-го бита

BEQ in2 ; Если не готово, то возвращаемся к ожиданию

LD (cnt)+ ; Увеличиваем кол-во числе на 1

IN 0x18 ; Ввод байта в AC

ST temp ; Загружаем значение чётного байта во временную ячейку

SWAB ; Загружаем в результат чётный байт

OR (res) ; Загружаем в результат чётный байт

ST (res) ; Загружаем в результат чётный байт

chek2: LD temp ;

CMP finish ;

BEQ exit ;

LD (res)+ ; увеличиваем ссылку на 1

JUMP in1 ; Возвращаемся к вводу нового символа

exit\_: LD -(res) ; Уменьшаем ссылку на 1 так как нам нужно чтобы сслыка указывала на сразу на символы, а не на стоп-слово

exit: LD -(cnt) ; Уменьшаем кол-во символов, так как конец ввода - это не символ изначальной последовательности

; ------------------ Вывод на принтер (ВУ-5) ------------------

LD cnt ; Загружаем в AC кол-во символов

AND #0x1 ; Обнуляем всё кроме первого бита

BEQ out2 ; Если получаем AC = 0, то число чётно и мы прыгаем на out2

out1: IN 0xD ; Ожидание разрешения на ввод нечётного символа

AND #0x40 ; Зануляем все биты, кроме 6-ого

BEQ out1 ; Если AC=0, то ВУ-5 не готово к обмену

LD (res) ; Записываем в AC первые два байта слова

AND bitmsk ; Оставляем только маладший байт

OUT 0xC ; Выводим на принтер (ВУ-5)

LD -(res) ; Просто уменьшаем ссылку на символы на 1

LD -(cnt) ; Уменьшаем кол-во символов на 1

LD cnt ; Загружаем cnt в AC

BEQ exit\_o ; Выходим если мы вывели все символы (AC = 0)

out2: IN 0xD ; Ожидание разрешения на ввод чётного символа

AND #0x40 ; Зануляем все биты, кроме 6-ого

BEQ out2 ; Если AC=0, то ВУ-5 не готово к обмену

LD (res) ; Вводим четный символ

SWAB ; Меняем местами старший и младший байты

AND bitmsk ; Оставляем только маладший байт

OUT 0xC ; Выводим на принтер (ВУ-5)

LD -(cnt) ; Уменьшаем кол-во символов на 1

LD cnt ; Загружаем cnt в AC

BEQ exit\_o ; Выходим если мы вывели все символы (AC = 0)

JUMP out1 ; Ввыводим новый символ

exit\_o: HLT ;

;------------------------------------------ДОП------------------------------------------

;Реализовать ввод с клавиатуры до введения стоп-символа, после окончания ввода вывести все введенное на текстовый

;принтер в обратном порядке.Все остальное согласно основному заданию.

; ------------------ Ввод с клавиатуры (ВУ-8) ------------------

ORG 0x1F3 ; Начинем программу

res: WORD 0x5C3 ; Адрес результата

finish: WORD 0x0A ; Стоп-символ

temp: WORD ? ; Временная ячейка для записи символов

cnt: WORD 0x0 ; Кол-во символов

bitmsk: WORD 0x00FF ; битмаска для обнуления старшего байта

START: CLA

in1: IN 0x19 ; Ожидание ввода нечётного символа

AND #0x40 ; Зануляем все биты, кроме 6-ого

BEQ in1 ; Если AC=0, то ВУ-3 не готово к обмену

LD (cnt)+ ; Увеличиваем кол-во числе на 1

IN 0x18 ; Ввод байта в AC

ST (res) ; Загружаем в результат значение нечётного символа

chek1: CMP finish ; Проверяем на стоп-слово

BEQ exit\_ ; Выходим если AC равен стоп-слову

in2: IN 0x19 ; Ожидание ввода чётного символа

AND #0x40 ; Зануляем все кроме 6-го бита

BEQ in2 ; Если не готово, то возвращаемся к ожиданию

LD (cnt)+ ; Увеличиваем кол-во числе на 1

IN 0x18 ; Ввод байта в AC

ST temp ; Загружаем значение чётного байта во временную ячейку

SWAB ; Загружаем в результат чётный байт

OR (res) ; Загружаем в результат чётный байт

ST (res) ; Загружаем в результат чётный байт

chek2: LD temp ;

CMP finish ;

BEQ exit ;

LD (res)+ ; увеличиваем ссылку на 1

JUMP in1 ; Возвращаемся к вводу нового символа

exit\_: LD -(res) ; Уменьшаем ссылку на 1 так как нам нужно чтобы сслыка указывала на сразу на символы, а не на стоп-слово

exit: LD -(cnt) ; Уменьшаем кол-во символов, так как конец ввода - это не символ изначальной последовательности

; ------------------ Вывод на принтер (ВУ-5) ------------------

LD cnt ; Загружаем в AC кол-во символов

AND #0x1 ; Обнуляем всё кроме первого бита

BEQ out2 ; Если получаем AC = 0, то число чётно и мы прыгаем на out2

out1: IN 0xD ; Ожидание разрешения на ввод нечётного символа

AND #0x40 ; Зануляем все биты, кроме 6-ого

BEQ out1 ; Если AC=0, то ВУ-5 не готово к обмену

LD (res) ; Записываем в AC первые два байта слова

AND bitmsk ; Оставляем только маладший байт

OUT 0xC ; Выводим на принтер (ВУ-5)

LD -(res) ; Просто уменьшаем ссылку на символы на 1

LD -(cnt) ; Уменьшаем кол-во символов на 1

LD cnt ; Загружаем cnt в AC

BEQ exit\_o ; Выходим если мы вывели все символы (AC = 0)

out2: IN 0xD ; Ожидание разрешения на ввод чётного символа

AND #0x40 ; Зануляем все биты, кроме 6-ого

BEQ out2 ; Если AC=0, то ВУ-5 не готово к обмену

LD (res) ; Вводим четный символ

SWAB ; Меняем местами старший и младший байты

AND bitmsk ; Оставляем только маладший байт

OUT 0xC ; Выводим на принтер (ВУ-5)

LD -(cnt) ; Уменьшаем кол-во символов на 1

LD cnt ; Загружаем cnt в AC

BEQ exit\_o ; Выходим если мы вывели все символы (AC = 0)

JUMP out1 ; Ввыводим новый символ

exit\_o: HLT ;