Федеральное государственное автономное образовательное учреждение

высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа по ОПД №5

Вариант 6504

Выполнил  
Пчелкин Илья Игоервич

P3106

Проверила

Ткешелашвили Н.М.

Санкт-Петербург 2025

Оглавление

[Текст задания 3](#_Toc194447003)

[Область представления 5](#_Toc194447004)

[ОДЗ 5](#_Toc194447005)

[Код программы на ассемблере БЭВМ 6](#_Toc194447006)

[Трассировка 7](#_Toc194447007)

[Вывод 8](#_Toc194447008)

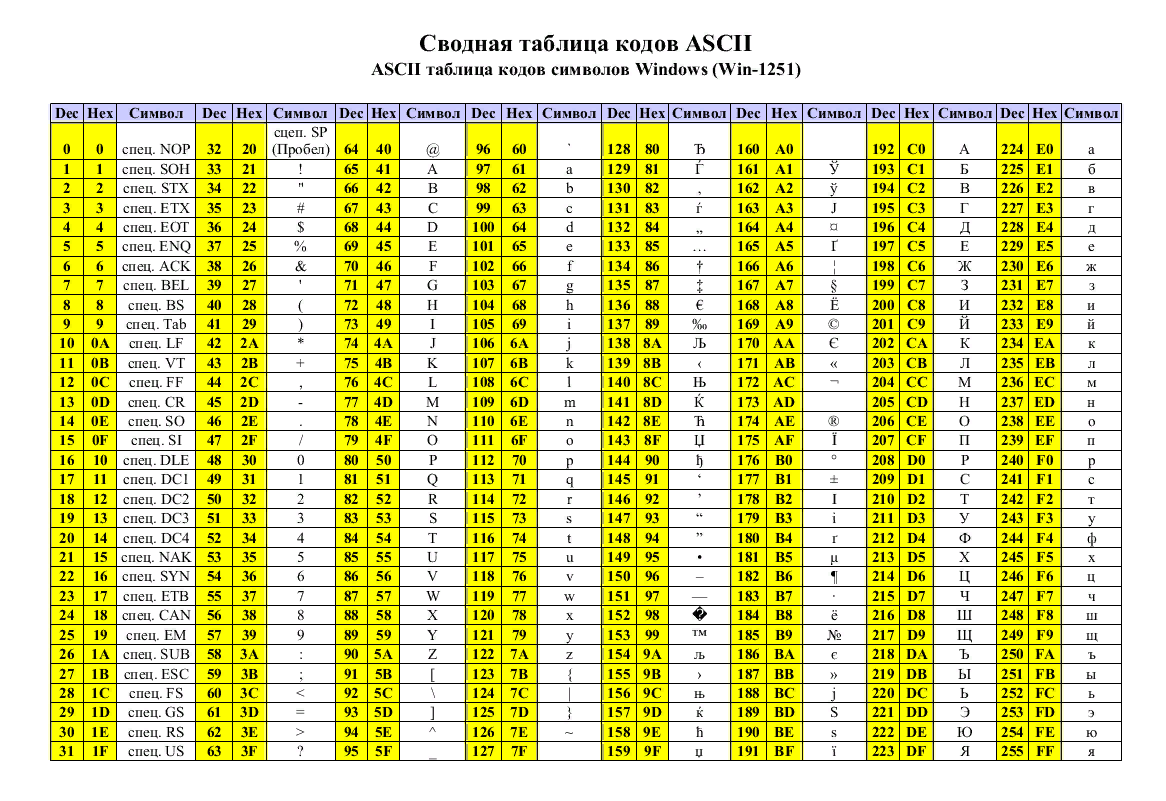
### Текст задания

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-1
2. Программа начинается с адреса 22816. Размещаемая строка находится по адресу 55216.
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 0D (CR). Стоп символ является обычным символом строки и подчиняется тем же правилам расположения в памяти что и другие символы строки.



### Область представления



first\_symbol\_address – 11-разрядный адрес первой ячейки с символами

current\_symbol – 11-разрядный адрес текущей ячейки с символами

encoded\_string ∊ [2116; FF16] – 16-разрядная ячейка для хранения текущих двух символов; Старший байт – код второго символа, младший байт – код первого символа.

stop\_symbol – 8-разрядный стоп символ

mask = FF16 – маска (нужна в ходе программы для сравнения текущего символа со стоп символом)

n – длина слова/2

Максимальная длина передаваемого слова = 2946

2\*(7FF – 23E) = 2\*5C1 = B8216 = 294610

### ОДЗ

[

старший байт encoded\_string ∊ [2116; FF16]

младший байт encoded\_string ∊ [2116; FF16]

mask = FF16

Передаваемое слово:

ЯСЕНЬ

Win-1251: DFD1 CDC5 0DDC

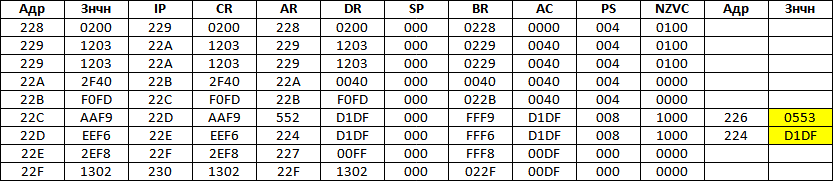
UTF-16LE: FFFE 2F04 2104 1504 1D04 2C04 000D

UTF-8: A1AF 9D95 0DAC

### Код программы на ассемблере БЭВМ

ORG 0x223  
word\_beginning: WORD 0x552  
encoded\_string: WORD ?  
stop\_symbol: WORD 0x000D  
current\_symbol: WORD 0x552  
mask: WORD 0x00FF  
  
  
start: CLA  
  
symbol\_1: IN 3  
 AND #0x40  
 BEQ symbol\_1  
 LD (current\_symbol)+  
 ST encoded\_string  
 AND mask  
 OUT 2  
 CMP stop\_symbol  
 BEQ end\_program  
  
symbol\_2: IN 3  
 AND #0x40  
 BEQ symbol\_2  
 LD encoded\_string  
 SWAB  
 AND mask  
 OUT 2  
 CMP stop\_symbol  
 BEQ end\_program  
 JUMP start  
  
end\_program: LD word\_beginning  
 ST current\_symbol  
 HLT  
  
  
ORG 0x552  
WORD 0xD1DF ; С, Я  
ORG 0x553  
WORD 0xCDC5 ; Н, Е  
ORG 0x554  
WORD 0x0DDC ; СТОП\_СИМВ, Ь

### Трассировка для первых двух символов С, Я: D1DF



### Дополнительное задание

Калькулятор. Умножение знаковых чисел. Ввод пары чисел с ВУ-9 (цифровая клавиатура), разделитель чисел - операция умножения. По нажатию "=" вывод результата на ВУ-7 (семисегментный индикатор).

### Код программы на ассемблере БЭВМ

a: word 0x0 ; первый множитель  
b: word 0x0 ; второй множитель  
res: word 0x0 ; резульат умножения  
  
t: word 0x0 ; количество десятков в числе  
  
sign: word 0x0 ; знак результата  
minus: word 0xA ; код символа "-"  
equate: word 0xF ; код символа "="  
multiplication: word 0xD ; код символа "\*"

START:  
 cla  
 ; очистка значений  
 ld sign  
 cla  
 st sign  
 ld t  
 cla  
 st t  
  
  
  
; цикл ввода a  
input\_a:  
 in 0x1C  
 cmp minus  
 beq handle\_minus\_a  
 cmp #0x0  
 blt input\_a ; если in < 0 то не цифра  
 cmp #0xB  
 bge input\_a ; если in >= B то не цифра и не минус  
 st a ; сохраняем первый символ - a  
 jump multiply\_input  
  
; изменение знака  
handle\_minus\_a:  
 ld sign  
 cmp #0x1  
 beq set\_zero\_a  
 ld #0x1 ; установка "-"  
 jump save\_sign\_a  
set\_zero\_a: ; установка "+"  
 ld #0x0  
save\_sign\_a: ; сохранение знака переход к вводу a  
 st sign  
 jump input\_a  
  
  
; цикл ввода "\*"  
multiply\_input:  
 in 0x1C ; ввод с клавиатуры  
 cmp multiplication  
 bne input\_a  
  
  
  
; цикл ввода b  
input\_b:  
 in 0x1C  
 cmp minus  
 beq handle\_minus\_b ;  
 cmp #0x0  
 blt input\_b ; если in < 0 то не цифра  
 cmp #0xB  
 bge input\_b ; если in >= B то не цифра и не минус  
 st b ; сохраняем первый символ - a  
 jump equate\_input  
  
; изменение знака  
handle\_minus\_b:  
 ld sign  
 cmp #0x1  
 beq set\_zero\_b  
 ld #0x1 ; установка "-"  
 jump save\_sign\_b  
  
set\_zero\_b: ; установка "+"  
 ld #0x0  
save\_sign\_b: ; сохранение знака переход к вводу b  
 st sign  
 jump input\_b  
  
  
  
; цикл ввода "="  
equate\_input:  
 in 0x1C  
 cmp equate  
 bne input\_b  
  
  
; умножение  
multiply:  
 ld a  
 push ; кладем а на стек => a = &2  
 ld b  
 push ; кладем b на стек => b = &1  
 call $func  
 pop  
 pop ; res -> AC  
 st res ; результат умножения записываем в res  
  
  
  
  
  
; вывод результата  
FINISH:  
 ld res  
  
; первод числа из 16 СС в 2-10 СС  
hex\_to\_bcd:  
 cmp #0x000A  
 blt done ; если res < 10 то завершаем подсчёт десятков  
 sub #0x000A  
 st res  
 ld t  
 inc  
 st t ; иначе вычитаем из него 10 и увеличиваем счетчик десятков на 1  
 jump FINISH ; переходим к следующей итерации  
  
done:  
 ld t ; обл. опред.: t E[0x0000, 0x0009]  
 asl  
 asl  
 asl  
 asl ; сдвигаем количество десятков из 1 тетрады во 2  
 or res ; объединяем с количеством единиц  
 st res  
  
 and #0x00f0 ; маска для сравнения 2 тетрады  
 cmp #0x0010  
 bge two\_digits ; если во второй тетраде число >= 1, значит двузначное число  
 blt one\_digit ; если во второй тетраде число < 1, значит цифра  
  
  
; вывод цифры  
one\_digit:  
 ld #0x002B  
 out 0x14 ; сброс 2 разряда индикатора  
 ld res  
 cmp #0x0  
 beq zero\_out  
  
 ld sign  
 cmp #0x1  
 beq negative\_res\_one\_digit ; если "-" то переходим к выводу цифры с минусом  
  
 ld #0x001B  
 out 0x14 ; сброс 1 разряда индикатора  
 ld res  
 out 0x14 ; иначе выводим цифру на позицию 0 и завершаем программу  
 jump START  
  
; вывод двузначного числа  
two\_digits:  
 ld sign  
 cmp #0x1  
 beq negative\_res\_two\_digits ; если "-" то переходим к выводу числа с минусом  
  
  
 ld #0x002B  
 out 0x14 ; сброс 2 разряда индикатора  
  
 ld res  
 asr  
 asr  
 asr  
 asr ; сдвигаем на 4 бита вправо => кол-во десятков t в первой тетраде => AC = 0x000t  
 or #0x0010 ; объединяем t и 0x0010 для вывода на 1 позицию индикатора => AC = 0x001t  
 out 0x14  
 ld res  
 and #0x000f ; оставляем только 1 тетраду res (кол-во единиц) и выводим на 0 позицию индикатора  
 out 0x14  
 jump START  
  
  
  
; вывод цифры с минусом  
negative\_res\_one\_digit:  
 ld #0x001A  
 out 0x14 ; выводим "-" на позицию 1  
 ld res  
 out 0x14 ; выводим цифру на позицию 0  
 jump START  
  
; вывод двузначного числа с минусом  
negative\_res\_two\_digits:  
 ld #0x002A  
 out 0x14 ; выводим "-" на позицию 2  
 ld res  
 asr  
 asr  
 asr  
 asr  
 or #0x0010  
 out 0x14  
 ld res  
 and #0x000f  
 out 0x14  
 jump START  
  
zero\_out:  
 ld #0x001B ;  
 out 0x14 ; сброс 1 разряда индикатора  
 ld #0x0  
 out 0x14  
 hlt  
  
  
  
  
  
  
; функция умножения  
org 0x200  
func:  
 ld &2 ; a -> AC  
 cmp #0x0  
 beq multiply\_X0  
 cmp #0x1  
 beq multiply\_X1  
 cmp #0x2  
 beq multiply\_X2  
 cmp #0x3  
 beq multiply\_X3  
 cmp #0x4  
 beq multiply\_X4  
 cmp #0x5  
 beq multiply\_X5  
 cmp #0x6  
 beq multiply\_X6  
 cmp #0x7  
 beq multiply\_X7  
 cmp #0x8  
 beq multiply\_X8  
 cmp #0x9  
 beq multiply\_X9  
  
  
  
  
; a = &2 - первый множитель, b = &1 - второй множитель  
multiply\_X0:  
 ld #0x0  
 st &2  
 ret  
  
multiply\_X1:  
 ld &1 ; в AC записываем b  
 cmp #0x1  
 beq return\_aX1  
 st &2 ; если b != 1, то записываем его в &2  
 ret ; и делаем ret (return b)  
return\_aX1:  
 ret  
  
  
  
multiply\_X2:  
 ld &1 ; в AC записываем b  
 cmp #0x2  
 beq return\_aX2  
 asl ;  
 st &2 ; если b != 2, то умножаем его на 2, записываем в 7FF = &2  
 ret ; и делаем ret (return b)  
return\_aX2:  
 ld &2  
 asl  
 st &2  
 ret  
  
  
multiply\_X3:  
 ld &1 ; в AC записываем b  
 cmp #0x3  
 beq return\_aX3  
 asl  
 add &1  
 st &2 ; если b != 3, то умножаем его на 3, записываем в 7FF = &2  
 ret ; и делаем ret (return b)  
return\_aX3:  
 ld &2  
 asl  
 add &2  
 st &2  
 ret  
  
  
multiply\_X4:  
 ld &1 ; в AC записываем b  
 cmp #0x4  
 beq return\_aX4  
 asl  
 asl  
 st &2 ; если b != 4, то умножаем его на 4, записываем в 7FF = &2  
 ret ; и делаем ret (return b)  
return\_aX4:  
 ld &2  
 asl  
 asl  
 st &2  
 ret  
  
  
multiply\_X5:  
 ld &1 ; в AC записываем b  
 cmp #0x5  
 beq return\_aX5  
 asl  
 asl  
 add &1  
 st &2 ; если b != 5, то умножаем его на 5, записываем в 7FF = &2  
 ret ; и делаем ret (return b)  
return\_aX5:  
 ld &2  
 asl  
 asl  
 add &2  
 st &2  
 ret  
  
  
multiply\_X6:  
 ld &1 ; в AC записываем b  
 cmp #0x6  
 beq return\_aX6  
 asl  
 add &1  
 asl  
 st &2 ; если b != 6, то умножаем его на 6, записываем в 7FF = &2  
 ret ; и делаем ret (return b)  
return\_aX6:  
 ld &2  
 asl  
 add &2  
 asl  
 st &2  
 ret  
  
  
multiply\_X7:  
 ld &1 ; в AC записываем b  
 cmp #0x7  
 beq return\_aX7  
 asl  
 add &1  
 asl  
 add &1  
 st &2 ; если b != 7, то умножаем его на 7, записываем в 7FF = &2  
 ret ; и делаем ret (return b)  
return\_aX7:  
 ld &2  
 asl  
 add &2  
 asl  
 add &2  
 st &2  
 ret  
  
  
multiply\_X8:  
 ld &1 ; в AC записываем b  
 cmp #0x8  
 beq return\_aX8  
 asl  
 asl  
 asl  
 st &2 ; если b != 8, то умножаем его на 8, записываем в 7FF = &2  
 ret ; и делаем ret (return b)  
return\_aX8:  
 ld &2  
 asl  
 asl  
 asl  
 st &2  
 ret  
  
multiply\_X9:  
 ld &1 ; в AC записываем b  
 cmp #0x8  
 beq return\_aX9  
 asl  
 asl  
 asl  
 add &1  
 st &2 ; если b != 9, то умножаем его на 9, записываем в 7FF = &2  
 ret ; и делаем ret (return b)  
return\_aX9:  
 ld &2  
 asl  
 asl  
 asl  
 add &2  
 st &2  
 ret