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**

по дисциплине «Основы профессиональной деятельности»

**Асинхронный обмен данными с ВУ**

Вариант №43889

Группа: P3112

Выполнил: Балин А. А.

Проверил: Осипов С. В.

**Оглавление**

[Введение 3](#_Toc136529963)

[Задание 4](#_Toc136529964)

[Текст программы 5](#_Toc136529965)

[Выполнение 6](#_Toc136529966)

[Задание на защиту 8](#_Toc136529967)

[Заключение 17](#_Toc136529968)

[Список литературы 18](#_Toc136529969)

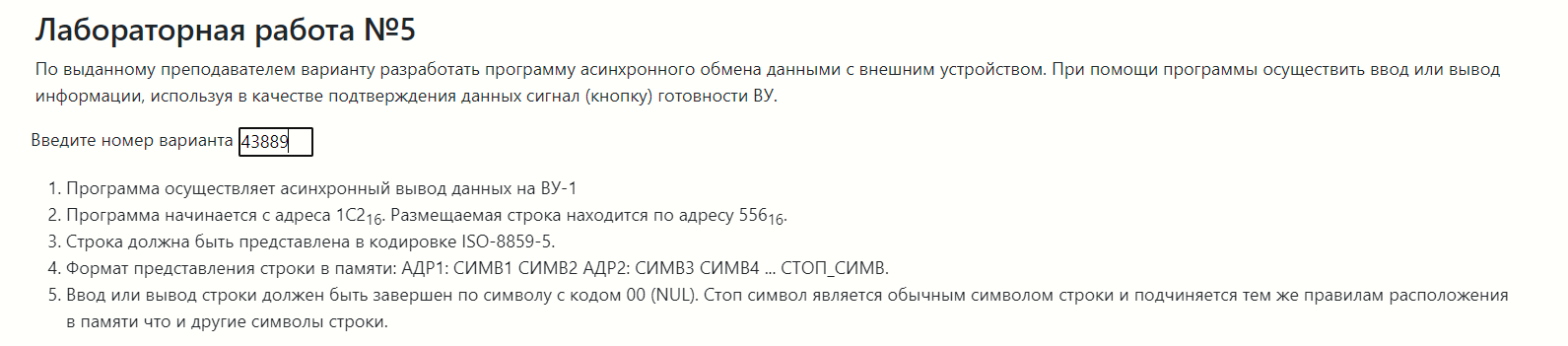
# Введение

В данной лабораторной работе я изучу реализацию комплекса программ в БЭВМ.

# 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.

Программа по моему варианту



# Текст программы

|  |  |
| --- | --- |
| **Текст программы** | **Описание** |
| ORG 0x1C2 | Регистр начала |
| LPART: WORD 0xFF00 | Для очистки младшей части слова |
| RPART: WORD 0x00FF | Для очистки старшей части слова |
| CHR: WORD 0x0556 | Ячейка с адресом текущих 2 символов |
|  |  |
|  |  |
| START: CLA | Очистка аккумулятора |
|  |  |
| WRITING: CALL $CHECK\_W1 | Вызов цикла spin-loop |
| LD (CHR) | Загрузка текущих 2 символов |
| AND $LPART | Очистка младшей части |
| BEQ STOP | Проверка на NUL (00) символ |
| SWAB | Старшая часть в младшую |
| OUT 2 | Вывод на ВУ-1 |
| CALL $CHECK\_W1 | Вызов цикла spin-loop |
| LD (CHR) | Загрузка текущих 2 символов |
| AND $RPART | Выделение младшей части |
| BEQ STOP | Проверка на NUL (00) символ |
| OUT 2 | Вывод на ВУ-1 |
| LD $CHR | Загрузка адреса текущей ячейки |
| INC | CHR++ |
| ST $CHR | Установка в CHR адрес следующей ячейки |
| JUMP $WRITING | Переход в новый цикл Writing |
|  |  |
| STOP: HLT | Останов. |
|  |  |
| CHECK\_W1: | Функция для проверки готовности записи в ВУ-1 |
| IN 3 | Бит готовности ВУ-1 в AC |
| AND #0x40 | Если бит был 1, то 0x40, иначе 0 |
| BEQ CHECK\_W1 | Если 0, в начало функции |
| RET | Возврат из функции |
|  |  |
|  |  |
| ORG 0x556 | Адрес первого символа |
| CHR12: WORD 0x77BB | Символы: wЛ |
| CHR34: WORD 0x3231 | Символы: 21 |
| CHR56: WORD 0x21DA | Символы: !к |
| CHR78: WORD 0x2300 | Символы: #NUL |

Строка: wЛ21!к#

Символы в ISO-8859-5: w: 7716, Л: BB16, 2: 3216, 1: 3116, ! : 2116, к: DA16, #: 2316

Символы в UTF-8: 7716, 41B16, 3216, 3116, 2116, 43A16, 2316

Символы в UTF-16: FEFF007716, FEFF041B16, FEFF003216, FEFF003116, FEFF002116, FEFF043A16, FEFF002316

# Выполнение

Строка: Q!ж$

ISO-8859-5:

Q: 5116

!: 2116

ж: D616

$: 2416

CHR12: WORD 0x5121

CHR34: WORD 0xD624

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Адрес | Знчн | IP | CR | AR | DR | SP | BR | AC | PS | NZVC | Адр | Знчн |
| 1C2 | FF00 | 1C3 | FF00 | 1C2 | FF00 | 0 | 01C2 | 0 | 4 | 100 |  |  |
| 1C3 | 00FF | 1C4 | 00FF | 1C3 | 00FF | 0 | 01C3 | 0 | 4 | 100 |  |  |
| 1C4 | 556 | 1C5 | 556 | 1C4 | 0 | 0 | 01C4 | 0 | 4 | 100 |  |  |
| 1C5 | 200 | 1C6 | 200 | 1C5 | 200 | 0 | 01C5 | 0 | 4 | 100 |  |  |
| 1C6 | D1D6 | 1D6 | D1D6 | 7FF | 01C7 | 7FF | D1D6 | 0 | 4 | 100 | 7FF | 01C7 |
| 1D6 | 1203 | 1D7 | 1203 | 1D6 | 1203 | 7FF | 01D6 | 0 | 4 | 100 |  |  |
| 1D7 | 2F40 | 1D8 | 2F40 | 1D7 | 40 | 7FF | 40 | 0 | 4 | 100 |  |  |
| 1D8 | F0FD | 1D6 | F0FD | 1D8 | F0FD | 7FF | FFFD | 0 | 4 | 100 |  |  |
| 1D9 | 0A00 | 1C7 | 0A00 | 7FF | 01C7 | 0 | 01D9 | 40 | 0 | 0 |  |  |
| 1C7 | A8FC | 1C8 | A8FC | 556 | 5121 | 0 | FFFC | 5121 | 0 | 0 |  |  |
| 1C8 | 21C2 | 1C9 | 21C2 | 1C2 | FF00 | 0 | 01C8 | 5100 | 0 | 0 |  |  |
| 1C9 | F00B | 1CA | F00B | 1C9 | F00B | 0 | 01C9 | 5100 | 0 | 0 |  |  |
| 1CA | 680 | 1CB | 680 | 1CA | 680 | 0 | 01CA | 51 | 0 | 0 |  |  |
| 1CB | 1302 | 1CC | 1302 | 1CB | 1302 | 0 | 01CB | 51 | 0 | 0 |  |  |
| 1CC | D1D6 | 1D6 | D1D6 | 7FF | 01CD | 7FF | D1D6 | 51 | 0 | 0 | 7FF | 01CD |
| 1D6 | 1203 | 1D7 | 1203 | 1D6 | 1203 | 7FF | 01D6 | 0 | 0 | 0 |  |  |
| 1D7 | 2F40 | 1D8 | 2F40 | 1D7 | 40 | 7FF | 40 | 0 | 4 | 100 |  |  |
| 1D8 | F0FD | 1D6 | F0FD | 1D8 | F0FD | 7FF | FFFD | 0 | 4 | 100 |  |  |
| 1D9 | 0A00 | 1CD | 0A00 | 7FF | 01CD | 0 | 01D9 | 40 | 0 | 0 |  |  |
| 1CD | A8F6 | 1CE | A8F6 | 556 | 5121 | 0 | FFF6 | 5121 | 0 | 0 |  |  |
| 1CE | 21C3 | 1CF | 21C3 | 1C3 | 00FF | 0 | 01CE | 21 | 0 | 0 |  |  |
| 1CF | F005 | 1D0 | F005 | 1CF | F005 | 0 | 01CF | 21 | 0 | 0 |  |  |
| 1D0 | 1302 | 1D1 | 1302 | 1D0 | 1302 | 0 | 01D0 | 21 | 0 | 0 |  |  |
| 1D1 | A1C4 | 1D2 | A1C4 | 1C4 | 556 | 0 | 01D1 | 556 | 0 | 0 |  |  |
| 1D2 | 700 | 1D3 | 700 | 1D2 | 700 | 0 | 01D2 | 557 | 0 | 0 |  |  |
| 1D3 | E1C4 | 1D4 | E1C4 | 1C4 | 557 | 0 | 01D3 | 557 | 0 | 0 | 1C4 | 557 |
| 1D4 | C1C6 | 1C6 | C1C6 | 1D4 | C1C6 | 0 | 01D4 | 557 | 0 | 0 |  |  |
| 1C6 | D1D6 | 1D6 | D1D6 | 7FF | 01C7 | 7FF | D1D6 | 557 | 0 | 0 | 7FF | 01C7 |
| 1D6 | 1203 | 1D7 | 1203 | 1D6 | 1203 | 7FF | 01D6 | 500 | 0 | 0 |  |  |
| 1D7 | 2F40 | 1D8 | 2F40 | 1D7 | 40 | 7FF | 40 | 0 | 4 | 100 |  |  |
| 1D8 | F0FD | 1D6 | F0FD | 1D8 | F0FD | 7FF | FFFD | 0 | 4 | 100 |  |  |
| 1D9 | 0A00 | 1C7 | 0A00 | 7FF | 01C7 | 0 | 01D9 | 40 | 0 | 0 |  |  |
| 1C7 | A8FC | 1C8 | A8FC | 557 | D624 | 0 | FFFC | D624 | 8 | 1000 |  |  |
| 1C8 | 21C2 | 1C9 | 21C2 | 1C2 | FF00 | 0 | 01C8 | D600 | 8 | 1000 |  |  |
| 1C9 | F00B | 1CA | F00B | 1C9 | F00B | 0 | 01C9 | D600 | 8 | 1000 |  |  |
| 1CA | 680 | 1CB | 680 | 1CA | 680 | 0 | 01CA | 00D6 | 0 | 0 |  |  |
| 1CB | 1302 | 1CC | 1302 | 1CB | 1302 | 0 | 01CB | 00D6 | 0 | 0 |  |  |
| 1CC | D1D6 | 1D6 | D1D6 | 7FF | 01CD | 7FF | D1D6 | 00D6 | 0 | 0 | 7FF | 01CD |
| 1D6 | 1203 | 1D7 | 1203 | 1D6 | 1203 | 7FF | 01D6 | 0 | 0 | 0 |  |  |
| 1D7 | 2F40 | 1D8 | 2F40 | 1D7 | 40 | 7FF | 40 | 0 | 4 | 100 |  |  |
| 1D8 | F0FD | 1D6 | F0FD | 1D8 | F0FD | 7FF | FFFD | 0 | 4 | 100 |  |  |
| 1D9 | 0A00 | 1C7 | 0A00 | 7FF | 01C7 | 0 | 01D9 | 40 | 0 | 0 |  |  |
| 1C7 | A8FC | 1C8 | A8FC | 558 | 0 | 0 | FFFC | 0 | 4 | 100 |  |  |
| 1C8 | 21C2 | 1C9 | 21C2 | 1C2 | FF00 | 0 | 01C8 | 0 | 4 | 100 |  |  |
| 1C9 | F00B | 1D5 | F00B | 1C9 | F00B | 0 | 000B | 0 | 4 | 100 |  |  |
| 1D5 | 100 | 1D6 | 100 | 1D5 | 100 | 0 | 01D5 | 0 | 4 | 100 |  |  |

# Задание на защиту

Разложение на простейшие. 😢

ORG 0x0000

ORG 0x0040

START:

JUMP $INPUT\_INIT

START\_FIND:CLA

CURRENT\_SIMPLE\_ID: WORD 0x0000

SUS\_SIMPLE: WORD 0x0000

NEEDED: WORD 0x003C

CLA

LD #2

ST (CURRENT\_SIMPLE\_ID)+

INC

ST $SUS\_SIMPLE

FINDING\_SIMPLE: LD $CURRENT\_SIMPLE\_ID

    CMP $NEEDED

    BEQ START\_DEREF

    LD $SUS\_SIMPLE

    PUSH

    LD $CURRENT\_SIMPLE\_ID

    PUSH

    CALL $IS\_SIMPLE

    SWAP

    POP

    SWAP

    POP

    BPL NEXT\_SUS

    LD $SUS\_SIMPLE

    ST (CURRENT\_SIMPLE\_ID)+

    NEXT\_SUS: LD $SUS\_SIMPLE

    ADD #2

    ST $SUS\_SIMPLE

    JUMP $FINDING\_SIMPLE

GET\_CURRENT\_SIMPLE:

    INSIDE\_DEREF: WORD 0x0000

    LD &1

    ST $INSIDE\_DEREF

    LD (INSIDE\_DEREF)

    RET

INPUT\_INIT:

    NUM: WORD 0x0000

    NEWNUM: WORD 0x0000

    INPUT\_NUM: IN 0x1D

    AND #0x40

    BEQ INPUT\_NUM

    IN 0x1C

    ST $NEWNUM

    PUSH

    CALL CHECK\_NUMERIC

    SWAP

    POP

    BMI ERR

    BEQ START\_FIND

    LD $NUM

    PUSH

    LD $NEWNUM

    PUSH

    CALL $NEW\_DEC\_NUM

    SWAP

    POP

    POP

    ST $NUM

    BMI ERR

    JUMP $INPUT\_NUM

CHECK\_NUMERIC:

    CMP #0xF

    BEQ START\_OUT

    CMP #0xA

    BEQ ERR\_OUT

    JUMP NUMERIC

    ERR\_OUT: LD #-1

    RET

    START\_OUT: LD #0

    RET

    NUMERIC: LD #1

    RET

DEREF: WORD ?

IS\_SIMPLE: CLA ;STACK: DIVIDED;TEMP;INDEX;RET;CURRENT\_SIMPLE;N

    PUSH

    PUSH

    PUSH

    CHECKING\_SIMPLE\_DEVISORS: LD &2

    CMP &4

    BPL IS\_SIMPLE\_OUT\_P

    LD &2

    ST $DEREF

    LD &5

    OTN: SUB (DEREF)

    ST &1

    LD &0

    INC

    ST &0

    LD &1

    BMI IS\_DIVISOR\_OUT

    BEQ IS\_DIVISOR\_OUT

    JUMP $OTN

    IS\_DIVISOR\_OUT: BEQ IS\_SIMPLE\_OUT\_N

    LD &2

    INC

    ST &2

    LD &1

    JUMP $CHECKING\_SIMPLE\_DEVISORS

    IS\_SIMPLE\_OUT\_P: SWAP

    POP

    SWAP

    POP

    SWAP

    POP

    LD #-1

    RET

    IS\_SIMPLE\_OUT\_N:

    LD &0

    SWAP

    POP

    SWAP

    POP

    SWAP

    POP

    RET

START\_DEREF:

    JUMP $STARTT

ERR: IN 0x15

    AND #0x40

    BEQ ERR

    LD #0

    OUT 0x14

    LD #0x1B

    OUT 0x14

    LD #0x2B

    OUT 0x14

    LD #0x3B

    OUT 0x14

    LD #0x4B

    OUT 0x14

    LD #0x5B

    OUT 0x14

    LD #0x6B

    OUT 0x14

    LD #0x7B

    OUT 0x14

    HLT

NEW\_DEC\_NUM:CLA

    PUSH;COUNT

    PUSH;TEMP

    LD #9

    ST &1

    LD &4

    ST &0

    ADDING: LD &4

    ADD &0

    ST &4

    LD &1

    DEC

    ST &1

    BEQ EXIT\_NEW\_DEC\_NUM

    JUMP $ADDING

    EXIT\_NEW\_DEC\_NUM: LD &4

    ADD &3

    ST &4

    SWAP

    POP

    SWAP

    POP

    RET

ORG 0x0100

STARTT: CLA

    VIVOD\_1: WORD 0x000B

    VIVOD\_2: WORD 0x001B

    VIVOD\_3: WORD 0x002B

    VIVOD\_4: WORD 0x003B

    VIVOD\_5: WORD 0x004B

    VIVOD\_6: WORD 0x005B

    VIVOD\_7: WORD 0x006B

    LD $NUM

    CHECK\_2\_AS\_DIVISOR: AND #1

    BEQ TWO\_IS\_DIVISOR

    JUMP $END\_CHECK\_2\_AS\_DIVISOR

    TWO\_IS\_DIVISOR: LD #2

    PUSH

    CALL $NEW\_NUM\_TO\_PRINT

    SWAP

    POP

    CALL $PRINT

    LD #0xB

    PUSH

    CALL $NEW\_NUM\_TO\_PRINT

    SWAP

    POP

    CALL $PRINT

    LD $NUM

    ASR

    ST $NUM

    JUMP $CHECK\_2\_AS\_DIVISOR

    END\_CHECK\_2\_AS\_DIVISOR:

    LD $NUM

    CMP #1

    BEQ HHLT\_DEREF

    NUM\_ON\_VIVOD: WORD 0x0000

    NUMERIC\_ON\_VIVOD: WORD 0x0000

    LD #1

    ST $CURRENT\_SIMPLE\_ID

    NEXT\_SIMPLE: LD $CURRENT\_SIMPLE\_ID

    PUSH

    CALL $GET\_CURRENT\_SIMPLE

    SWAP

    POP

    CUR\_SIM\_DIVIDED: PUSH

    LD $NUM

    PUSH

    CALL $CHECK\_IF\_DIVIDED

    BMI CLEAR\_AND\_NEXT\_SIMPLE

    ST $NUM

    POP

    POP

    ST $NUM\_ON\_VIVOD

    NEXT\_NUMERIC: PUSH

    CALL $GET\_LAST\_NUM

    ST $NUMERIC\_ON\_VIVOD

    POP

    ST $NUM\_ON\_VIVOD

    LD $NUMERIC\_ON\_VIVOD

    PUSH

    CALL $SET\_T

    SWAP

    POP

    LD $NUM\_ON\_VIVOD

    BNE PRINT\_NUMERIC\_AND\_GET\_NEXT

    BEQ PRINT\_NUMERIC\_AND\_GO

    PRINT\_NUMERIC\_AND\_GET\_NEXT:

    LD $NUM\_ON\_VIVOD

    JUMP $NEXT\_NUMERIC

    PRINT\_NUMERIC\_AND\_GO:

    T\_GETTING: CALL $GET\_T

    PUSH

    CALL $NEW\_NUM\_TO\_PRINT

    SWAP

    POP

    LD $CURRENT\_T

    BEQ LAST\_NUM

    JUMP $T\_GETTING

    LAST\_NUM:

    LD #0xB

    PUSH

    CALL $NEW\_NUM\_TO\_PRINT

    SWAP

    POP

    CALL $PRINT

    LD $NUM

    CMP #1

    BEQ HHLT\_DEREF

    LD $CURRENT\_SIMPLE\_ID

    PUSH

    CALL $GET\_CURRENT\_SIMPLE

    SWAP

    POP

    JUMP $CUR\_SIM\_DIVIDED

    CLEAR\_AND\_NEXT\_SIMPLE: SWAP

    POP

    SWAP

    POP

    LD $CURRENT\_SIMPLE\_ID

    INC

    ST $CURRENT\_SIMPLE\_ID

    CMP $NEEDED

    BEQ PRINT\_WHAT\_OTHER\_AND\_HLT

    JUMP $NEXT\_SIMPLE

    PRINT\_WHAT\_OTHER\_AND\_HLT:

    LD $NUM

    BEQ HHLT\_DEREF

    PRINTING\_LAST:

    PUSH

    CALL $GET\_LAST\_NUM

    ST $NUMERIC\_ON\_VIVOD

    POP

    ST $NUM\_ON\_VIVOD

    LD $NUMERIC\_ON\_VIVOD

    PUSH

    CALL $SET\_T

    SWAP

    POP

    LD $NUM\_ON\_VIVOD

    BEQ PRINT\_LAST

    JUMP $PRINTING\_LAST

PRINT\_LAST:

    NOP

    T\_GETTING\_NEW: CALL $GET\_T

    PUSH

    CALL $NEW\_NUM\_TO\_PRINT

    SWAP

    POP

    LD $CURRENT\_T

    BEQ HHLT\_PRINT

    JUMP $T\_GETTING\_NEW

    HHLT\_PRINT: CALL $PRINT

    JUMP $HHLT\_DEREF

HHLT\_DEREF: JUMP $HHLT

NEW\_NUM\_TO\_PRINT: LD $VIVOD\_6

    ADD #0x10

    ST $VIVOD\_7

    LD $VIVOD\_5

    ADD #0x10

    ST $VIVOD\_6

    LD $VIVOD\_4

    ADD #0x10

    ST $VIVOD\_5

    LD $VIVOD\_3

    ADD #0x10

    ST $VIVOD\_4

    LD $VIVOD\_2

    ADD #0x10

    ST $VIVOD\_3

    LD $VIVOD\_1

    ADD #0x10

    ST $VIVOD\_2

    LD &1

    ST $VIVOD\_1

    RET

GET\_LAST\_NUM:

    CLA

    PUSH

    PUSH

    LD &3

    ST &0

    CYCLE\_GET\_LAST\_NUM:

    SUB #10

    BMI RETURN\_GET\_LAST\_NUM

    ST &0

    LD &1

    INC

    ST &1

    LD &0

    JUMP $CYCLE\_GET\_LAST\_NUM

    RETURN\_GET\_LAST\_NUM: LD &1

    ST &3

    LD &0

    SWAP

    POP

    SWAP

    POP

    RET

PRINT:

    NOP

    OSTANOV:IN 0x1D

    AND #0x40

    BEQ OSTANOV

    LD $VIVOD\_1

    OUT 0x14

    LD $VIVOD\_2

    OUT 0x14

    LD $VIVOD\_3

    OUT 0x14

    LD $VIVOD\_4

    OUT 0x14

    LD $VIVOD\_5

    OUT 0x14

    LD $VIVOD\_6

    OUT 0x14

    LD $VIVOD\_7

    OUT 0x14

    RET

HHLT: HLT

CHECK\_IF\_DIVIDED: CLA

    PUSH

    LD &2

    CYCLE\_CHECK\_IF\_DIVIDED:

    SUB &3

    BMI CHECK\_IF\_DIVIDED\_BAD

    ST &2

    LD &0

    INC

    ST &0

    LD &2

    BEQ CHECK\_IF\_DIVIDED\_GOOD

    JUMP $CYCLE\_CHECK\_IF\_DIVIDED

    CHECK\_IF\_DIVIDED\_BAD: LD #-1

    SWAP

    POP

    RET

    CHECK\_IF\_DIVIDED\_GOOD: LD &0

    SWAP

    POP

    RET

ORG 0x01FF

T0: WORD 0xB

T1: WORD ?

T2: WORD ?

T3: WORD ?

T4: WORD ?

T5: WORD ?

SET\_T:

    LD START\_T

    ADD $CURRENT\_T

    ST $DEREFT

    LD $CURRENT\_T

    INC

    ST $CURRENT\_T

    LD &1

    ST (DEREFT)

    RET

GET\_T:

    LD START\_T

    ADD $CURRENT\_T

    DEC

    ST $DEREFT

    LD $CURRENT\_T

    DEC

    ST $CURRENT\_T

    LD (DEREFT)

    RET

DEREFT: WORD ?

CURRENT\_T: WORD ?

START\_T: WORD 0x0200

# Заключение

Я изучил работу БЭВМ с ВУ, создал программу для вывода данных на ВУ-1.

# Список литературы

**Методические указания к лабораторным работам по курсу "Основы профессиональной деятельности"** [В Интернете] / авт. В. В. Кириллов А. А. Приблуда, С. В. Клименков, Д. Б. Афанасьев. - https://se.ifmo.ru/documents/10180/38002/Методические+указания+к+выполнению+лабораторных+работ+и+рубежного+контроля+БЭВМ+2019+bcomp-ng.pdf/d5a1be02-ad3f-4c43-8032-a2a04d6db12e.