Отчет

**по лабораторной работе №4**

**«Исследование работы БЭВМ»**

по дисциплине «Основы профессиональной деятельности» вариант 10673

Выполнил: Караганов П.Э., группа P3110

Преподаватель: Блохина Е.Н.

Оглавление

[Текст задания 3](#_Toc195138826)

[Подпрограмма 4](#_Toc195138827)

[График функции, реализуемой подпрограммой 4](#_Toc195138828)

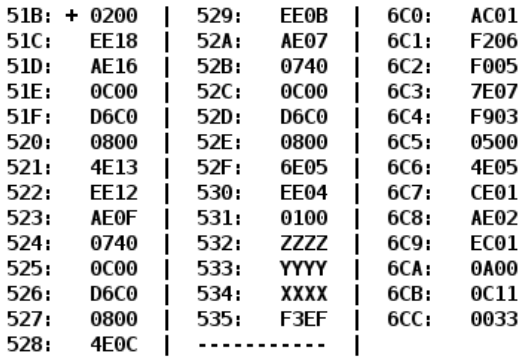
[Описание комплекса программ 5](#_Toc195138829)

[Таблица трассировки 6](#_Toc195138830)

[Вывод 7](#_Toc195138831)

# Текст задания

По выданному преподавателем варианту восстановить текст заданного варианта программы и подпрограммы (программного комплекса), определить предназначение и составить его описание, определить область представления и область допустимых значений исходных данных и результата, выполнить трассировку программного комплекса.

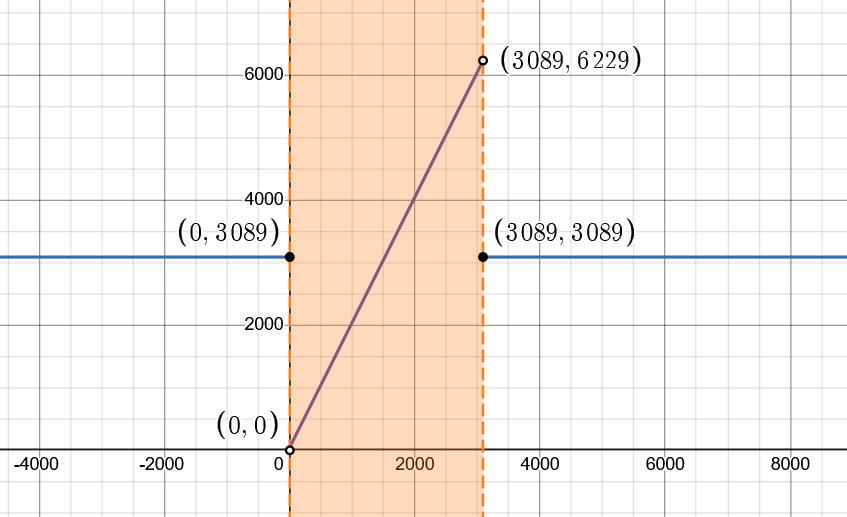


|  |  |  |  |
| --- | --- | --- | --- |
| **Адрес** | **Код команды** | **Мнемоника** | **Комментарии** |
| 51B: | 0200 | CLA | Очистка аккумулятора  0 → AC |
| 51C: | EE18 | ST (IP + 24) | AC → R |
| 51D: | AE16 | LD (IP + 23) | X → AC |
| 51E: | 0C00 | PUSH | AC → -(SP)  SP стал равен 7FF и в эту ячейку записалось значение AC |
| 51F: | D6C0 | CALL (6C0) | Вызываем подпрограмму по адресу (6С0) |
| 520: | 0800 | POP | SP → AC (записали в AC результат работы подпрограммы); SP увеличиваем на 1 (SP = 0000) |
| 521: | 4E13 | ADD (IP + 19) | AC + R → AC |
| 522: | EE12 | ST (IP + 18) | AC → R |
| 523: | AE0F | LD (IP +15) | Y → AC |
| 524: | 0740 | DEC | AC – 1 → AC |
| 525: | 0C00 | PUSH | AC→ -(SP) |
| 526: | D6C0 | CALL (6CO) | Вызываем подпрограмму по адресу (6С0) |
| 527: | 0800 | POP | SP → AC (записали в AC результат работы подпрограммы); SP увеличиваем на 1 (SP = 0000) |
| 528: | 4E0C | ADD (IP +12) | AC + R → AC |
| 529: | EE0B | ST (IP + 11) | AC → R |
| 52A: | AE07 | LD (IP +7) | Z → AC |
| 52B: | 0740 | DEC | AC – 1 → AC |
| 52C: | 0C00 | PUSH | AC → -(SP) |
| 52D: | D6C0 | CALL (6C0) | Вызываем подпрограмму по адресу (6С0) |
| 52E: | 0800 | POP | SP → AC (записали в AC результат работы подпрограммы); SP увеличиваем на 1 (SP = 0000) |
| 52F: | 6E05 | SUB (IP +5) | AC – R → AC |
| 530: | EE04 | ST (IP +4) | AC → R |
| 531: | 0100 | HLT | Остановка |

# Подпрограмма

|  |  |  |  |
| --- | --- | --- | --- |
| **Адрес** | **Код команды** | **Мнемоника** | **Комментарии** |
| 6C0: | AC01 | LD (SP + 1) | Загрузка в аккумулятор предпоследнего значения в стеке (второго сверху). |
| 6C1: | F206 | BMI (IP +6) | Переходим на 6C8 если AC ≤ 0 |
| 6C2: | F005 | BEQ (IP +5) |  |
| 6C3: | 7E07 | CMP (IP + 7) | Устанавливаем знаки по AC – 0С1116 (=308910) |
| 6C4: | F903 | BGE (IP + 3) | Переходим на 6С8 если AC ≥ 0С1116 (=308910) |
| 6C5: | 0500 | ASL | На 2 умножили AC:  AC→ BR  AC + BR → AC |
| 6C6: | 4E05 | ADD (IP +5) | AC + 003316 (= 5110) → AC |
| 6C7: | CE01 | JUMP (IP +1) | Перепрыгиваем одну строчку |
| 6C8: | AE02 | LD (IP + 2) | 0С1116 (=308910) → AC |
| 6C9: | EC01 | ST (SP + 1) | Запись содержимого аккумулятора в предпоследнюю ячейку стека. (Вторую сверху, теперь это результат работы подпрограммы) |
| 6CA: | 0A00 | RET | Возвращение  (SP)+ → IP |

# График функции, реализуемой подпрограммой



# Описание комплекса программ

**Описание подпрограммы:**

Пусть t - аргумент

Тогда подпрограмма выполняет следующее:

if 0000 < t < 0C11:

return t\*2 + 0033

return 0C11

**Описание программы:**

Назовем подпрограмму функцией f(t), где t – передаваемый на вход аргумент.

Тогда программа выполняет расчёт по формуле:

R = f(z-1) – f(x) – f(y-1)

**Описание комплекса программ:**

Комплекс программ состоит из основной программы и одной подпрограммы. Подпрограмма вызывается 3 раза, обмен аргументами и возвращаемыми значениями между программой и подпрограммой происходит через стек.

**Область представления:**

x, y, z, r и значения, возвращаемые подпрограммой – 15-разрядные целые знаковые числа

-215 ≤ f(z-1) – f(x) – f(y-1) ≤ 215 - 1

8000 ≤ x,y,z ≤ 7FFF

**ОДЗ:**

Рассмотрим какой результат вернёт f(t) при различных t

При t ∊ [8000; 0000] ∪[0C11; 7FFF] результат будет 0C11

То есть при таких t result будет 0C11 = 308910 → при использовании любого значения из данного промежутка в функции и результате не возникает переполнения.

При t ∊ (0000; 0C11) результат будет 2t + 0033

На промежутке (0000; 0C11) функция монотонно возрастающая → можем взять максимальное значение и минимальное

fmax(t) = f(0C10) = 185316 = 622710

fmin(t) = f(0001) = 3516 = 5310

Следовательно, Rmax = 6227 – 53 – 53 = 612110 = 17E916; Rmin = 53 – 6227 – 6227 = -12401 = CF8F16, следовательно, переполнения не возникает при:

f(t) [0035; 1853]

Z, Y [8001; 7FFF]

X [8000; 7FFF]

R [CF8F; 17E9]

**Расположение в памяти ЭВМ:**

* Подпрограммы: Ячейки (6C0) – (6CA)
* Данные для подпрограммы: Ячейки (6CB) – (6CC)
* Программы: Ячейки (51B) – (531)
* Данныe: Ячейки (532) – (534)
* Результат: Ячейка (535)
* Адрес первой выполняемой команды: (51B)
* Адрес последней выполняемой команды: (531)

# Таблица трассировки

Z = 05DC16 = 150010

Y = FFA716 = -8910

X = 3CF016 = 1560010

Ожидаемый результат R = F3C716 = -312910

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |
| 51B | 0200 | 51C | 0200 | 51B | 0200 | 000 | 051B | 0000 | 0100 |  |  |
| 51C | EE18 | 51D | EE18 | 535 | 0000 | 000 | 0018 | 0000 | 0100 | 535 | 0000 |
| 51D | AE16 | 51E | AE16 | 534 | 3CF0 | 000 | 0016 | 3CF0 | 0000 |  |  |
| 51E | 0C00 | 51F | 0C00 | 7FF | 3CF0 | 7FF | 051E | 3CF0 | 0000 | 7FF | 3CF0 |
| 51F | D6C0 | 6C0 | D6C0 | 7FE | 0520 | 7FE | D6C0 | 3CF0 | 0000 | 7FE | 0520 |
| 6C0 | AC01 | 6C1 | AC01 | 7FF | 3CF0 | 7FE | 0001 | 3CF0 | 0000 |  |  |
| 6C1 | F206 | 6C2 | F206 | 6C1 | F206 | 7FE | 06C1 | 3CF0 | 0000 |  |  |
| 6C2 | F005 | 6C3 | F005 | 6C2 | F005 | 7FE | 06C2 | 3CF0 | 0000 |  |  |
| 6C3 | 7E07 | 6C4 | 7E07 | 6CB | 0C11 | 7FE | 0007 | 3CF0 | 0001 |  |  |
| 6C4 | F903 | 6C8 | F903 | 6C4 | F903 | 7FE | 0003 | 3CF0 | 0001 |  |  |
| 6C8 | AE02 | 6C9 | AE02 | 6CB | 0C11 | 7FE | 0002 | 0C11 | 0001 |  |  |
| 6C9 | EC01 | 6CA | EC01 | 7FF | 0C11 | 7FE | 0001 | 0C11 | 0001 | 7FF | 0C11 |
| 6CA | 0A00 | 520 | 0A00 | 7FE | 0520 | 7FF | 06CA | 0C11 | 0001 |  |  |
| 520 | 0800 | 521 | 0800 | 7FF | 0C11 | 000 | 0520 | 0C11 | 0001 |  |  |
| 521 | 4E13 | 522 | 4E13 | 535 | 0000 | 000 | 0013 | 0C11 | 0000 |  |  |
| 522 | EE12 | 523 | EE12 | 535 | 0C11 | 000 | 0012 | 0C11 | 0000 | 535 | 0C11 |
| 523 | AE0F | 524 | AE0F | 533 | FFA7 | 000 | 000F | FFA7 | 1000 |  |  |
| 524 | 0740 | 525 | 0740 | 524 | 0740 | 000 | 0524 | FFA6 | 1001 |  |  |
| 525 | 0C00 | 526 | 0C00 | 7FF | FFA6 | 7FF | 0525 | FFA6 | 1001 | 7FF | FFA6 |
| 526 | D6C0 | 6C0 | D6C0 | 7FE | 0527 | 7FE | D6C0 | FFA6 | 1001 | 7FE | 0527 |
| 6C0 | AC01 | 6C1 | AC01 | 7FF | FFA6 | 7FE | 0001 | FFA6 | 1001 |  |  |
| 6C1 | F206 | 6C8 | F206 | 6C1 | F206 | 7FE | 0006 | FFA6 | 1001 |  |  |
| 6C8 | AE02 | 6C9 | AE02 | 6CB | 0C11 | 7FE | 0002 | 0C11 | 0001 |  |  |
| 6C9 | EC01 | 6CA | EC01 | 7FF | 0C11 | 7FE | 0001 | 0C11 | 0001 | 7FF | 0C11 |
| 6CA | 0A00 | 527 | 0A00 | 7FE | 0527 | 7FF | 06CA | 0C11 | 0001 |  |  |
| 527 | 0800 | 528 | 0800 | 7FF | 0C11 | 000 | 0527 | 0C11 | 0001 |  |  |
| 528 | 4E0C | 529 | 4E0C | 535 | 0C11 | 000 | 000C | 1822 | 0000 |  |  |
| 529 | EE0B | 52A | EE0B | 535 | 1822 | 000 | 000B | 1822 | 0000 | 535 | 1822 |
| 52A | AE07 | 52B | AE07 | 532 | 05DC | 000 | 0007 | 05DC | 0000 |  |  |
| 52B | 0740 | 52C | 0740 | 52B | 0740 | 000 | 052B | 05DB | 0001 |  |  |
| 52C | 0C00 | 52D | 0C00 | 7FF | 05DB | 7FF | 052C | 05DB | 0001 | 7FF | 05DB |
| 52D | D6C0 | 6C0 | D6C0 | 7FE | 052E | 7FE | D6C0 | 05DB | 0001 | 7FE | 052E |
| 6C0 | AC01 | 6C1 | AC01 | 7FF | 05DB | 7FE | 0001 | 05DB | 0001 |  |  |
| 6C1 | F206 | 6C2 | F206 | 6C1 | F206 | 7FE | 06C1 | 05DB | 0001 |  |  |
| 6C2 | F005 | 6C3 | F005 | 6C2 | F005 | 7FE | 06C2 | 05DB | 0001 |  |  |
| 6C3 | 7E07 | 6C4 | 7E07 | 6CB | 0C11 | 7FE | 0007 | 05DB | 1000 |  |  |
| 6C4 | F903 | 6C5 | F903 | 6C4 | F903 | 7FE | 06C4 | 05DB | 1000 |  |  |
| 6C5 | 0500 | 6C6 | 0500 | 6C5 | 05DB | 7FE | 06C5 | 0BB6 | 0000 |  |  |
| 6C6 | 4E05 | 6C7 | 4E05 | 6CC | 0033 | 7FE | 0005 | 0BE9 | 0000 |  |  |
| 6C7 | CE01 | 6C9 | CE01 | 6C7 | 06C9 | 7FE | 0001 | 0BE9 | 0000 |  |  |
| 6C9 | EC01 | 6CA | EC01 | 7FF | 0BE9 | 7FE | 0001 | 0BE9 | 0000 | 7FF | 0BE9 |
| 6CA | 0A00 | 52E | 0A00 | 7FE | 052E | 7FF | 06CA | 0BE9 | 0000 |  |  |
| 52E | 0800 | 52F | 0800 | 7FF | 0BE9 | 000 | 052E | 0BE9 | 0000 |  |  |
| 52F | 6E05 | 530 | 6E05 | 535 | 1822 | 000 | 0005 | F3C7 | 1000 |  |  |
| 530 | EE04 | 531 | EE04 | 535 | F3C7 | 000 | 0004 | F3C7 | 1000 | 535 | F3C7 |
| 531 | 0100 | 532 | 0100 | 531 | 0100 | 000 | 0531 | F3C7 | 1000 |  |  |

РАЗОБРАТЬСЯ КАК РАБОТЕТ:

* *CALL*: DR→BR; IP→DP; BR→IP; ~0 + SP→SP, **AR**; DR→MEM(AR);
* *PUSH*: AC→DR; ~0 + SP→SP, **AP**; DR→MEM(AR);
* *POP*: SP→AR; MEM(AR) →DR; DR→AC; SP + 1→SP;
* *RET*: SP→AR; MEM(AR) →DR: DR→IP; SP + 1→SP;
* *SWAP*: SP→AR; MEM(AR) → DR; DR→BR; AC→DR; BR→AC; DR→MEM(AR)

# Вывод

В процессе выполнения лабораторной работы я углубил понимание ключевых аспектов архитектуры БЭВМ, включая:

* применение подпрограмм в рамках вычислительного процесса,
* определение областей допустимых значений программы,
* использование условных переходов и команд ветвления,
* принципы организации стека и работы с ним.

Полученные навыки позволили лучше освоить взаимодействие между модулями программы и управление данными через стек.

Написать программ, которая суммирует 3 элемента массива длиной 32-бита и получает 32-битный результат.

ORG 0x10

FIRST\_H: WORD 0x0000

FIRST\_L: WORD 0x7FFF

SECND\_H: WORD 0x0000

SECND\_L: WORD 0x7FFF

THIRD\_H: WORD 0x0000

THIRD\_L: WORD 0x7FFF

R\_H: WORD 0

R\_L: WORD 0

START: CLA

ST R\_H ; Очищаем R\_H

ST R\_L ; Очищаем R\_L

LD FIRST\_L ; Загружаем в AC младший байт первого эл.

ADD SECND\_L ; Складываем с AC младший байт второго эл. (При переполнении C=1)

ST R\_L ; Загружаем в результат в R\_L (C не меняется)

LD FIRST\_H ; Загружаем в AC старший байт (C не меняется)

ADC SECND\_H ; SECND\_H + AC + C → AC

ADC R\_H ; Доб. результату значение AC (необязательно так как R\_H=0)

ST R\_H ; Загружаем значение в R\_H

LD THIRD\_L ; Загружаем в AC THIRD\_L

ADD R\_L ; Доб. R\_L в AC (при переполнении С= 1)

ST R\_L ; Загружаем в R\_L AC (C не меняется)

LD THIRD\_H ; Загружаем в AC THIRD\_H (C не меняются)

ADC R\_H ; R\_H + AC + C → AC

ST R\_H ; Загружаем AC в R\_H

HLT