Федеральное государственное автономное образовательное учреждение

высшего образования «Национальный исследовательский университет

ИТМО»

*Факультет программной инженерии и компьютерной техники*

**Лабораторная работа №3**

**По дисциплине**

**“Основы профессиональной деятельности”**

Вариант: 1080

Выполнил:

Ахроров Кароматуллохон Фирдавсович

Группа: Р3110

Преподаватель:

Блохина Елена Николаевна

Санкт-Петербург, 2024г

Содержание

[ЗАДАНИЕ 2](#_Задание)

[Ход работы 3](#_Ход_работы)

[Описание Программы 3](#_1._Описание_программы_1)

[Область представления 4](#_2._Область_представления)

[Область допустимых значений 4](#_3._Область_допустимых)

[Расположение данных в памяти 5](#_Расположение_данных_в)

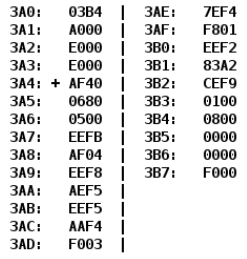
[Адреса первой и последней выполняемой команды 5](#_Toc191547624)

[Таблица трассировки 5](#_Таблица_трассировки)

[Вывод 6](#_Вывод)

## Задание

По выданному преподавателем варианту определить функцию, вычисляемую программой, область представления и область допустимых значений исходных данных и результата, выполнить трассировку программы, предложить вариант с меньшим числом команд. При выполнении работы представлять результат и все операнды арифметических операций знаковыми числами, а логических операций набором из шестнадцати логических значений.



## Ход работы

### Текст исходной программы

|  |  |  |  |
| --- | --- | --- | --- |
| **Адрес** | **Код команды** | **Мнемоника** | **Комментарий** |
| 3A4 | AF40 | LD 40 | Прямая загрузка 0040 в AC |
| 3A5 | 0680 | SWAB | Обмен |
| 3A6 | 0500 | ASL | Сдвиг влево |
| 3A7 | EEFB | ST (IP-5) | Прямое относительное Сохранение AC в ячейку по адресу  IP-5(мин число) |
| 3A8 | AF04 | LD 04 | Прямая загрузка 0004 в AC |
| 3A9 | EEF8 | ST (IP-8) | Прямое относительное Сохранение AC в ячейку по адресу IP+1-8(Кол-во элементов массива 4) |
| 3AA | AEF5 | LD (IP­-11) | Прямая относительная загрузка в AC по адресу IP+1-11 |
| 3AB | EEF5 | ST (IP-11) | Прямое относительное Сохранение AC в ячейку по адресу  IP-11(в R) |
| 3AC | AAF4 | LD (IP-12) + | Косвенная авто инкрементальная загрузка:  MEM(IP-12) +=1; MEM(M) -> AC (в адрес текущего элемента) |
| 3AD | F003 | BEQ (IP+1) | Если Z == 1, то IP = IP + 1 + 1 -> IP |
| 3AE | 7EF4 | CMP (IP-12) | Флаги по результату АС-R |
| 3AF | F801 | BLT | Если (NꚛV == 1 / N!=V), то IP = IP + 1 -> IP |
| 3B0 | EEF2 | ST (IP-14) | Прямое относительное Сохранение AC в ячейку по адресу  IP-14(в R) |
| 3B1 | 83A2 | LOOP 3A2 | MEM(3A2) – 1 -> MEM(3A2); Если MEM(3A2) <= 0, то IP + 1 -> IP |
| 3B2 | CEF9 | JUMP (IP-6) | Прямой относительный прыжок IP-6+1 -> IP |
| 3B3 | 0100 | HLT | Остановка |

## Описание программы

Программа ищет **максимальный ненулевой элемент** массива из n элементов (хранящихся в памяти по некоторому указателю).

* Все элементы массива — 16-битные целые (знаковые).
* В ячейке 3A3 хранится текущее «максимальное найденное» (изначально 0x8000 = −32768).
* Программа перебирает элементы один за другим, **пропуская** те, которые равны нулю, и сравнивая остальные с текущим «максимумом». Если элемент ≥ хранимого значения, программа обновляет 3A3 новым элементом.
* По завершении цикла в **MEM(3A3)** содержится:

max( {−32768} ∪ {xi​∣xi ≠0})

Иначе говоря, если **все** элементы массива были равны нулю, результат остаётся −32768.

* Счётчик n (число элементов) хранится в 3A2, автоматически уменьшается на каждой итерации. Когда он достигает 0, программа останавливается.

### Формула результата

Пусть массив x1, x2,…,xn. Тогда

result = max( {−32768} ∪ {xi​∣xi ≠0}) , если  существует  хотя  бы  один  xi≠0 ,если все xi=0.

## Область представления

1. **arr\_length** (ячейка 3A2)
   * 16‑разрядное целое беззнаковое (или знаковое, но программа использует только положительные значения).
   * Хранит число n — количество элементов массива, которые надо обработать.
2. **arr\_ptr** (ячейка 3A1)
   * 16‑разрядный указатель (адрес), указывающий на первый элемент массива.
   * В цикле автоинкрементируется, чтобы «сдвигаться» по элементам.
3. **result** (ячейка 3A3)
   * 16‑разрядное знаковое целое, в котором хранится «текущий максимум» среди уже просмотренных ненулевых элементов.
   * Изначально 0x8000 (−32768).
4. **arr[i]**
   * Каждый элемент массива — 16‑разрядное знаковое целое ([−32768.. +32767]).
   * Физически находится в памяти по адресу, который формируется из начального указателя (arr\_ptr) и смещения.
5. **Программа** хранится в области памяти (3A4…3B3). Сами переменные 3A0…3A3 соседствуют либо внутри этой области, либо рядом, в зависимости от точной схемы размещения.

## Область допустимых значений

1. **arr\_length** ∈[1..Nmax]
   * Чтобы цикл корректно завершался и не выходил за границы памяти, n должно быть в разумном диапазоне, скажем 1≤n≤4.
   * Если n=0, программа может завершиться сразу, но тогда результат останется −32768.
2. **arr[i]** ∈[−32768..+32767]
   * При любом значении элемент остаётся валидным, так как программа лишь проверяет «равно ли 0» и «AC ≥ result».
3. **result** (в 3A3) ∈[−32768..+32767]
   * По ходу работы это «плавающее» знаковое число в 16 битах. Изначально −32768.
   * На выходе оно либо останется −32768, если все элементы были 0, либо будет равно какому‑то ненулевому значению массива.
4. **Указатель arr\_ptr** (3A1) должен указывать на область памяти, где лежат nnn элементов. То есть arr\_ptr+(n−1) не выходит за «легальную» зону памяти машины (например, [0x0000..0xFFFF] в 16‑битной адресации).

**ОДЗ** включает:

* n≥0 (или >0, если хотим хотя бы один элемент),
* Любые 16‑битные значения массива,
* Корректный указатель arr\_ptr, чтобы не «вылететь» за границы памяти.

Программа не накладывает иных ограничений: любые элементы, отрицательные, положительные, нулевые, — обрабатываются без ошибок в пределах 16 бит.

# Расположение данных в памяти

* 3A0, 3A2, 38A, 3B4, 3B5,3B6,3B7 – исходные данные;
* 3A1 – промежуточный результат;
* 3A3 – итоговый результат;
* 3A4 – 3B3 – команды

# Адреса первой и последней выполняемой команды

* Адрес первой команды: 3A4
* Адрес последней команды: 3B3

# Таблица трассировки

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Выполняемая команда** | | **Содержание регистров в процессоре после выпо лнения команды** | | | | | | | | | **Ячейка, содержимое которой изменилось после выполнения команды** | |
| Адрес | Код | IP | CR | AR | DR | SP | BR | AC | PS | NZVC | Адрес | Новый код |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

# Вывод

В ходе выполнения данной лабораторной работы познакомился с устройством БЭВМ. Изучил её структуру, принцип функционирования БЭВМ на уровне машинных команд, систему команд БЭВМ, познакомился с представлением логической информации и чисел, научился выполнять трассировку собственной программы. Проанализировал программу для базовой ЭВМ и разработал вариант с меньшим числом команд.