

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
Ход работы.....	3
Описание Программы.....	3
Область представления	4
Область допустимых значений	4
Расположение данных в памяти.....	5
Адреса первой и последней выполняемой команды.....	5
Таблица трассировки	5
Вывод	6

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.

3A0:	03B4		3AE:	7EF4
3A1:	A000		3AF:	F801
3A2:	E000		3B0:	EEF2
3A3:	E000		3B1:	83A2
3A4:	+ AF40		3B2:	CEF9
3A5:	0680		3B3:	0100
3A6:	0500		3B4:	0800
3A7:	EEFB		3B5:	0000
3A8:	AF04		3B6:	0000
3A9:	EEF8		3B7:	F000
3AA:	AEF5			
3AB:	EEF5			
3AC:	AAF4			
3AD:	F003			

1. Ход работы

Текст исходной программы

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

2. 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 \geq хранимого значения, программа обновляет 3A3 новым элементом.
- По завершении цикла в **MEM(3A3)** содержится:

$\max(\{-32768\} \cup \{x_i | x_i \neq 0\})$

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 x_1, x_2, \dots, x_n . Тогда

$\text{result} = \max(\{-32768\} \cup \{x_i | x_i \neq 0\})$, если существует хотя бы один $x_i \neq 0$, если все $x_i = 0$.

3. 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.

4. Область допустимых значений

1. **arr_length** $\in [1..N_{\max}]$
 - Чтобы цикл корректно завершался и не выходил за границы памяти, n должно быть в разумном диапазоне, скажем $1 \leq n \leq 4$.
 - Если $n=0$, программа может завершиться сразу, но тогда результат останется -32768 .
2. **arr[i]** $\in [-32768..+32767]$
 - При любом значении элемент остаётся валидным, так как программа лишь проверяет «равно ли 0» и « $AC \geq \text{result}$ ».
3. **result** (в $3A3$) $\in [-32768..+32767]$

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).

- $n \geq 0$ (или > 0 , если хотим хотя бы один элемент),
- Любые 16-битные значения массива,
- Корректный указатель `arr_ptr`, чтобы не «вылететь» за границы памяти.

5. Расположение данных в памяти

- #### 6. Адреса первой и последней выполняемой команды

- ## 7. Таблица трассировки

[illegible]

8. 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.