МИНИСТРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное автономное

образовательное учреждение

высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «Московский Политех»)

Факультет: «Машиностроение»

Кафедра: «Автоматика и управление»

Дисциплина: «Микропроцессорные системы управления»

Курсовая работа

Тема: «Работа с массивами и числами со знаками на intel 8086»

Группа 201-251

Выполнил:

Зарубин Илья Александрович

Проверил:

Палагута Константин Алексеевич

Москва 2023

Оглавление

1 Формулировка задания 3
2 Теоретическая часть 4
3 Выполнение задания 8
3.1 Описание программы блок-схемы 8
3.2 Результаты проверки программы 10
Заключение 13
Список использованных источников и литературы 14
Приложение 1 15

1 Формулировка задания

Цель курсовой работы: научиться работать с массивами и обрабатывать числа со знаком на микроконтроллере intel 8086 и выполнить задание.

Формулировка задания: для массива из 12 16-разрядных чисел со знаком разработать блок-схему алгоритма, программу на языке ассемблера и в машинных кодах микропроцессора К1810ВМ86, которая сортирует массив слов на четные и нечетные. Также требуется:

— расположить четные числа в порядке убывания модуля,

— расположить нечетные числа в порядке возрастания модуля,

— рассчитывает среднее арифметическое всех чисел, кратных 3.

2 Теоретическая часть

Для выполнения задания, необходимо было изучить и использовать команды приведенный в таблице 1.

Таблица 2.1. Команды, которые использовались при выполнении задания

Мнемокод	Команда	Что она делает
MOV opr1, opr2	Пересылка	Копирует opr2 в opr1
XCHG opr1, opr2	Перестановка операндов	opr1, opr2 = opr2, opr1
ADD opr1, opr2	Сложение	opr1 = opr1 + opr2
SUB opr1, opr2	Вычитание	opr1 = opr1 – opr2
INC opr1	Инкремент	opr1 = opr1 + 1
DEC opr1	Декремент	opr1 = opr1 - 1
NEG opr1	Отрицание	Меняет знак у opr1
IDIV opr1	Деление со знаком	Если opr1 – байт: AL = AX / opr1 AH = остаток Если opr1 – слово: AX = (DX AX) / операнд DX = остаток
CMP opr1, opr2	Сравнение	opr1 – opr2 Результат никуда не записывается. Флаги устанавливаются (OF, SF, ZF, AF, PF, CF) в соответствии с результатом.
TEST opr1, opr2	Логическое И между всеми битами двух операндов	Не изменяет результирующий операнд, а влияет

		только на флаги. Задействованы следующие флаги: ZF, SF, PF.
JNZ label	Переход если не ноль	Если ZF=0, то выполняется переход по метке label
JZ label	Переход по ноль	Если ZF=1, то выполняется переход по метке label
JGE label	Короткий переход, если первый операнд больше или равен второму	Если SF=OF, то выполняется переход по метке label
JE label	Переход если первый операнд равен второму	Если ZF=1, то выполняется переход по метке label
JMP label	Безусловный переход	Выполнить переход в любом случае
JNB label	Переход если первый операнд не меньше второго операнда. Беззнаковый.	Если CF = 0, то выполняется переход по метке label
JNA label	Переход если первый операнд не больше второго операнда. Беззнаковый.	Если (CF = 1 OR ZF = 1), то выполняется переход по метке label
LOOP label	Переход если CX<>0	CX=CX-1 Если CX<>0, то выполняется переход по метке label
JCXZ label	Переход если CX=0	Если CX=0, то выполняется переход по метке label

Для сортировки массивов я использовал алгоритм сортировки пузырьком.

Основная идея алгоритма сортировки пузырьком – последовательно сравнивать значение соседних элементов и менять их местами, если предыдущий элемент больше последующего. Таким образом, наибольший оказывается в конце массива, а наименьший в начале.

Сортировка заключается в том, чтобы за каждый проход последовательно сравнивать соседние элементы. Если предыдущий элемент больше последующего, то происходит обмен значения элемента. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на свое место в конце неотсортированной части массива рядом с предыдущим наибольшим элементов, а наименьший на одну позицию к началу массива. Если после прохода по внутреннему циклу не было ни одной перестановки, это значит, что массив отсортирован.

Также этот алгоритм можно использовать не только для сортировки чисел по возрастанию, но и по убыванию тоже.

Блок-схема алгоритма сортировки пузырьком чисел по возрастанию представлена на рис. 2.1.

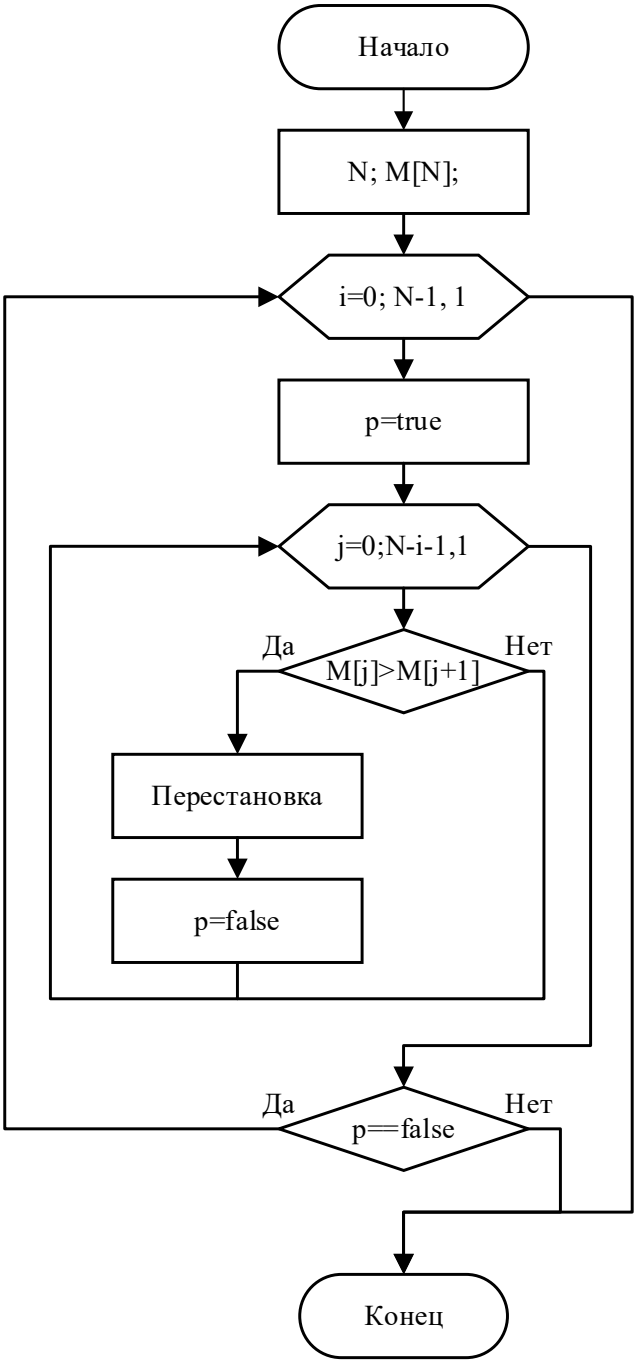

Рис. 2.1. Блок-схема алгоритма сортировки пузырьком

3 Выполнение задания

Параметры программы

— [0x0292] - с этой ячейки начинается исходный массив чисел.

— [0x0010] – с этой ячейки начинается массив четных чисел.

— [0x0030] – с этой ячейки начинается массив нечетных чисел.

— [0x0050] – с этой ячейки начинается массив чисел кратных 3.

— [0x0080] – ячейка для записи среднего арифметического всех чисел кратных 3.

— [0x0082] – ячейка для записи остатка среднего арифметического всех чисел кратных 3.

3.1 Описание программы блок-схемы

Описание программы блок-схемы, представленной на рис. 3.1.

1. Инициализация всех перемен и регистров.
2. Проход по массиву, распределением чисел в два массива по четности. Проверка числа на кратность 3, запись этого числа в массив кратных 3.
3. Сортировка нечетного массива по убыванию, брав число с модулем.
4. Сортировка четного массива по возрастанию, брав число с модулем.
5. Проход по массиву, деля каждое число из массива кратных 3 на количество элементов массива и суммируя частное и остаток в две ячейки памяти.
6. Разделить ячейку с остатком на его количество и прибавить результат деление остатка в ячейку с частным, а результат остатка после деление записать в ячейку с остатком.

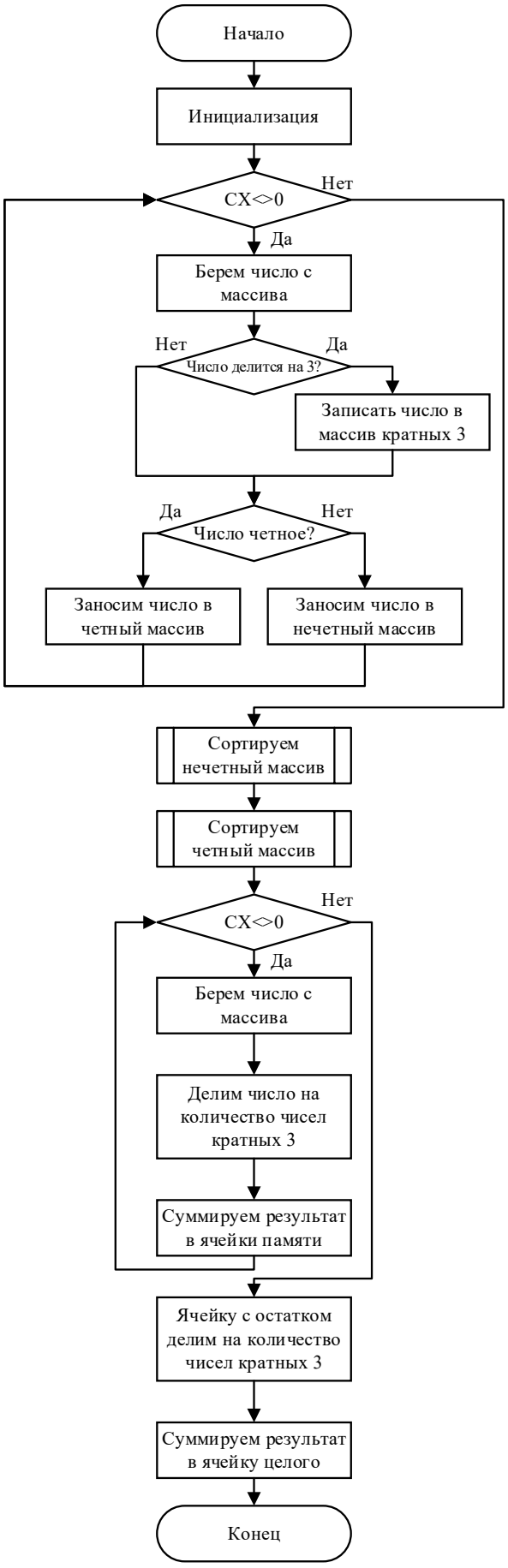

Рис. 3.1. Блок-схема программы

3.2 Результаты проверки программы

Результаты теста программы №1 приведены в таблице 3.1.

Таблица 3.1. Тест программы с набором №1

Входные значения	04D2, 162E, 2334, 0D80, E12E, F6D7, 1538, 2694, 10E1, 223D, F7C3, E58B
Итоговый четный массив в порядке убывания модуля	2694, 2334, E12E, 162E, 1538, 0D80, 04D2
Итоговый нечетный массив в порядке возрастания модуля	F7C3, F6D7, 10E1, E58B, 223D;
Среднее арифметическое всех чисел, кратных 3	09A5

Результаты теста программы №2 приведены в таблице 3.2.

Таблица 3.2. Тест программы с набором №2

Входные значения	DDDD, BCDA, BCDF, 12A3, 574F, 0000, 1357, 8888, 1337, 4488, 228C, DEF1
Итоговый четный массив в порядке убывания модуля	8888, 4488, BCDA, 228C, 0000
Итоговый нечетный массив в порядке возрастания модуля	12A3, 1337, 1357, DEF1, DDDD, BCDF, 574F
Среднее арифметическое всех чисел, кратных 3	FACA

Результаты теста программы №3 приведены в таблице 3.3.

Таблица 3.3. Тест программы с набором №3

Входные значения	FFFF, 0000, 8000, 0001, 4000, C000, F000, 0FFF, 8FFF, 8765, 1122, FFFF
Итоговый четный массив в порядке убывания модуля	8000, C000, 4000, 1122, F000, 0000

Итоговый нечетный массив в порядке возрастания модуля	FFFF, FFFF, 0001, 0FFF, 8FFF, 8765
Среднее арифметическое всех чисел, кратных 3	0B0B

Результаты теста программы №4 приведены в таблице 3.4.

Таблица 3.4. Тест программы с набором №4

Входные значения	FFFF, 0000, 8001, 0001, 40C3, C000, F000, 0FFF, 8FCF, 8765, 1122, FFFF
Итоговый четный массив в порядке убывания модуля	C000, 1122, F000, 0000
Итоговый нечетный массив в порядке возрастания модуля	FFFF, 0001, FFFF, 0FFF, 40C3, 8FCF, 8765, 8001
Среднее арифметическое всех чисел, кратных 3	0B0B

Результаты теста программы №5 приведены в таблице 3.5.

Таблица 3.5. Тест программы с набором №5

Входные значения	0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002
Итоговый четный массив в порядке убывания модуля	0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002
Итоговый нечетный массив в порядке возрастания модуля	-
Среднее арифметическое всех чисел, кратных 3	-

Результаты теста программы №6 приведены в таблице 3.6.

Таблица 3.6. Тест программы с набором №6

Входные значения	0000, 0003, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002

Итоговый четный массив в порядке убывания модуля	0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0002, 0000
Итоговый нечетный массив в порядке возрастания модуля	0003
Среднее арифметическое всех чисел, кратных 3	0001
Остаток среднего арифметического всех чисел, кратных 3	0001

Заключение

Выполняя данную курсовую работу, я научился работать с массивами, научился выполнять операции над числами со знаком, проверять четность чисел, сортировать числа по модулю, проверять делимость числа на 3.

Список использованных источников и литературы

1. Полный набор команд процессора 8086. [Электронный ресурс]. - Режим доступа: http://www.avprog.narod.ru/progs/emu8086/8086_instruction_set.html.

2. Роберт Седжвик. Алгоритмы на C++. - Москва: Издательский дом «Вильямс», 2016

3. К. А. Палагута. Микропроцессор INTEL 8086 (K1810BM86) и его программирование. – Москва: МГИУ, 2007.

Приложение 1

Листинг программы

```
ORG 100h

;array DW 004D2h, 0162Eh, 02334h, 00D80h, 0E12Eh, 0F6D7h, 01538h,
02694h, 010E1h, 0223Dh, 0F7C3h, 0E58Bh
;array DW 0DDDDh, 0BCDAh, 0BCDFh, 012A3h, 0574Fh, 00000h, 01357h,
08888h, 01337h, 04488h, 0228Ch, 0DEF1h
;array DW 0FFFFh, 00000h, 08000h, 00001h, 04000h, 0C000h, 0F000h,
00FFFh, 08FFFh, 08765h, 01122h, 0FFFFh
;array DW 07FFBh, 07FFBh, 07FFBh, 07FFBh, 07FFBh, 07FFBh, 07FFBh,
07FFBh, 07FFBh, 07FFBh, 07FFBh, 07FFBh

;[0x0010] - с этой ячейки начинается массив четных чисел.
;[0x0030] - с этой ячейки начинается массив нечетных чисел.
;[0x0050] - с этой ячейки начинается массив чисел кратных 3.
;[0x0080] - ячейка для записи среднего арифметического всех чисел кратных 3.
;[0x0082] - ячейка для записи остатка среднего арифметиче-ского всех чисел кратных 3.

count_even DW 0
addres_array_even DW 00010h
index_array_even DW 0

count_odd DW 0
addres_array_odd DW 00030h
index_arrar_odd DW 0
```

```
count_value_3 DW 0
addres_avg_value DW 00080h
addres_array_avg_value_3 DW 00050h
index_array_avg_value_3 DW 0

start:
    MOV SI, 0
    MOV BP, 0
    MOV CX, 12
    MOV BX, 3
    OUTPUT:
        MOV AX, array[SI]
        MOV DX, 0
        TEST AX, 08000h
        JZ NO_NEGATIVE_AX
        MOV DX, 0FFFFh
        NO_NEGATIVE_AX:
        IDIV BX
        CMP DX, 0
        JNE NO_IDIV3
        INC [count_value_3]
        MOV AX, array[SI]
        MOV BP, [addres_array_avg_value_3]
        MOV DI, [index_array_avg_value_3]
        MOV [BP+DI], AX
        ADD [index_array_avg_value_3], 2
        NO_IDIV3:
        MOV AX, array[SI]
        TEST AX, 1
```

```
        JNZ NUMBER_ODD
        JZ NUMBER_EVEN
    NUMBER_ODD:
        MOV DI, [index_arrar_odd]
        MOV BP, [addres_array_odd]
        MOV [BP+DI], AX
        ADD SI, 2
        INC [count_odd]
        ADD [index_arrar_odd], 2
    LOOP OUTPUT
    JCXZ NEXT_ARRAY_SORT
    NUMBER_EVEN:
        MOV DI, [index_array_even]
        MOV BP, [addres_array_even]
        MOV [BP+DI], AX
        ADD SI, 2
        INC [count_even]
        ADD [index_array_even], 2
    LOOP OUTPUT
    JCXZ NEXT_ARRAY_SORT
NEXT_ARRAY_SORT:
; ================================
; SORT ARRAY ODD
; ================================
MOV AX, [count_odd]
CMP AX, 0
JE EXIT_SORT_ODD
MOV [00001h], AX
DEC [00001h]
```

```
MOV BX, 0
MOV BP, [addres_array_odd]
LOOOP_START_ODD:
   CMP BX, [00001h]
   JGE EXIT_SORT_ODD
   MOV CX, [00001h]
   SUB CX, BX
   MOV SI, 0
   MOV DX, 0
   MOV DI, 0
   LOOP_SORT_ODD:
      MOV DI, [BP+SI+2]
      TEST DI, 08000h
      JZ NO_NEGATIVE_WORD2_ODD
      NEG DI
      NO_NEGATIVE_WORD2_ODD:
      MOV AX, [BP+SI]
      TEST AX, 08000h
      JZ NO_NEGATIVE_WORD1_ODD
      NEG AX
      NO_NEGATIVE_WORD1_ODD:
      CMP AX, DI
      JNA NO_SWAP_ODD
      MOV AX, [BP+SI]
      XCHG AX, [BP+SI+2]
      MOV [BP+SI], AX
      MOV DX, 1
      NO_SWAP_ODD:
      ADD SI, 2
```

```
        LOOP LOOP_SORT_ODD
    INC BX
    CMP DX, 0
    JE EXIT_SORT_ODD
    JMP LOOOP_START_ODD
EXIT_SORT_ODD:
; ================================
; SORT ARRAY EVEN
; ================================

MOV AX, [count_even]
CMP AX, 0
JE EXIT_SORT_EVEN
MOV [00001h], AX
DEC [00001h]
MOV BX, 0
MOV BP, [addres_array_even]
LOOOP_START_EVEN:
    CMP BX, [00001h]
    JGE EXIT_SORT_EVEN
    MOV CX, [00001h]
    SUB CX, BX
    MOV SI, 0
    MOV DX, 0
    LOOP_SORT_EVEN:
        MOV DI, [BP+SI+2]
        TEST DI, 08000h
        JZ NO_NEGATIVE_WORD2_EVEN
        NEG DI
```

```
        NO_NEGATIVE_WORD2_EVEN:
        MOV AX, [BP+SI]
        TEST AX, 08000h
        JZ NO_NEGATIVE_WORD1_EVEN
        NEG AX
        NO_NEGATIVE_WORD1_EVEN:
        CMP AX, DI
        JNB NO_SWAP_EVEN
        MOV AX, [BP+SI]
        XCHG AX, [BP+SI+2]
        MOV [BP+SI], AX
        MOV DX, 1
        NO_SWAP_EVEN:
        ADD SI, 2
        LOOP LOOP_SORT_EVEN
    INC BX
    CMP DX, 0
    JE EXIT_SORT_EVEN
    JMP LOOOP_START_EVEN
EXIT_SORT_EVEN:
; ================================
; AVG VALUE 3
; ================================
MOV SI, 0
MOV BP, [addres_array_avg_value_3]
MOV DI, [addres_avg_value]
MOV CX, [count_value_3]
MOV BX, CX
MOV w. [DI], 00000h
```

```
MOV w. [DI+2], 00000h
CMP CX, 0
JE EXIT_AVG_VALUE
AVG_ARRAY_LOOP:
   MOV AX, [BP + SI]
   MOV DX, 0
   TEST AX, 08000h
   JZ NO_NEGATIVE_IDIV
   MOV DX, 0FFFFh
   NO_NEGATIVE_IDIV:
   IDIV BX
   ADD [DI],   AX
   ADD [DI+2], DX
   ADD SI, 2
LOOP AVG_ARRAY_LOOP
MOV AX, [DI+2]
MOV DX, 0
TEST AX, 08000h
JZ NO_NEGATIVE_IDIV_2
NEG AX
NO_NEGATIVE_IDIV_2:
IDIV BX
TEST [DI], 08000h
JZ NO_NEGATIVE_IDIV_3
NEG AX
NO_NEGATIVE_IDIV_3:
ADD [DI],   AX
MOV [DI+2], DX
EXIT_AVG_VALUE:
```

HLT

;array DW 0FFFFh, 00000h, 08001h, 00001h, 040C3h, 0C000h, 0F000h, 00FFFh, 08FCFh, 08765h, 01122h, 0FFFFh

array DW 00000h, 00003h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h

;array DW 00002h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h, 00002h

HLT