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](#_Toc129904850)

[2 Теоретическая часть 4](#_Toc129904851)

[3 Выполнение задания 7](#_Toc129904852)

[3.1 Описание программы блок-схемы 7](#_Toc129904853)

[3.2 Результаты проверки программы 9](#_Toc129904854)

[Заключение 10](#_Toc129904855)

[Список использованных источников и литературы 11](#_Toc129904856)

[Приложение 1 12](#_Toc129904857)

# 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 |
| JNBE label | Переход если первый операнд больше. Беззнаковый. | Если (CF = 0 and ZF = 0), то выполняется переход по метке label |
| JNAE label | Переход если первый операнд больше. Беззнаковый. | Если CF = 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

# 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.
3. Сортировка нечетного массива по убыванию, брав число с модулем.
4. Сортировка четного массива по возрастанию, брав число с модулем.
5. Инициализация всех перемен и регистров.
6. Проход по массиву, проверяя делимость числа на 3, если делится, то суммировав его в ячейку памяти и добавив данное число в массив.
7. Разделить сумму чисел делимых на 3, на их количество.



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 |

# 

# 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

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\_value\_sum\_3 DW 00050h

addres\_array\_avg\_value\_3 DW 00050h

index\_array\_avg\_value\_3 DW 0

start:

MOV SI, 0

MOV BP, 0

MOV CX, 12

OUTPUT:

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]

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

JNAE 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]

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

JNBE 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 CX, 12

MOV BX, 3

AVG\_OUTPUT:

MOV AX, array[SI]

MOV DX, 0

TEST AX, 08000h

JZ NO\_NEGATIVE

MOV DX, 0FFFFh

NO\_NEGATIVE:

IDIV BX

CMP DX, 0

JNE NO\_SUM3

INC [count\_value\_3]

MOV DI, [addres\_value\_sum\_3]

MOV AX, array[SI]

ADD [DI], AX

MOV DI, [index\_array\_avg\_value\_3]

MOV [BP+DI], AX

ADD [index\_array\_avg\_value\_3], 2

NO\_SUM3:

ADD SI, 2

LOOP AVG\_OUTPUT

MOV AX, 0

MOV BX, 0

MOV BP, [addres\_value\_sum\_3]

MOV AX, [BP]

MOV DX, 0

TEST AX, 08000h

JZ NO\_NEGATIVE\_IDIV

MOV DX, 0FFFFh

NO\_NEGATIVE\_IDIV:

MOV BX, [count\_value\_3]

CMP BX, 0

JE NO\_IDIV\_AVG

IDIV BX

NO\_IDIV\_AVG:

HLT