ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

ассистент		И.Д. Свеженин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

ОБРАБОТКА МАССИВОВ

по курсу: Архитектура ЭВМ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4329		Д.С. Шаповалова
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

1. Цель работы:

Изучение работы с массивами, организации арифметических циклов в языке ассемблера.

2. Задание:

1. Сформировать исходные данные в соответствии с вариантом.
2. Составить алгоритм программы для решения поставленной задачи.
3. Провести трассировку заданного алгоритма с использованием заданных исходных данных.
4. Составить программу заданного алгоритма в мнемокодах.
5. Оформить отчет по лабораторной работе.
6. В учебной лаборатории проверить результаты выполнения программы в программе-отладчике, сравнивая их с результатами ручной трассировки алгоритма.

Для всех заданий исходный массив хранится в сегменте данных, результаты необходимо сохранить в РОНы.

Вариант 1:

Найти логическую сумму положительных элементов массива и записать ее в Rg AX, и логическую сумму отрицательных элементов массива, записать ее в Rg BX (формат элементов массива - байт).

3. Ход работы:

1. Определение исходных данных:

Были выбраны 10 чисел в размерности байт, в пределах от -128 до +127:

15, -30, 7, 0, -5, 12, -1, 8, -100, 44

2. Ручная трассировка алгоритма с использованием исходных данных:

Положительные элементы:

15 = 00001111

7 = 00000111

12 = 00001100

8 = 00001000

44 = 00101100

00001111 (15) **OR** 00000111 (7) = 00001111

OR 00001100 (12) = 00001111

OR 00001000 (8) = 00001111

OR 00101100 (44) = 00101111

Итого: 00101111

Нули – игнорируются.

Отрицательные в доп. коде:

-30 = 11100010

-5 = 11111011

-1 = 11111111

-100 = 10011100

11100010 (-30) **OR** 11111011 (-5) = 11111011

OR 11111111 (-1) = 11111111

OR 10011100 (-100) = 11111111

3. Описание алгоритма:

Загружаем в память массив, в CX загружаем кол-во элементов.

Идём по каждому элементу.

Сравниваем с 0:

Если равно 0, то игнорируем, идём к следующему элементу.

Если меньше нуля – обрабатываем как негативное – применяем операцию OR текущего числа в AL, к тому что хранится в DL.

Если больше – применяем операцию OR текущего числа AL к тому, что хранится в DH.

Минусуем кол-во циклов (CX).

Записываем результат по положительным в AL, по отрицательным в BL.

4. Алгоритм в мнемокодах

```
.MODEL SMALL
.STACK 256

.DATA
    Mas DB 15, -30, 7, 0, -5, 12, -1, 8, -100, 44

.CODE
MAIN PROC FAR

    PUSH DS
    PUSH AX

    MOV AX, @DATA
    MOV DS, AX

    MOV CX, 10              ;hm loops
    LEA SI, Mas             ;load effective address

LOOP_START:
    MOV AL, [SI]

    CMP AL, 0               ;compare
    JE NEXT                 ;jump if equal

    JS NEGATIVE             ;jump if sign (s=1)

    OR DL, AL
    JMP NEXT                ;jump to NEXT

NEGATIVE:
    OR DH, AL               ; DH or(logic +) AL

NEXT:
    INC SI                  ;to next element (SI++)
```

```
    LOOP LOOP_START


    MOV AX, 0
    MOV AL, DL
    MOV BX, 0
    MOV BL, DH

    RET                     ;back to DOS
MAIN ENDP

END MAIN
```

4. Вывод:

В ходе выполнения работы была разработана и отлажена программа на языке ассемблера для микропроцессора Intel 8086, реализующая обработку массива из 10 байтов со знаком. Программа корректно вычисляет логическую сумму (побитовое ИЛИ) всех положительных элементов массива и записывает результат в регистр AX, а также логическую сумму всех отрицательных элементов — в регистр BX.

Реализация выполнена в соответствии с методическими указаниями ГУАП: использованы сегменты данных, стека и кода, применена косвенная адресация через регистр SI, организован арифметический цикл с помощью команды LOOP, а ветвление — с использованием команд условного перехода (CMP, JE, JS).

Результаты ручной трассировки совпали с результатами выполнения программы в эмуляторе emu8086, что подтверждает корректность алгоритма и программной реализации.