**Лабораторная работа по информатике №2**

**Задание 1**

На вход поступает массив, состоящий из целых чисел. Необходимо посчитать количество отрицательных элементов в массиве.

**Листинг**

section .text

global CMAIN

CMAIN:

MOV EBP, ESP

XOR ECX, ECX

LEA EBX, [a]

MOV EDI, 6

MOV ESI, 0

Next:

CMP ESI, EDI

JE End

MOV AL, [EBX + ESI]

CMP AL, 0

JG Increment

INC ECX

INC ESI

JMP Next

Increment:

INC ESI

JMP Next

End:

PRINT\_DEC 1, ECX

ret

section .data

a: DB 5, -2, 4, -7, -1, 10

**Анализ**

Мы проходимся по заданному массиву, затем сравниваем ESI и EDI (длину массива и число пройденных элементов); в метке END – конечный переход и запись количества отрицательных элементов, в регистр AL мы записываем сумму элемента массива и 0, при этом если число отрицательное, то произойдёт переход к метке Increment, в которой считаются отрицательные элементы, иначе – происходит переход через этот элемент и сравнение следующего элемента. Таким образом, для заданного массива [5, -2, 4, -7, -1, 10] результатом работы будет число 3, которое соответствует количеству отрицательных элементов массива

**Задание 2**

На вход поступает массив из положительных чисел, меньших 64 (т.к. заданная разрядность элементов 6 бит). Необходимо написать программу, реализующую доступ к упакованному массиву с указанной разрядностью элементов.

**Листинг**

section .text

global CMAIN

CMAIN:

LEA ESI, [a]

**;Для извлечения первого элемента**

MOV AL, [ESI]

MOV DL, 0xBC

AND AL, DL

SHR AL, 2

PRINT\_UDEC 1, AL

NEWLINE

**; Для извлечения второго элемента**

**; достаём старшие 2 бита**

MOV AL, [ESI]

MOV DL, 0x03

AND AL, DL

SHL AL, 4

MOV BL, AL

**; сохраняем старшие 2 бита**

**; достаем младшие 4 бита**

MOV AL, [ESI + 1]

MOV DL, 0xF0

AND AL, DL

SHR AL, 4

OR DL, AL

**; накладываем старшие и младшие биты, чтобы получить число полностью**

PRINT\_UDEC 1, AL

NEWLINE

**; Для извлечения третьего элемента**

**; достаем старшие 4 бита**

MOV AL, [ESI + 1]

MOV DL, 0x0F

AND AL, DL

SHL AL, 2

MOV BL, AL

**; достаем младшие 2 бита**

MOV AL, [ESI + 2]

MOV DL, 0xC0

AND AL, DL

SHR AL, 6

OR AL, BL

PRINT\_UDEC 1, AL

NEWLINE

**; Для извлечения четвертого элемента**

MOV AL, [ESI + 2]

MOV DL, 0x3F

AND AL, DL

PRINT\_UDEC 1, AL

NEWLINE

ret

section .data

**; packed array [47, 1, 60, 29]**

a: DB 0xBC, 0x1F, 0x1D

**Анализ**

Мы берём несколько чисел, меньших 64, переводим их в двоичную систему по 6 бит, затем делим на восьмёрки, переводим эти двоичные числа в 16-ричную СС и забиваем эти числа в массив. Затем для извлечения, например, 2-ого элемента, 2 старших бита, переносим их на 4 влево, чтобы они встали на свои места, то же самое с 4 младшими битами, затем накладываем биты друг друга операцией OR, чтобы получить искомое число, и выводим это число с помощью PRINT\_UDEC в регистре AL.