

**Министерство науки и высшего образования РФ**  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Вятский государственный университет»

Факультет автоматики и вычислительной техники  
Кафедра радиоэлектронных средств

Отчет по лабораторной работе №1  
«Принципы выполнения команд ветвления,  
организация циклов и подпрограмм»

Дисциплина «Цифровые устройства и микропроцессоры»

Вариант № 19

Выполнил: студент группы ИНБс-3301-01-00 \_\_\_\_\_ В.О. Игнатович  
Проверил: преподаватель кафедры РЭС \_\_\_\_\_ М.А. Земцов

Киров 2025

## 1 Цели и задачи

Цель работы — изучение принципов выполнения команд ветвления, циклов и функций на языке ассемблере.

Задачи:

- изучить команды ветвления, написания циклов, подпрограмм;
- решить предложенные по варианту задания посредством Microsoft Macro Assembler.

## 2 Ход работы

Задания по варианту №19 приведены на рисунке 1.

Дано: X=9189 Y=714B Z=CD12 (расположены в памяти один за другим)		ЗАДАНИЕ № 19
В цикле произвести обнуление старших байтов данных X,Y,Z (результат X', Y', Z')	Вычислить $M = X' + Y' \& Z'$ мл. $M > 0$ переход к п/п 1 ( $R = M - 4101$ ) мл. $M \leq 0$ переход к п/п 2 ( $R = M \text{ xor } E130$ )	Если в мл. R четное количество единиц, то переход к АДР1 (R or 1024), иначе переход к АДР2 (R/2)

Рисунок 1 — Задания по варианту

Для решения поставленных задач был создан проект в среде Visual Studio 2019 для ЯП C++ с настройками для работы с ASM.



Заданные переменные представлены в таблице 2.1.




Этапы решения поставленных задач с помощью калькулятора и masm представлены в таблице 2.2.

Таблица 2.1 — Представление значений заданных переменных.

Переменная	В 16 CC	В 10 CC	В 2 CC (16 бит)
X	9189h	37257	10010001 10001001
Y	714Bh	29003	01110001 01001011
Z	CD12h	52498	11001101 00010010

Таблица 2.2 — Этапы расчёта.

№ операции	Операция	Результат	
		Калькулятор	masm
1	X' (zero first byte)	Hex result 89 = 137	
2	Y' (zero first byte)	Hex result 4B = 75	

3	Z` (zero first byte)	Hex result 12 = 18	 Z 0x0012
4	Y` & Z`	Hex result 2 = 2	 ebx 0x00000002
5	M = X` + Y` & Z`	Hex result 8B = 139	 ebx 0x0000008b
6	R = M - 4101h	Hex result -4076 = -16502 = = -100000001110110b  Перевод в обратный код:  + 1 = = 11111111111111111111111111111111001b = = FFFFBF8Ah = 4294950794	 R 0xffffbf8a
7	число единиц в мл. R	мл. R (выделенная часть) =  = 10001010 число единиц = 3 (нечёт.)	 cnt 3 (нечёт.)
8	нечётн. $\Rightarrow$ $\Rightarrow R = R / 2$	Hex result 7FFDFC5 = 2147475397	 R 2147475397

Полученный код изображён на рисунке 2.

```

.686
.model flat,stdcall
.stack 100h          ; stack size
.data                ; vars here
X      word 37257
Y      word 29003
Z      word 52498
M      word 0
R      dword 0
two    dword 2
cnt     word 0

.code
ExitProcess PROTO STDCALL :DWORD
Start:
    xor eax, eax
    xor ebx, ebx
    xor ecx, ecx
    xor edx, edx
    mov dl, 0
    mov eax, offset X    ; getting address of X
    mov cx, 3
    mov si, 0

; Task 1 there:
loop_s:
    inc eax
    mov [eax], dl
    inc eax
    inc si
    loop loop_s
; Task 2 there:
    mov bx, Y
    and bx, Z
    add bx, X
    mov M, bx
    cmp ebx, 0
    ja on_larger_then_zero
    ; if (M <= 0) {
    xor ebx, 0E130h
    jmp endd
    ; }
    ; if (M > 0) {
on_larger_then_zero:
    sub ebx, 04101h
    ; }
endd:
    mov R, ebx
; Task 3 there:
    mov cx, 8    ; 8 times for 8 bit in 1 byte
                ; R[0] is in bx
    xor eax, eax
    and ebx, 11111111b ; keeping ebx[0] only
    mov ax, bx
loop_s2:
    div two
    add cnt, dx
    ;sar bx, 1
    loop loop_s2
    ; now amount of ones is in cnt
    mov ebx, R
    test cnt, 1
    jz on_even    ; if (cnt % 2 == 0) jmp on_even;
                    ; else {
    call on_not_even_proc
    jmp endd2
                    ; }
on_even:
    call on_even_proc
    jmp endd2

on_even_proc PROC
    or ebx, 1024
    ret
on_even_proc ENDP

on_not_even_proc PROC
    shr ebx, 1    ; div by 2
    ret
on_not_even_proc ENDP

endd2:
    mov R, ebx
exit:
Invoke ExitProcess,1
End Start

```

Рисунок 2 — Код программы

### **3 Вывод**

Нами были изучены способы перемещения позиции текущей инструкции посредством условных и безусловных переходов. Также были изучены методы создания циклов и подпрограмм.

Поставленные задачи были выполнены в полной мере, результаты вычисления программы полностью совпали с результатами, полученными посредством стороннего инструмента.