МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

## «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №6

по дисциплине: Архитектура вычислительных систем тема: «Логические команды и команды сдвига»

# Выполнил: ст. группы ПВ-223

# Дмитриев Андрей Александрович

Проверил:

# Осипов Олег Васильевич

Белгород 2024 г.

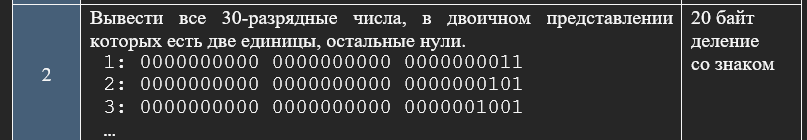
# Вариант 2

Цель работы: изучение команд поразрядной обработки данных.

1. Написать программу для вывода чисел на экран согласно варианту задания. При выполнении задания №1 все числа считать беззнаковыми. Написать и использовать функцию **output(a)** для вывода числа **a** на экран или в файл. Функция должна удовлетворять соглашению о вызовах. В функцию для вывода **output** передавать в качестве аргумента переменную размерности 32 или 64 бита, которой достаточно для хранения числа. К примеру, если в задании число указано как 15-разрядное, то аргументом функции должно быть число размером двойное слово, если 40-разрядное, то учетверённое слово. Функция должна выводить столько разрядов числа, сколько указано в задании, даже если старшие разряды равны нулю. Не допускается прямой перебор всех чисел с проверкой, удовлетворяет ли оно условию вывода (за исключением вариантов № 8, 12, 13). Числа выводить в порядке, который является удобным. Проверить количество выведенных чисел с помощью формул комбинаторики. В отчёт включить вывод формул и результаты работы программы.
2. Написать подпрограмму для умножения (**multiplication**) или деления (**division**) большого целого числа на 2*n* (в зависимости от варианта задания) с использованием команд сдвига. Подпрограммы должны иметь следующие заголовки:

**multiplication(char\* a, int n, char\* res); division(char\* a, int n, char\* res).**

Входные параметры: *a* – адрес первого числа в памяти, *n* – степень двойки. Выходные параметры: *res* – адрес массива, куда записывается результат. В случае операции умножения, для массива *res* зарезервировать в два раза больше памяти, чем для множителей *a* и *b*. Числа *a*, *b*, *res* вывести на экран в 16-ричном виде. Подобрать набор тестовых данных для проверки правильности работы подпрограммы.



# Задание №1.

.data

    mask1 dd 1b

    fmt db "%d", 0

    fntNewLine db 10,13, 0

    fmt\_number db "%d :", 9, 0

.code

output proc uses eax ebx edx ecx a\_ : dword

    mov eax, a\_

    shl eax, 12

    clc

mov ecx, 20

output\_number:

    rol eax, 1

    mov edx, eax

    and edx, 1b

    push eax

    push ecx

    push edx

    push edx

    push offset fmt

    call crt\_printf

    add esp, @WordSize \* 2

    pop edx

    pop ecx

    pop eax

    loop output\_number

    push offset fntNewLine

    call crt\_printf

    add esp, @WordSize

    ret

output endp

START:

    mov ecx, 19

    mov esi, 1

first\_loop:

    push ecx

    mov ebx, mask1

    shl ebx, cl

second\_loop:

    dec ecx

    mov edx, mask1

    shl edx, cl

    xor eax, eax

    or eax, ebx

    or eax, edx

    push ecx

    push eax

    push ebx

    push esi

    push offset fmt\_number

    call crt\_printf

    add esp, @WordSize

    pop esi

    inc esi

    pop ebx

    pop eax

    pop ecx

    invoke output, eax

    cmp ecx, 0

    jne second\_loop

    pop ecx

    loop first\_loop

    push 0

    call ExitProcess

END START

# Вывод программы:

# 

Задание №2.

.data

    fmtBit db "%d", 0        ; Формат для вывода одного бита

    fmtSpace db " ", 0       ; Формат для вывода пробела

    fmtF db "%d", 0

    fmtNewLine db 10, 13, 0

    number db 19 dup(77h), 000h

    ; number db 19 dup(0FFh), 0F8h

.code

output\_big\_number proc uses eax ebx ecx edx esi, number\_: dword

    mov esi, number\_

    ; add esi, 19

    mov ecx, 20

output\_loop:

    push ecx                  ; Сохраняем ecx

    mov ecx, 8                ; Устанавливаем счетчик для 8 бит

    mov ebx, 010000000b                ; Маска для выделения бита

bit\_loop:

    dec ecx

    mov eax, [esi]            ; Загружаем текущий байт в eax

    and eax, ebx              ; Выделяем текущий бит

    shr eax, cl

    push ecx                  ; Сохраняем ecx

    push eax                  ; Передаем бит в качестве аргумента

    push offset fmtBit        ; Передаем формат для вывода бита

    call crt\_printf           ; Вызываем функцию printf

    add esp, 8                ; Восстанавливаем стек

    pop ecx                   ; Восстанавливаем ecx

    shr ebx, 1                ; Сдвигаем маску для следующего бита

    cmp ecx, 0

    jne bit\_loop             ; Повторяем цикл для оставшихся бит

    push offset fmtSpace      ; Передаем формат для вывода пробела

    call crt\_printf           ; Вызываем функцию printf

    add esp, 4                ; Восстанавливаем стек

    pop ecx                   ; Восстанавливаем ecx

    inc esi                   ; Переходим к следующему байту

    loop output\_loop          ; Повторяем цикл для оставшихся байтов

    ; Вывод новой строки

    push offset fmtNewLine    ; Передаем формат для вывода новой строки

    call crt\_printf           ; Вызываем функцию printf

    add esp, 4                ; Восстанавливаем стек

    ret

output\_big\_number endp

div\_2 proc number\_ptr: dword, k\_ : dword

    pushad

    mov ecx, k\_

    mov ebx, number\_ptr

loop\_start:

    push ecx

    mov esi, ebx

    clc

    mov ecx, 20

loop\_shift:

    sar byte ptr [esi], 1

    pushf

    add esi, 1

    popf

    loop loop\_shift

    pop ecx

    loop loop\_start

    popad

    ret

div\_2 endp

START:

invoke output\_big\_number, offset number

invoke div\_2, offset number, 2

invoke output\_big\_number, offset number

; invoke div\_2, offset number, 1

; invoke output\_big\_number, offset number

; invoke div\_2, offset number, 10

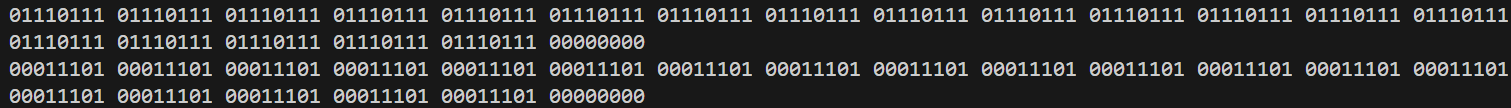
; invoke output\_big\_number, offset number

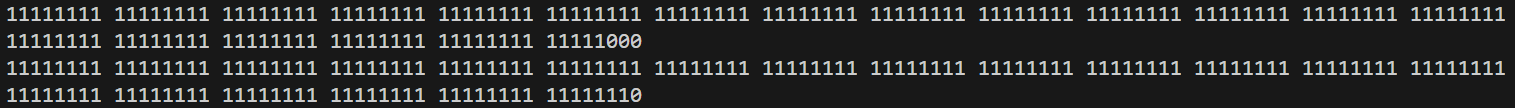
push 0

call ExitProcess

END START

Вывод программы:





**Вывод:** Входе лабораторной работы изучены команды поразрядной обработки данных.