

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

**ЛАБОРАТОРНАЯ РАБОТА №4**

**ОТЧЁТ**

студента 2 курса 251 группы  
направления 09.03.04 — Программная инженерия  
факультета КНиИТ  
Григорьева Даниила Евгеньевича

Проверено:

Старший преподаватель

\_\_\_\_\_

Е. М. Черноусова

## СОДЕРЖАНИЕ

1 Задание .....	3
2 Алгоритм .....	4
3 Исходный код .....	5
4 Скриншот запуска программы .....	10
5 Ответы на контрольные вопросы .....	11

## **1 Задание**

**Вариант 5** Массив из 20 чисел заполнить последовательностью, состоящей наполовину из чисел кратных 3 и наполовину из квадратов этих чисел; организовать вывод массива на экран в виде таблицы 2x10 с фиксированной шириной столбцов.

## **2 Алгоритм**

1. Вывод фамилии и номера группы
2. Вывод массива
  1. Вывод первой строки (обход 0-9 индексов)
    1. Вывод числа
      1. Приведение к цифре и сохранение в стеке
      2. Извлечение цифры из стека, её приведение к символу и вывод на экран
    2. Вывод второй строки (обход 10-19 индексов)
      1. Аналогично выводу первой строки
3. Завершение работы программы

### 3 Исходный код

```
.model small
.stack 100h

NUMBERS_COUNT = 10
NUMBER_SIZE   = 2
COLUMN_WIDTH  = 5

.data
    ; Фамилия и номер группы
    info db "Grigorev Danya 251$"
    ; Массив чисел
    numbers db 2 dup (NUMBER_SIZE * NUMBERS_COUNT dup (?))

.code
main:
    ; Инициализация сегмента данных
    mov ax, @data
    mov ds, ax
    ; Выводим фамилию и номер группы
    mov ah, 09h
    lea dx, info
    int 21h
    ; Переход на новую строку
    call NewLine
    ; Генерируем массив
    call GenerateArray
    ; Выводим массив чисел
    call PrintArray
    ; Завершаем программу
    mov ah, 4Ch
    int 21h

GenerateArray proc
    mov bx, 10
    lea ax, numbers
    call GenerateTwos
    lea ax, numbers + NUMBER_SIZE * NUMBERS_COUNT
    call GenerateThrees
    ret
GenerateArray endp

; Генерирует bx первых степеней двойки и кладёт их в массив с
началом в [ax]
```

```

GenerateTwos proc
    push si
    push cx
    push dx
    mov si, ax
    mov cx, bx
    mov dx, 2
    generateTwo:
        mov [si], dx
        sal dx, 1
        add si, NUMBER_SIZE
        loop generateTwo
    pop dx
    pop cx
    pop si
    ret
GenerateTwos endp

```

; Генерирует bx первых степеней тройки и кладёт их в массив с началом в [ax]

```

GenerateThrees proc
    push ax
    push bx
    push cx
    push dx
    push si
    mov si, ax
    mov cx, bx
    mov ax, 3
    mov bx, 3
    generateThree:
        mov [si], ax
        mul bx
        add si, NUMBER_SIZE
        loop generateThree
    pop si
    pop dx
    pop cx
    pop bx
    pop ax
    ret
GenerateThrees endp

```

; Переход на новую строку

NewLine proc

```
push ax
push dx
mov ah, 02h
mov dl, 0Ah
int 21h
mov dl, 0Dh
int 21h
pop dx
pop ax
ret
```

NewLine endp

; Процедура для вывода массива чисел в виде таблицы 2x5

PrintArray proc

```
push si
; Первый ряд чисел
lea si, numbers
call PrintRow
```

```
; Переход на новую строку
call NewLine
```

```
; Второй ряд чисел (числа с индексами 5-9)
lea si, numbers + NUMBER_SIZE * NUMBERS_COUNT
call PrintRow
```

```
pop si
ret
```

PrintArray endp

; Процедура для вывода одного ряда чисел

PrintRow proc

```
push ax
push cx
push dx
push si
```

```
; Цикл вывода чисел с правильными отступами
mov cx, NUMBERS_COUNT ; Количество элементов
```

PrintLoop:

```
; Загружаем число из массива
```

```
mov ax, [si]
```

```
; Выводим число с отступом
```

```
call PrintNum
```

```

        ; Переход к следующему числу
        add si, NUMBER_SIZE
        mov dx, 1
        cmp dx, cx
        jz continuePrintLoop
        ; Печатаем пробел между числами
        mov ah, 02h
        mov dl, 09h ; табуляция
        ;mov dl, '_' ; табуляция
        int 21h
        continuePrintLoop:
loop PrintLoop
pop si
pop dx
pop cx
pop ax
ret
PrintRow endp

; Процедура для вывода числа в виде строки
PrintNum proc
    push ax
    push bx
    push cx
    push dx

    xor cx, cx
    mov bx, 10
    calculation:
        xor dx, dx
        div bx
        push dx
        inc cx
        cmp ax, 0
        jne calculation

    ; Вывод ведущих пробелов
    push cx
    mov dx, COLUMN_WIDTH
    sub dx, cx
    jna start_printing
    mov cx, dx
    spacing:
        mov ah, 02h

```



```

        mov dl, ' '
        int 21h
        loop spacing

; Вывод непосредственно числа
start_printing:
pop cx
printing:
    pop bx
    call PrintSingleDigit
loop printing

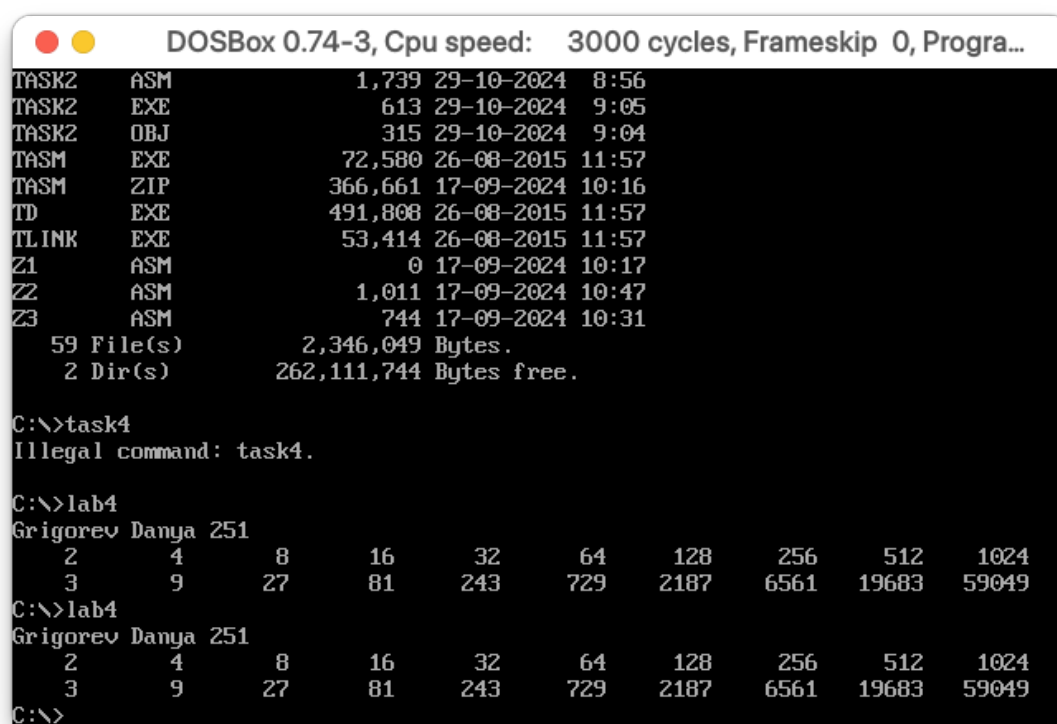
pop dx
pop cx
pop bx
pop ax
ret

PrintSingleDigit proc
    push ax
    push bx
    push dx
    add bl, '0'          ; Преобразуем в символ
    mov dl, bl
    mov ah, 02h
    int 21h
    pop dx
    pop bx
    pop ax
    ret
PrintNum endp

end main

```

#### 4 Скриншот запуска программы



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

TASK2	ASM	1,739	29-10-2024	8:56
TASK2	EXE	613	29-10-2024	9:05
TASK2	OBJ	315	29-10-2024	9:04
TASM	EXE	72,580	26-08-2015	11:57
TASM	ZIP	366,661	17-09-2024	10:16
TD	EXE	491,808	26-08-2015	11:57
TLINK	EXE	53,414	26-08-2015	11:57
Z1	ASM	0	17-09-2024	10:17
Z2	ASM	1,011	17-09-2024	10:47
Z3	ASM	744	17-09-2024	10:31

59 File(s) 2,346,049 Bytes.  
2 Dir(s) 262,111,744 Bytes free.

C:\>task4  
Illegal command: task4.

C:\>lab4  
Grigorev Danya 251

2	4	8	16	32	64	128	256	512	1024
3	9	27	81	243	729	2187	6561	19683	59049

C:\>lab4  
Grigorev Danya 251

2	4	8	16	32	64	128	256	512	1024
3	9	27	81	243	729	2187	6561	19683	59049

C:\>

## 5 Ответы на контрольные вопросы

**Вопрос 1** Какой командой можно выделить в памяти место под одномерный массив байтов array размерностью 20?

Для выделения памяти существуют директивы db, dw, dd, dq, dt, выделяющие, в зависимости от конкретной директивы, байт, слово, двойное слово... К ним применим оператор dup, позволяющий выделить несколько экземпляров:

Array DB 20 DUP (?)

(?) – ячейки выделенной памяти не будут инициализироваться конкретными значениями.

В результате выполнения вышеприведённой директивы будет выделено 20 байт памяти с начальным адресом под меткой .

Перечислив значения через запятую, можно проинициализировать конкретные байты, слова, ...

**Вопрос 2** Опишите команды умножения на байт и на слово.

Для умножения значения регистра AL или AX (в зависимости от размерности второго множителя) на число используется инструкция mul или imul (в зависимости от знаковости, mul беззнаковый): mul <второй множитель>

В зависимости от разрядности результат (частное и остаток соответственно) помещаются либо в AX, либо в DX:AX.

**Вопрос 3** Какое максимальное беззнаковое число можно хранить в элементе массива размером в 1 байт?

Число 255.

**Вопрос 4** Пусть имеется массив: array DW 50 DUP(?). Для доступа к отдельным элементам массива используется адресное выражение array[SI]. Как называется этот способ адресации и как с его помощью будет вычисляться адрес элементов массива?

Прямая адресация с индексированием. array определяет начало массива, а значение в SI — индекс элемента, прибавляющийся к адресу.

**Вопрос 5** Каким образом осуществляется перебор элементов некоторого массива *A* с помощью адресного выражения *A[SI]*, если массив состоит из байтов, слов или двойных слов?

Увеличивая значение *SI* в соответствии с размерностью элементов массива, эквивалентно *i += sizeof a[0]* в Си.

**Вопрос 6** Для некоторого массива *A* каким будет результат выполнения команды *mov DI, A* и команды *mov DI, offset A*?

В первом случае в *DI* будет помещено значение элемента массива *A*, а во втором адрес элемента *A*. *mov DI, A* работает со значением по адресу *A*; *mov DI, offset A* работает непосредственно с адресом.