

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

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

ЛАБОРАТОРНАЯ РАБОТА №4

Отчёт о практике

студента 2 курса 251 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Тюменцева Радомира Александровича

Проверено:

Старший преподаватель

Е. М. Черноусова

Саратов 2025

1 Задание 4

Вариант 1. Массив из 20 чисел заполнить последовательностью, состоящей наполовину из чётных чисел (2, 4, ...) и наполовину из квадратов этих чисел; организовать вывод массива на экран в виде таблицы 2x10 с фиксированной шириной столбцов:

2	4	6	8	10	12	14	16	18	20
4	16	36	64	100	144	196	256	324	400

Рис. 1 – Вывод в виде таблицы 2x10 массива из 20 чисел.

1.1 Краткий словестный алгоритм программы

1. Вывод фамилии, имени и номера группы.
2. Заполнение массива `simple` через вложенный цикл:
 1. В первой итерации цикла `fill_rows` в цикле `fill_cols` заполняются первые 10 элементов чётными числами.
 2. Во второй итерации цикла `fill_rows` в цикле `fill_cols` заполняются следующие 10 элементов их квадратами.
3. Вывод массива `simple` в две строки через вложенный цикл с переводом чисел в строки ASCII фиксированной длины.
4. Завершение работы программы.

1.2 Текст программы на языке ассемблера с комментариями

```
1 .model small
2 .stack 100h
3 .186
4
5 .data
6 my_name db "Tyumentsev Radomir, 251$"
7 simple dw 20 dup (?) ; Массив из 20 слов (неинициализированный)
8 result db 5 dup (' '), '$' ; Буфер для вывода одного числа: 5 символов +
   '$'
9 nl      db 0Ah, 0Dh, '$' ; Символы перехода на новую строку
10
11 .code
12 start:
13 mov AX, @data
14 mov DS, AX
15
16 ; Вывод фамилии, имени и номера группы
17 lea DX, my_name
18 mov AH, 09h
19 int 21h
```

```

20
21  call print_new_line
22
23  mov CX, 2; Количество строк
24  mov SI, 0; Смещение в байтах (0)
25
26  ; Заполнение массива
27  ; Первые 10 элементов - чётные числа, следующие 10 - их квадраты
28  fill_rows:
29      mov BX, 2; Первое число
30
31      fill_cols:
32          mov AX, BX
33          cmp CX, 2
34          je even
35          mul AX; AX = AX * AX (возведение в квадрат)
36          even:
37          mov simple[SI], AX; Занесение числа в массив
38          add SI, 2
39          add BX, 2
40          cmp SI, 20
41          je fill_rows_loop
42          cmp SI, 40
43          je fill_rows_loop
44          jl fill_cols
45          fill_rows_loop:
46              loop fill_rows
47
48  mov CX, 2; Количество строк
49  mov SI, 0; Смещение в байтах (0)
50
51  ; Вывод массива в две строки
52  print_rows:
53      print_cols:
54          mov AX, simple[SI]; Занесение числа из массива в AX
55          call word_asc; Перевод числа в строку ASCII
56          lea DX, result
57          call print_string; Вывод строки
58          add SI, 2
59          cmp SI, 20
60          je print_rows_loop
61          cmp SI, 40
62          je print_rows_end
63          jl print_cols
64          print_rows_loop:
65              call print_new_line
66              loop print_rows
67          print_rows_end:
68
69  ; завершение программы

```

```

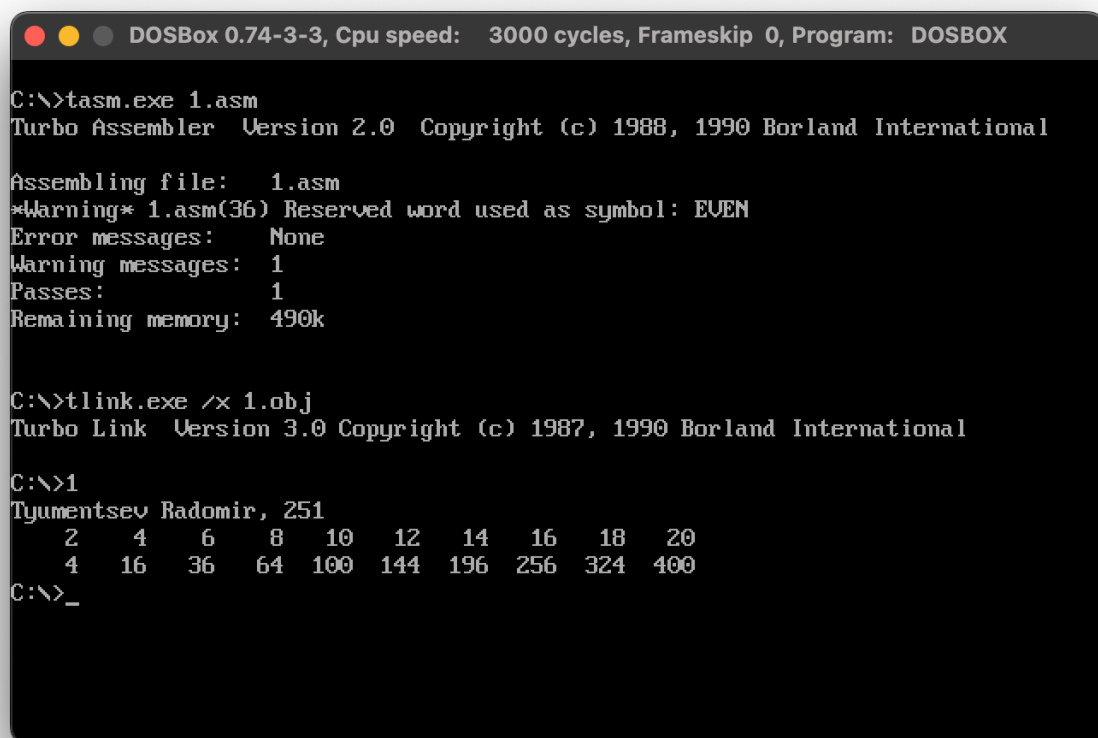
70  mov AX, 4C00h
71  int 21h
72
73  ; Процедура перевода числа в строку ASCII фиксированной длины
74  word_asc proc
75      pusha
76      mov BX, 10; Основание системы счисления
77      mov SI, 0; Смещение в байтах, изначально 0, затем увеличивается до 5
78      mov CX, 5; Длина строки result (5 символов)
79
80      ; Заполнение буфера result пробелами для его очистки
81      fill_spaces:
82          mov result[SI], ' '
83          inc SI
84          loop fill_spaces
85
86      ; SI = 5
87      ; Заполнение буфера result символами
88      convert_loop:
89          dec SI
90          mov DX, 0; Обнуление прошлого остатка от деления
91          div BX; AX = частное, DX = остаток
92          add DL, '0' ; Добавление кода символа 0 в ASCII
93          mov result[SI], DL; Занесение символа в буфер
94          cmp AX, 0
95          jne convert_loop
96
97      popa
98      ret
99  word_asc endp
100
101  ; Процедура вывода строки, хранящейся в DX
102  print_string proc
103      push AX
104
105      mov AH, 09h
106      int 21h
107
108      pop AX
109      ret
110  print_string endp
111
112  ; Процедура переноса строки
113  print_new_line proc
114      push AX
115      push DX
116
117      lea DX, nl
118      mov AH, 09h
119      int 21h

```

```
120
121     pop DX
122     pop AX
123     ret
124 print_new_line endp
125
126 end start
```

Текст программы

1.3 Скриншоты запуска программ



The screenshot shows a DOSBox window with the following text:

```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

C:\>tasm.exe 1.asm
Turbo Assembler Version 2.0 Copyright (c) 1988, 1990 Borland International

Assembling file: 1.asm
*Warning* 1.asm(36) Reserved word used as symbol: EVEN
Error messages: None
Warning messages: 1
Passes: 1
Remaining memory: 490k

C:\>tlink.exe /x 1.obj
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International

C:\>1
Tyumentsev Radomir, 251
  2   4   6   8  10  12  14  16  18  20
  4  16  36  64 100 144 196 256 324 400
C:\>_
```

Запуск программы

2 Ответы на контрольные вопросы

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, и так далее. К ним применим оператор `dup`, позволяющий выделить несколько экземпляров:

```
array db 20 dup (?)
```

При использовании `(?)` ячейки выделенной памяти не будут инициализироваться конкретными значениями.

В результате выполнения приведённой в примере директивы будет выделено 20 байт памяти с начальным адресом под меткой `array`.

Перечислив значения через запятую, можно проинициализировать конкретные байты, слова, и так далее, соответственно.

2. Опишите команды умножения на байт и на слово.

Для умножения значения регистра `AL` или `AX` на число используется инструкция `mul` для беззнаковых чисел или `imul` для знаковых:

```
1 mul <второй множитель>
2 mul BL; AX = AL * BL
3 mul BX; DX:AX = AX * BX
```

Команда умножения может устанавливать только флаги переноса и переполнения.

Команда `mul` устанавливает оба флага, если старшая половина результата не нулевая. Если умножаются два байта, установка флагов переполнения и переноса показывает, что результат умножения больше 255 и не может содержаться в одном байте. В случае умножения слов флаги устанавливаются, если результат больше 65535.

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

Число $255 = 2^8 - 1$.

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 адрес начала массива, а значение в `SI` - смещение (индекс элемента * размер данных).

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

Задаётся `CX`, равный количеству элементов массива, а затем в цикле постепенно увеличивается `SI` на размер элемента массива (байт, слово или двойное слово, соответственно), и происходит постепенное смещение по элементам.

```
1 mov CX, 5; Если в массиве 5 элементов
2 mov SI, 0
3 array_loop:
4     mov AX, A[SI]
5     add SI, 2; Для слова
6     loop array_loop
```

6. Для некоторого массива `array`, объявленного следующим образом: `array DW 20 DUP(?)`, каким будет результат выполнения команды `mov DI, array` и команды `mov DI, offset array`?

В первом случае в `DI` будет помещено значение первого элемента массива `array`, а во втором - адрес начала массива `array`.