

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

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

ЛАБОРАТОРНАЯ РАБОТА №3

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 3

Сначала программы должны печатать фамилию, имя и номер группы студента и переходить на новую строку. Используя рассмотренное упражнение, выполните следующие задания:

1.1 Задание 3.1

В регистре AX задано число от 0 до 65535. Выведите это число на экран. (Проверить программу для числа более 2600.)

1.2 Задание 3.2

Используя 32-битные регистры процессора (EAX, EBX, EDX), напишите программу, выводящую на экран число 65536. Число 65536 изначально поместить в регистр EAX.

1.3 Тексты программ на языке ассемблера с комментариями

```
1  .model small
2  .stack 100h
3
4  .data
5  my_name db 'Tyumentsev Radomir, 251', 0Dh, 0Ah, '$'
6
7  .386 ; Разрешение трансляции команд процессора 386
8  .code
9
10 start:
11  mov AX, @data; Помещение указателя на сегмент данных в AX
12  mov DS, AX; Помещение указателя на сегмент данных в DS
13
14  ; Вывод фамилии, имени и номера группы
15  mov DX, offset my_name
16  mov AH, 09h
17  int 21h
18
19  mov AX, 3456; Занесение числа
20  mov BX, 10; Занесение основания системы счисления (делителя)
21  mov CX, 0; Обнуление счётчика
22
23  loop_first: ; Занесение в стек цифр числа
24      inc CX; Увеличение счётчика
25      mov DX, 0; Обнуление остатка от деления в DX
26      div BX; Деление AX на BX
27      push DX; Занесение остатка от деления в стек
28      cmp AX, 0; Сравнение частного с нулём
29      jne loop_first; Если AX != 0, то возвращаемся к loop_first
30
```

```

31  mov AH, 02h ; Занесение в AH кода команды вывода символа
32
33  loop_second: ; Вывод цифр числа из стека
34      pop DX
35      call out_digit
36      loop loop_second
37
38  ; Завершение программы
39  mov AX, 4C00h
40  int 21h
41
42  ; Процедура вывода цифры
43  out_digit proc
44      add DX, 30h; Перевод цифры в ASCII
45      int 21h
46      ret
47  out_digit endp
48
49 end start

```

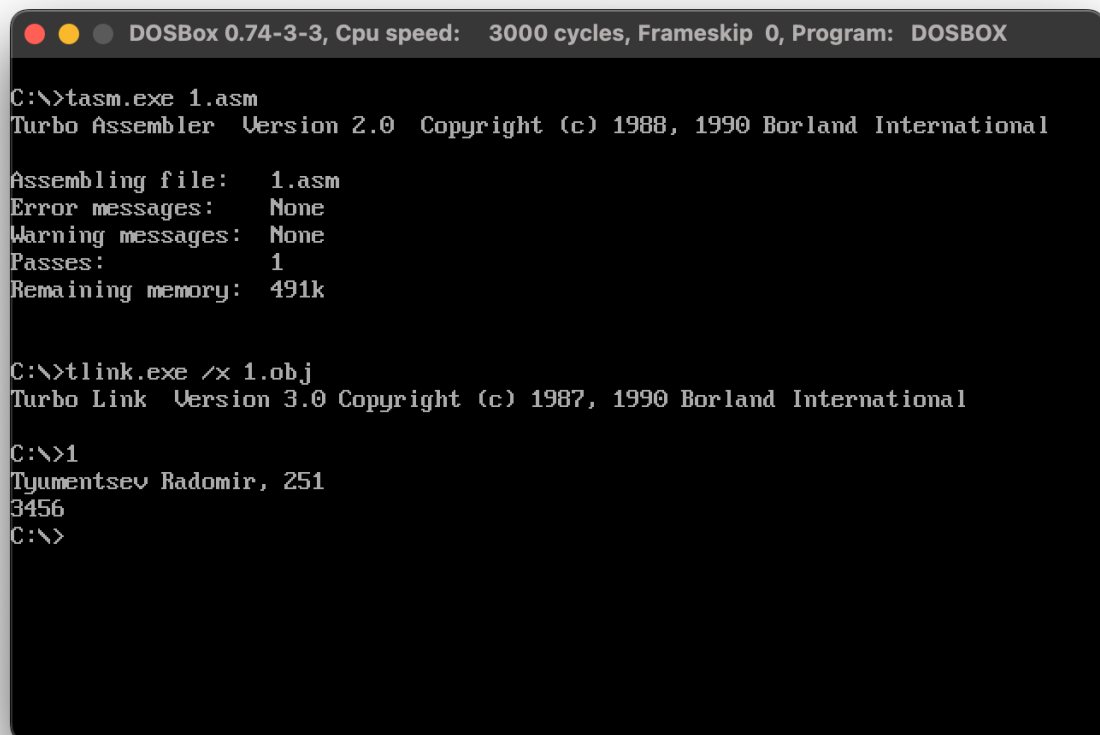
Текст программы 1

```

1  .model small
2  .stack 100h
3
4  .data
5  my_name db 'Tyumentsev Radomir, 251', 0Dh, 0Ah, '$'
6
7  .386 ; Разрешение трансляции команд процессора 386
8  .code
9
10 start:
11  mov AX, @data; Помещение указателя на сегмент данных в AX
12  mov DS, AX; Помещение указателя на сегмент данных в DS
13
14  ; Вывод фамилии, имени и номера группы
15  mov DX, offset my_name
16  mov AH, 09h
17  int 21h
18
19  mov EAX, 65536; Занесение числа
20  mov EBX, 10; Занесение основания системы счисления (делителя)
21  mov CX, 0; Обнуление счётчика
22
23  loop_first: ; Занесение в стек цифр числа
24      inc CX; Увеличение счётчика
25      mov EDX, 0; Обнуление остатка от деления в DX
26      div EBX; Деление EAX на EBX
27      push EDX; Занесение остатка от деления в стек
28      cmp EAX, 0; Сравнение частного с нулём
29      jne loop_first; Если EAX != 0, то возвращаемся к loop_first
30
31  mov AH, 02h ; Занесение в AH кода команды вывода символа
32
33  loop_second: ; Вывод цифр числа из стека
34      pop EDX
35      call out_digit
36      loop loop_second
37
38  ; Завершение программы
39  mov AX, 4C00h
40  int 21h
41
42  ; Процедура вывода цифры
43  out_digit proc
44      add EDX, 30h; Перевод цифры в ASCII
45      int 21h
46      ret
47  out_digit endp
48
49 end start

```

1.4 Скриншоты запуска программ




The screenshot shows a DOSBox window with a dark background and white text. The title bar at the top reads "DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX". The terminal content shows the following commands and output:

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

Assembling file: 1.asm
Error messages:  None
Warning messages: None
Passes:         1
Remaining memory: 491k

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

C:\>1
Tyumentsev Radomir, 251
3456
C:\>
```



The image shows a DOSBox terminal window with a dark background and light-colored text. The window title bar at the top reads "DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX". The terminal content shows the following sequence of commands and output:

```
C:\>tasm.exe 2.asm
Turbo Assembler Version 2.0 Copyright (c) 1988, 1990 Borland International

Assembling file: 2.asm
Error messages:  None
Warning messages: None
Passes:         1
Remaining memory: 491k

C:\>tlink.exe /x 2.obj
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International

C:\>2
Tyumentsev Radomir, 251
65536
C:\>
```

Запуск программы 2

2 Ответы на контрольные вопросы

1. Чем отличается деление на байт от деления на слово? (где должно располагаться делимое, куда попадут частное от деления и остаток от деления)

Деление на байт:

- Делимое должно располагаться в регистре AX (для 16-битных операций) или AL (для 8-битных операций).
- Частное от деления помещается в AL, а остаток — в AH.

Деление на слово:

- Делимое должно располагаться в регистре DX:AX (для 32-битных операций) или в AX (для 16-битных операций).
- Частное помещается в AX, а остаток — в DX.

2. Каков механизм действия команды cmp? В паре с какими командами она обычно используется?

Команда CMP выполняет сравнение двух операндов путем вычитания одного из другого. Результаты вычитания не сохраняются, но устанавливаются соответствующие флаги в регистре EFLAGS:

- ZF устанавливается, если операнды равны.
- CF устанавливается, если первый операнд меньше второго (для беззнаковых чисел).
- SF устанавливается, если результат отрицательный (для знаковых чисел).
- OF устанавливается при переполнении знакового результата.

Часто команда CMP используется перед командами условного перехода (JE, JNE, JG, JL и т.д.), которые принимают решения на основе установленных флагов.

3. На какие флаги реагируют команды условного перехода для чисел со знаком и для чисел без знака?

Для чисел со знаком:

- JG (Jump if Greater): SF = OF и ZF = 0
- JL (Jump if Less): SF ≠ OF
- JE или JZ (Jump if Equal): ZF = 1

Для чисел без знака:

- JA (Jump if Above): CF = 0 и ZF = 0
- JB (Jump if Below): CF = 1
- JE или JZ: ZF = 1

4. С помощью команд условного и безусловного перехода выполните программную реализацию алгоритма ветвления для определения наименьшего числа из двух заданных. Алгоритм изображен в виде блок-схемы, приведенной на Рис. 1.

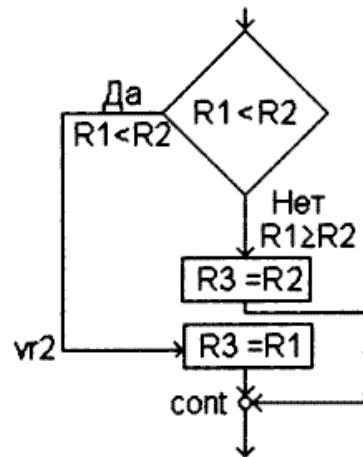


Рис. 1 – Организация ветвления на машинном уровне:

R1 - первое число хранится в регистре AX;

R2 - второе число хранится в регистре BX;

R3 - результат заносится в регистр DX;

vr2, cont - метки команд.

```

1  .model small
2  .stack 100h
3
4  .code
5  start:
6      ; Задание значений R1 и R2
7      mov AX, 5; R1
8      mov BX, 3; R2
9
10     cmp AX, BX; Сравнение R1 с R2
11     jl vr2; Если R1 < R2, переход к метке vr2
12
13     ; Если R1 >= R2
14     mov DX, BX; Сохранение R2 как результат
15     jmp cont; Переход к завершению
16
17     vr2:
18         mov DX, AX; Сохранение R1 как результат
19
20     cont: ; Завершение программы
21         mov AX, 4C00h
22         int 21h
23
24     end start
  
```


5. Каков механизм работы команды организации цикла LOOP?

Команда LOOP уменьшает значение регистра CX на единицу и выполняет переход по указанной метке, если CX не равен нулю. Это позволяет организовать циклы с фиксированным количеством итераций. Пример:

```
1 mov CX, 5; Установка количество итераций
2 loop_start:
3     ; Ваш код здесь
4     loop loop_start; Переход к loop_start, пока CX не станет равным нулю.
```

6. Как с помощью команды сдвига можно умножить знаковое число, хранящееся в AX, на 2 в n-ой степени?

Чтобы умножить число в регистре AX на 2^n , можно использовать команду сдвига влево (SHL). Например:

```
1 mov AX, 3; Пример для числа 3
2 shl AX, 1; Умножаем на 2 (3 * 2 = 6)
3 shl AX, 2; Умножаем на 4 (6 * 4 = 24)
```

Каждый сдвиг влево увеличивает степень двойки.

7. Как с помощью команды сдвига проверить содержимое регистра BX на четность?

Чтобы проверить четность числа в регистре BX, можно использовать команду сдвига вправо (SHR) и проверять младший бит:

```
1 mov BX, some_value; Значение для проверки четности
2 shr BX, 1; Сдвигаем вправо на один бит
3 jnc is_even; Если нет переноса, число четное
4
5 is_odd:
6 ; Код для обработки нечетного числа
7
8 is_even:
9 ; Код для обработки четного числа
```

Если после сдвига младший бит равен нулю (нет переноса), то число четное.