

# **Отчёт по лабораторной работе 6**

**Архитектура компьютеров**

Сухоруков Сергей Андреевич

# **Содержание**

<b>1 Цель работы</b>	<b>5</b>
<b>2 Выполнение лабораторной работы</b>	<b>6</b>
2.1 Символьные и численные данные в NASM . . . . .	6
2.2 Выполнение арифметических операций в NASM . . . . .	12
2.3 Задание для самостоятельной работы . . . . .	17
<b>3 Выводы</b>	<b>20</b>

# Список иллюстраций

2.1 Код программы lab6-1.asm . . . . .	7
2.2 Запуск программы lab6-1.asm . . . . .	7
2.3 Код программы lab6-1.asm с числами . . . . .	8
2.4 Запуск программы lab6-1.asm с числами . . . . .	9
2.5 Код программы lab6-2.asm . . . . .	10
2.6 Запуск программы lab6-2.asm . . . . .	10
2.7 Код программы lab6-2.asm с числами . . . . .	11
2.8 Запуск программы lab6-2.asm с числами . . . . .	12
2.9 Запуск программы lab6-2.asm без переноса строки . . . . .	12
2.10 Код программы lab6-3.asm . . . . .	13
2.11 Запуск программы lab6-3.asm . . . . .	13
2.12 Код программы lab6-3.asm с новым выражением . . . . .	14
2.13 Запуск программы lab6-3.asm с новым выражением . . . . .	14
2.14 Код программы variant.asm . . . . .	15
2.15 Запуск программы variant.asm . . . . .	16
2.16 Код программы task.asm . . . . .	18
2.17 Запуск программы task.asm . . . . .	19

# **Список таблиц**

# **1 Цель работы**

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## **2 Выполнение лабораторной работы**

### **2.1 Символьные и численные данные в NASM**

Создаю папку для программ лабораторной работы № 6, перехожу в неё и создаю файл lab6-1.asm.

Рассмотрим примеры программ, которые выводят символы и числа. Программы будут выводить значения из регистра eax.

В этой программе в регистр eax записывается символ „6“ (инструкция mov eax,,„6“), а в регистр ebx – символ „4“ (инструкция mov ebx,,„4“). Затем к значению в eax добавляется значение из ebx (инструкция add eax,ebx), и результат записывается обратно в eax. После этого выводим результат.

The screenshot shows a window titled "lab06-1.asm" with a menu bar including File, Edit, Selection, View, Go, Projects, LSP Client, Sessions, and a Help icon. Below the menu is a toolbar with New, Open, Save, Save As, and Undo buttons. The main area displays assembly code with line numbers on the left:

```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7     mov eax, '6'
8     mov ebx, '4'
9     add eax, ebx
10    mov [buf1], eax
11    mov eax, buf1
12    call sprintLF
13    call quit
14
```

Рисунок 2.1: Код программы lab6-1.asm

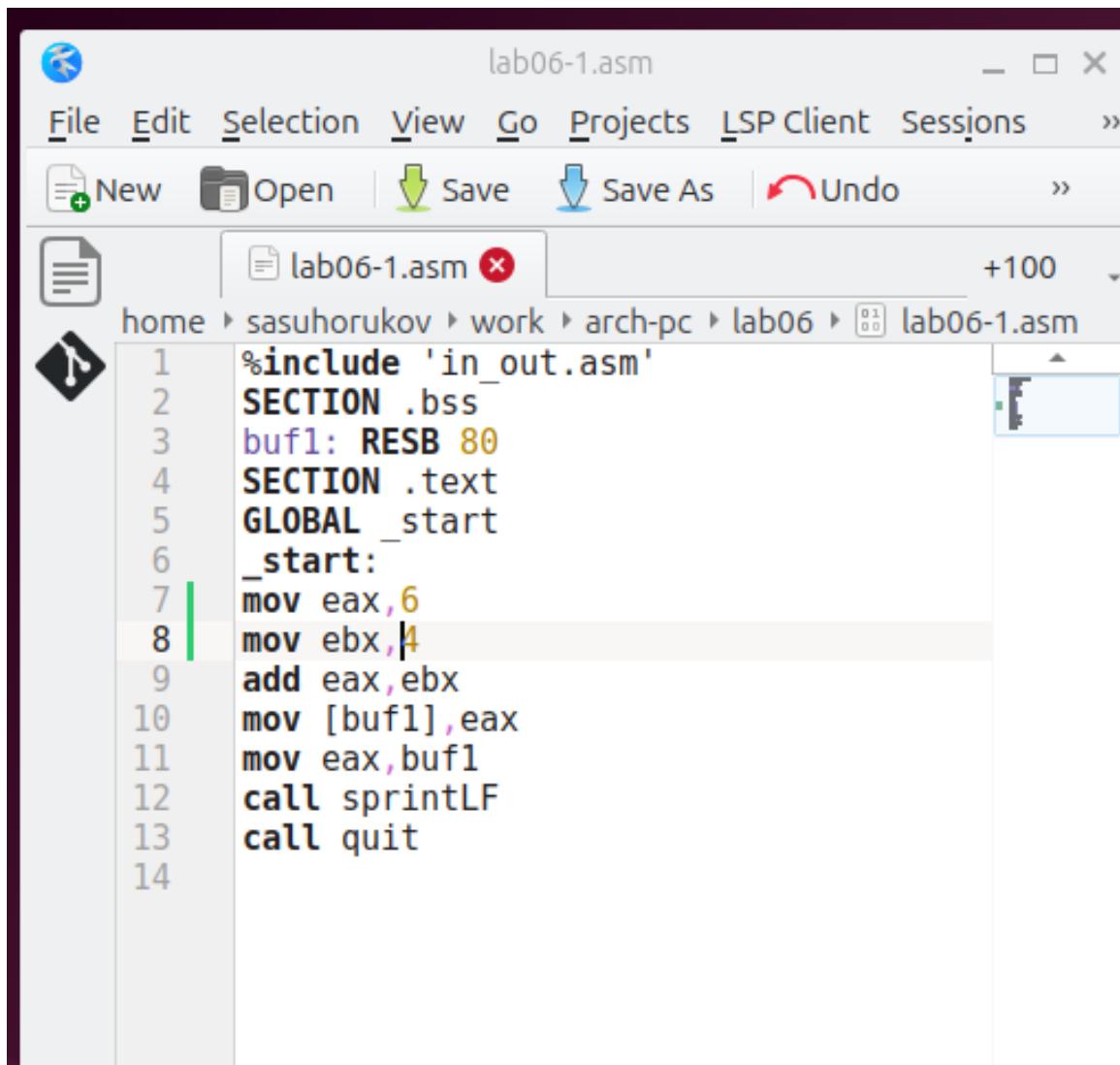
```
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ./lab06-1
j
sasuhorukov@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.2: Запуск программы lab6-1.asm

При выводе значения из eax ожидаем увидеть число 10. Однако вместо это-

го выводится символ „j“. Это связано с тем, что код символа „6“ в двоичном формате – 00110110 (54 в десятичной системе), а код символа „4“ – 00110100 (52). После сложения в eax получаем 01101010 (106), что соответствует символу „j“.

Теперь изменим программу и вместо символов запишем в регистры числа.



The screenshot shows a Microsoft Notepad window titled "lab06-1.asm". The menu bar includes File, Edit, Selection, View, Go, Projects, LSP Client, Sessions, and a Help icon. Below the menu is a toolbar with New, Open, Save, Save As, Undo, and a Help button. The status bar shows "+100". The file path is displayed as "home > sasuhorukov > work > arch-pc > lab06 > lab06-1.asm". The code editor contains the following assembly code:

```
%include 'in_out.asm'  
SECTION .bss  
buf1: RESB 80  
SECTION .text  
GLOBAL _start  
  
_start:  
    mov eax, 6  
    mov ebx, 4  
    add eax, ebx  
    mov [buf1], eax  
    mov eax, buf1  
    call sprintLF  
    call quit
```

Рисунок 2.3: Код программы lab6-1.asm с числами

```
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ./lab06-1

sasuhorukov@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.4: Запуск программы lab6-1.asm с числами

Как и в предыдущем случае, при выполнении программы не получаем число 10. В этот раз выводится символ с кодом 10, что означает конец строки (возврат каретки). В консоли он не отображается, но добавляет пустую строку.

Для работы с числами в файле in\_out.asm есть подпрограммы, которые преобразуют символы ASCII в числа и обратно. Изменяем программу, используя эти функции.

The screenshot shows a window titled "lab06-2.asm" with a menu bar including File, Edit, Selection, View, Go, Projects, LSP Client, Sessions, and a toolbar with New, Open, Save, Save As, Undo, and other icons. Below the toolbar is a file list showing "lab06-1.asm" and "lab06-2.asm" with a status "+100". The main area displays assembly code:

```
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
    mov eax, '6'  
    mov ebx, '4'  
    add eax, ebx  
    call iprintLF  
    call quit
```

Рисунок 2.5: Код программы lab6-2.asm

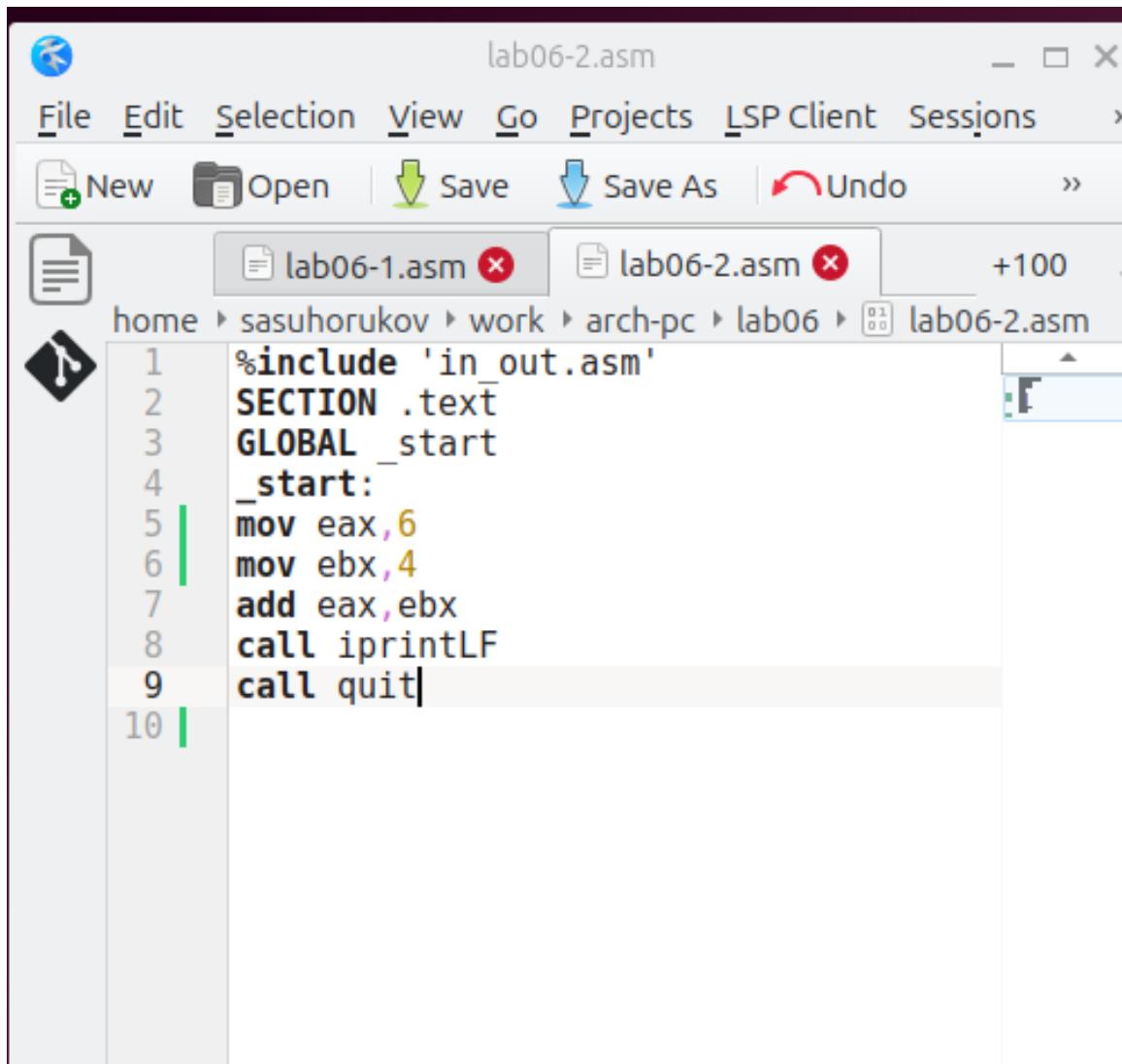
```
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm  
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2  
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ./lab06-2  
106  
sasuhorukov@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.6: Запуск программы lab6-2.asm

В результате выполнения программы мы получим число 106. Здесь команда add складывает коды символов „6“ и „4“ ( $54 + 52 = 106$ ). Но, в отличие от

предыдущей программы, функция iprintLF выводит число, а не соответствующий ему символ.

Теперь снова изменим символы на числа.



The screenshot shows a window titled "lab06-2.asm" with a menu bar including File, Edit, Selection, View, Go, Projects, LSP Client, Sessions, and a toolbar with New, Open, Save, Save As, and Undo buttons. Below the toolbar is a tab bar with "lab06-1.asm" and "lab06-2.asm" (the current file). The main area displays assembly code:

```
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
.start:  
    mov eax, 6  
    mov ebx, 4  
    add eax, ebx  
    call iprintLF  
    call quit
```

Рисунок 2.7: Код программы lab6-2.asm с числами

Функция iprintLF позволяет вывести число, так как operandами являются числа. Поэтому получаем число 10.

```
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ./lab06-2
10
sasuhorukov@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.8: Запуск программы lab6-2.asm с числами

Заменил функцию iprintLF на iprint. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки.

```
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ./lab06-2
10sasuhorukov@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.9: Запуск программы lab6-2.asm без переноса строки

## 2.2 Выполнение арифметических операций в NASM

Для примера выполнения арифметических операций в NASM рассмотрим программу, вычисляющую выражение  $f(x) = (5 * 2 + 3)/3$ .

The screenshot shows a code editor window with the file `lab6-3.asm` open. The code is written in NASM assembly language. It includes an include directive for `in_out.asm`, defines sections for data and text, and contains labels for division results. The assembly instructions perform calculations on registers `eax` and `ebx`, and use system calls for printing results.

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:

    mov eax,5
    mov ebx,2
    mul ebx
    add eax,3
    xor edx,edx
    mov ebx,3
    div ebx
    mov edi,eax
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF
    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF
    call quit
```

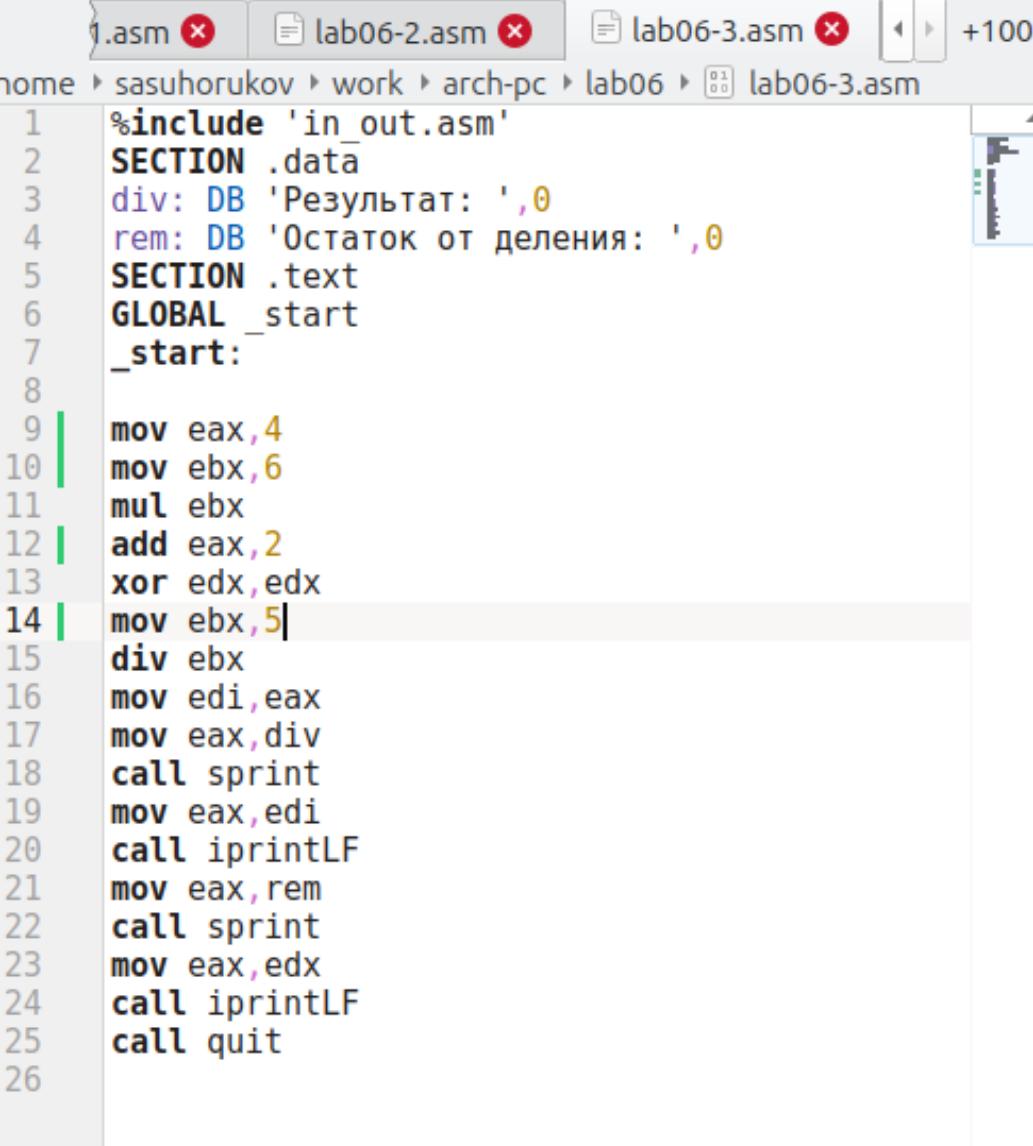
Рисунок 2.10: Код программы lab6-3.asm

```
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
sasuhorukov@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.11: Запуск программы lab6-3.asm

Изменяю программу для вычисления выражения  $f(x) = (4 * 6 + 2) / 5$ . Создаю

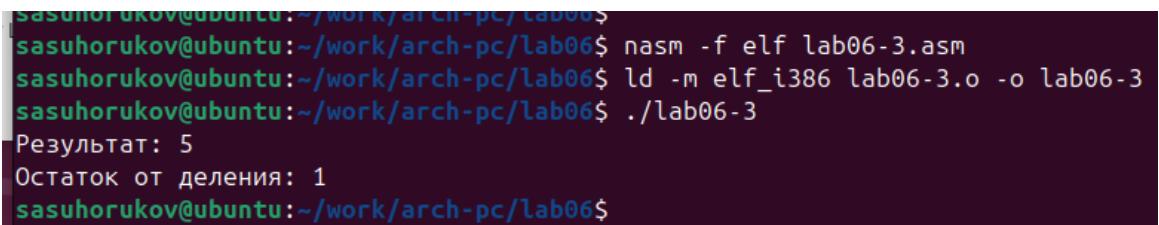
исполняемый файл и проверяю его работу.



The screenshot shows a code editor window with several tabs at the top: 1.asm (closed), lab06-2.asm (closed), lab06-3.asm (active), and +100. The file path below the tabs is: home > sasuhorukov > work > arch-pc > lab06 > lab06-3.asm. The code in the editor is as follows:

```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8
9 mov eax,4
10 mov ebx,6
11 mul ebx
12 add eax,2
13 xor edx,edx
14 mov ebx,5|           ; Line 14 with cursor
15 div ebx
16 mov edi,eax
17 mov eax,div
18 call sprint
19 mov eax,edi
20 call iprintLF
21 mov eax,rem
22 call sprint
23 mov eax,edx
24 call iprintLF
25 call quit
26
```

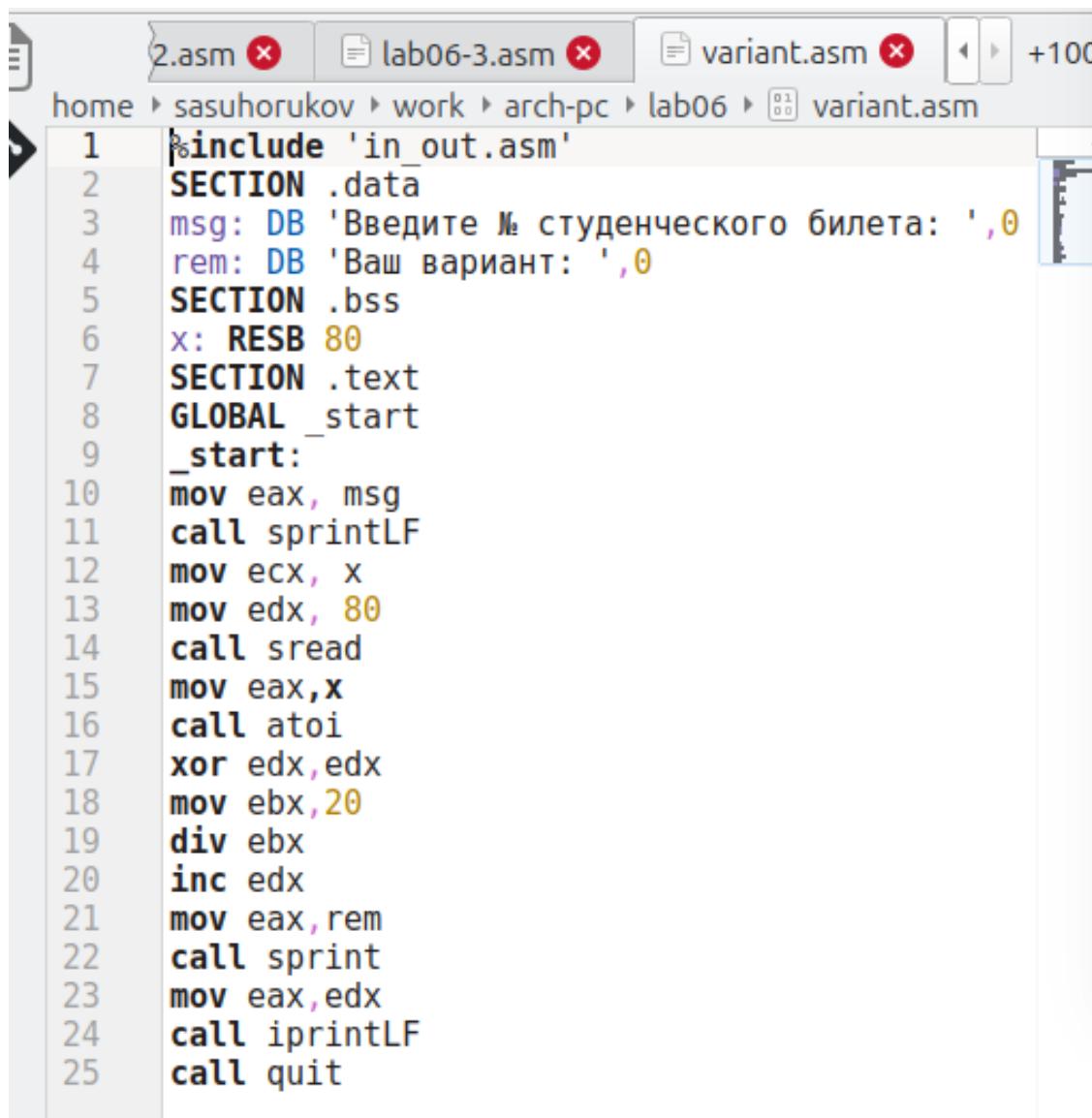
Рисунок 2.12: Код программы lab6-3.asm с новым выражением



```
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ./lab06-3
Результат: 5
Остаток от деления: 1
sasuhorukov@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.13: Запуск программы lab6-3.asm с новым выражением

Рассмотрим ещё одну программу, вычисляющую вариант задания по номеру студенческого билета.



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № студенческого билета: ',0
4 rem: DB 'Ваш вариант: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintLF
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax,x
16 call atoi
17 xor edx,edx
18 mov ebx,20
19 div ebx
20 inc edx
21 mov eax,rem
22 call sprint
23 mov eax,edx
24 call iprintLF
25 call quit
```

Рисунок 2.14: Код программы variant.asm

```
sasuhorukov@ubuntu:~/work/arch-pc/Lab06$ nasm -f elf variant.asm
sasuhorukov@ubuntu:~/work/arch-pc/Lab06$ ld -m elf_i386 variant.o -o variant
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032242470
Ваш вариант: 11
sasuhorukov@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.15: Запуск программы variant.asm

Здесь число, над которым нужно выполнять арифметические операции, вводится с клавиатуры. Поскольку ввод осуществляется в символьном виде, символы нужно преобразовать в числа. Для этого можно использовать функцию atoi из файла in\_out.asm.

### 2.2.1 Ответы на вопросы

1. Какие строки отвечают за вывод сообщения „Ваш вариант:“?
  - Инструкция mov eax, rem загружает значение переменной с фразой „Ваш вариант:“ в регистр eax.
  - Инструкция call sprint вызывает подпрограмму для вывода строки.
2. Для чего нужны следующие инструкции?
  - Инструкция mov esx, x перемещает значение переменной x в регистр esx.
  - Инструкция mov edx, 80 перемещает значение 80 в регистр edx.
  - Инструкция call sread вызывает подпрограмму для считывания номера студенческого билета из консоли.
3. Для чего нужна инструкция call atoi?
  - Инструкция call atoi используется для преобразования введенных символов в числовой формат.
4. Какие строки отвечают за вычисления варианта?

- Инструкция xor edx, edx обнуляет регистр edx.
  - Инструкция mov ebx, 20 загружает значение 20 в регистр ebx.
  - Инструкция div ebx делит номер студенческого билета на 20.
  - Инструкция inc edx увеличивает значение регистра edx на 1. Здесь происходит деление номера студенческого билета на 20, а в регистре edx хранится остаток, к которому прибавляется 1.
5. В какой регистр записывается остаток от деления при выполнении div ebx?
- Остаток от деления записывается в регистр edx.
6. Для чего нужна инструкция inc edx?
- Инструкция inc edx увеличивает значение в регистре edx на 1, как это предусмотрено формулой для вычисления варианта.
7. Какие строки отвечают за вывод результата вычислений на экран?
- Инструкция mov eax, edx помещает результат вычислений в регистр eax.
  - Инструкция call iprintLF вызывает подпрограмму для вывода значения на экран.

## 2.3 Задание для самостоятельной работы

Напишите программу для вычисления выражения  $y = f(x)$ . Программа должна выводить формулу, запрашивать ввод значения x, вычислять выражение в зависимости от введенного x и выводить результат. Форму функции  $f(x)$  выберите из таблицы 6.3 вариантов заданий в соответствии с номером, полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его для значений x1 и x2 из 6.3. Получили вариант  $11 - 10(x + 1) - 10$  для  $x = 1, x = 7$ .

The screenshot shows a window titled "task.asm" with the following assembly code:

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите X ',0
rem: DB 'выражение = : ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax, msg
    call sprintLF
    mov ecx, x
    mov edx, 80
    call sread
    mov eax,x
    call atoi
    add eax,1
    mov ebx,10
    mul ebx
    sub eax,10
    mov ebx,eax
    mov eax,rem
    call sprint
    mov eax,ebx
    call iprintLF
    call quit
```

Рисунок 2.16: Код программы task.asm

При  $x = 1$  результат — 10.

При  $x = 7$  результат — 70.

```
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf task.asm
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 task.o -o task
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ./task
Введите X
1
выражение = : 10
sasuhorukov@ubuntu:~/work/arch-pc/lab06$ ./task
Введите X
7
выражение = : 70
sasuhorukov@ubuntu:~/work/arch-pc/lab06$
```

Рисунок 2.17: Запуск программы task.asm

Программа работает корректно.

## **3 Выводы**

Изучили работу с арифметическими операциями.