

# **Отчёт по лабораторной работе 6**

**Архитектура компьютеров и операционные системы**

Сатторов Икромджон Абдувохидович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Выводы</b>	<b>21</b>

## Список иллюстраций

2.1	Редактирование файла lab6-1.asm . . . . .	7
2.2	Компиляция и проверка программы программы lab6-1.asm . . . .	7
2.3	Редактирование файла lab6-1.asm . . . . .	8
2.4	Компиляция и проверка программы программы lab6-1.asm . . . .	9
2.5	Редактирование файла lab6-2.asm . . . . .	10
2.6	Компиляция и проверка программы программы lab6-2.asm . . . .	10
2.7	Редактирование файла lab6-2.asm . . . . .	11
2.8	Компиляция и проверка программы программы lab6-2.asm . . . .	11
2.9	Компиляция и проверка программы программы lab6-2.asm . . . .	12
2.10	Редактирование файла lab6-3.asm . . . . .	13
2.11	Компиляция и проверка программы программы lab6-3.asm . . . .	13
2.12	Редактирование файла lab6-3.asm . . . . .	14
2.13	Компиляция и проверка программы программы lab6-3.asm . . . .	15
2.14	Редактирование файла variant.asm . . . . .	16
2.15	Компиляция и проверка программы программы variant.asm . . . .	16
2.16	Редактирование файла task.asm . . . . .	19
2.17	Компиляция и проверка программы программы task.asm . . . . .	20

## **Список таблиц**

# 1 Цель работы

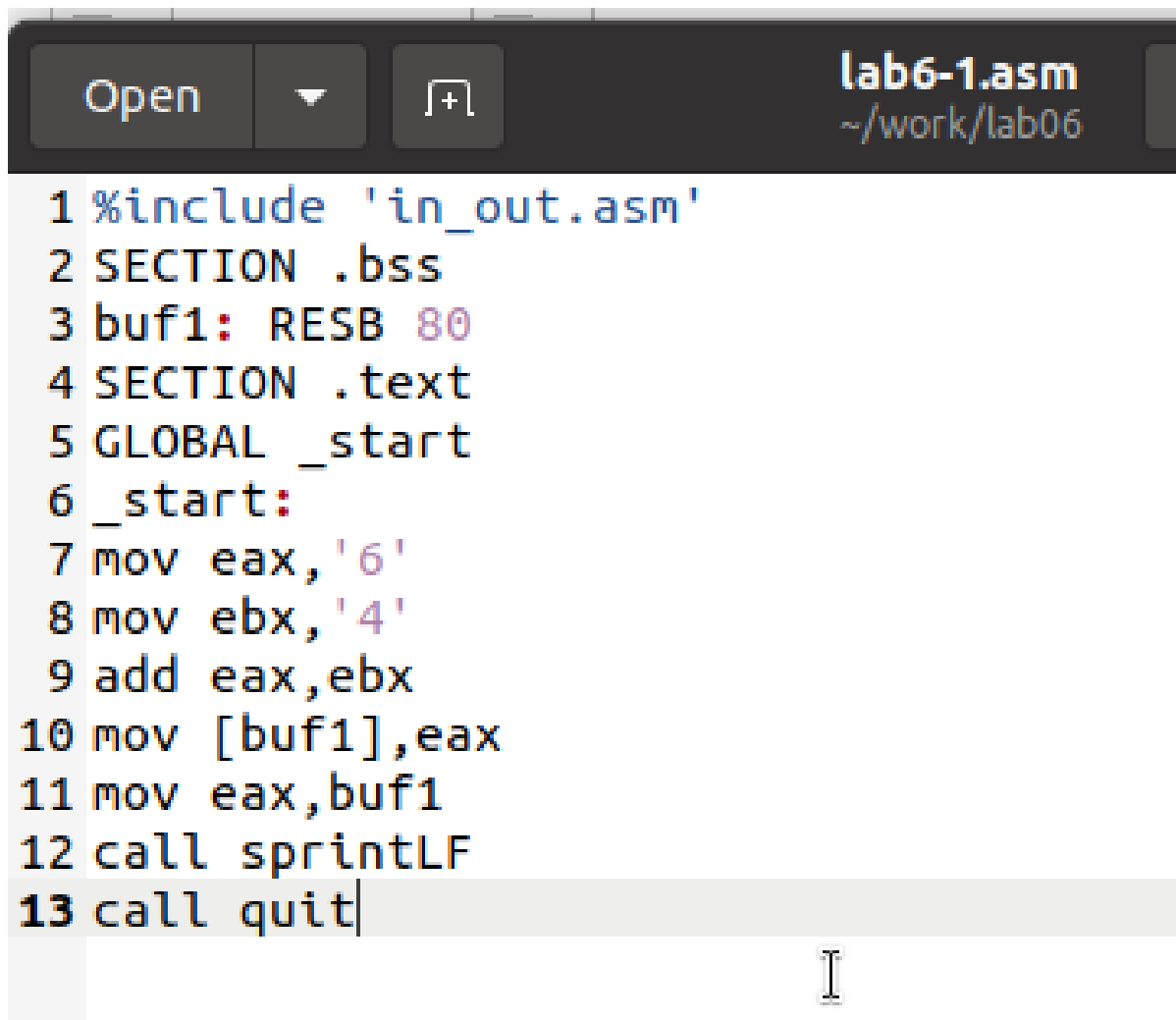
Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

Я создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab6-1.asm.

Давайте рассмотрим примеры программ, которые выводят символьные и числовые значения. В этих программах значения будут записываться в регистр еах.

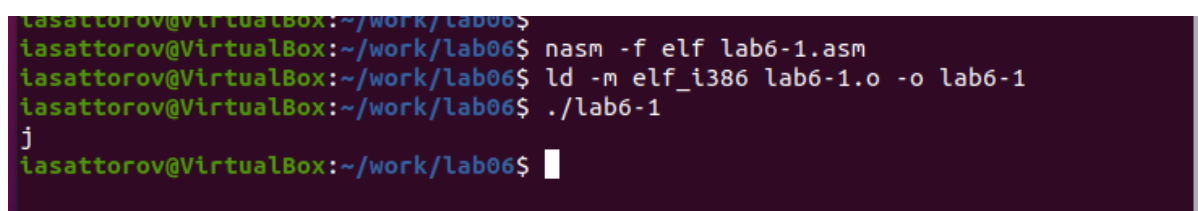
В данной программе мы записываем символ '6' в регистр еах (`mov еах, '6'`), а символ '4' в регистр ебх (`mov ебх, '4'`). Затем мы добавляем значение регистра ебх к значению в регистре еах (`add еах, ебх`, результат сложения записывается в регистр еах). После этого мы выводим результат. Однако, для работы функции `sprintf`, необходимо, чтобы в регистре еах был записан адрес, поэтому мы используем дополнительную переменную. Мы записываем значение регистра еах в переменную `buf1` (`mov [buf1], еах`), а затем записываем адрес переменной `buf1` в регистр еах (`mov еах, buf1`) и вызываем функцию `sprintf`.



```
lab6-1.asm
~/work/lab06

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 printf
13 call _exit
```

Рис. 2.1: Редактирование файла lab6-1.asm



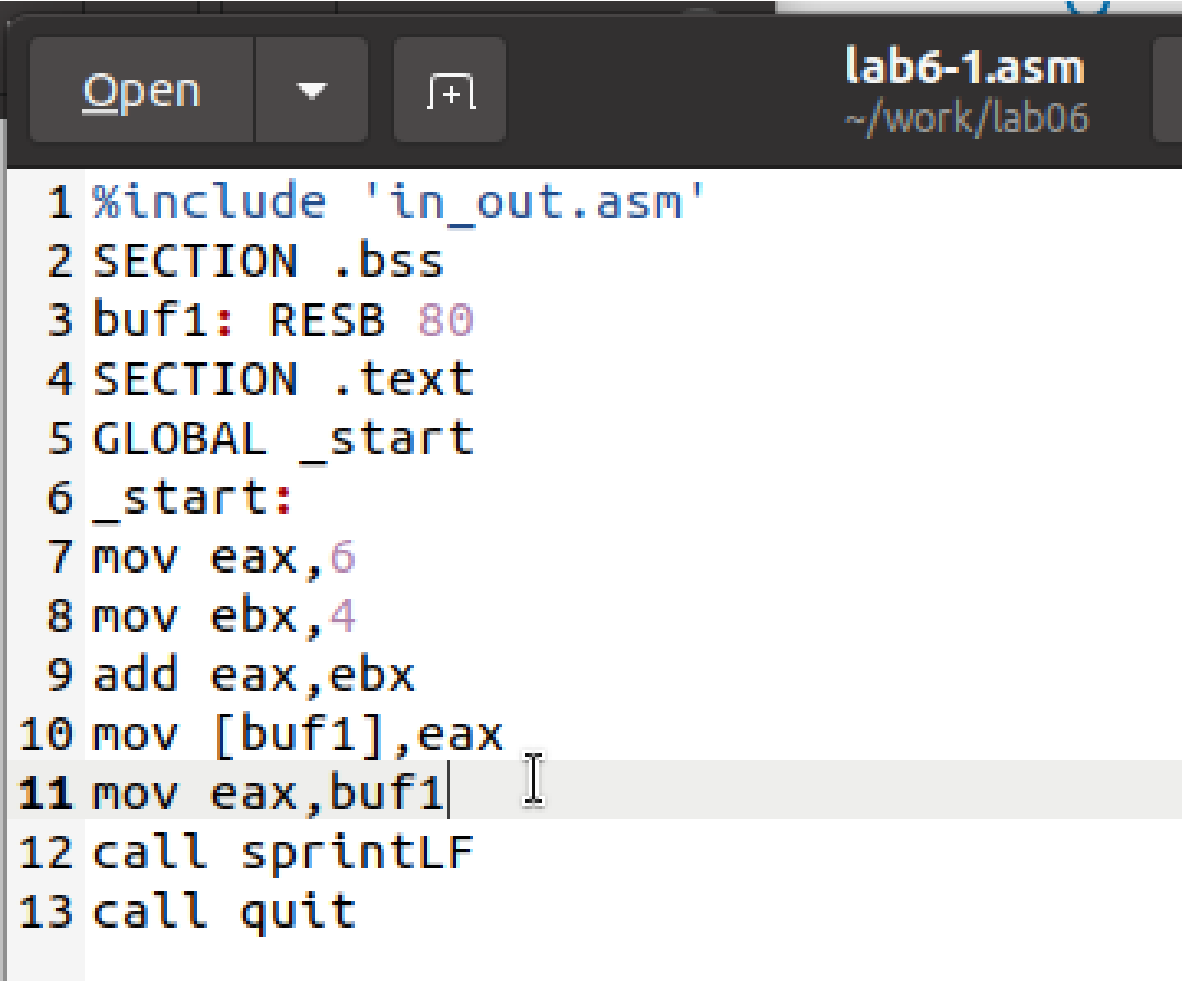
```
iasattorov@VirtualBox:~/work/lab06$ nasm -f elf lab6-1.asm
iasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
iasattorov@VirtualBox:~/work/lab06$ ./lab6-1
j
iasattorov@VirtualBox:~/work/lab06$
```

Рис. 2.2: Компиляция и проверка программы программы lab6-1.asm

В данном случае, когда мы ожидаем увидеть число 10 при выводе значения регистра `eax`, фактическим результатом будет символ `'j'`. Это происходит из-за

того, что код символа '6' равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа '4' равен 00110100 (или 52 в десятичном представлении). Когда мы выполняем команду `add eax, ebx`, результатом будет сумма кодов - 01101010 (или 106 в десятичном представлении), который соответствует символу 'j'.

Далее изменяю текст программы и вместо символов, запишем в регистры числа.



```
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 sprintf
13 call quit
```

Рис. 2.3: Редактирование файла lab6-1.asm

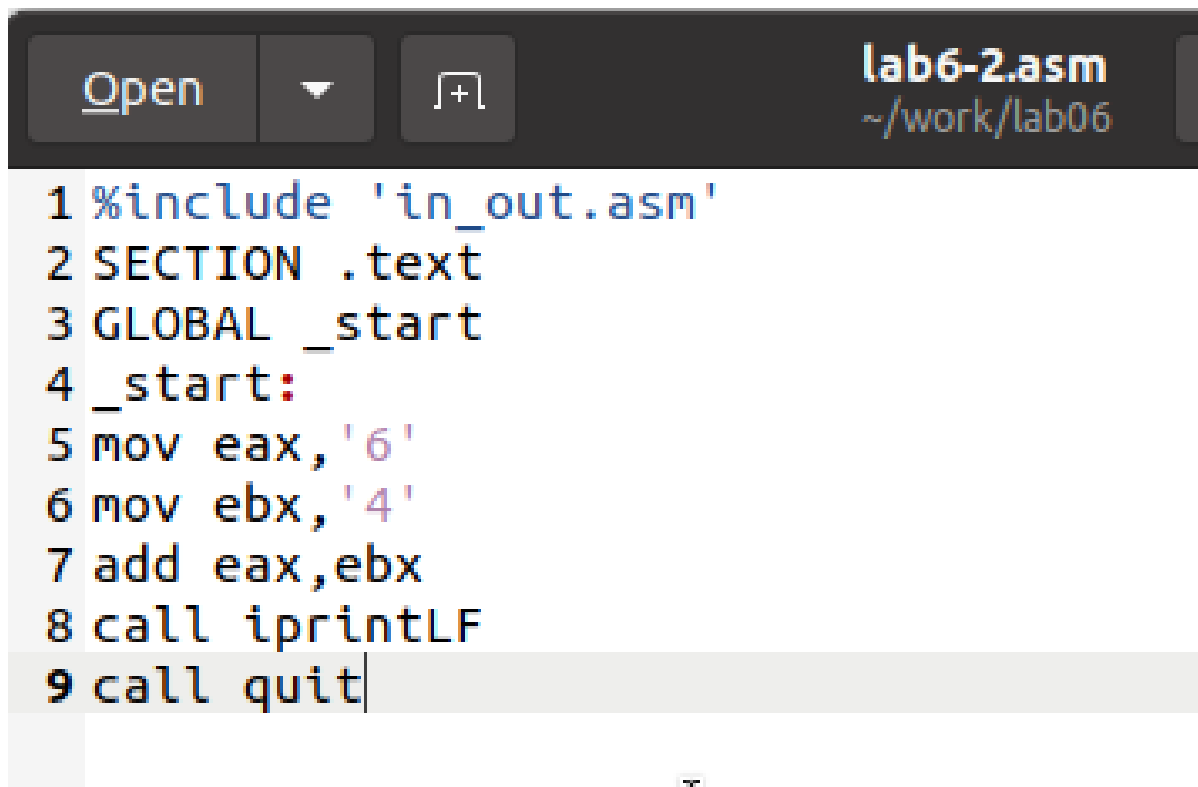


```
iasattorov@VirtualBox:~/work/lab06$  
iasattorov@VirtualBox:~/work/lab06$ nasm -f elf lab6-1.asm  
iasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1  
iasattorov@VirtualBox:~/work/lab06$ ./lab6-1  
j  
iasattorov@VirtualBox:~/work/lab06$  
iasattorov@VirtualBox:~/work/lab06$ nasm -f elf lab6-1.asm  
iasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1  
iasattorov@VirtualBox:~/work/lab06$ ./lab6-1  
  
iasattorov@VirtualBox:~/work/lab06$ █
```

Рис. 2.4: Компиляция и проверка программы программы lab6-1.asm

При изменении текста программы и записи чисел в регистры, мы также не получим число 10 при выполнении программы. Вместо этого будет выведен символ с кодом 10, который представляет собой символ конца строки (возврат каретки). В консоли этот символ не отображается, но он добавляет пустую строку.

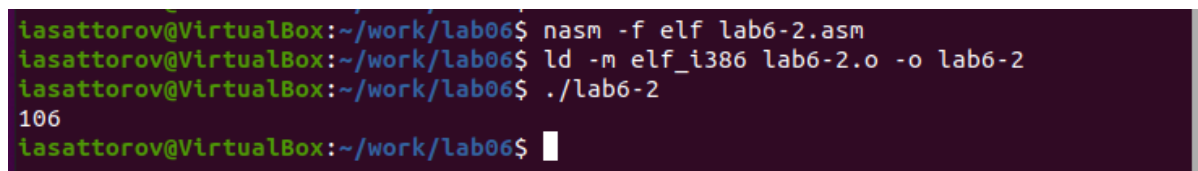
Как уже было отмечено ранее, в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Я преобразовал текст программы, используя эти функции.



```
lab6-2.asm
~/work/lab06

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, '6'
6 mov ebx, '4'
7 add eax, ebx
8 call iprintLF
9 call quit
```

Рис. 2.5: Редактирование файла lab6-2.asm

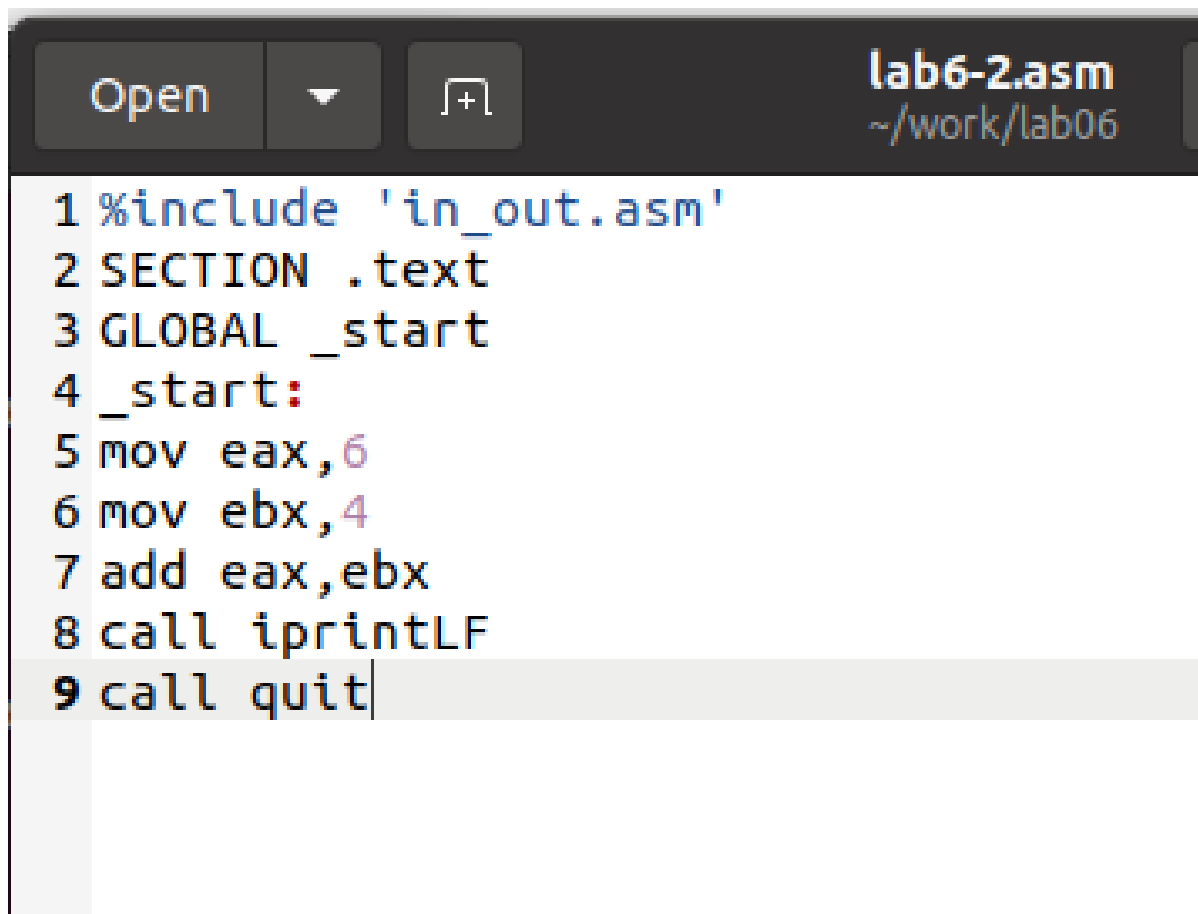


```
iasattorov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
iasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
iasattorov@VirtualBox:~/work/lab06$ ./lab6-2
106
iasattorov@VirtualBox:~/work/lab06$
```

Рис. 2.6: Компиляция и проверка программы программы lab6-2.asm

В результате выполнения программы мы получим число 106. В данном случае, как и в первом случае, команда `add` складывает коды символов '6' и '4' ( $54+52=106$ ). Однако, в отличие от предыдущей программы, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа.

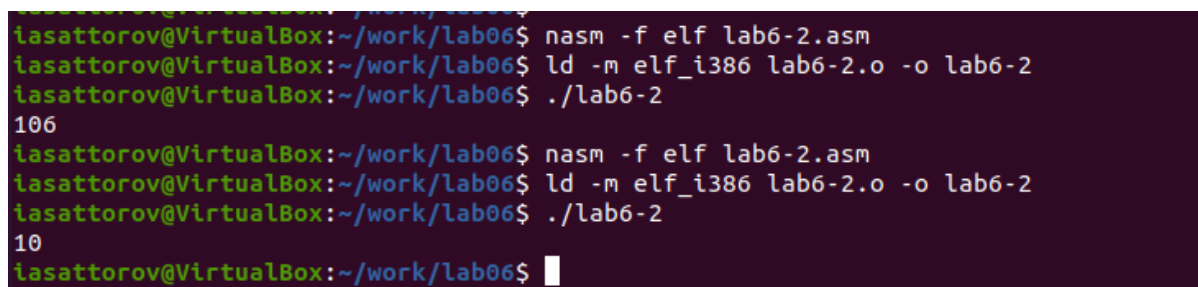


```
lab6-2.asm
~/work/lab06

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintLF
9 call quit
```

Рис. 2.7: Редактирование файла lab6-2.asm

Функция `iprintLF` позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10.



```
lasattorov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
lasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
lasattorov@VirtualBox:~/work/lab06$ ./lab6-2
106
lasattorov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
lasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
lasattorov@VirtualBox:~/work/lab06$ ./lab6-2
10
lasattorov@VirtualBox:~/work/lab06$
```

Рис. 2.8: Компиляция и проверка программы программы lab6-2.asm

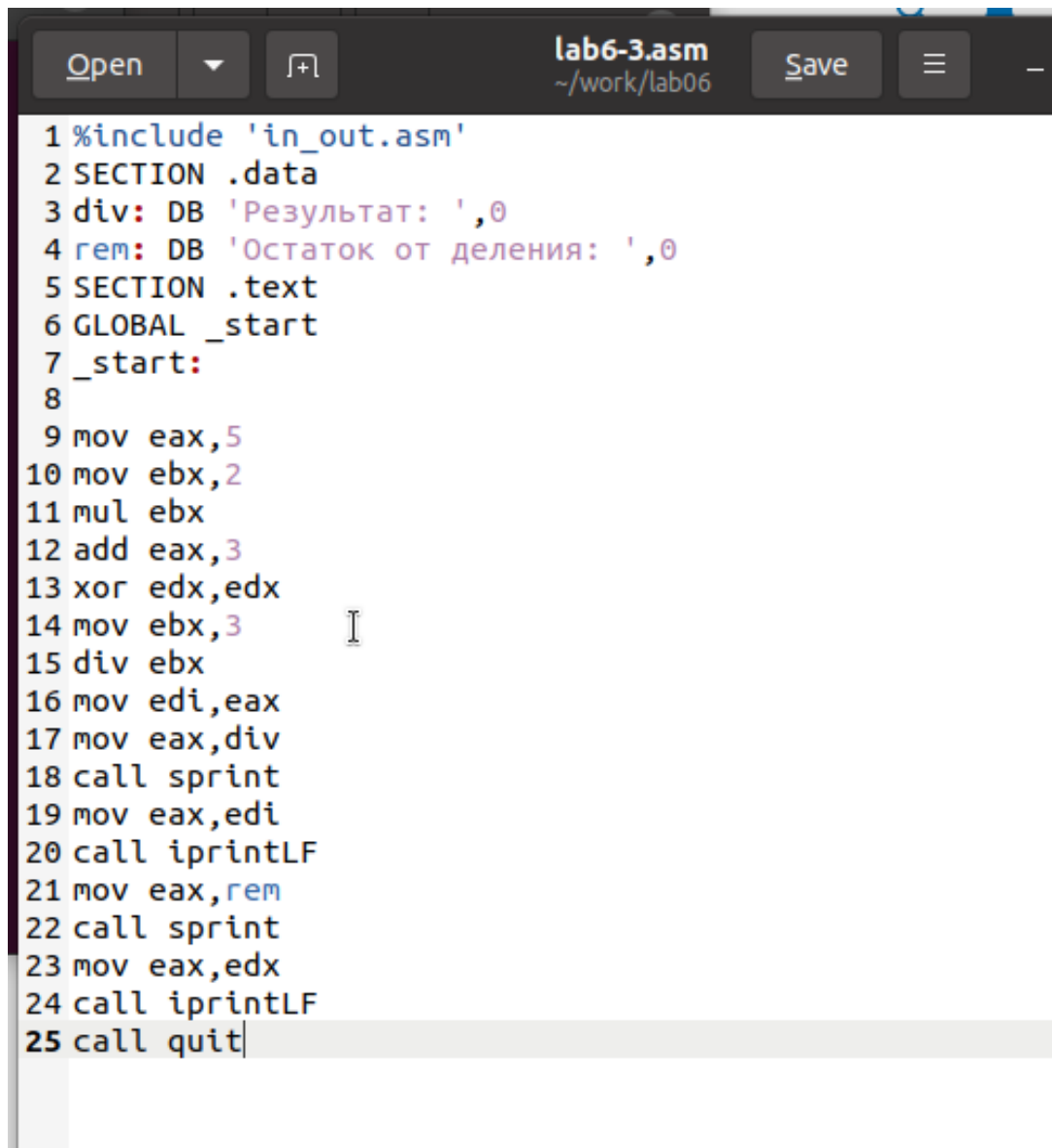
Заменяю функцию `iprintLF` на `iprint`. Создал исполняемый файл и запустил его.

Вывод отличается тем, что нет переноса строки.

```
iasattorov@VirtualBox:~/work/lab06$  
iasattorov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm  
iasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2  
iasattorov@VirtualBox:~/work/lab06$ ./lab6-2  
106  
iasattorov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm  
iasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2  
iasattorov@VirtualBox:~/work/lab06$ ./lab6-2  
10  
iasattorov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm  
iasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2  
iasattorov@VirtualBox:~/work/lab06$ ./lab6-2  
10iasattorov@VirtualBox:~/work/lab06$  
iasattorov@VirtualBox:~/work/lab06$  
iasattorov@VirtualBox:~/work/lab06$
```

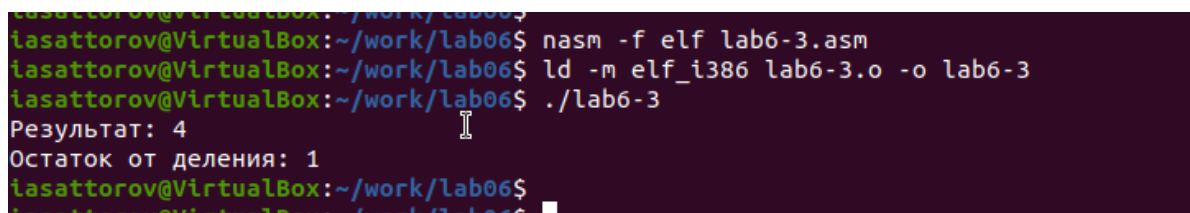
Рис. 2.9: Компиляция и проверка программы программы lab6-2.asm

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения  $f(x) = (5 * 2 + 3) / 3$ .



```
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,5
10 mov ebx,2
11 mul ebx
12 add eax,3
13 xor edx,edx
14 mov ebx,3
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
```

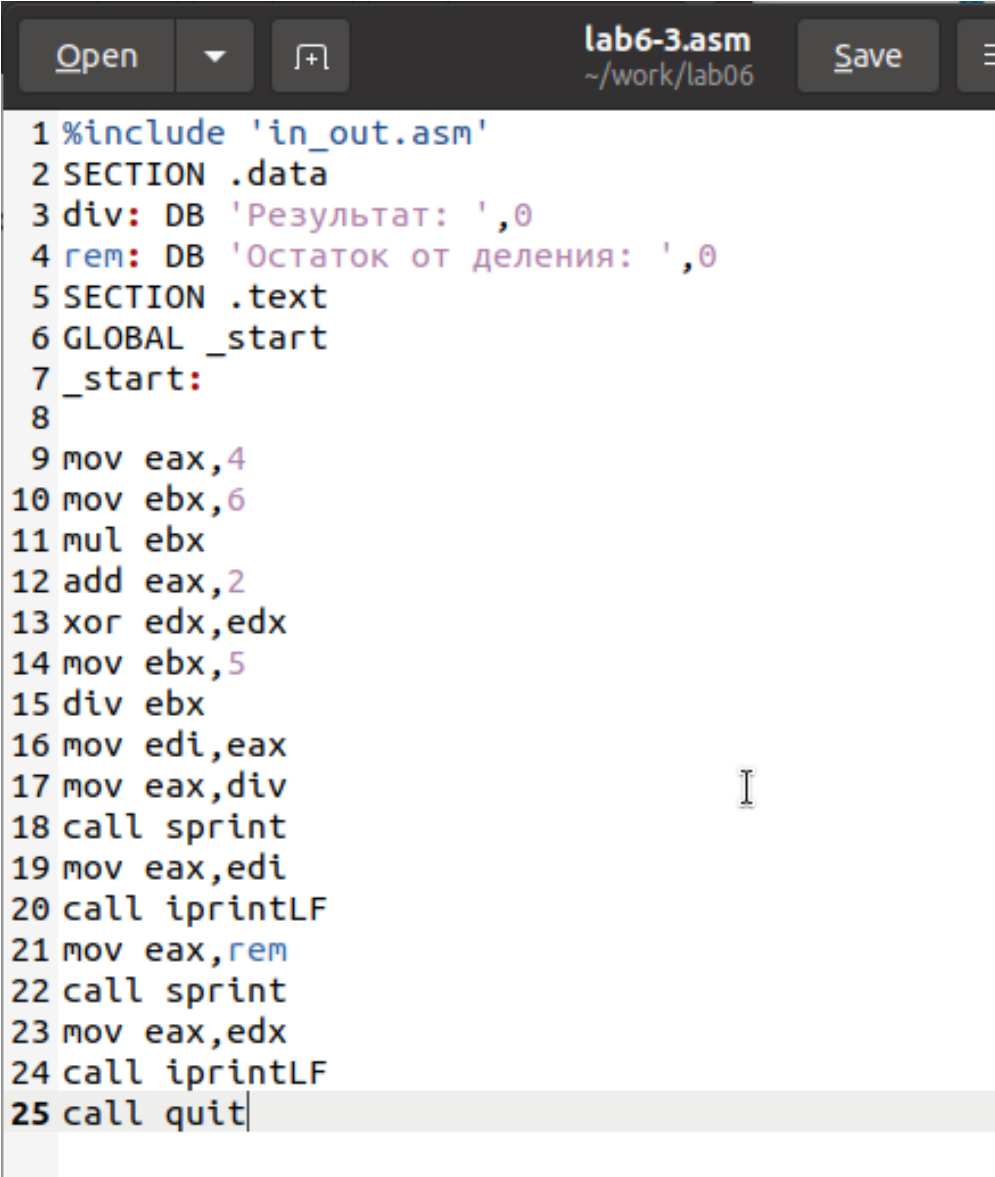
Рис. 2.10: Редактирование файла lab6-3.asm



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

Рис. 2.11: Компиляция и проверка программы программы lab6-3.asm

Изменил текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$ .  
Создал исполняемый файл и проверил его работу.



```
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
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
```

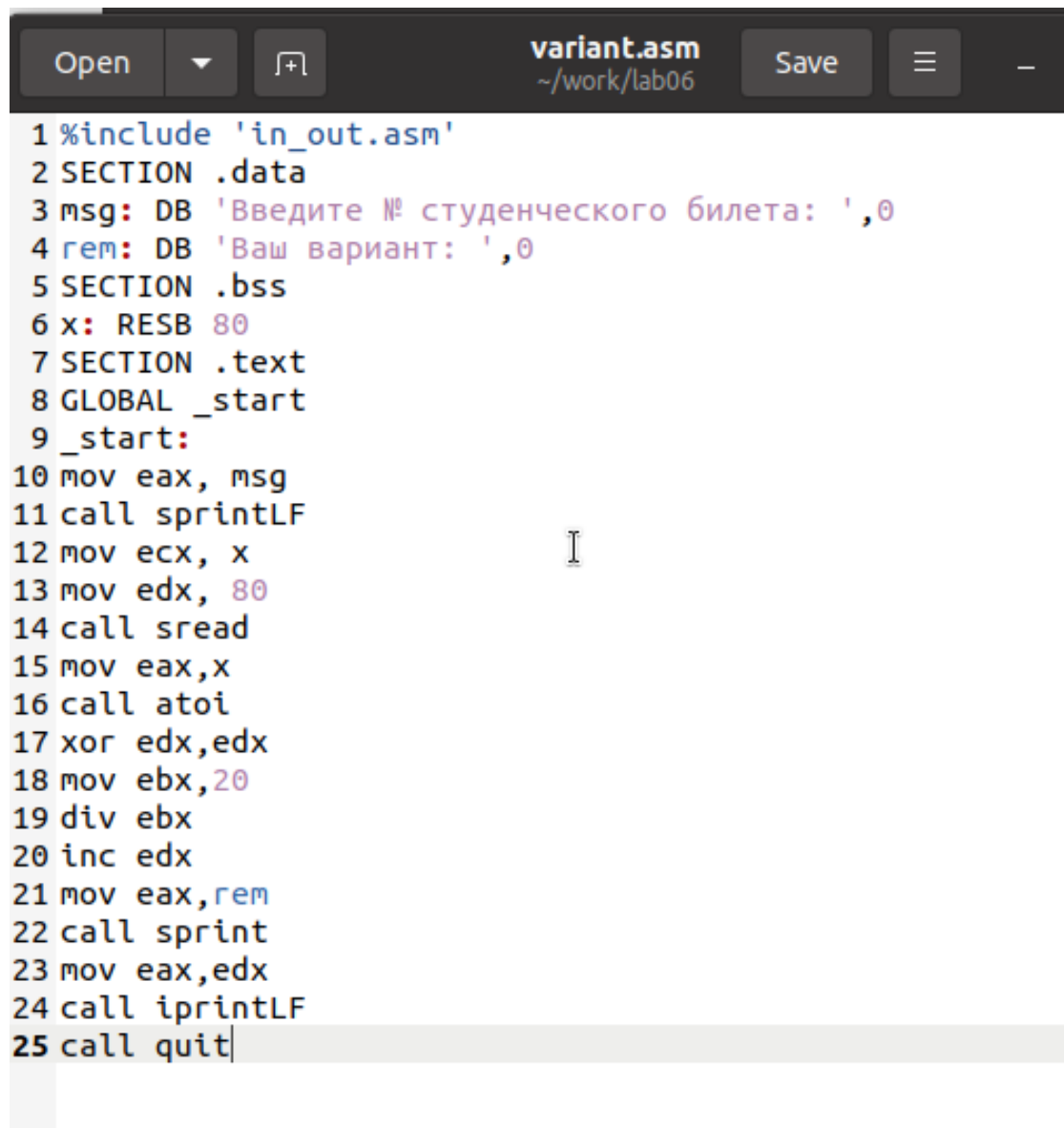
Рис. 2.12: Редактирование файла lab6-3.asm

```
iasattorov@VirtualBox:~/work/lab06$  
iasattorov@VirtualBox:~/work/lab06$ nasm -f elf lab6-3.asm  
iasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3  
iasattorov@VirtualBox:~/work/lab06$ ./lab6-3  
Результат: 5  
Остаток от деления: 1  
iasattorov@VirtualBox:~/work/lab06$  
iasattorov@VirtualBox:~/work/lab06$
```

Рис. 2.13: Компиляция и проверка программы программы lab6-3.asm

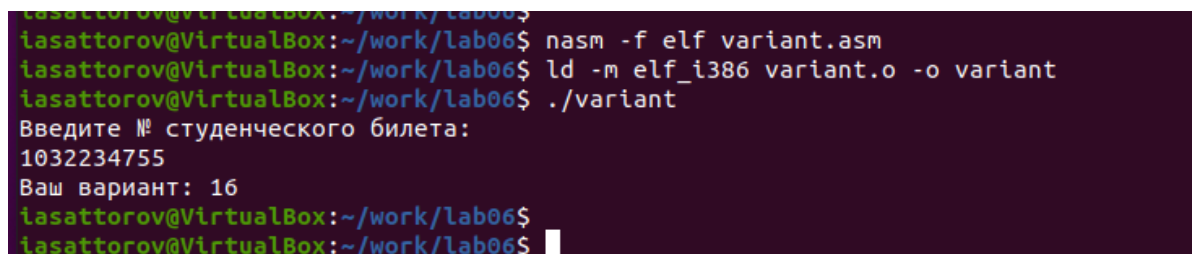
В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета.

В данном случае число, над которым нужно выполнить арифметические операции, вводится с клавиатуры. Как было отмечено ранее, ввод с клавиатуры осуществляется в символьном виде, и для правильной работы арифметических операций в NASM символы должны быть преобразованы в числа. Для этой цели можно использовать функцию `atoi` из файла `in_out.asm`.



```
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 sprintf
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 sprintf
23 mov eax, edx
24 call iprintLF
25 call quit
```

Рис. 2.14: Редактирование файла variant.asm



```
iasattorov@VirtualBox:~/work/lab06$ nasm -f elf variant.asm
iasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 variant.o -o variant
iasattorov@VirtualBox:~/work/lab06$ ./variant
Введите № студенческого билета:
1032234755
Ваш вариант: 16
iasattorov@VirtualBox:~/work/lab06$
iasattorov@VirtualBox:~/work/lab06$
```

Рис. 2.15: Компиляция и проверка программы программы variant.asm



## ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения 'Ваш вариант:'?

Строка `"mov eax, get"` перекладывает значение переменной с фразой 'Ваш вариант:' в регистр `eax`.

Строка `"call sprint"` вызывает подпрограмму для вывода строки.

2. Для чего используются следующие инструкции?

Инструкция `"mov ecx, x"` используется для перемещения значения переменной `x` в регистр `ecx`.

Инструкция `"mov edx, 80"` используется для перемещения значения 80 в регистр `edx`.

Инструкция `"call sread"` вызывает подпрограмму для считывания значения студенческого билета из консоли `sread`

3. Для чего используется инструкция `"call atoi"`?

Инструкция `"call atoi"` используется для преобразования введенных символов в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?

Строка `"xor edx, edx"` обнуляет регистр `edx`.

Строка `"mov ebx, 20"` записывает значение 20 в регистр `ebx`.

Строка `"div ebx"` выполняет деление номера студенческого билета на 20.

Строка `"inc edx"` увеличивает значение регистра `edx` на 1.

5. В какой регистр записывается остаток от деления при выполнении инструкции `"div ebx"`?

Остаток от деления записывается в регистр `edx`.

6. Для чего используется инструкция “inc edx”?

Инструкция “inc edx” используется для увеличения значения в регистре edx на 1, согласно формуле вычисления варианта.

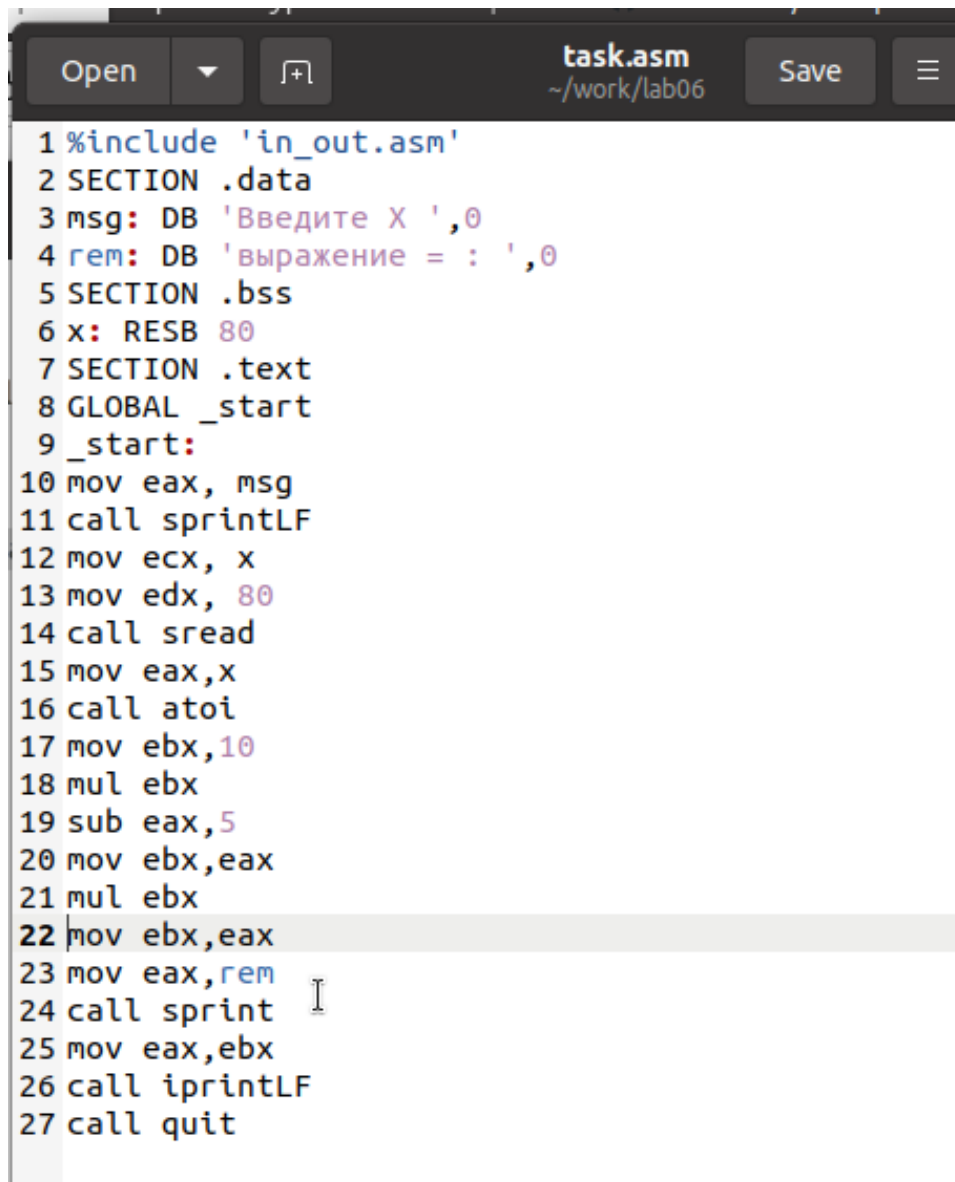
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

Строка “mov eax, edx” перекладывает результат вычислений в регистр eax.

Строка “call iprintLF” вызывает подпрограмму для вывода значения на экран.

Написать программу вычисления выражения  $y = f(x)$ . Программа должна выводить выражение для вычисления, выводить запрос на ввод значения  $x$ , вычислять заданное выражение в зависимости от введенного  $x$ , выводить результат вычислений. Вид функции  $f(x)$  выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений  $x_1$  и  $x_2$  из 6.3.

Получили вариант  $16 - (10x - 5)^2$  для  $x = 3, x = 1$



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите X ',0
4 rem: DB 'выражение = : ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x
16 call atoi
17 mov ebx, 10
18 mul ebx
19 sub eax, 5
20 mov ebx, eax
21 mul ebx
22 mov ebx, eax
23 mov eax, rem
24 call sprintf
25 mov eax, ebx
26 call iprintLF
27 call quit
```

Рис. 2.16: Редактирование файла task.asm

Также размещаю код программы в отчете.

```
iasattorov@VirtualBox:~/work/lab06$  
iasattorov@VirtualBox:~/work/lab06$ nasm -f elf task.asm  
iasattorov@VirtualBox:~/work/lab06$ ld -m elf_i386 task.o -o task  
iasattorov@VirtualBox:~/work/lab06$ ./task  
Введите X  
3  
выражение = : 625  
iasattorov@VirtualBox:~/work/lab06$ ./task  
Введите X  
1  
выражение = : 25  
iasattorov@VirtualBox:~/work/lab06$
```

Рис. 2.17: Компиляция и проверка программы программы task.asm

## **3 Выводы**

Изучили работу с арифметическими операциями.