

Отчёт по лабораторной работе 6

Архитектура ЭВМ

Саруханов Артур Евгеньевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	22

Список иллюстраций

2.1	Программа lab6-1.asm	7
2.2	Запуск программы lab6-1.asm	7
2.3	Программа lab6-1.asm	8
2.4	Запуск программы lab6-1.asm	9
2.5	Программа lab6-2.asm	10
2.6	Запуск программы lab6-2.asm	10
2.7	Программа lab6-2.asm	11
2.8	Запуск программы lab6-2.asm	12
2.9	Программа lab6-2.asm	12
2.10	Запуск программы lab6-2.asm	13
2.11	Программа lab6-3.asm	14
2.12	Запуск программы lab6-3.asm	14
2.13	Программа lab6-3.asm	15
2.14	Запуск программы lab6-3.asm	16
2.15	Программа variant.asm	17
2.16	Запуск программы variant.asm	17
2.17	Программа prog.asm	20
2.18	Запуск программы prog.asm	21

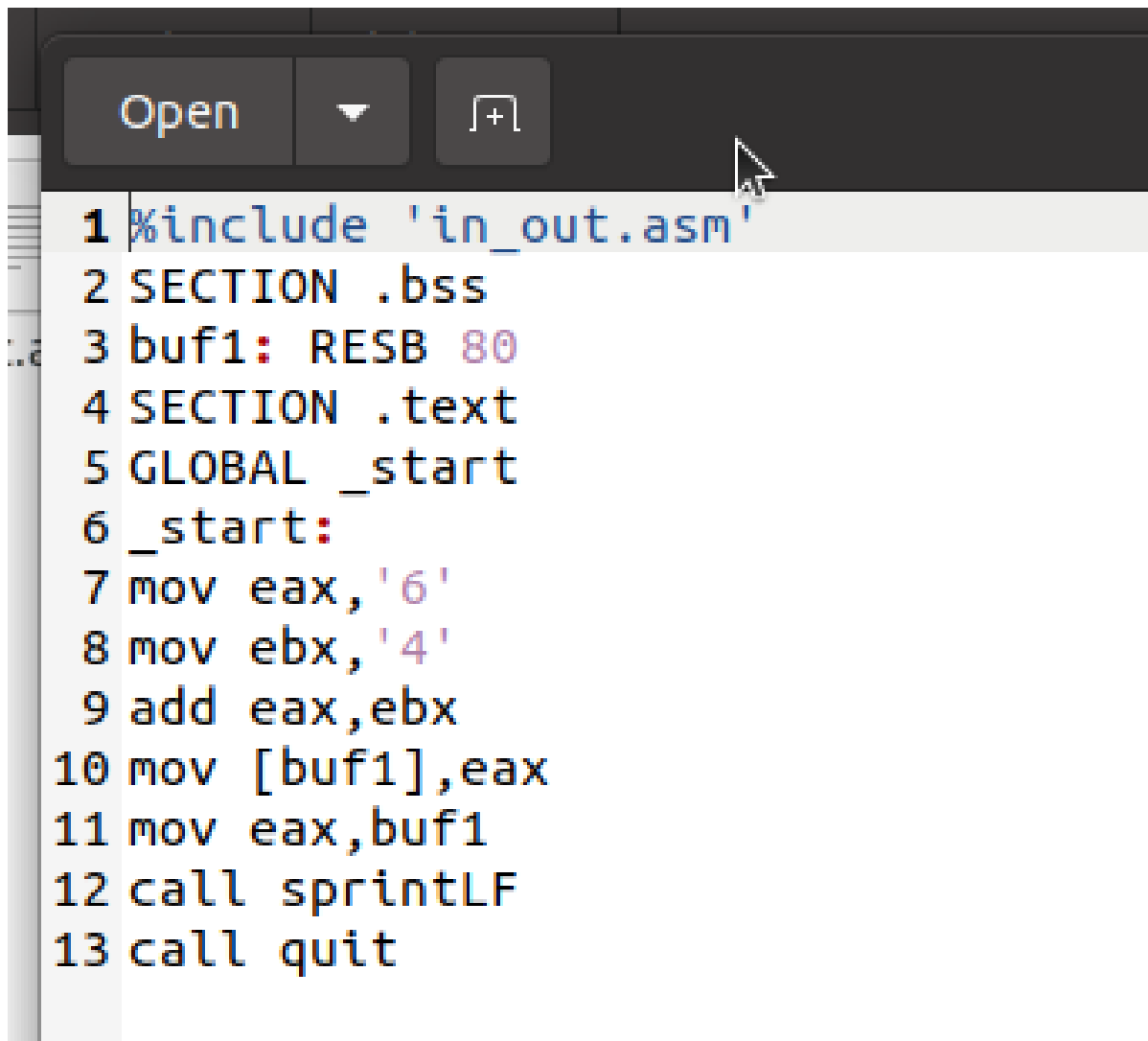
Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

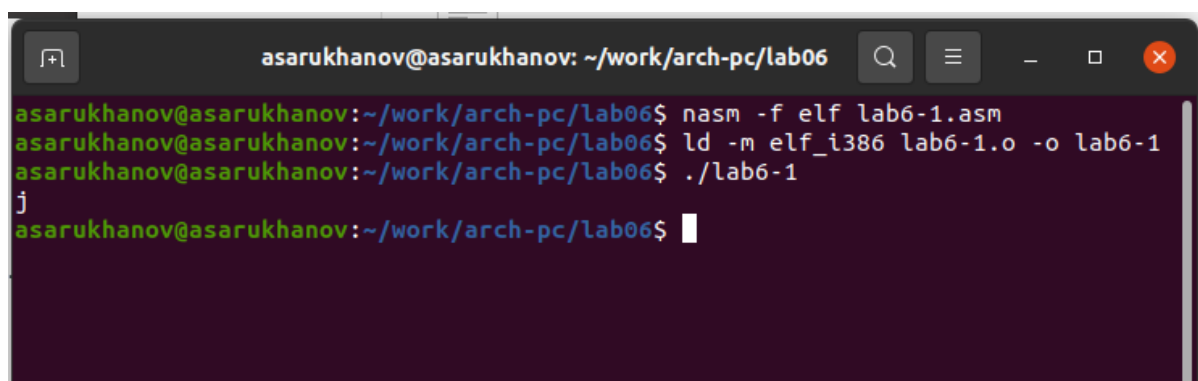
2 Выполнение лабораторной работы

1. Создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab6-1.asm.
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистрах еах.



```
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
```

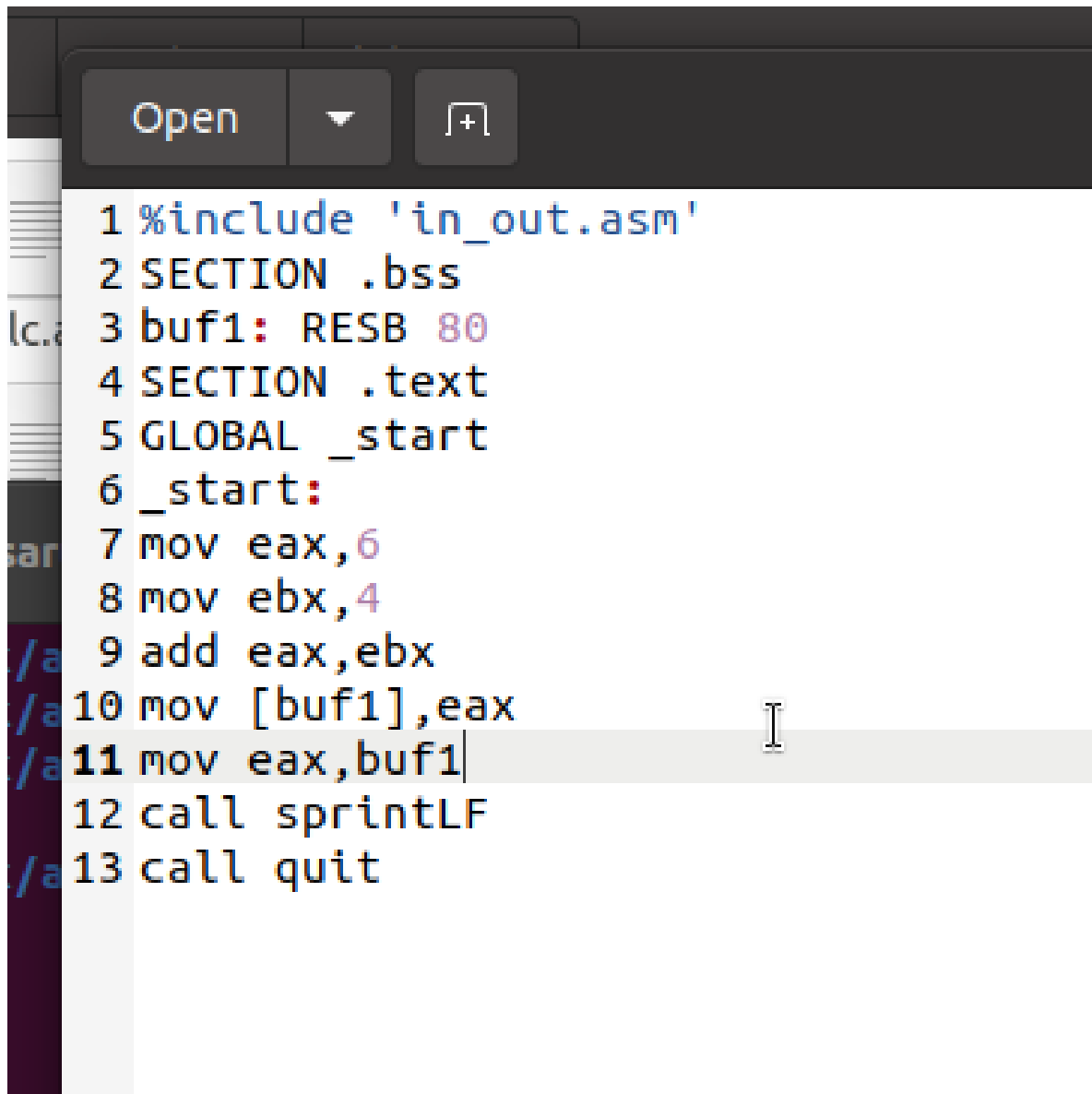
Рис. 2.1: Программа lab6-1.asm



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

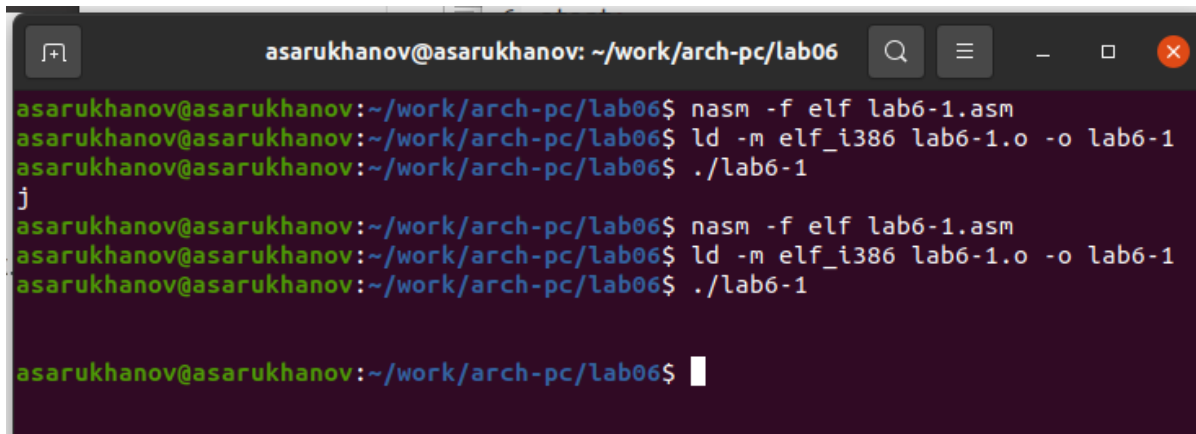
Рис. 2.2: Запуск программы lab6-1.asm

3. 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 sprintfLF
13 call quit
```

Рис. 2.3: Программа lab6-1.asm

A terminal window with a dark background and light-colored text. The window title is 'asarukhanov@asarukhanov: ~/work/arch-pc/lab06'. The terminal shows the following commands and output:

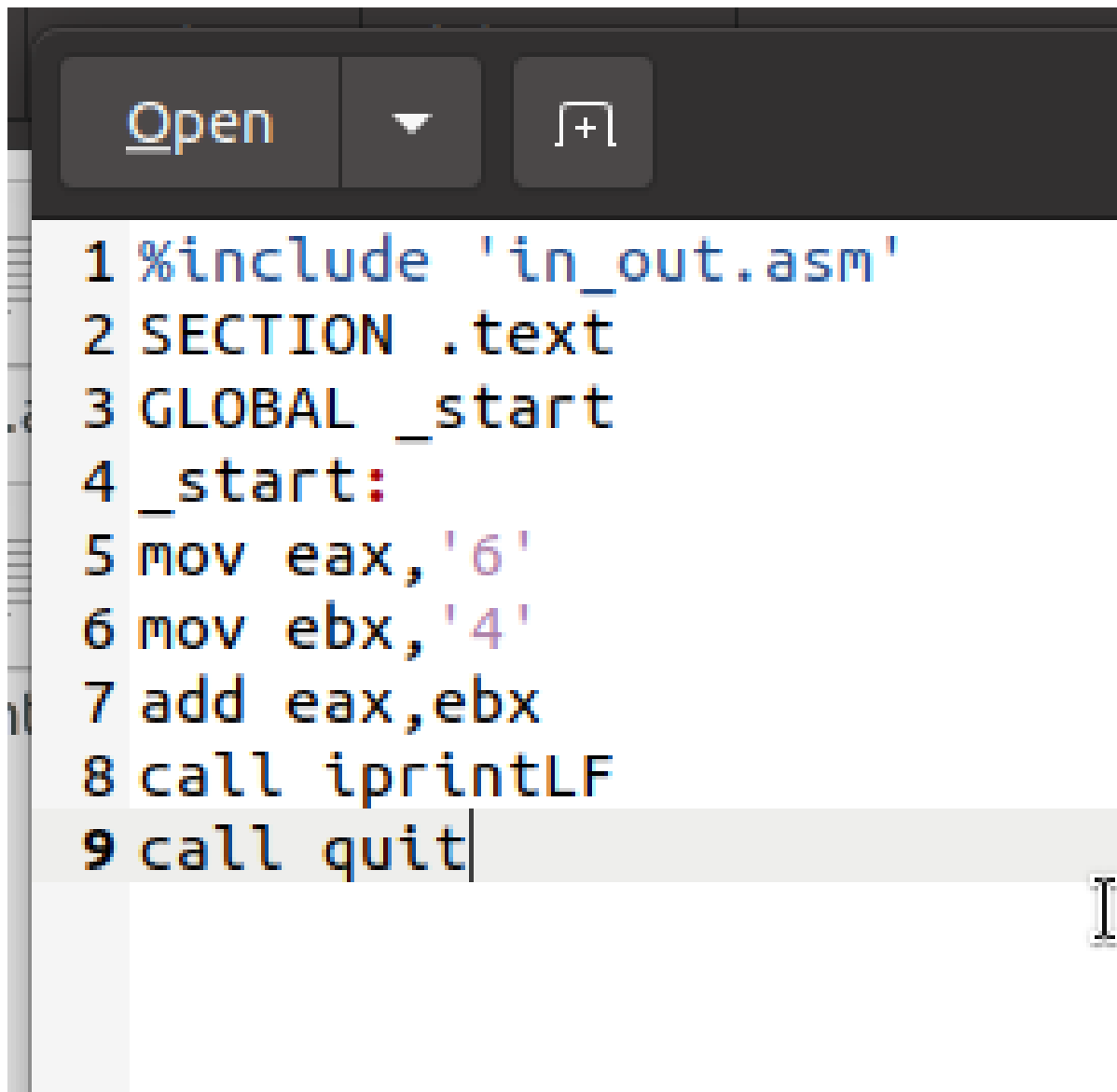
```
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./lab6-1
j
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
asarukhanov@asarukhanov:~/work/arch-pc/lab06$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

asarukhanov@asarukhanov:~/work/arch-pc/lab06$
```

Рис. 2.4: Запуск программы lab6-1.asm

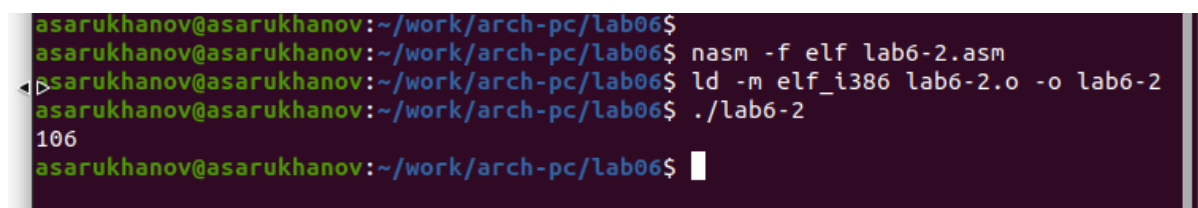
Никакой символ не виден, но он есть. Это возврат каретки LF.

4. Как отмечалось выше, для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовал текст программы с использованием этих функций.



```
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: Программа lab6-2.asm



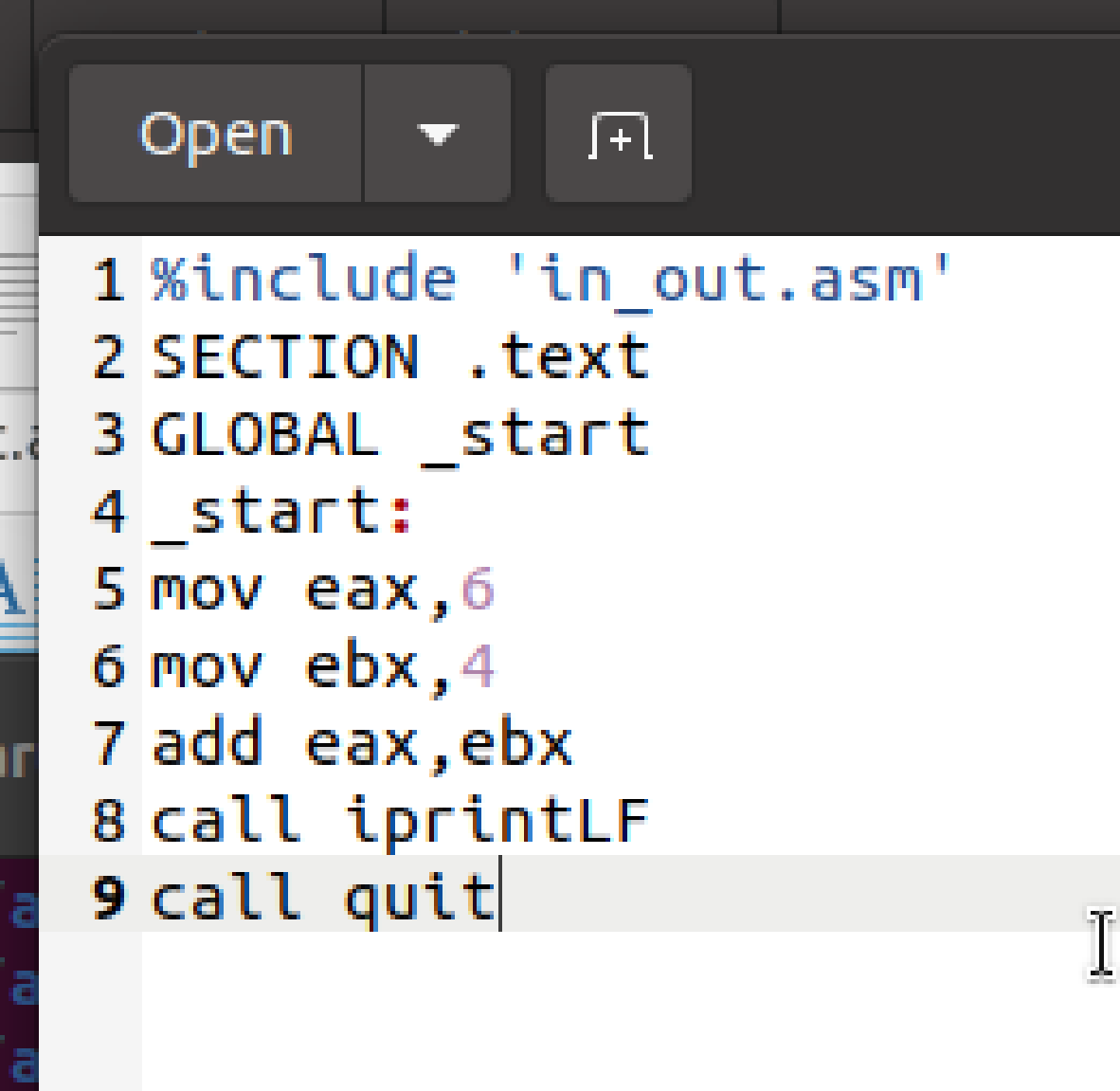
```
asarukhanov@asarukhanov:~/work/arch-pc/lab06$
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./lab6-2
106
asarukhanov@asarukhanov:~/work/arch-pc/lab06$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

Рис. 2.6: Запуск программы lab6-2.asm

В результате работы программы мы получим число 106. В данном случае, как

и в первом, команда `add` складывает коды символов '6' и '4' ($54+52=106$). Однако, в отличие от прошлой программы, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

5. Аналогично предыдущему примеру изменим символы на числа.



```
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: Программа lab6-2.asm

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

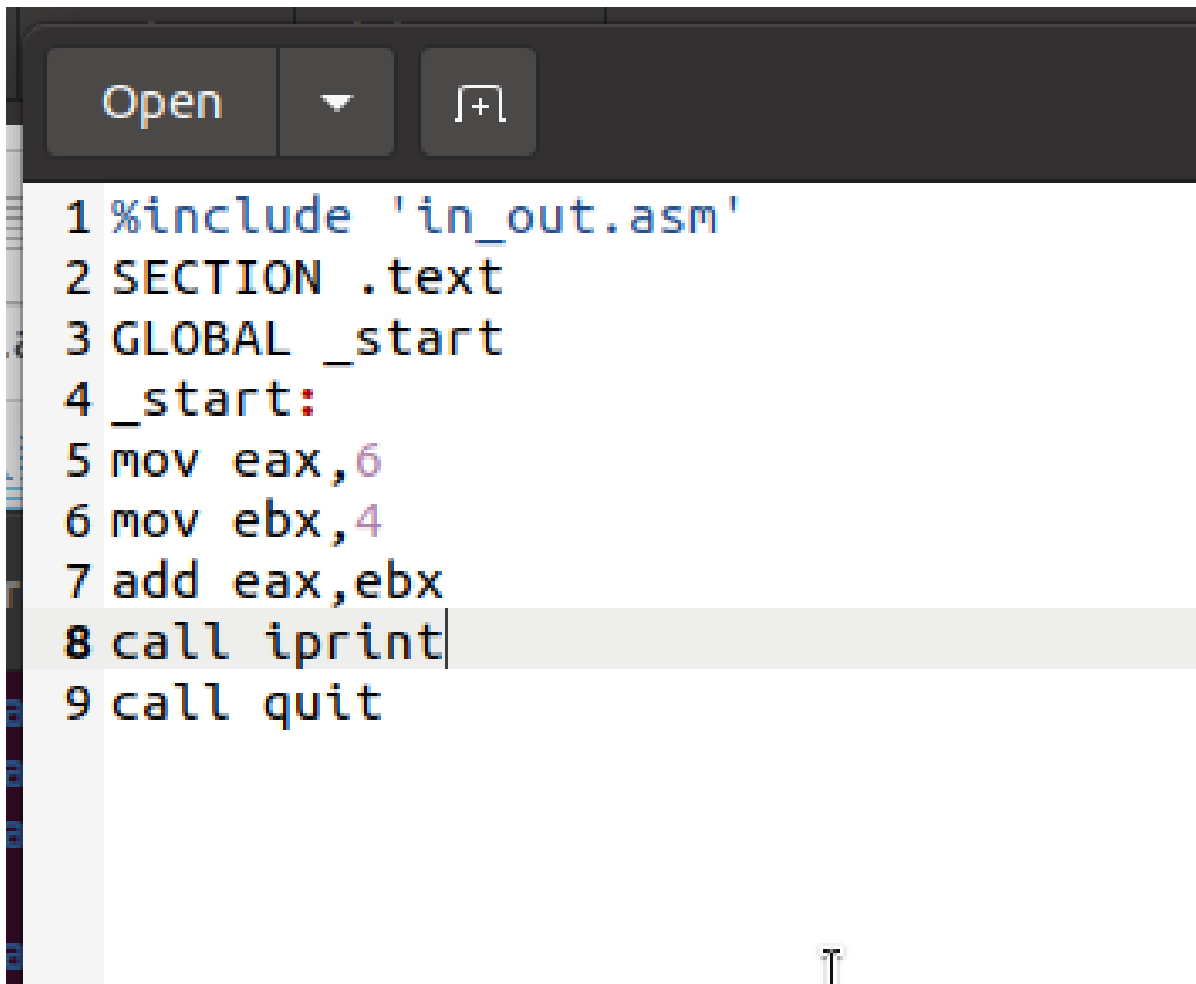
```

asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./lab6-2
106
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./lab6-2
10
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ █

```

Рис. 2.8: Запуск программы lab6-2.asm

Заменял функцию `iprintLF` на `iprint`. Вывод отличается тем, что нет переноса строки.



```

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 iprint
9 call quit

```

Рис. 2.9: Программа lab6-2.asm

```

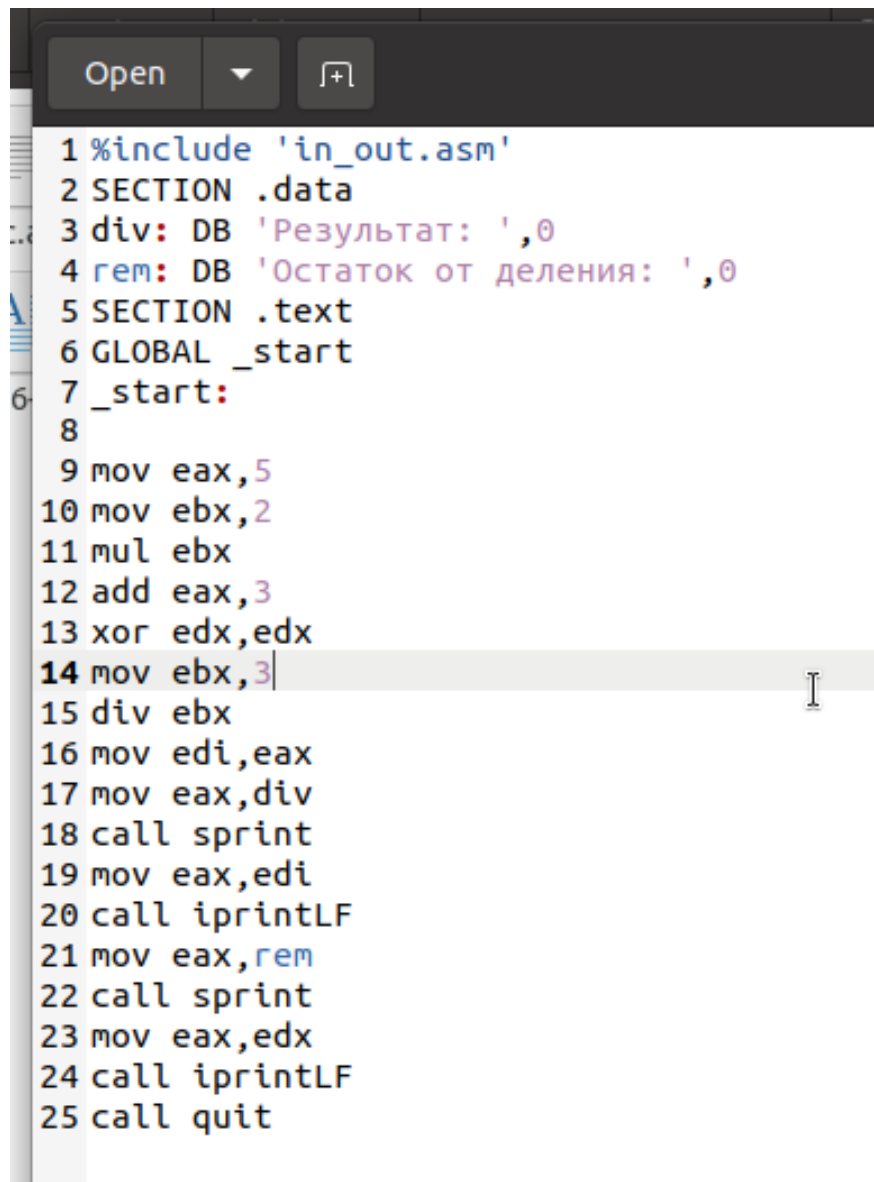
asarukhanov@asarukhanov:~/work/arch-pc/lab06$
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./lab6-2
106
asarukhanov@asarukhanov:~/work/arch-pc/lab06$
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./lab6-2
10
asarukhanov@asarukhanov:~/work/arch-pc/lab06$
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./lab6-2
10
asarukhanov@asarukhanov:~/work/arch-pc/lab06$

```

Рис. 2.10: Запуск программы lab6-2.asm

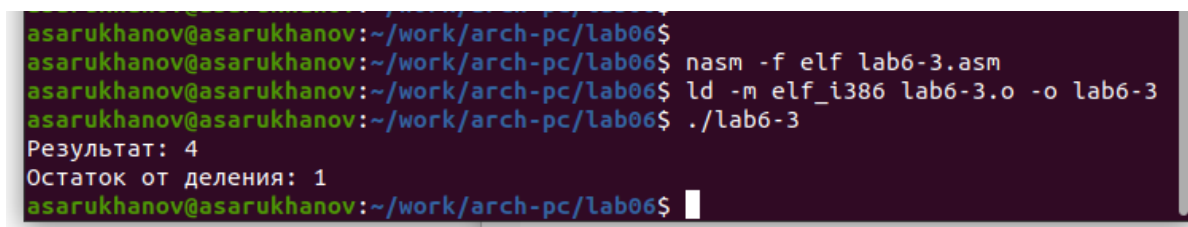
6. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения

$$f(x) = (5 * 2 + 3) / 3$$



```
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. 2.11: Программа lab6-3.asm



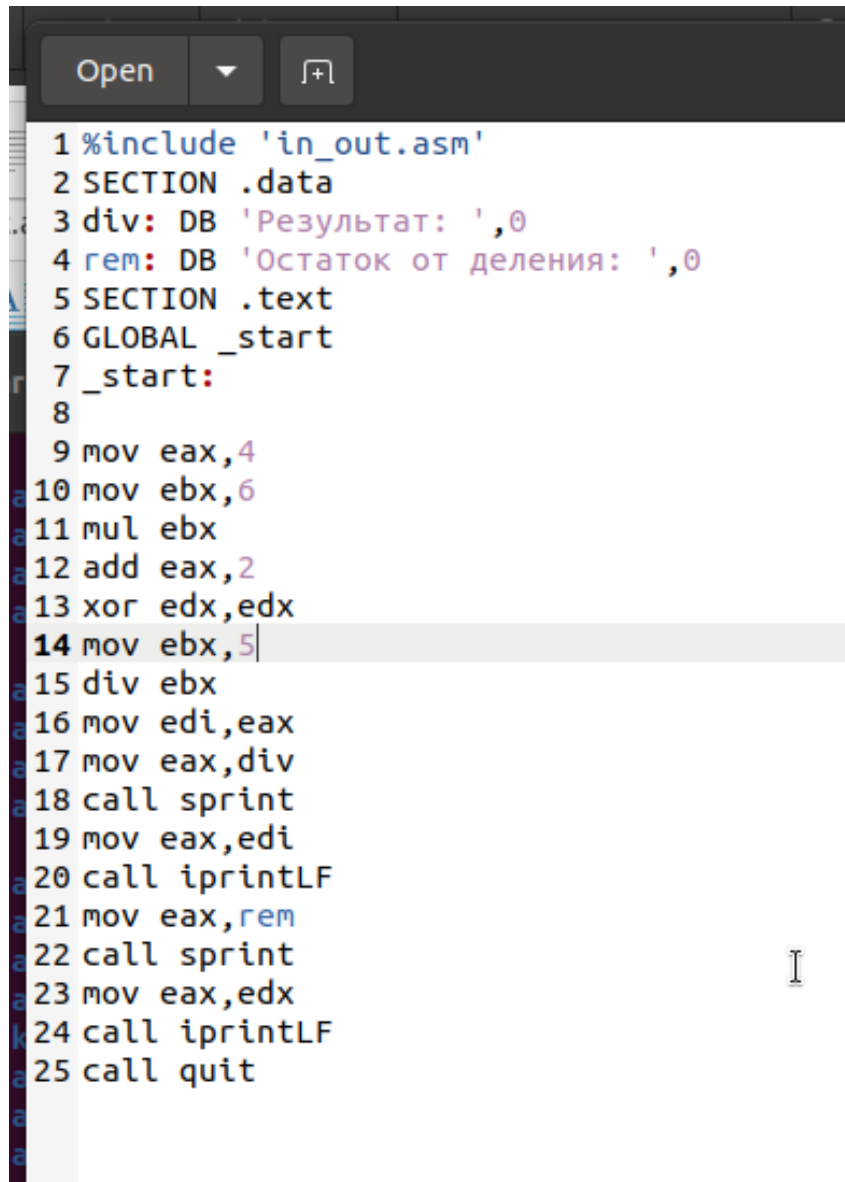
```
asarukhanov@asarukhanov:~/work/arch-pc/lab06$
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
asarukhanov@asarukhanov:~/work/arch-pc/lab06$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

Рис. 2.12: Запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения

$$f(x) = (4 * 6 + 2) / 5$$

. Создал исполняемый файл и проверил его работу.



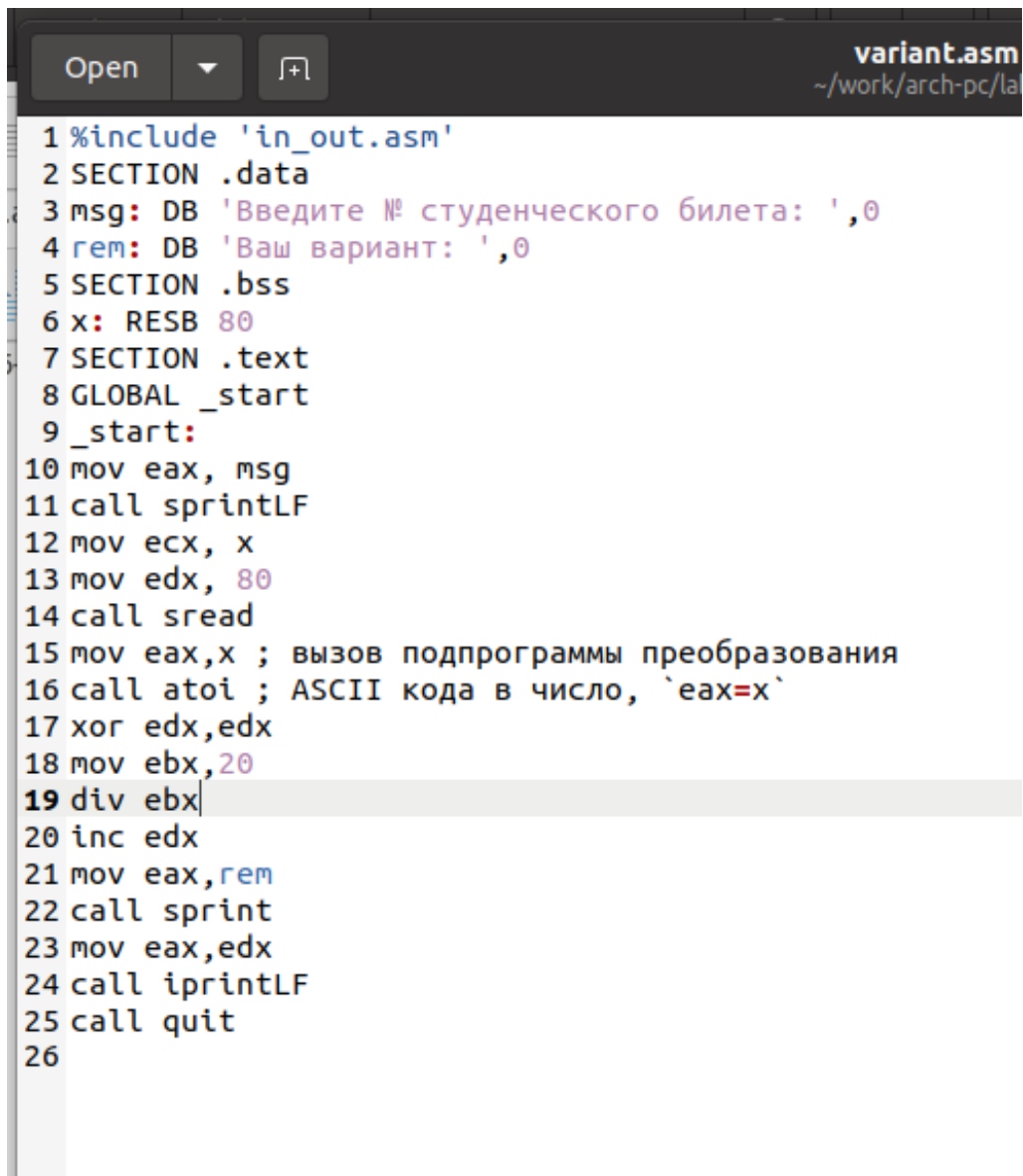
```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8
9 mov eax,4
10 mov ebx,6
11 mul ebx
12 add eax,2
13 xor edx,edx
14 mov ebx,5
15 div ebx
16 mov edi,eax
17 mov eax,div
18 call sprint
19 mov eax,edi
20 call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

Рис. 2.13: Программа lab6-3.asm

```
asarukhanov@asarukhanov:~/work/arch-pc/lab06$  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./lab6-3  
Результат: 4  
Остаток от деления: 1  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./lab6-3  
Результат: 5  
Остаток от деления: 1  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$
```

Рис. 2.14: Запуск программы lab6-3.asm

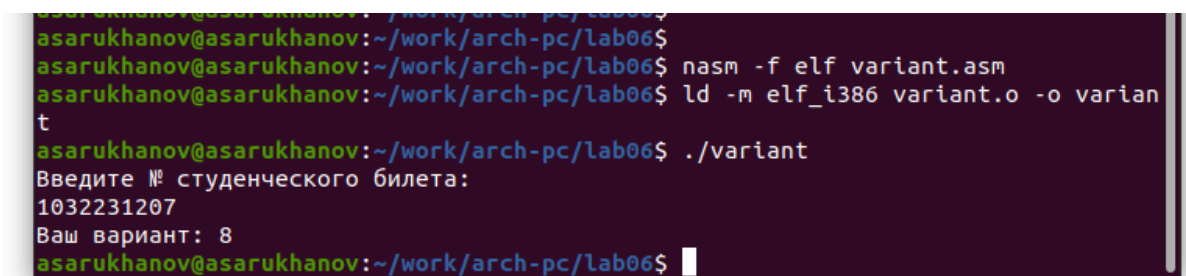
7. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета.



```
variant.asm
~/work/arch-pc/la

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 ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, `eax=x`
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
26
```

Рис. 2.15: Программа variant.asm



```
asarukhanov@asarukhanov:~/work/arch-pc/lab06$
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf variant.asm
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032231207
Ваш вариант: 8
asarukhanov@asarukhanov:~/work/arch-pc/lab06$
```

Рис. 2.16: Запуск программы variant.asm

ответы на вопросы

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

Значение переменной с фразой ‘Ваш вариант:’ перекладывается в регистр `eax` с помощью строки `mov eax, rem`. Вызывается подпрограмма `sprint` для вывода строки.

2. Для чего используются следующие инструкции?

```
mov ecx, x  
mov edx, 80  
call sread
```

Инструкция `mov ecx, x` перемещает значение переменной `X` в регистр `ecx`.

Инструкция `mov edx, 80` устанавливает значение 80 в регистр `edx`.

Инструкция `call sread` вызывает подпрограмму для чтения значения с консоли.

3. Для чего используется инструкция “`call atoi`”?

Инструкция `call atoi` вызывает подпрограмму, которая преобразует введенные символы в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?

Инструкция `xor edx, edx` обнуляет регистр `edx`.

Инструкция `mov ebx, 20` устанавливает значение 20 в регистр `ebx`.

Инструкция `div ebx` выполняет деление номера студенческого билета на 20.

Инструкция `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 регистра `edx` на 1. В данном случае, она используется для выполнения формулы вычисления варианта, где требуется добавить 1 к остатку от деления.

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

Результат вычислений перекладывается в регистр `eax` с помощью строки `mov eax, edx`. Вызывается подпрограмма `iprintLF` для вывода результата на экран.

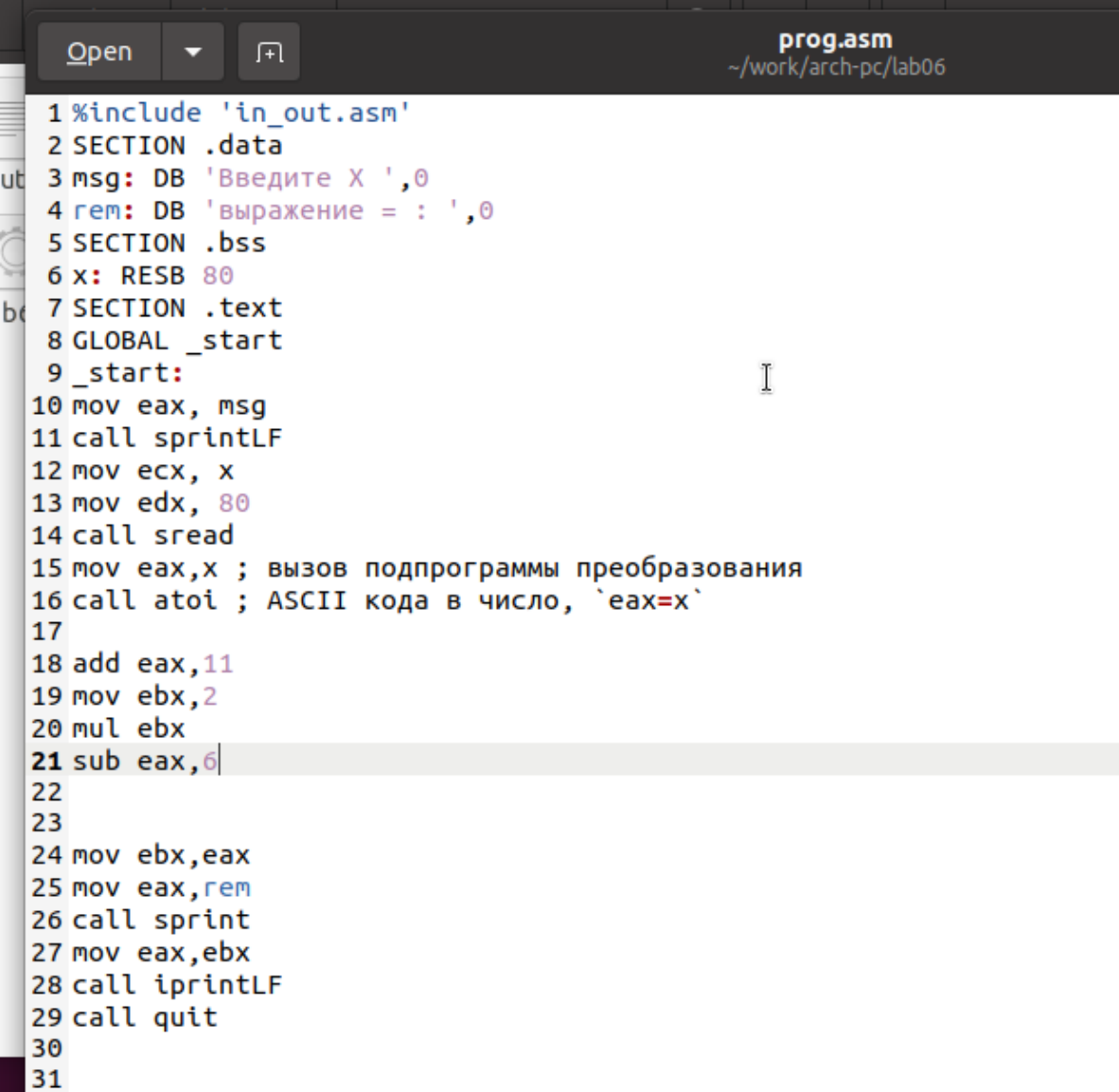
8. Написать программу вычисления выражения $y = f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии 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 8 -

$$(11 + x) * 2 - 6$$

для

$$x = 1, x = 9$$



```
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 sprintLF
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax,x ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, `eax=x`
17
18 add eax,11
19 mov ebx,2
20 mul ebx
21 sub eax,6
22
23
24 mov ebx,eax
25 mov eax,rem
26 call sprint
27 mov eax,ebx
28 call iprintLF
29 call quit
30
31
```

Рис. 2.17: Программа prog.asm

```
asarukhanov@asarukhanov:~/work/arch-pc/lab06$  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ nasm -f elf prog.asm  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ld -m elf_i386 prog.o -o prog  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./prog  
Введите X  
1  
выражение = : 18  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ ./prog  
Введите X  
9  
выражение = : 34  
asarukhanov@asarukhanov:~/work/arch-pc/lab06$ █
```

Рис. 2.18: Запуск программы prog.asm

3 Выводы

Изучили работу с арифметическими операциями.