

Отчёт по лабораторной работе 6

Архитектура компьютера

Ушаков Данила Алексеевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Символьные и численные данные в NASM	6
2.2	Выполнение арифметических операций в NASM	12
2.3	Задание для самостоятельной работы	18
3	Выводы	21

Список иллюстраций

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

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

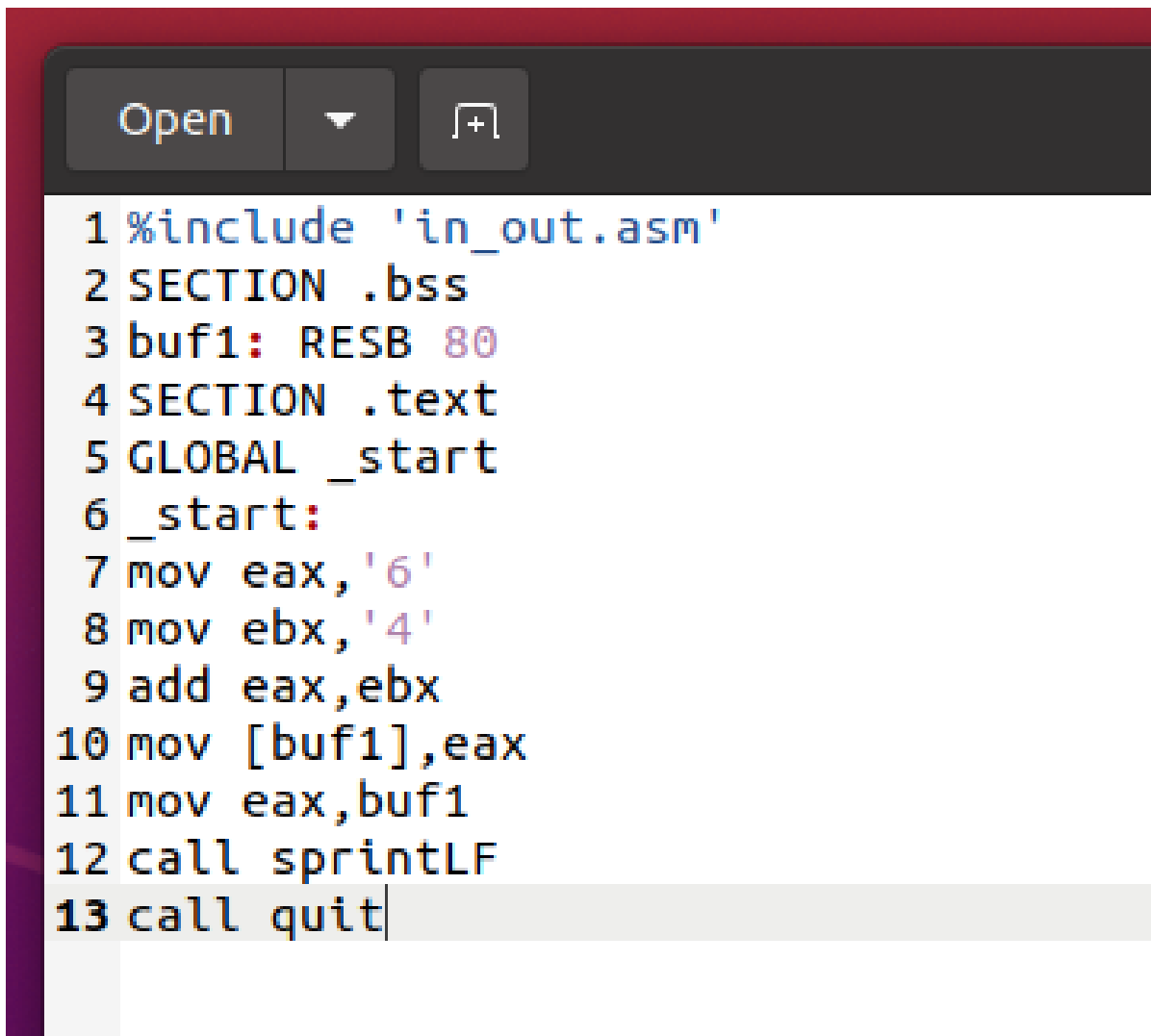
2.1 Символьные и численные данные в NASM

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

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр еах.

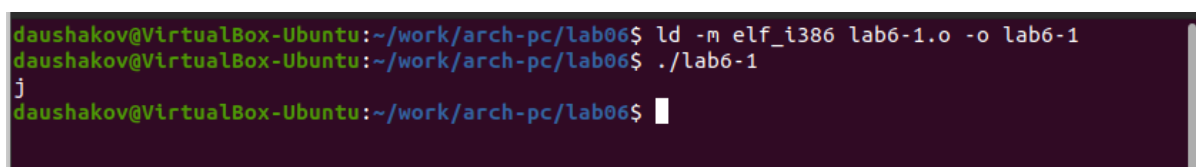
В данной программе (рис. 2.1) в регистр еах записывается символ 6 (mov еах,'6'), в регистр ебх символ 4 (mov ебх,'4'). Далее к значению в регистре еах прибавляем значение регистра ебх (add еах,ебх, результат сложения запишется в регистр еах). Далее выводим результат. (рис. 2.2)

Так как для работы функции sprintLF в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра еах в переменную buf1 (mov [buf1],еах), а затем запишем адрес переменной buf1 в регистр еах (mov еах,buf1) и вызовем функцию sprintLF.



```
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: Код программы lab6-1.asm



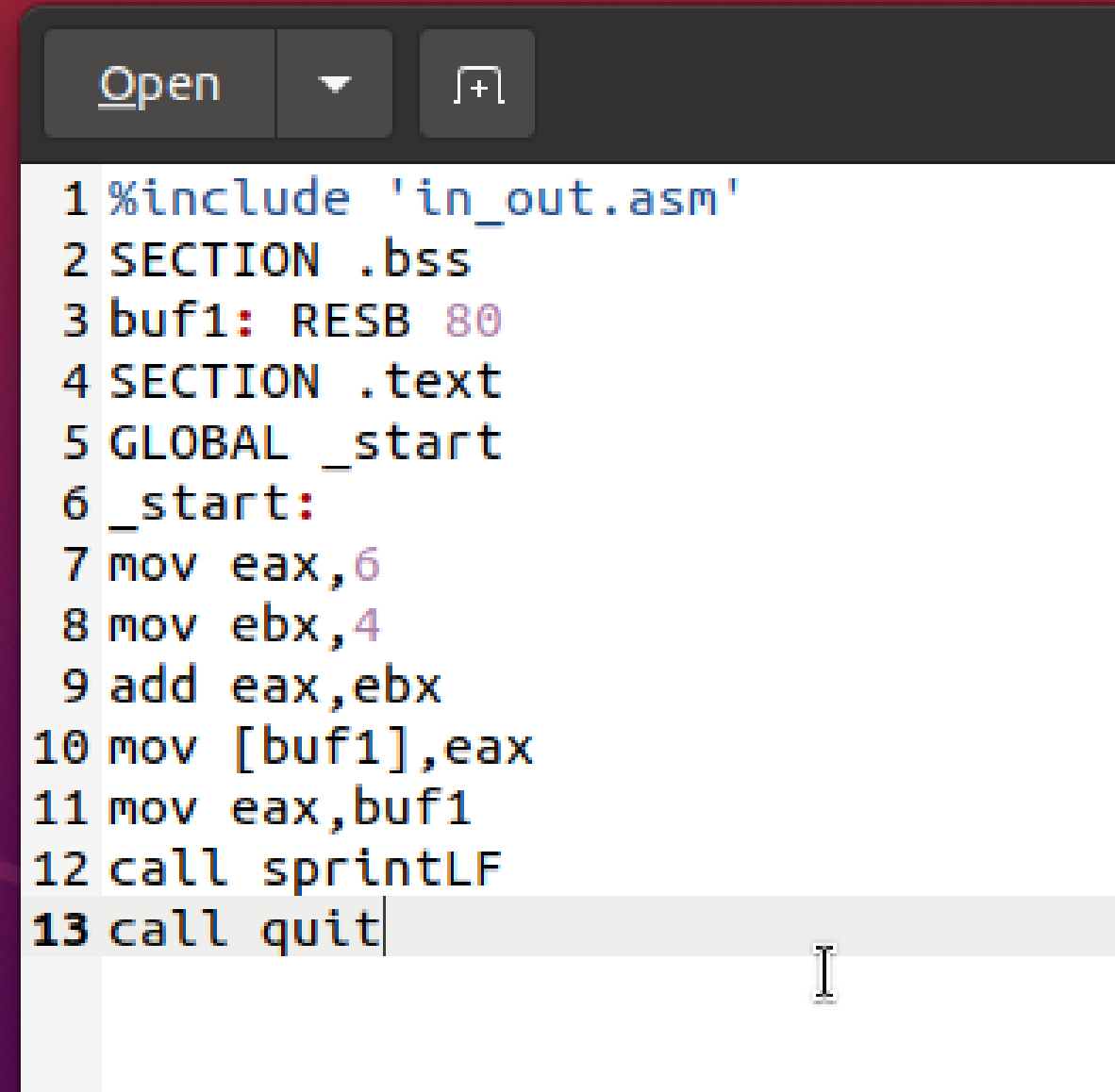
```
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ./lab6-1
j
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$
```

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

В данном случае при выводе значения регистра `eax` мы ожидаем увидеть число 10. Однако результатом будет символ `j`. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда `add eax,ebx` запишет в ре-

гистр еах сумму кодов – 01101010 (106), что в свою очередь является кодом символа j.

Далее изменяю текст программы и вместо символов, запишем в регистры числа. (рис. 2.3)



```
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: Код программы lab6-1.asm

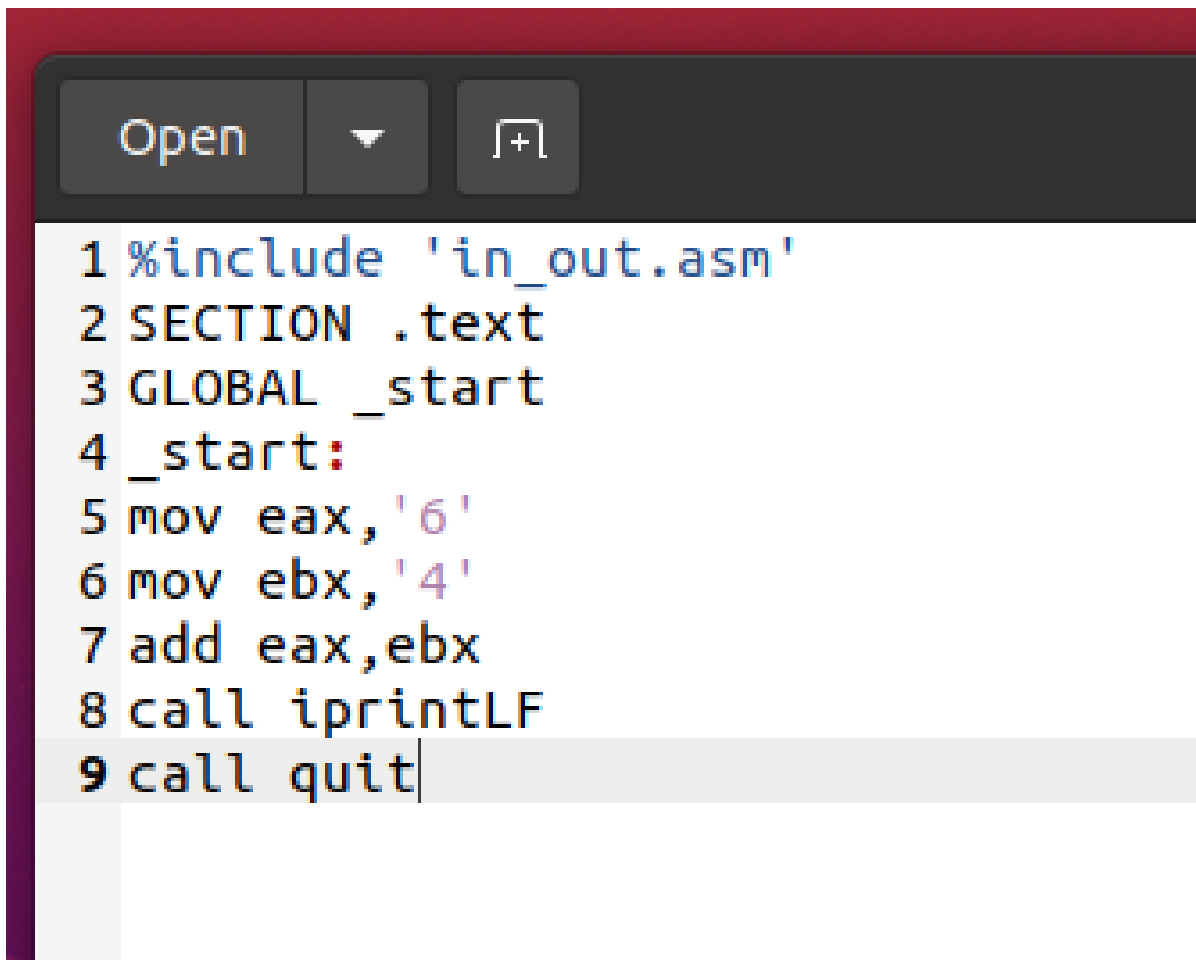

```
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ./lab6-1

daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ █
```

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

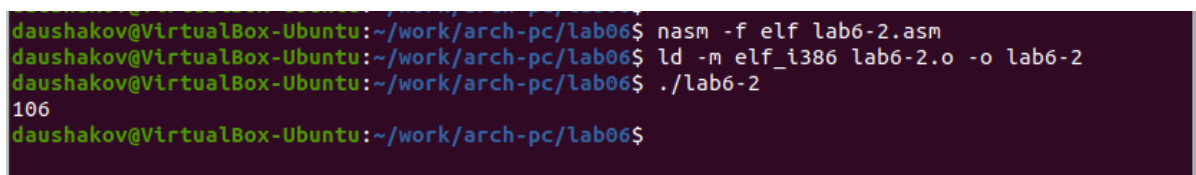
Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10 (рис. 2.4). Это символ конца строки (возврат каретки). В консоле он не отображается, но добавляет пустую строку.

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. (рис. 2.5)



```
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: Код программы lab6-2.asm

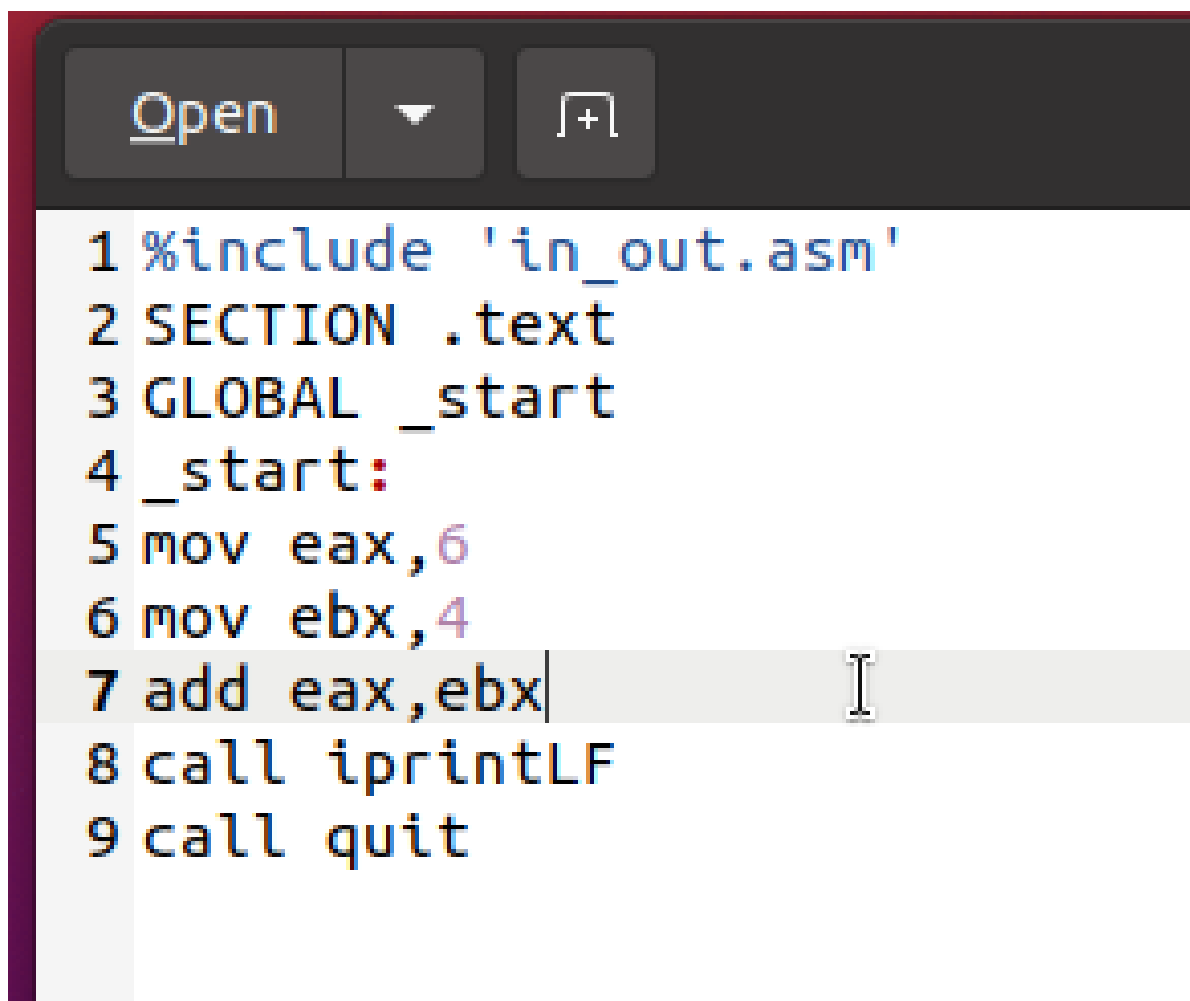


```
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ./lab6-2
106
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$
```

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

В результате работы программы мы получим число 106.(рис. 2.6) 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.

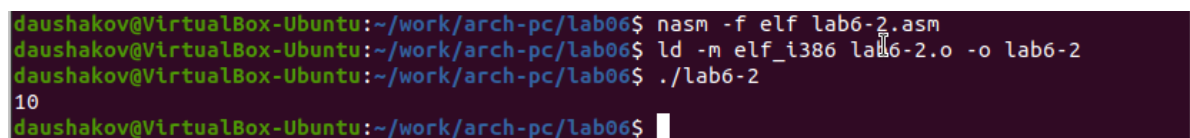
Аналогично предыдущему примеру изменим символы на числа.(рис. 2.7)



```
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: Код программы lab6-2.asm

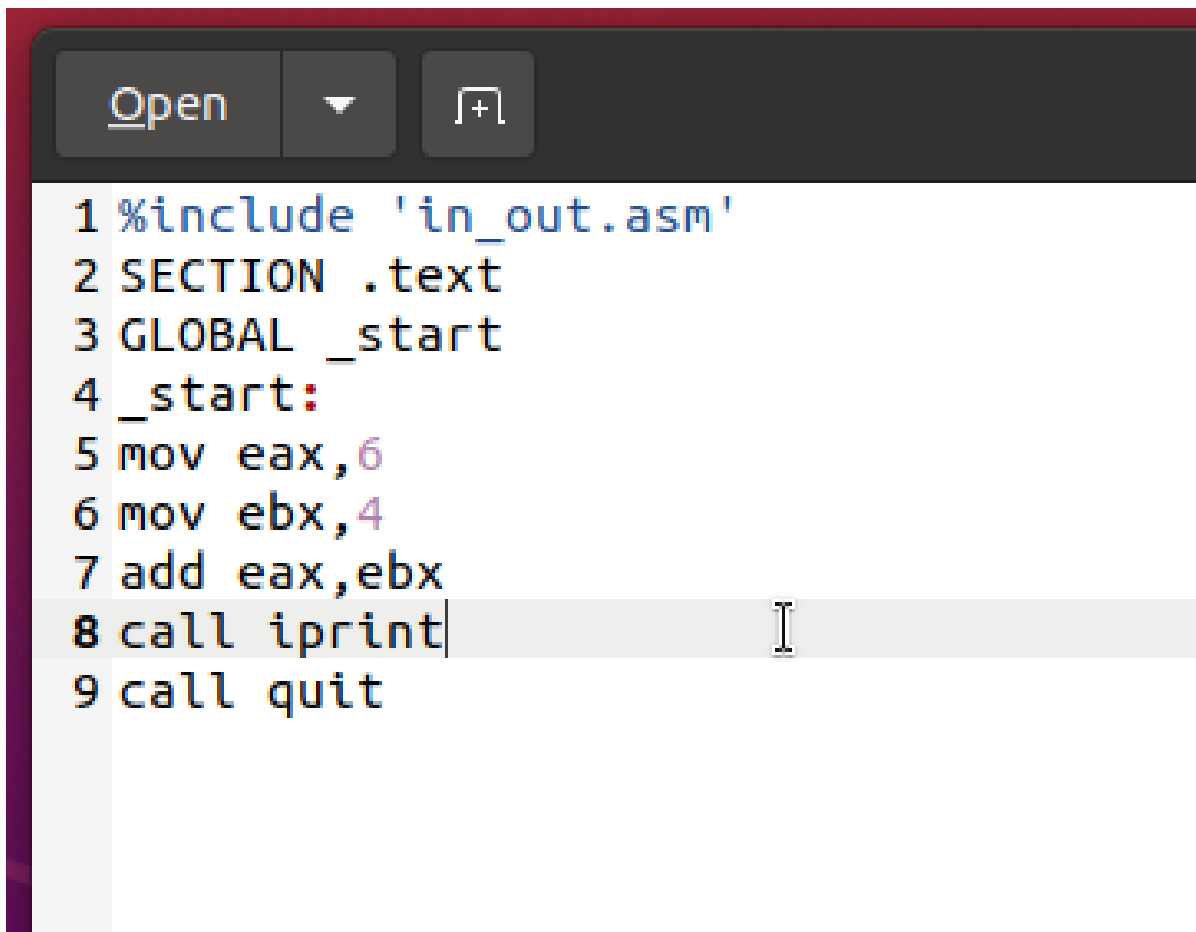
Функция iprintLF позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10. (рис. 2.8)



```
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ./lab6-2
10
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$
```

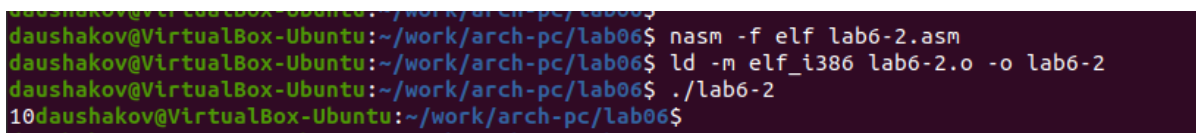
Рис. 2.8: Компиляция и запуск программы lab6-2.asm

Заменяю функцию iprintLF на iprint. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки. (рис. 2.9)



```
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: Код программы lab6-2.asm



```
daushakov@VirtualBox-Ubuntu: ~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
daushakov@VirtualBox-Ubuntu: ~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
daushakov@VirtualBox-Ubuntu: ~/work/arch-pc/lab06$ ./lab6-2
10daushakov@VirtualBox-Ubuntu: ~/work/arch-pc/lab06$
```

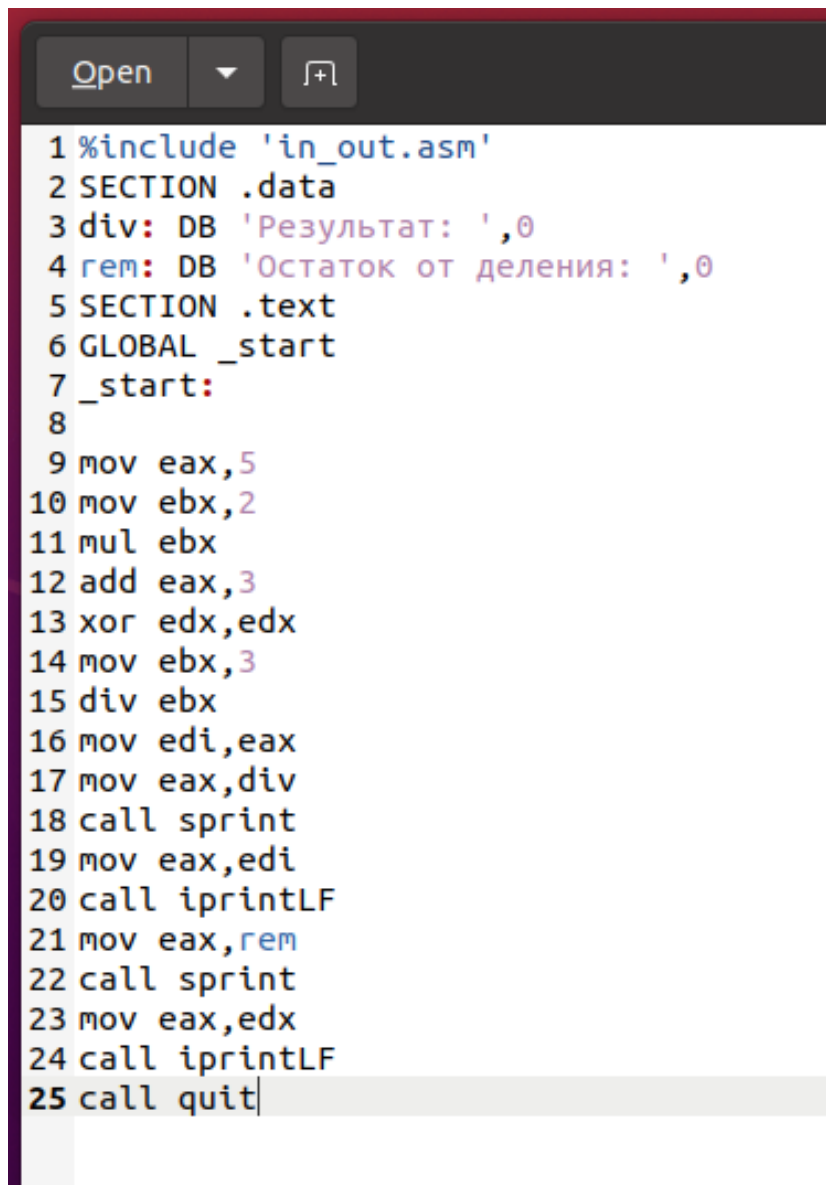
Рис. 2.10: Компиляция и запуск программы lab6-2.asm

2.2 Выполнение арифметических операций в NASM

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

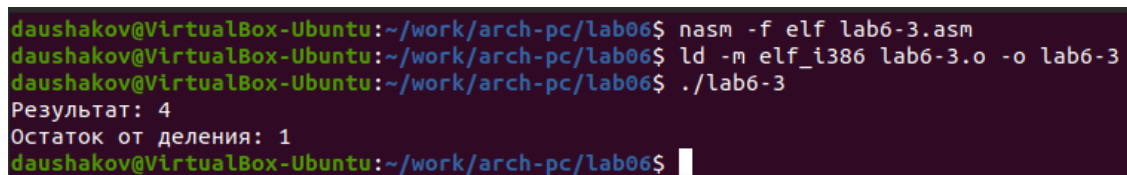
$$f(x) = (5 * 2 + 3) / 3$$

. (рис. 2.11) (рис. 2.12)



```
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: Код программы lab6-3.asm



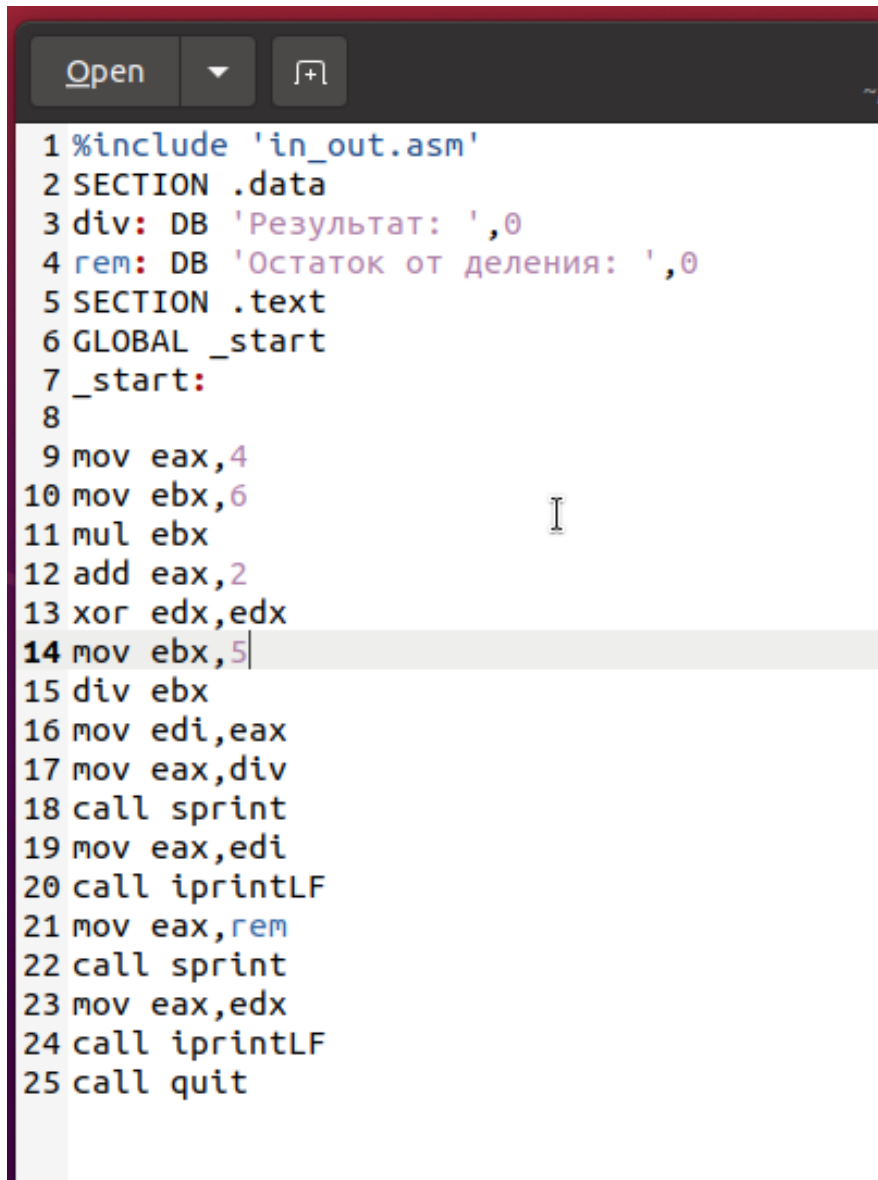
```
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$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

Рис. 2.12: Компиляция и запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения

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

. Создал исполняемый файл и проверил его работу. (рис. 2.13) (рис. 2.14)



```
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
```

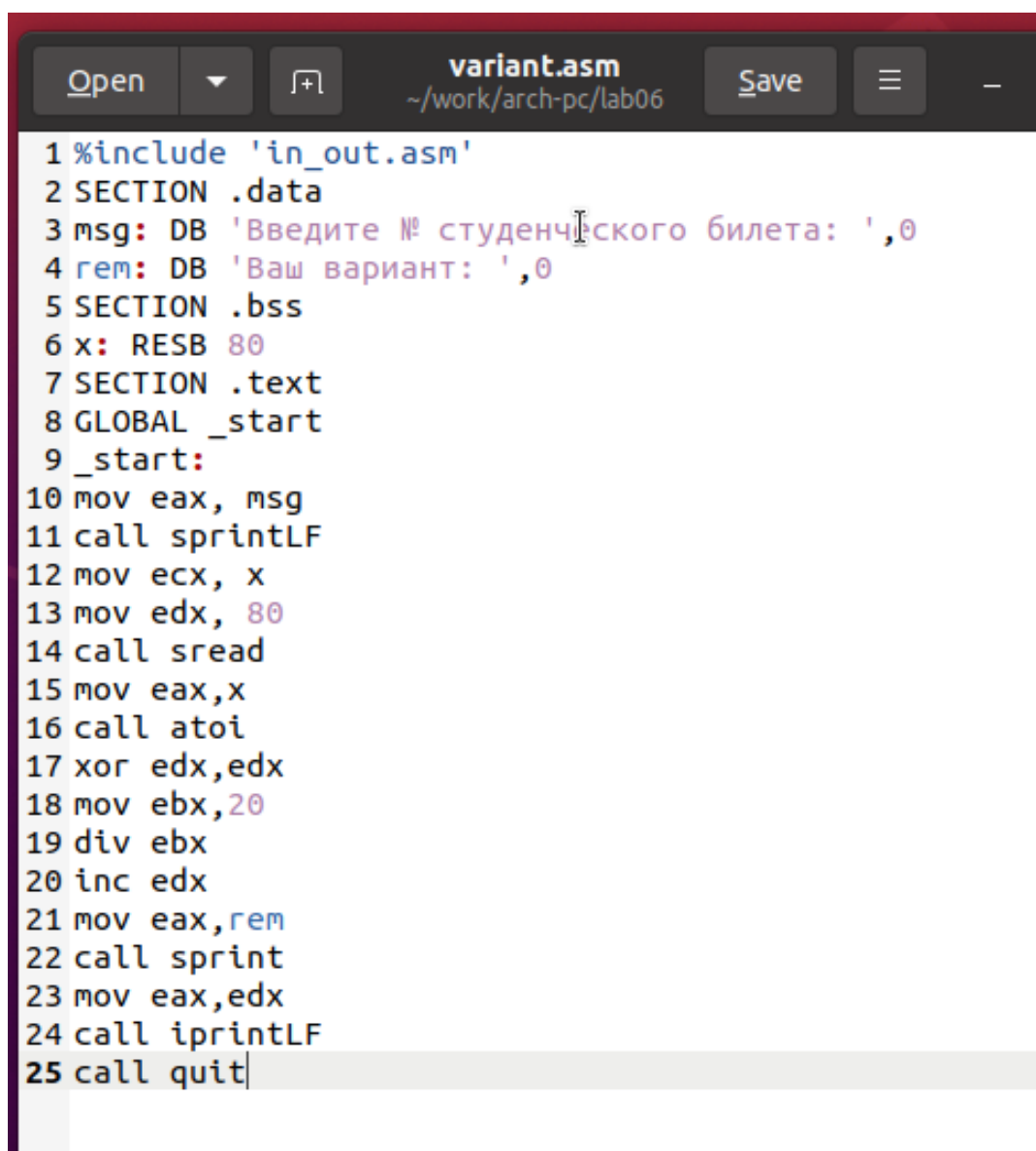
Рис. 2.13: Код программы lab6-3.asm

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

Рис. 2.14: Компиляция и запуск программы lab6-3.asm

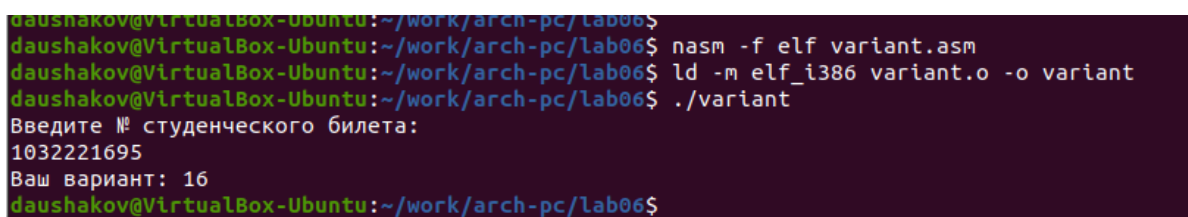
В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета.

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция `atoi` из файла `in_out.asm`. (рис. 2.15) (рис. 2.16)



```
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. 2.15: Код программы variant.asm



```
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf variant.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032221695
Ваш вариант: 16
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.16: Компиляция и запуск программы variant.asm

ответы на вопросы

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

Переменная с фразой “Ваш вариант:” перекладывается в регистр `eax` с помощью строки `mov eax, ptr`.

Для вызова подпрограммы вывода строки используется строка `call sprint`.

2. Для чего используются следующие инструкции?

- `mov ecx, x` - перекладывает регистр `ecx` в переменную
- `mov edx, 80` - устанавливает значение 80 в регистр `edx`.
- `call sread` - вызывает подпрограмму для считывания значения из консоли.

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?

- `xor edx, edx` - обнуляет регистр `edx`.
- `mov ebx, 20` - устанавливает значение 20 в регистр `ebx`.
- `div ebx` - выполняет деление номера студенческого билета на 20.
- `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.

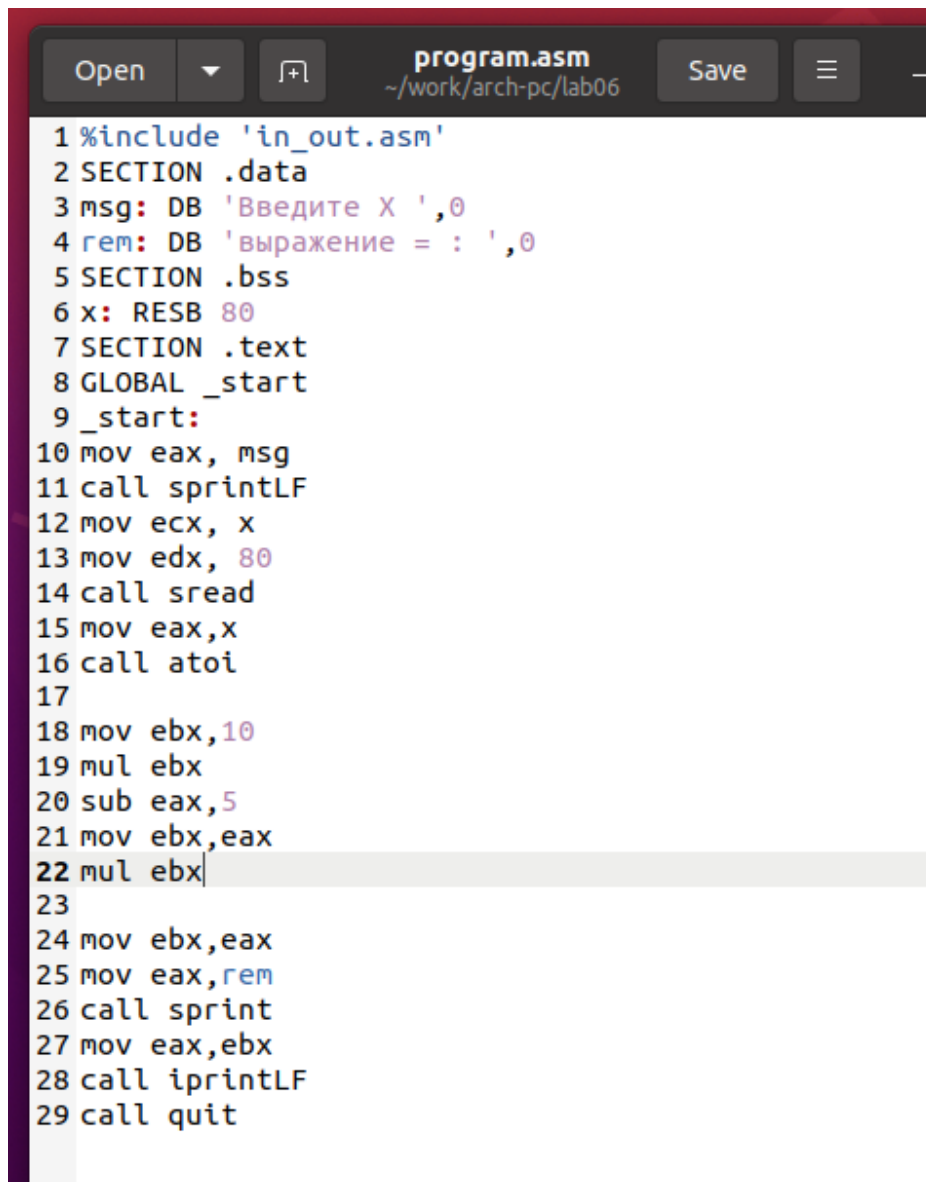
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

- Результат вычислений перекладывается в регистр `eax` с помощью строки `mov eax, edx`.
- Для вызова подпрограммы вывода используется строка `call iprintLF`.

2.3 Задание для самостоятельной работы

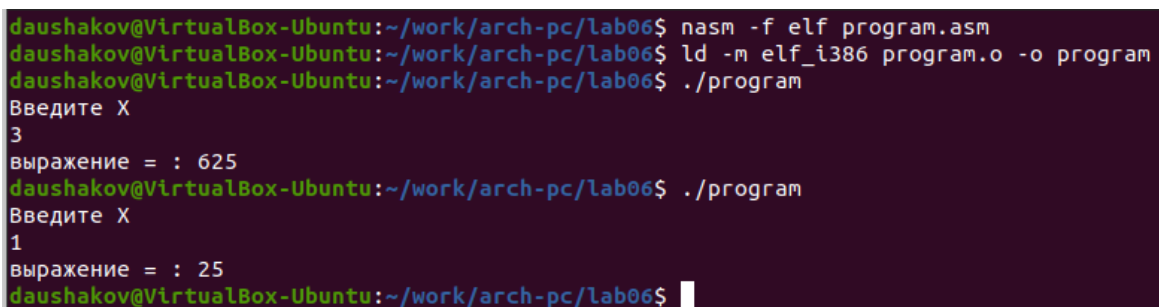
Написать программу вычисления выражения $y = f(x)$. Программа должна вводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. (рис. 2.17) (рис. 2.18) Создайте исполняемый файл и проверьте его работу для значений x_1 и x_2 из 6.3.

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
18 mov ebx, 10
19 mul ebx
20 sub eax, 5
21 mov ebx, eax
22 mul ebx
23
24 mov ebx, eax
25 mov eax, rem
26 call sprintf
27 mov eax, ebx
28 call iprintLF
29 call quit
```

Рис. 2.17: Код программы program.asm



```
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf program.asm
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 program.o -o program
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ./program
Введите X
3
выражение = : 625
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$ ./program
Введите X
1
выражение = : 25
daushakov@VirtualBox-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.18: Компиляция и запуск программы program.asm

Программа считает верно.

3 Выводы

Изучили работу с арифметическими операциями.