

Отчёт по лабораторной работе 6

Архитектура компьютеров и операционные системы

Старикова Владислава Александровна НММбд-03-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Ответы на вопросы по программе variant.asm	17
2.2	Самостоятельное задание	18
3	Выводы	21

Список иллюстраций

2.1	Подготовила каталог	6
2.2	Программа в файле lab6-1.asm	7
2.3	Запуск программы lab6-1.asm	8
2.4	Программа в файле lab6-1.asm	9
2.5	Запуск программы lab6-1.asm	9
2.6	Программа в файле lab6-2.asm	10
2.7	Запуск программы lab6-2.asm	10
2.8	Программа в файле lab6-2.asm	11
2.9	Запуск программы lab6-2.asm	12
2.10	Запуск программы lab6-2.asm	12
2.11	Программа в файле lab6-3.asm	13
2.12	Запуск программы lab6-3.asm	13
2.13	Программа в файле lab6-3.asm	14
2.14	Запуск программы lab6-3.asm	15
2.15	Программа в файле variant.asm	16
2.16	Запуск программы variant.asm	17
2.17	Программа в файле task.asm	19
2.18	Запуск программы task.asm	20

Список таблиц

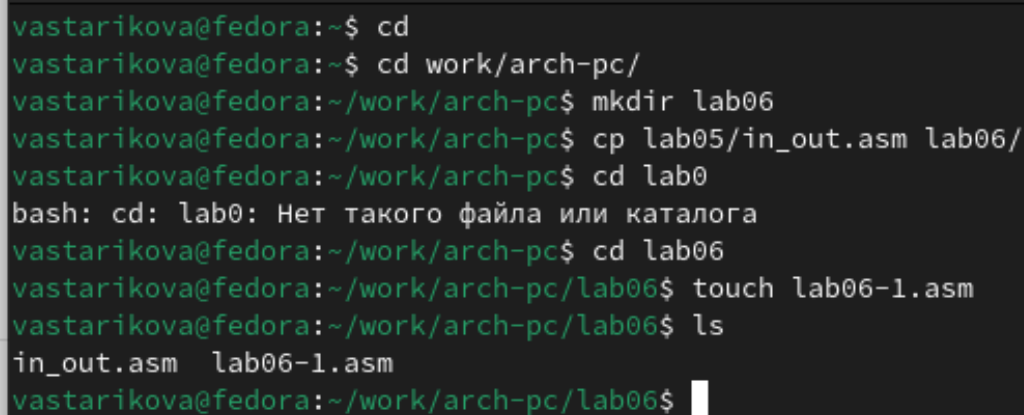
1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

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

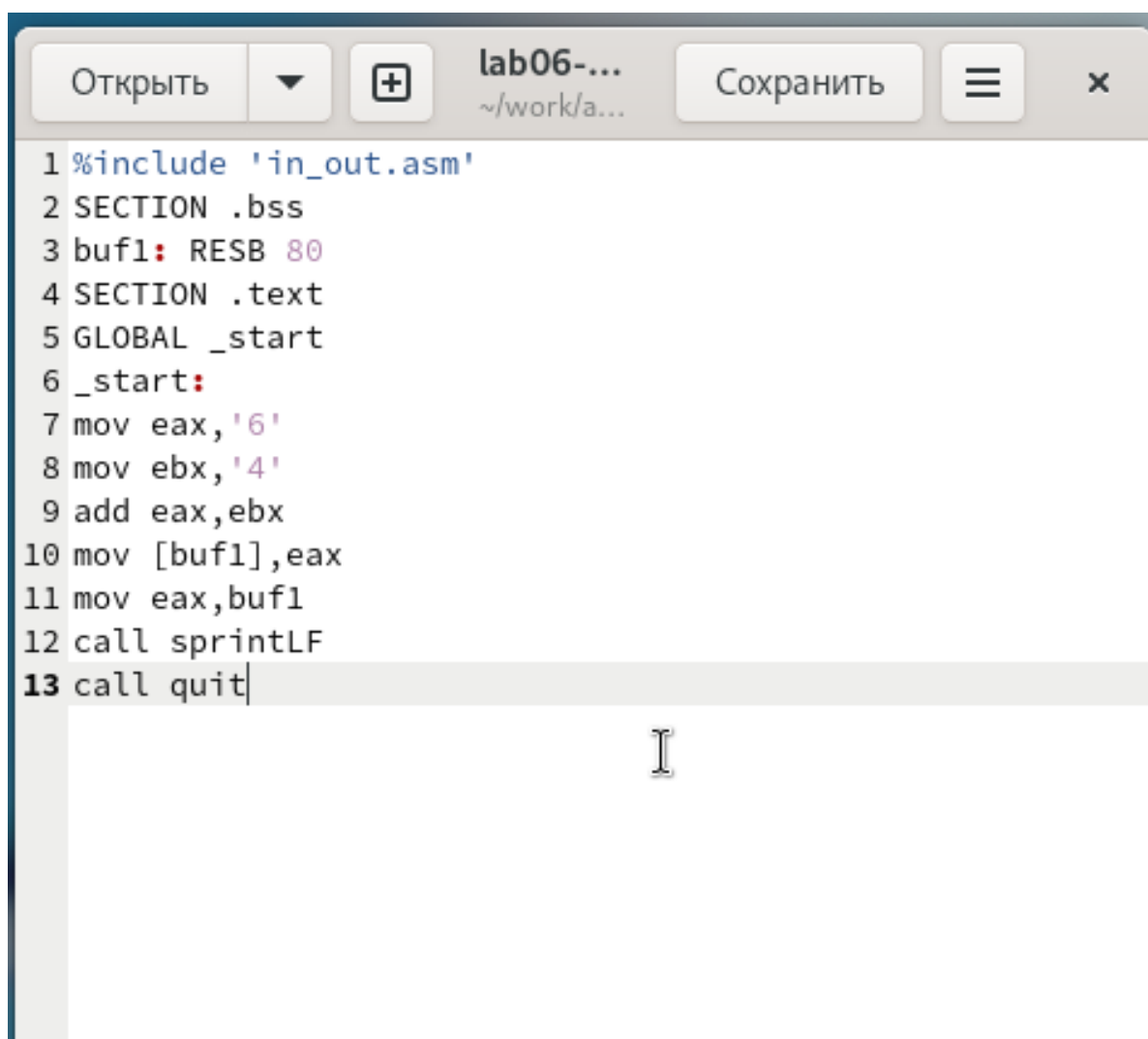
(рис. 2.1)



```
vastarikova@fedora:~$ cd
vastarikova@fedora:~$ cd work/arch-pc/
vastarikova@fedora:~/work/arch-pc$ mkdir lab06
vastarikova@fedora:~/work/arch-pc$ cp lab05/in_out.asm lab06/
vastarikova@fedora:~/work/arch-pc$ cd lab0
bash: cd: lab0: Нет такого файла или каталога
vastarikova@fedora:~/work/arch-pc$ cd lab06
vastarikova@fedora:~/work/arch-pc/lab06$ touch lab06-1.asm
vastarikova@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab06-1.asm
vastarikova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.1: Подготовила каталог

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистры `eax`.



```
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 quit
```

Рис. 2.2: Программа в файле lab6-1.asm

В данной программе (рис. 2.2) я записываю символ '6' в регистр `eax` (`mov eax, '6'`), а символ '4' в регистр `ebx` (`mov ebx, '4'`).

Затем я добавляю значение регистра `ebx` к значению в регистре `eax` (`add eax, ebx`, результат сложения записывается в регистр `eax`).

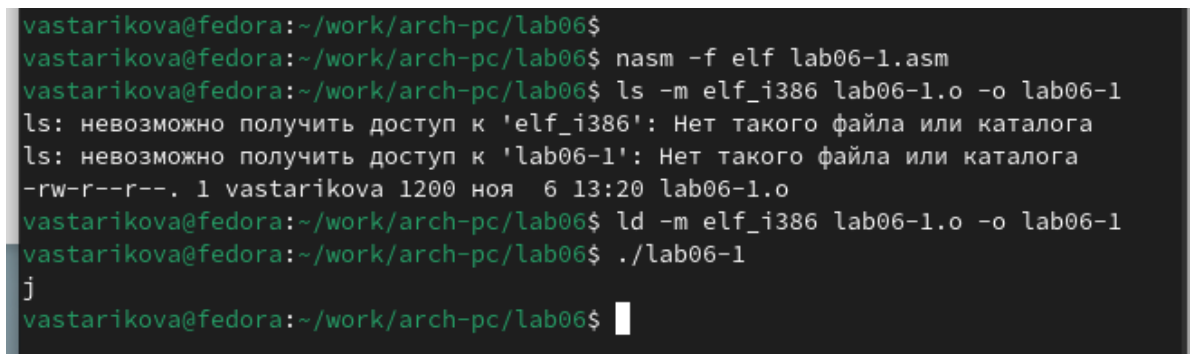
После этого я вывожу результат.

Однако, для использования функции `sprintf`, необходимо, чтобы в регистре `eax` был записан адрес, поэтому я использую дополнительную переменную.

Я записываю значение регистра `eax` в переменную `buf1` (`mov [buf1], eax`),

а затем записываю адрес переменной `buf1` в регистр `eax` (`mov eax, buf1`) и вызываю

функцию `sprintLF`.



```
vastarikova@fedora:~/work/arch-pc/lab06$  
vastarikova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm  
vastarikova@fedora:~/work/arch-pc/lab06$ ls -m elf_i386 lab06-1.o -o lab06-1  
ls: невозможно получить доступ к 'elf_i386': Нет такого файла или каталога  
ls: невозможно получить доступ к 'lab06-1': Нет такого файла или каталога  
-rw-r--r--. 1 vastarikova 1200 ноя  6 13:20 lab06-1.o  
vastarikova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1  
vastarikova@fedora:~/work/arch-pc/lab06$ ./lab06-1  
j  
vastarikova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.3: Запуск программы `lab6-1.asm`

В данном случае, когда мы ожидаем увидеть число 10 при выводе значения регистра `eax`, фактическим результатом будет символ ‘j’.

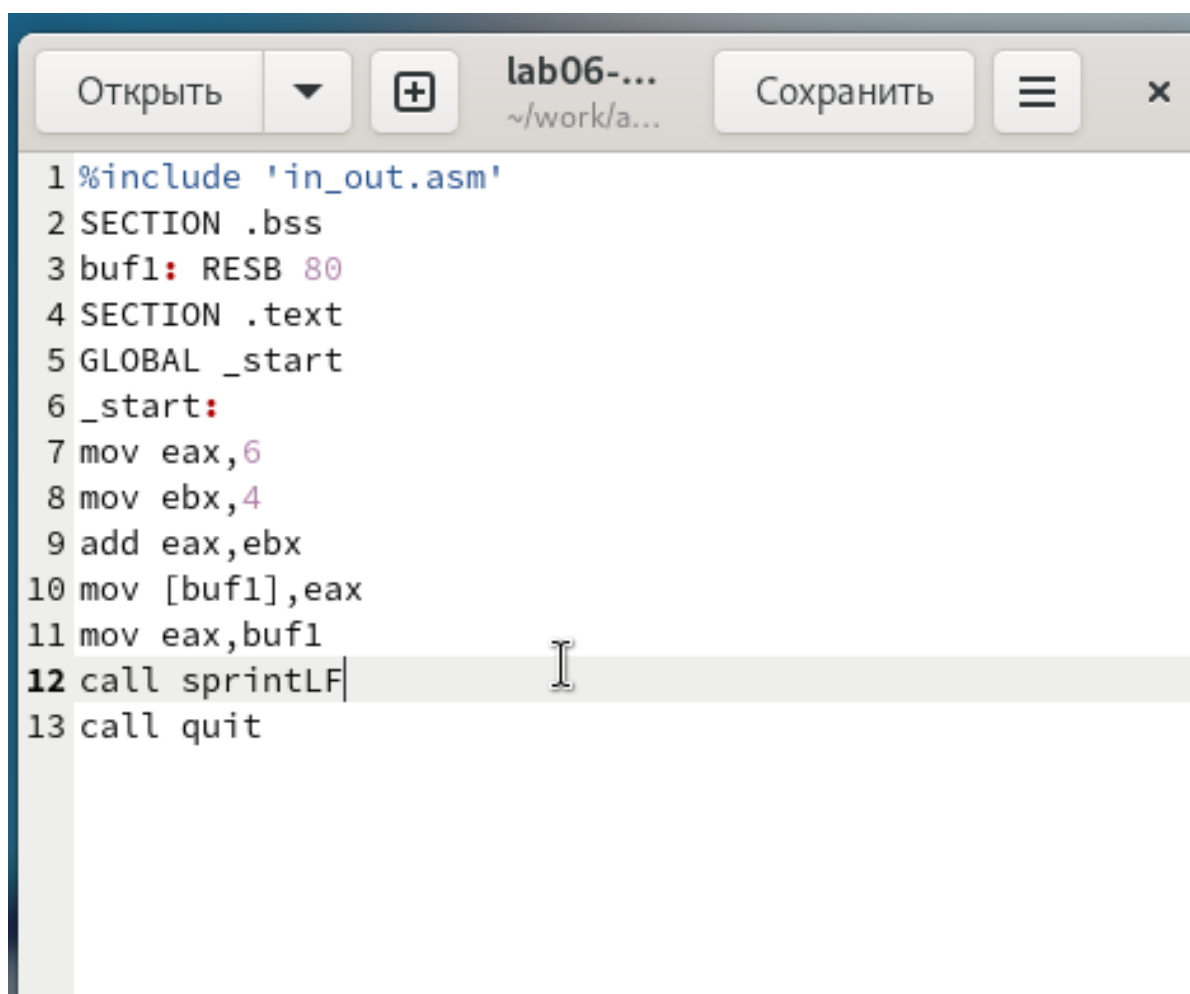
Это происходит из-за того, что код символа ‘6’ равен 00110110 в двоичном представлении (или 54 в десятичном представлении),

а код символа ‘4’ равен 00110100 (или 52 в десятичном представлении).

Когда я выполняю команду `add eax, ebx`, результатом будет сумма кодов - 01101010 (или 106 в десятичном представлении),

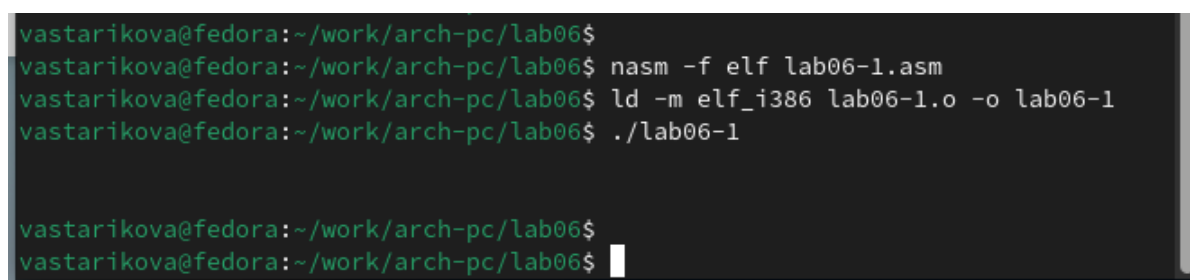
который соответствует символу ‘j’. (рис. 2.3)

Далее я изменила текст программы и вместо символов записала в регистры числа. (рис. 2.4)



```
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. 2.4: Программа в файле lab6-1.asm



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

vastarikova@fedora:~/work/arch-pc/lab06$
vastarikova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.5: Запуск программы lab6-1.asm

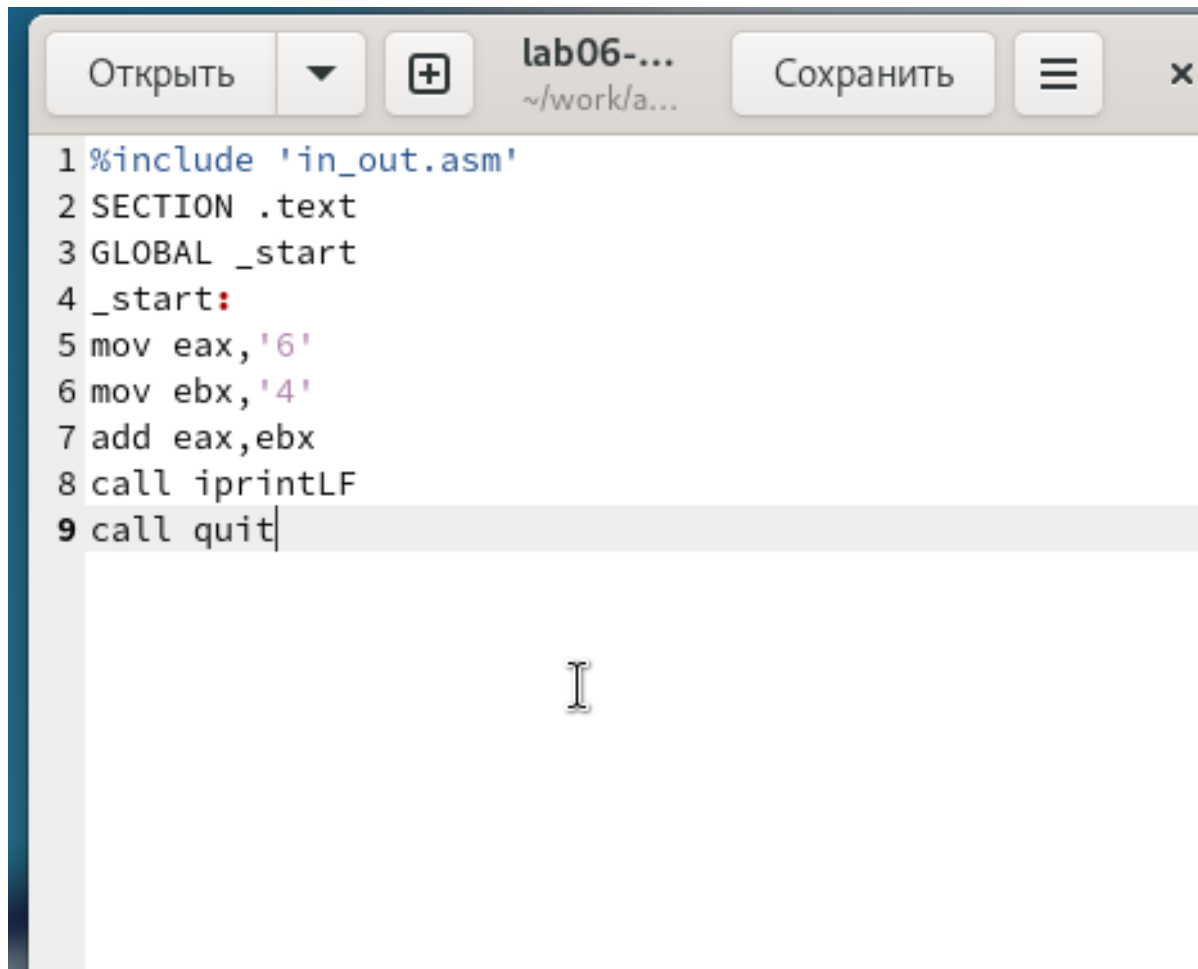
Как и в предыдущем случае, при выполнении программы мы не получим число 10. Вместо этого выводится символ с кодом 10, который представляет собой символ

конца строки (возврат каретки).

Этот символ не отображается в консоли, но он добавляет пустую строку.

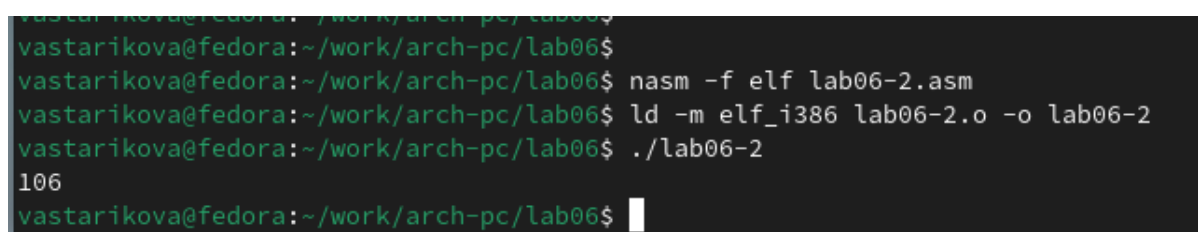
Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно.

Я преобразовала текст программы с использованием этих функций. (рис. 2.6)



```
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. 2.6: Программа в файле `lab6-2.asm`



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

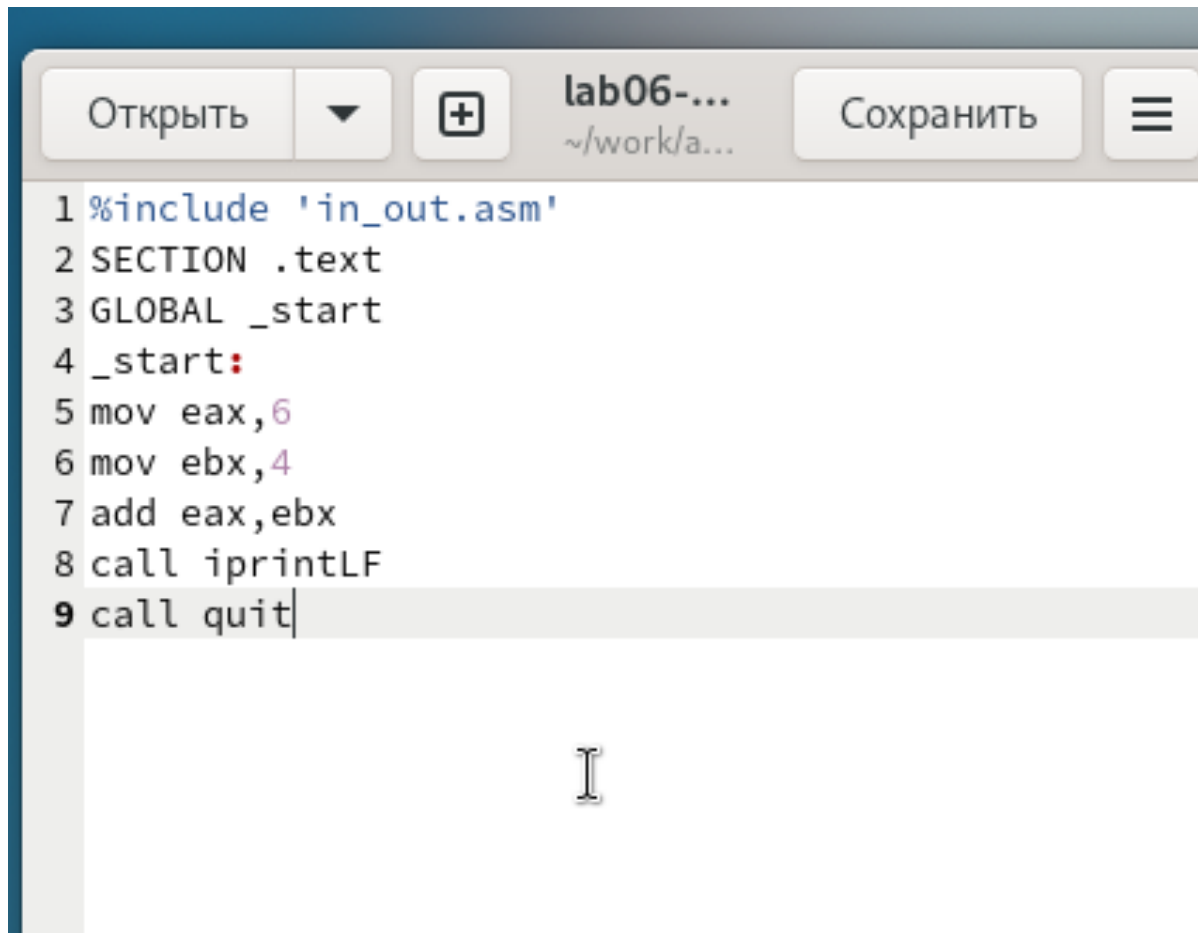
Рис. 2.7: Запуск программы `lab6-2.asm`

В результате выполнения программы я получаю число 106. (рис. 2.7)

В данном случае, как и в первом примере, 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, я изменила символы на числа. (рис. 2.8)



```
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. 2.8: Программа в файле lab6-2.asm

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

Поэтому получаем число 10. (рис. 2.9)

```

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

```

Рис. 2.9: Запуск программы lab6-2.asm

Я заменила функцию `iprintLF` на `iprint`. Создала исполняемый файл и запустила его.

Вывод отличается тем, что нет переноса строки. (рис. 2.10)

```

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

```

Рис. 2.10: Запуск программы lab6-2.asm

В качестве примера выполнения арифметических операций в NASM привожу программу вычисления арифметического выражения (рис. 2.11) (рис. 2.12)

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

```
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 в файле lab6-3.asm

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

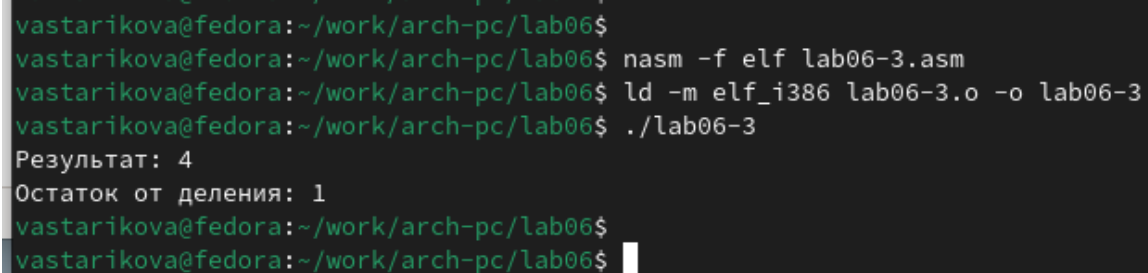
Рис. 2.12: Запуск программы lab6-3.asm

Я изменила текст программы для вычисления выражения

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

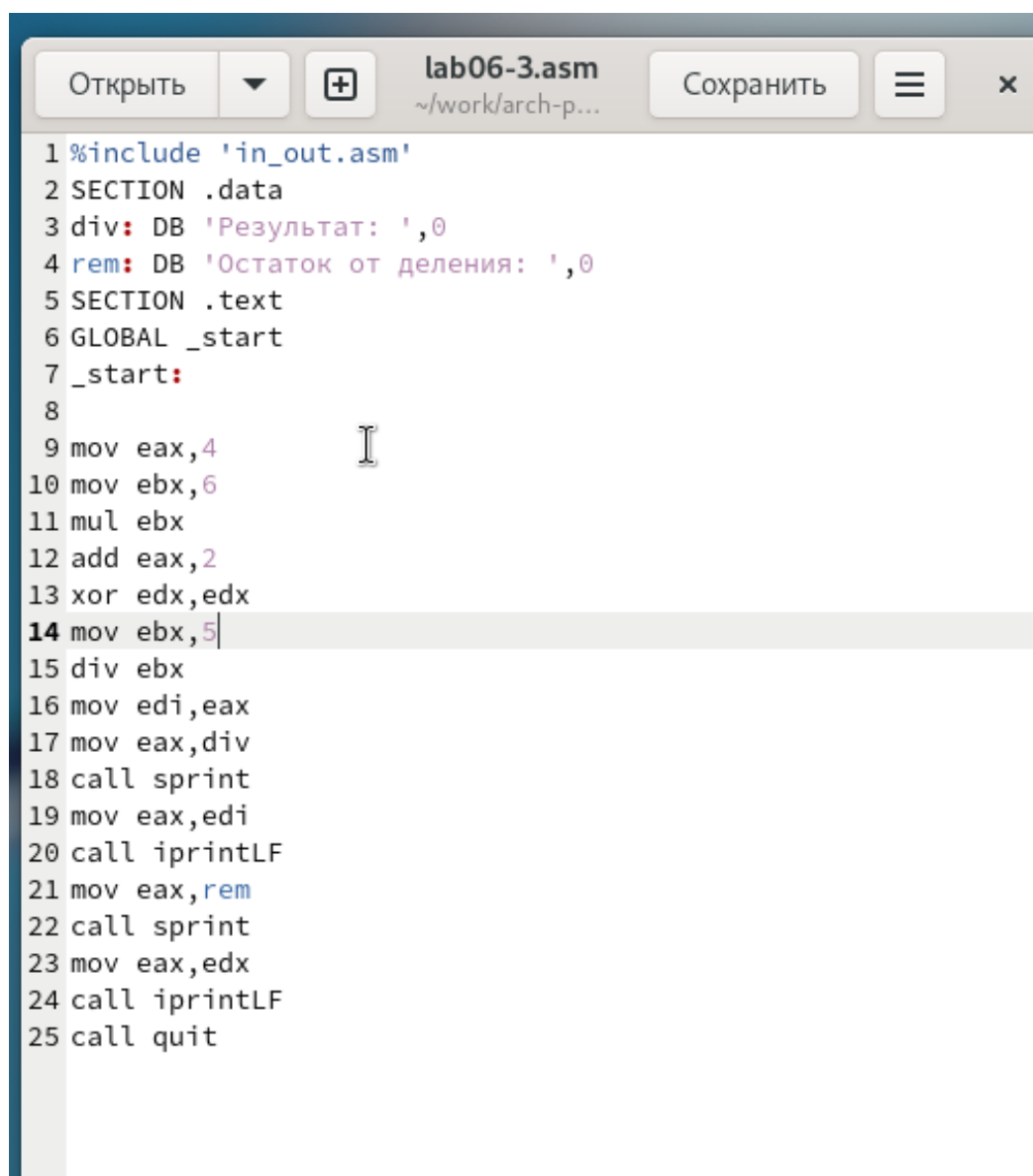
.

Создала исполняемый файл и проверила его работу. (рис. 2.13) (рис. 2.14)



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

Рис. 2.13: Программа в файле lab6-3.asm



```
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
```

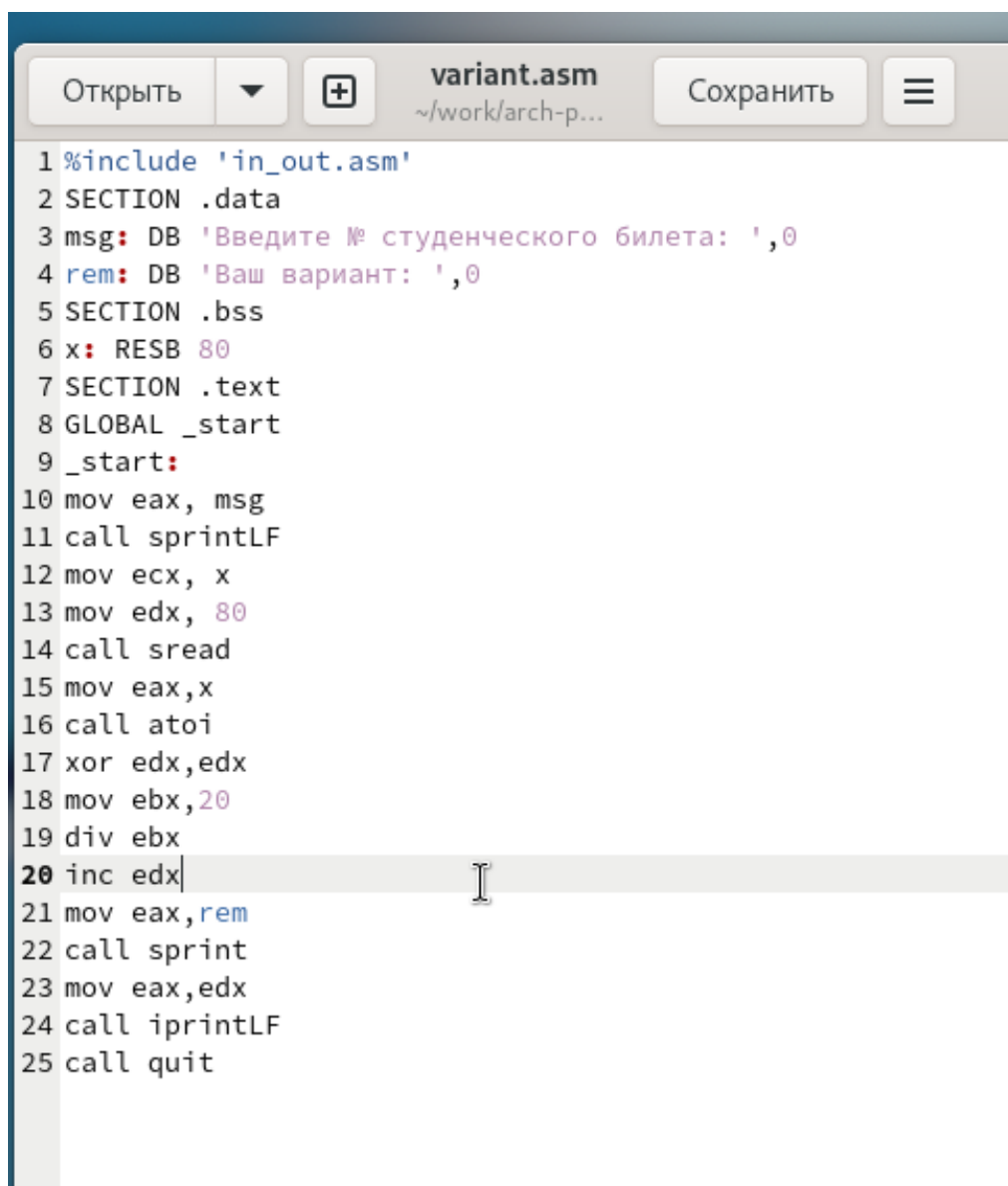
Рис. 2.14: Запуск программы lab6-3.asm

В качестве другого примера рассматриваю программу вычисления варианта задания по номеру студенческого билета. (рис. 2.15) (рис. 2.16)

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`.



```
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
```

Рис. 2.15: Программа в файле `variant.asm`


```
vastarikova@fedora:~/work/arch-pc/lab06$  
vastarikova@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm  
vastarikova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant  
vastarikova@fedora:~/work/arch-pc/lab06$ ./variant  
Введите № студенческого билета:  
1132246796  
Ваш вариант: 17  
vastarikova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.16: Запуск программы variant.asm

2.1 Ответы на вопросы по программе variant.asm

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

Строка “mov eax, mem” перекладывает в регистр значение переменной с фразой “Ваш вариант:”.

Строка “call sprint” вызывает подпрограмму вывода строки.

2. **Для чего используются следующие инструкции?**

Инструкция “nasm” используется для компиляции кода на языке ассемблера NASM.

Инструкция “mov esx, x” используется для перемещения значения переменной x в регистр esx.

Инструкция “mov edx, 80” используется для перемещения значения 80 в регистр edx.

Инструкция “call sread” вызывает подпрограмму для считывания значения студенческого билета 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?**

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, в соответствии с формулой 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.

2.2 Самостоятельное задание

Я написала 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 $f(x)$ выбрала из таблицы 6.3 вариантов заданий в соответствии с номером, полученным при выполнении лабораторной работы.

Создала исполняемый файл и проверила его работу для значений x_1 и x_2 из 6.3.

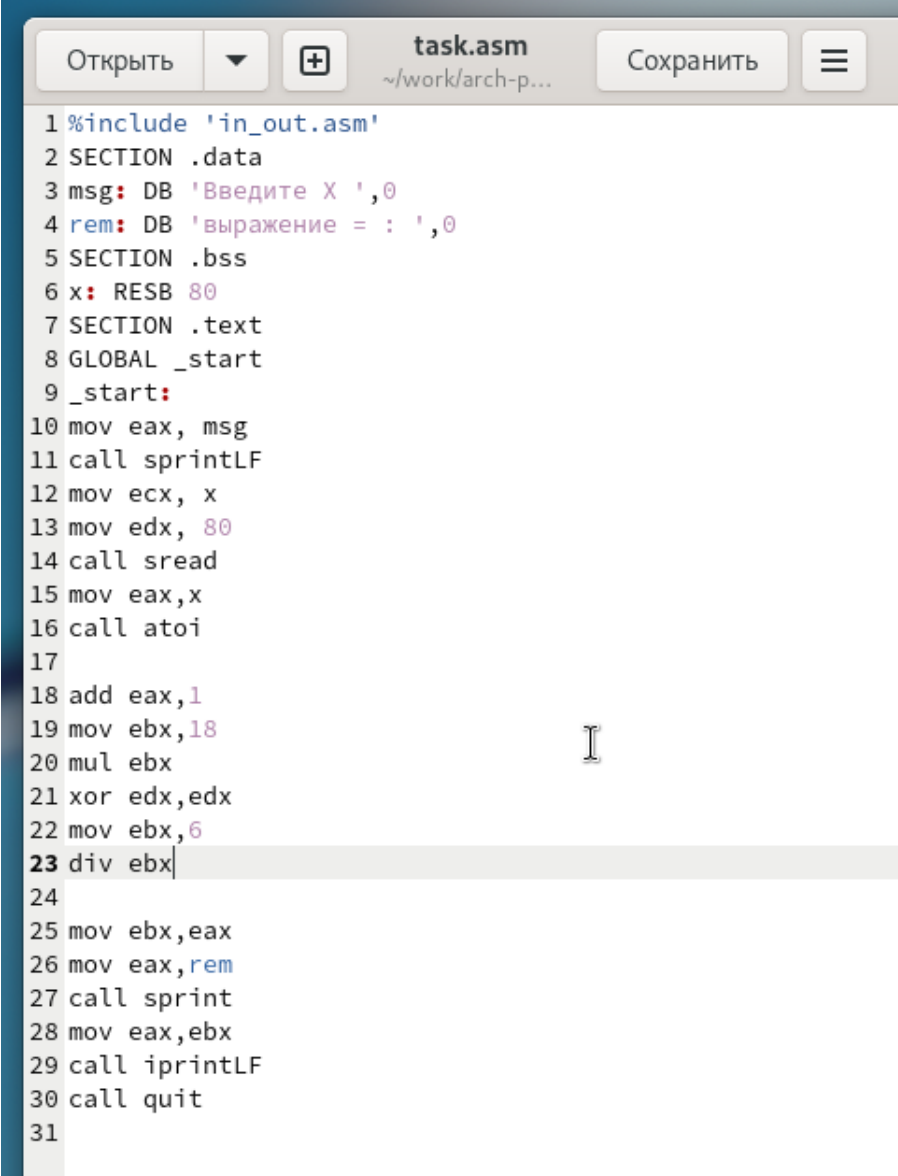
Получили вариант 17 -

$$18(x + 1)/6$$

для

$$x_1 = 1, x_2 = 5$$

(рис. 2.17) (рис. 2.18)



```
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 add eax, 1
19 mov ebx, 18
20 mul ebx
21 xor edx, edx
22 mov ebx, 6
23 div ebx
24
25 mov ebx, eax
26 mov eax, rem
27 call sprintf
28 mov eax, ebx
29 call iprintLF
30 call quit
31
```

Рис. 2.17: Программа в файле task.asm

```
vastarikova@fedora:~/work/arch-pc/lab06$  
vastarikova@fedora:~/work/arch-pc/lab06$ nasm -f elf task.asm  
vastarikova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 task.o -o task  
vastarikova@fedora:~/work/arch-pc/lab06$ ./task  
Введите X  
1  
выражение = : 6  
vastarikova@fedora:~/work/arch-pc/lab06$ ./task  
Введите X  
4  
выражение = : 15  
vastarikova@fedora:~/work/arch-pc/lab06$ ./task  
Введите X  
5  
выражение = : 18  
vastarikova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.18: Запуск программы task.asm

3 Выводы

Изучили работу с арифметическими операциями.