

Отчёт по лабораторной работе 6

дисциплина: Архитектура компьютера

Кайнова Екатерина Андреевна НПИбд-03-24

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	11
2.3	Ответы на вопросы	15
2.4	Задание для самостоятельной работы	16
3	Выводы	19

Список иллюстраций

2.1	Программа lab6-1.asm	7
2.2	Запуск программы lab6-1.asm	7
2.3	Программа lab6-1.asm с числами	8
2.4	Запуск программы lab6-1.asm с числами	9
2.5	Программа lab6-2.asm	9
2.6	Запуск программы lab6-2.asm	10
2.7	Программа lab6-2.asm с числами	10
2.8	Запуск программы lab6-2.asm с числами	11
2.9	Запуск программы lab6-2.asm без переноса строки	11
2.10	Программа lab6-3.asm	12
2.11	Запуск программы lab6-3.asm	12
2.12	Программа lab6-3.asm с другим выражением	13
2.13	Запуск программы lab6-3.asm с другим выражением	13
2.14	Программа variant.asm	14
2.15	Запуск программы variant.asm	15
2.16	Программа task.asm	17
2.17	Запуск программы task.asm	18

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

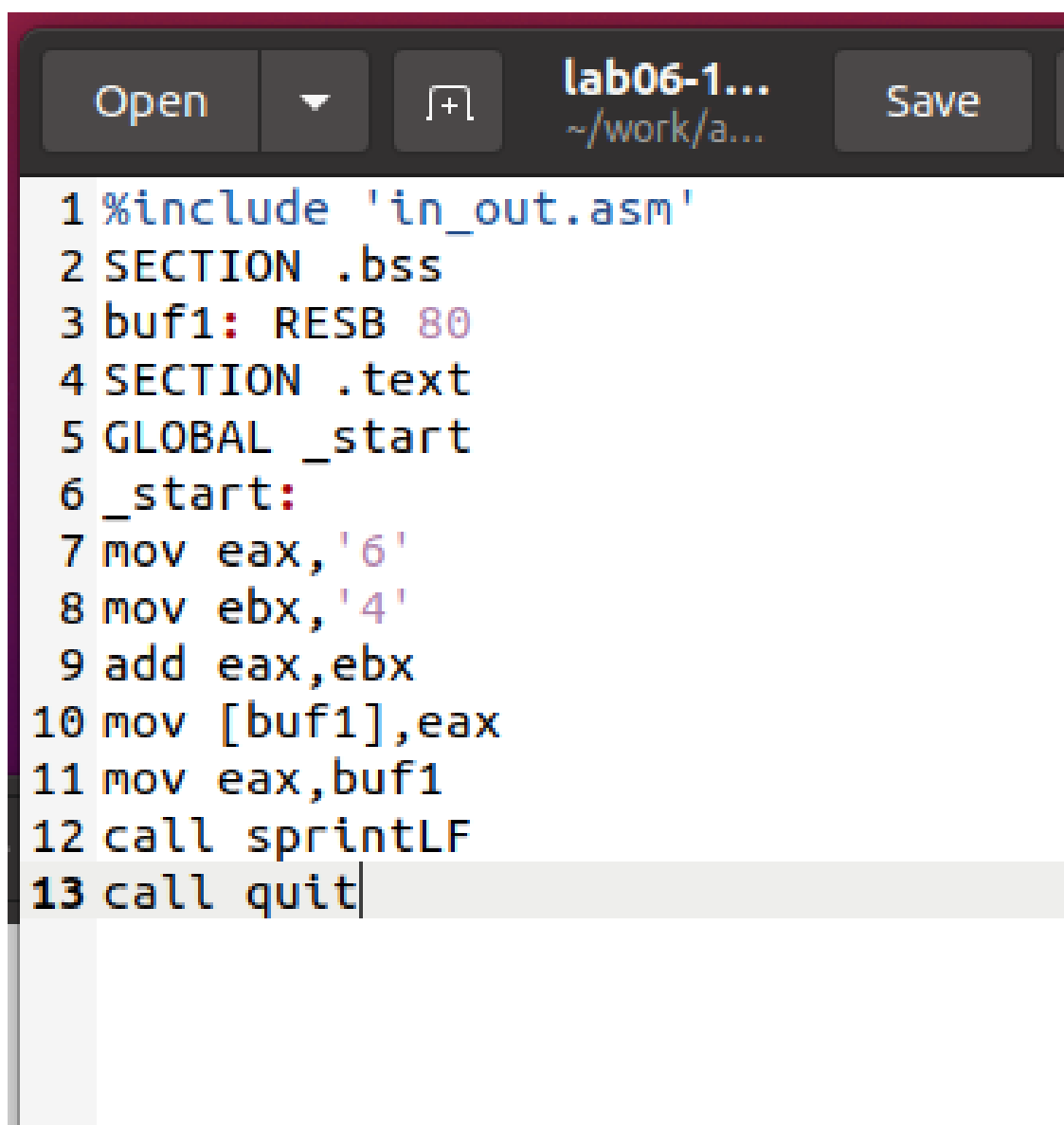
2.1 Символьные и численные данные в NASM

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

Программы, которые будут приведены далее, демонстрируют вывод символьных и численных значений, записанных в регистр еах.

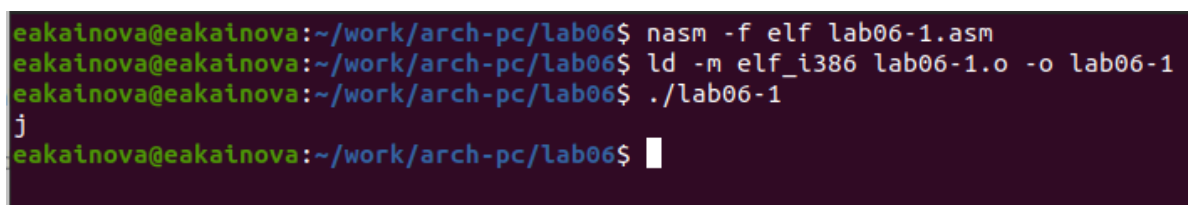
В первой программе в регистр еах записывается символ 6 с помощью команды `mov еах, '6'`, а в регистр ебх – символ 4 с помощью команды `mov ебх, '4'`. Далее, к значению в регистре еах прибавляется значение регистра ебх командой `add еах, ебх`. Результат сложения записывается в регистр еах. После этого выводится результат.

Так как функция `sprintf` требует, чтобы в регистр еах был записан адрес, используется дополнительная переменная. Для этого записываю значение из регистра еах в переменную `buf1` командой `mov [buf1], еах`, затем записываю адрес переменной `buf1` в регистр еах командой `mov еах, buf1` и вызываю функцию `sprintf`.



```
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. 2.1: Программа lab6-1.asm



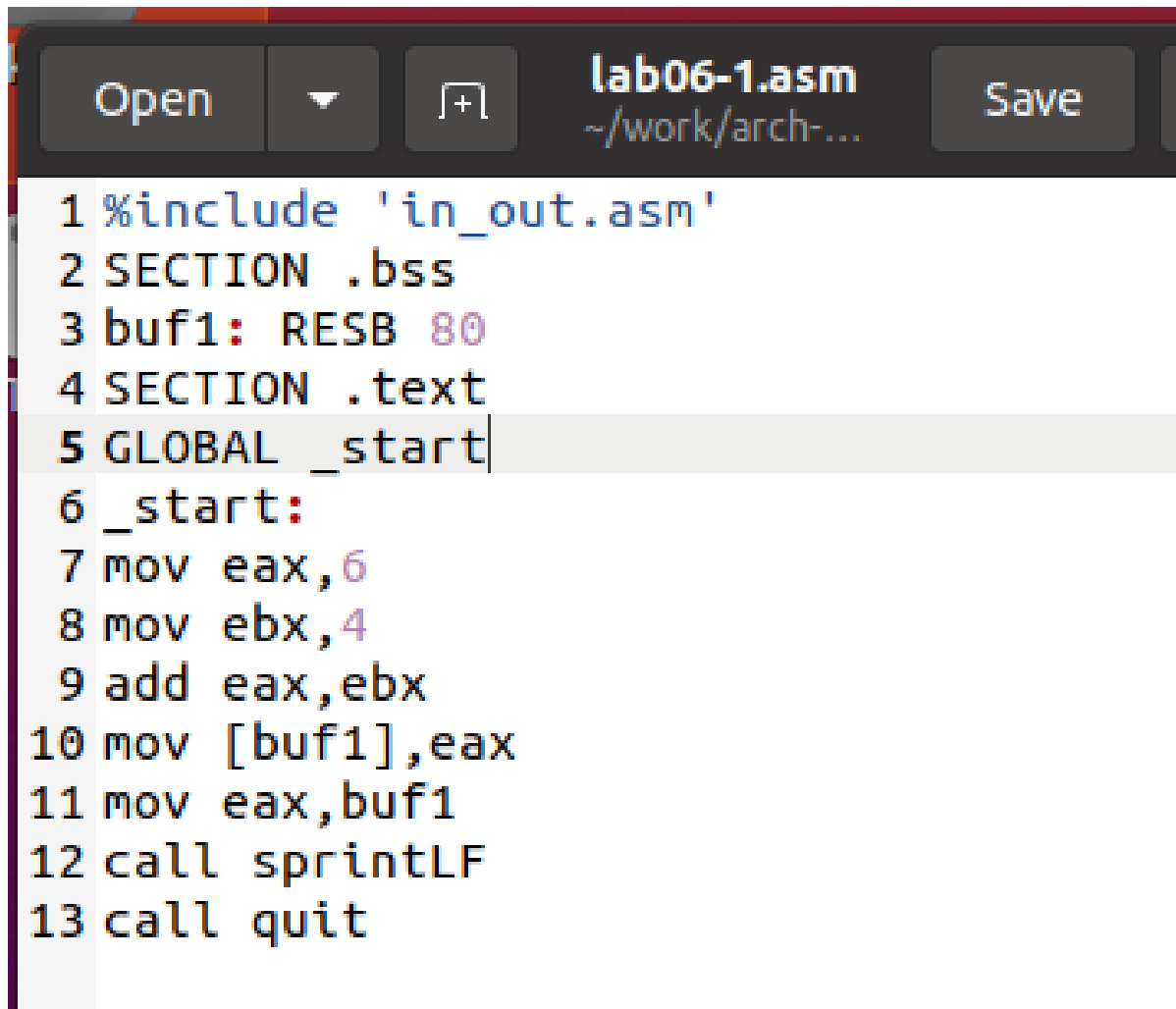
```
eakainova@eakainova:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
eakainova@eakainova:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
eakainova@eakainova:~/work/arch-pc/lab06$ ./lab06-1
j
eakainova@eakainova:~/work/arch-pc/lab06$
```

Рис. 2.2: Запуск программы lab6-1.asm

При выводе значения из регистра `eax` мы ожидаем увидеть число 10. Однако

результатом будет символ j. Это происходит потому, что код символа 6 равен 54 в десятичной системе (или 00110110 в двоичной), а код символа 4 – 52 в десятичной системе (или 00110100 в двоичной). После выполнения команды add eax, ebx, в регистр eax записывается сумма кодов, равная 106, что соответствует символу j.

Далее, в программе вместо символов записываю 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: Программа lab6-1.asm с числами

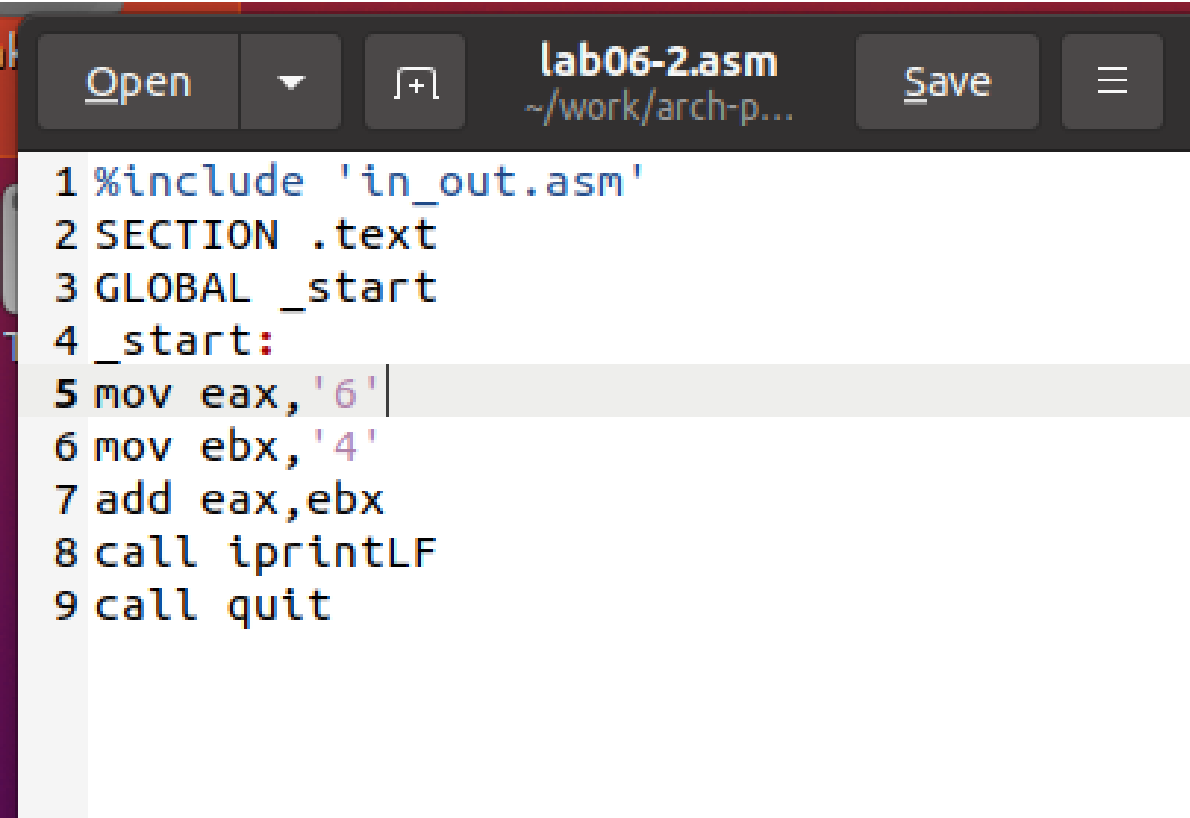

```
eakainova@eakainova:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
eakainova@eakainova:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
eakainova@eakainova:~/work/arch-pc/lab06$ ./lab06-1

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

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

Как и в предыдущем примере, выводится не число 10, а символ с кодом 10, который представляет собой символ конца строки. Этот символ не отображается в консоли, но добавляет пустую строку.

Для работы с числами в файле in_out.asm реализованы функции для преобразования символов ASCII в числа и наоборот. Преобразую программу с использованием этих функций.



```
lab06-2.asm
~/work/arch-p...

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
```

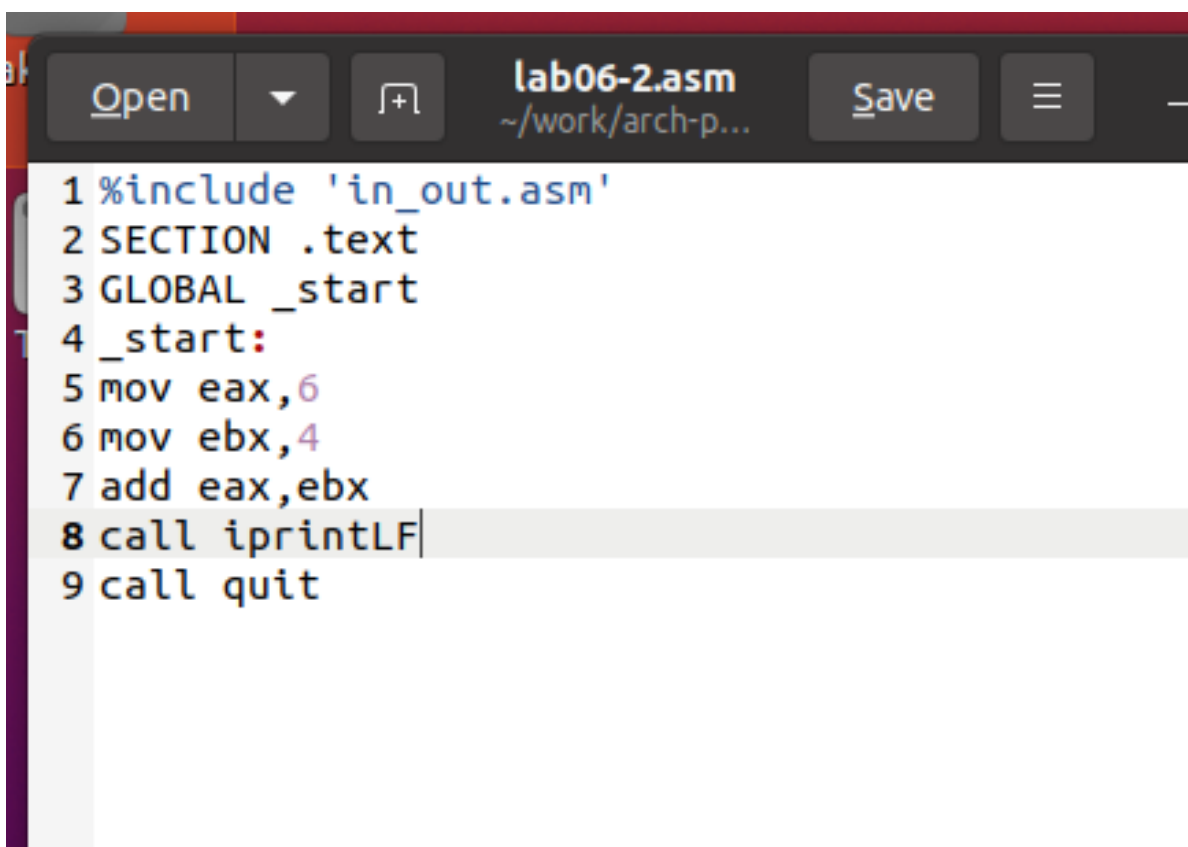
Рис. 2.5: Программа lab6-2.asm

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

Рис. 2.6: Запуск программы lab6-2.asm

Результатом работы программы будет число 106. Здесь, 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 именно число, а не символ, соответствующий данному коду.

Заменяю 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 с числами

В данном случае, благодаря функции `iprintLF`, выводится число 10, так как операндами являются числа.

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

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

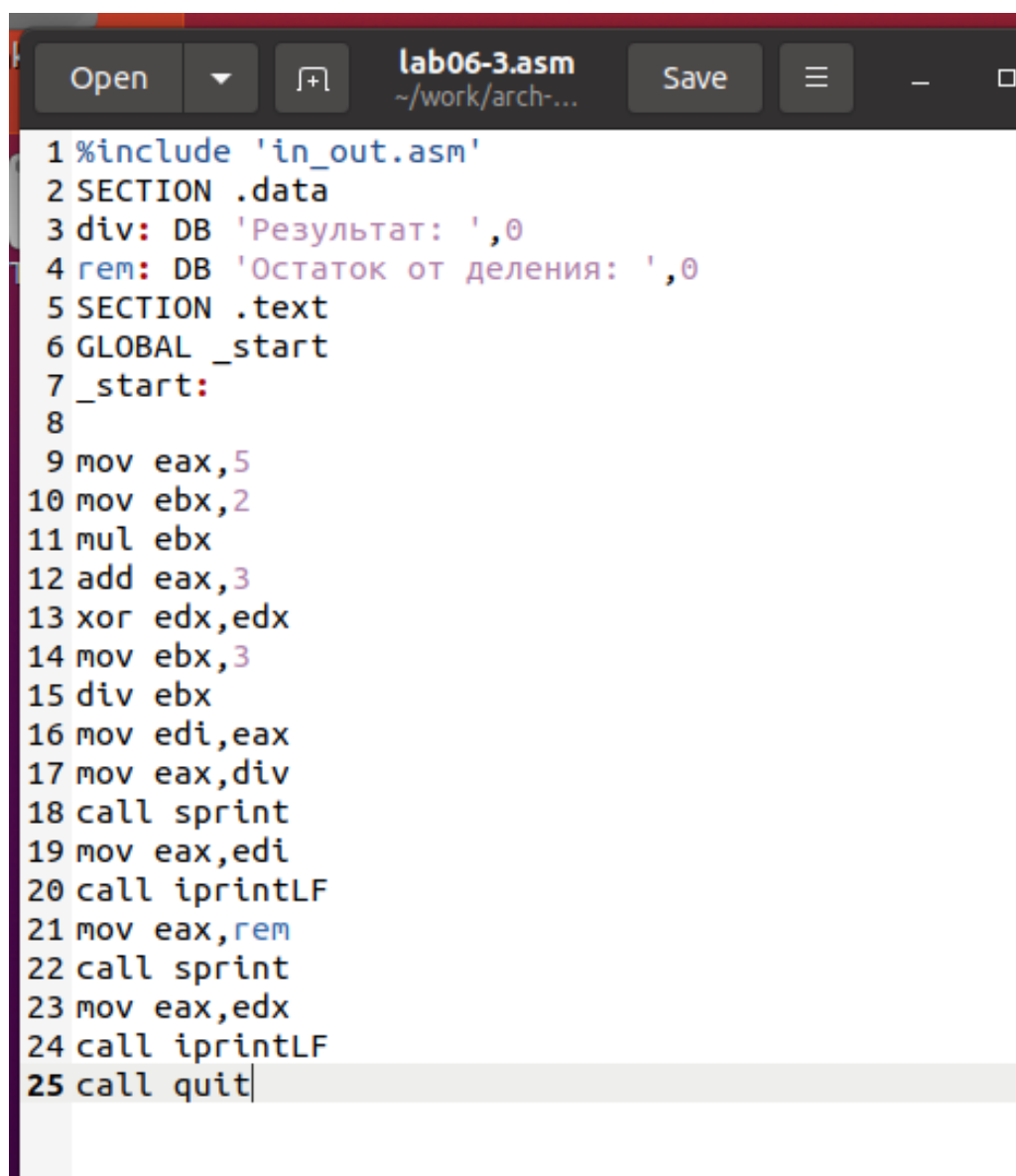
Заменяю функцию `iprintLF` на `iprint` и создаю исполняемый файл, затем запускаю программу. Вывод отличается тем, что теперь нет переноса строки.

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

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

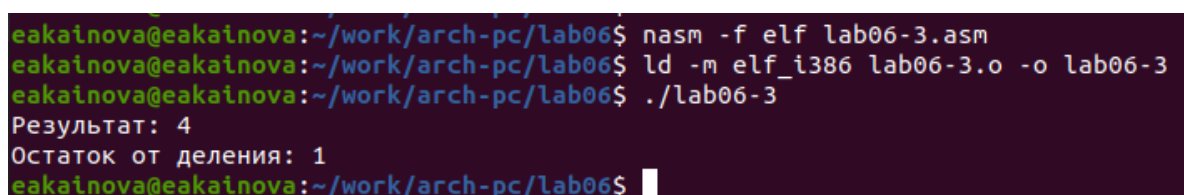
2.2 Выполнение арифметических операций в NASM

Примером арифметических операций в NASM будет программа для вычисления 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: Программа lab6-3.asm

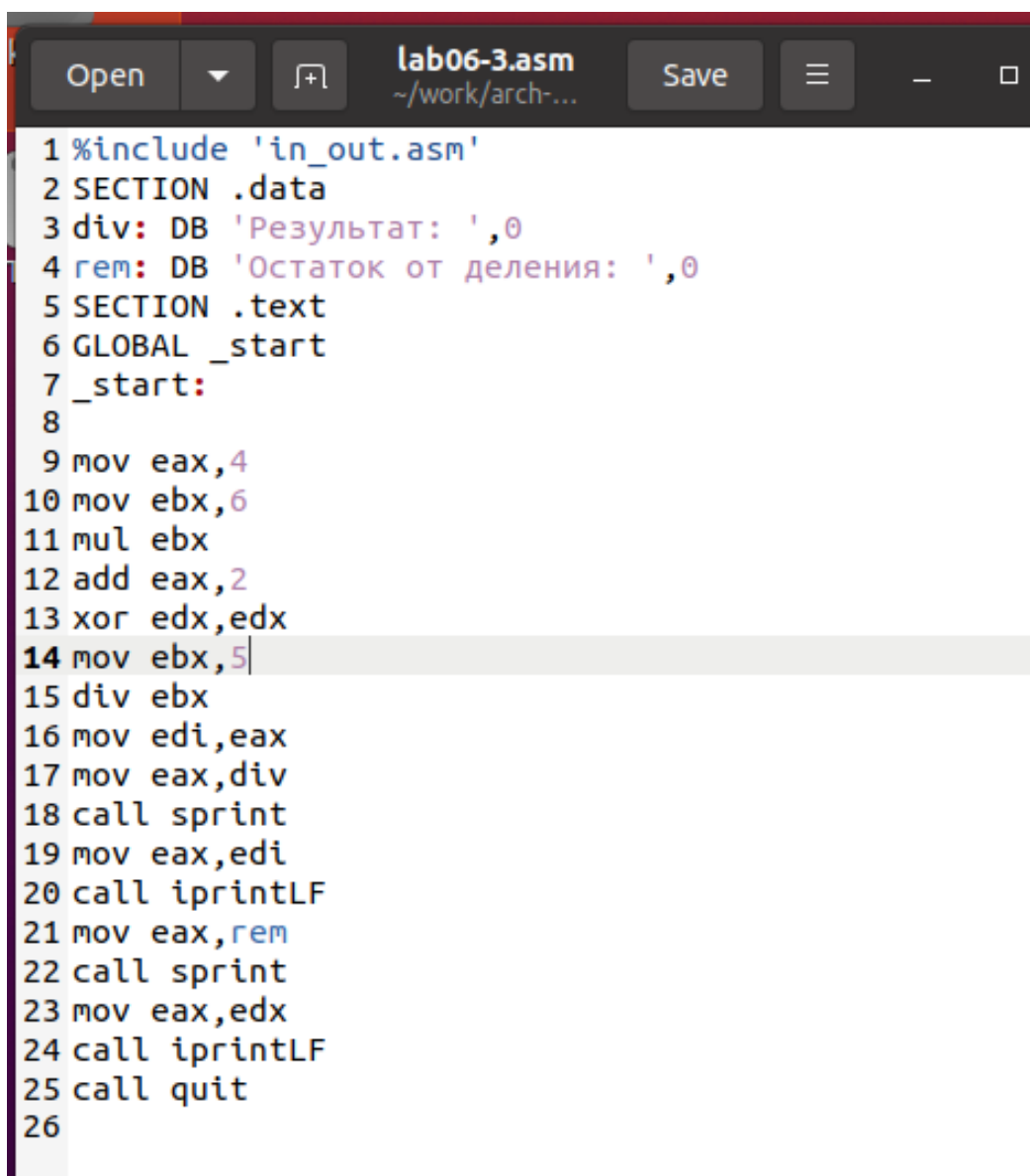


```
eakainova@eakainova:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
eakainova@eakainova:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
eakainova@eakainova:~/work/arch-pc/lab06$ ./lab06-3
Результат: 4
Остаток от деления: 1
eakainova@eakainova:~/work/arch-pc/lab06$
```

Рис. 2.11: Запуск программы lab6-3.asm

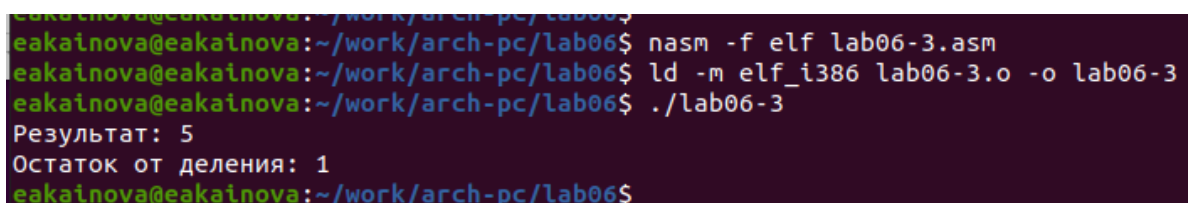
Изменяю программу для вычисления выражения $f(x) = (4 * 6 + 2) / 5$. Создаю

исполняемый файл и проверяю его работу.



```
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
26
```

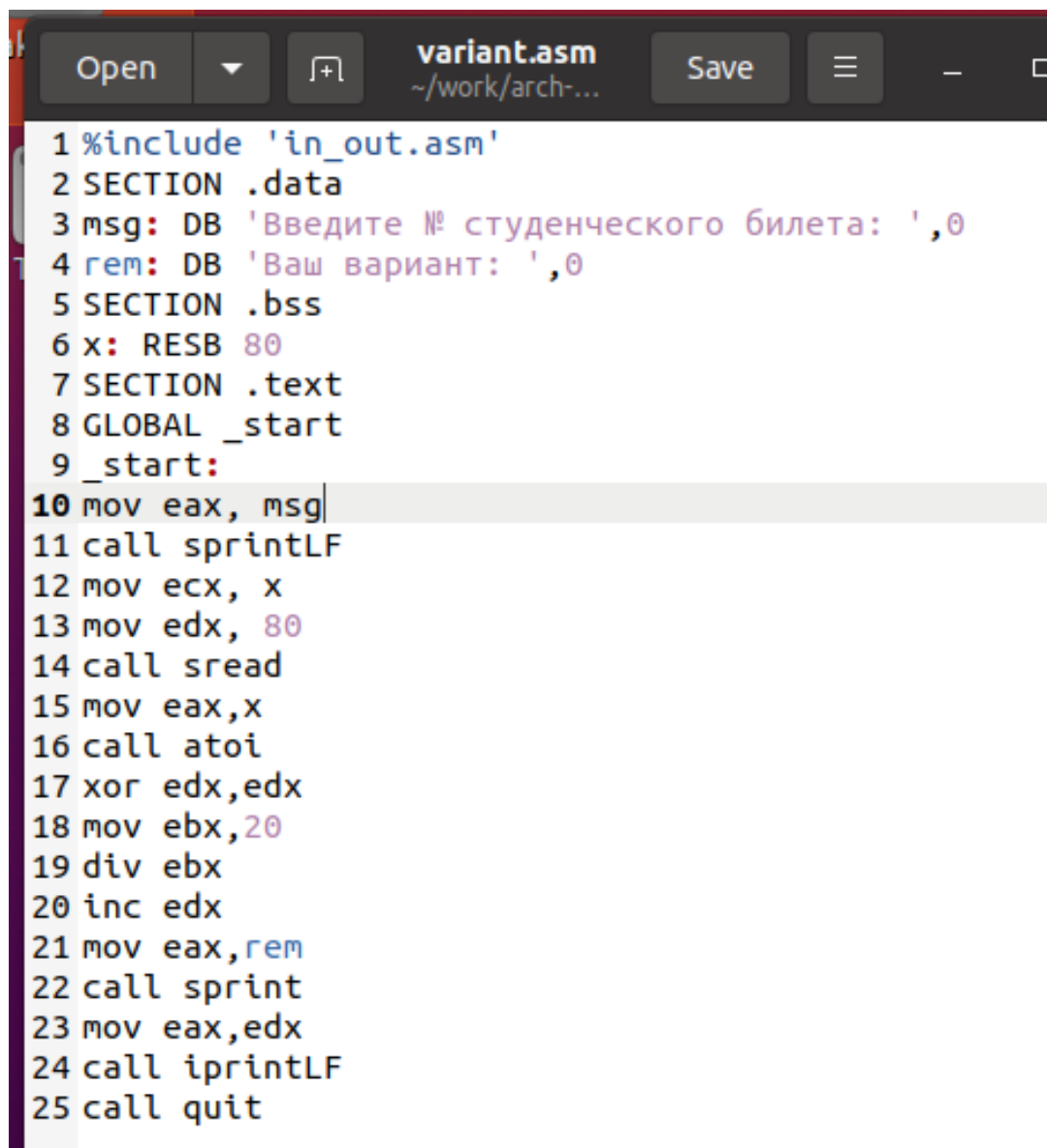
Рис. 2.12: Программа lab6-3.asm с другим выражением



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

Рис. 2.13: Запуск программы lab6-3.asm с другим выражением

Другим примером будет программа для вычисления варианта задания по номеру студенческого билета. В этом случае число, с которым производятся арифметические операции, вводится с клавиатуры. Для корректной работы с числами, введенные символы необходимо преобразовать в числовой формат, для чего используется 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. 2.14: Программа `variant.asm`

```
eakainova@eakainova:~/work/arch-pc/lab06$  
eakainova@eakainova:~/work/arch-pc/lab06$  
eakainova@eakainova:~/work/arch-pc/lab06$ nasm -f elf variant.asm  
eakainova@eakainova:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant  
eakainova@eakainova:~/work/arch-pc/lab06$ ./variant  
Введите № студенческого билета:  
1132243110  
Ваш вариант: 11  
eakainova@eakainova:~/work/arch-pc/lab06$
```

Рис. 2.15: Запуск программы variant.asm

2.3 Ответы на вопросы

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

- Инструкция `mov eax, ret` записывает значение переменной с фразой ‘Ваш вариант:’ в регистр `eax`.
- Инструкция `call sprint` вызывает подпрограмму для вывода строки.

2. Для чего используются следующие инструкции?

- `mov ecx, x` — записывает значение переменной `x` в регистр `ecx`.
- `mov edx, 80` — записывает значение 80 в регистр `edx`.
- `call sread` — вызывает подпрограмму для считывания значения студенческого билета.

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` — записывает значение 20 в регистр `ebx`.
- `div ebx` — выполняет деление номера студенческого билета на 20.

- `inc edx` — увеличивает значение в регистре `edx` на 1.

Здесь происходит деление номера студенческого билета на 20. Остаток записывается в регистр `edx`, к которому добавляется 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`” увеличивает значение в регистре `edx` на 1, согласно формуле вычисления варианта.

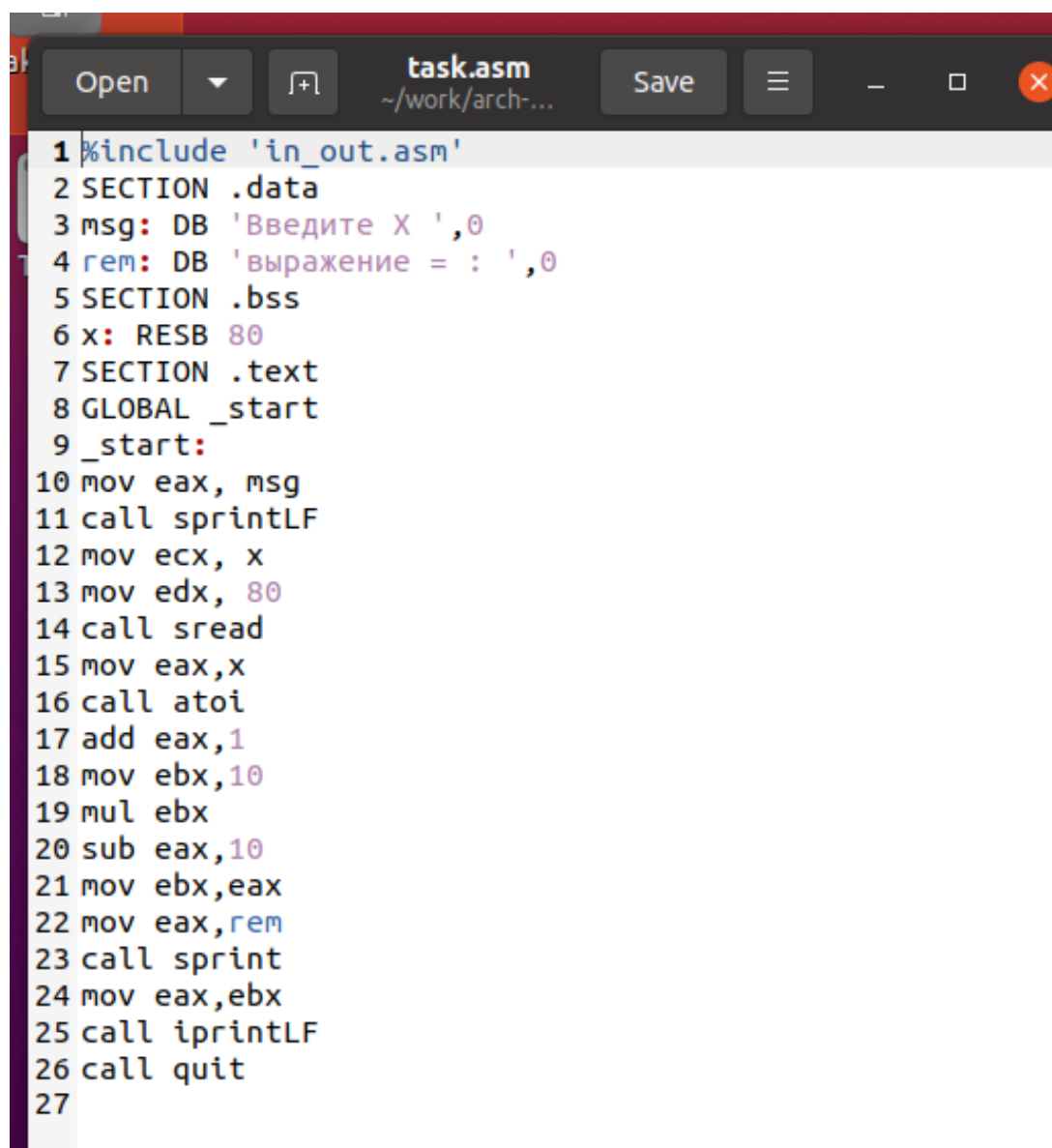
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

- Инструкция `mov eax, edx` записывает результат в регистр `eax`.
- Инструкция `call iprintLF` вызывает подпрограмму для вывода значения на экран.

2.4 Задание для самостоятельной работы

Необходимо написать программу для вычисления выражения $y = f(x)$, которая должна выводить выражение для вычисления, запросить ввод значения x , вычислить выражение в зависимости от введенного значения и вывести результат. Вид функции $f(x)$ должен быть выбран согласно таблице 6.3 вариантов заданий.

Мы получили вариант 11 для выражения $10(x+1)-10$ с $x=1, x=7$.



```
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
16 call atoi
17 add eax, 1
18 mov ebx, 10
19 mul ebx
20 sub eax, 10
21 mov ebx, eax
22 mov eax, rem
23 call sprint
24 mov eax, ebx
25 call iprintLF
26 call quit
27
```

Рис. 2.16: Программа task.asm

При \$ x=1 \$ результат равен 10.

При \$ x=7 \$ результат равен 70.

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

Рис. 2.17: Запуск программы task.asm

Программа работает правильно.

3 Выводы

Изучили работу с арифметическими операциями.