

Отчёта по лабораторной работе 6

Арифметические операции в NASM.

Тукаев Тимур Ильшатovich НММбд-03-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	27

Список иллюстраций

2.1	Программа в файле lab6-1.asm	7
2.2	Запуск программы lab6-1.asm	8
2.3	Программа в файле lab6-1.asm	9
2.4	Запуск программы lab6-1.asm	10
2.5	Программа в файле lab6-2.asm	11
2.6	Запуск программы lab6-2.asm	12
2.7	Программа в файле lab6-2.asm	13
2.8	Запуск программы lab6-2.asm	14
2.9	Запуск программы lab6-2.asm	14
2.10	Программа в файле lab6-3.asm	15
2.11	Запуск программы lab6-3.asm	17
2.12	Программа в файле lab6-3.asm	18
2.13	Запуск программы lab6-3.asm	19
2.14	Программа в файле variant.asm	20
2.15	Запуск программы variant.asm	22
2.16	Программа в файле task.asm	24
2.17	Запуск программы task.asm	26

Список таблиц

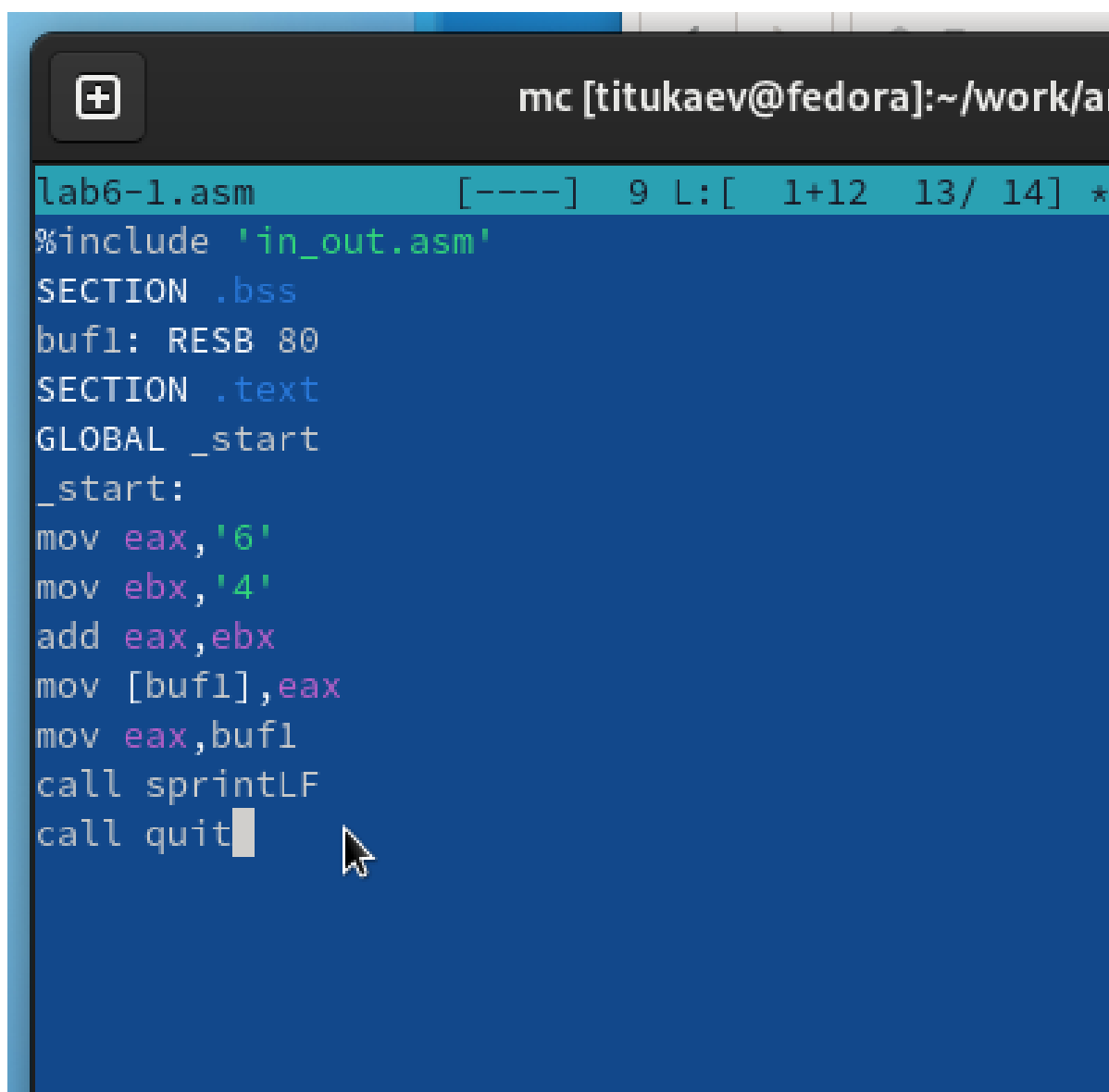
1 Цель работы

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 регистр еах.

В данной программе 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. Так как для работы функции `sprintf` 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 `sprintf`.



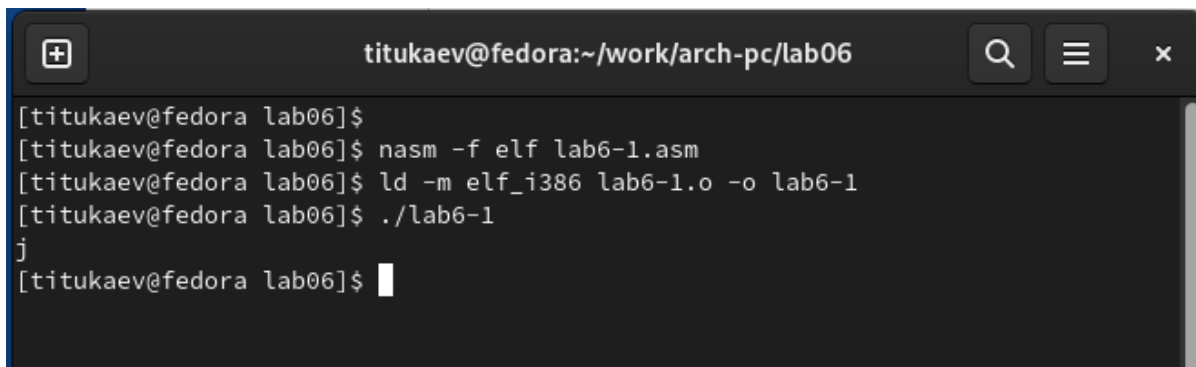
```
mc [titukaev@fedora]:~/work/a
lab6-1.asm  [----]  9  L:[  1+12  13/ 14]  *
#include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintLF
call quit
```

Рис. 2.1: Программа в файле lab6-1.asm

Привожу код программы в отчете

```
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
```

```
_start:
mov  eax,'6'
mov  ebx,'4'
add  eax,ebx
mov  [buf1],eax
mov  eax,buf1
call sprintLF
call quit
```

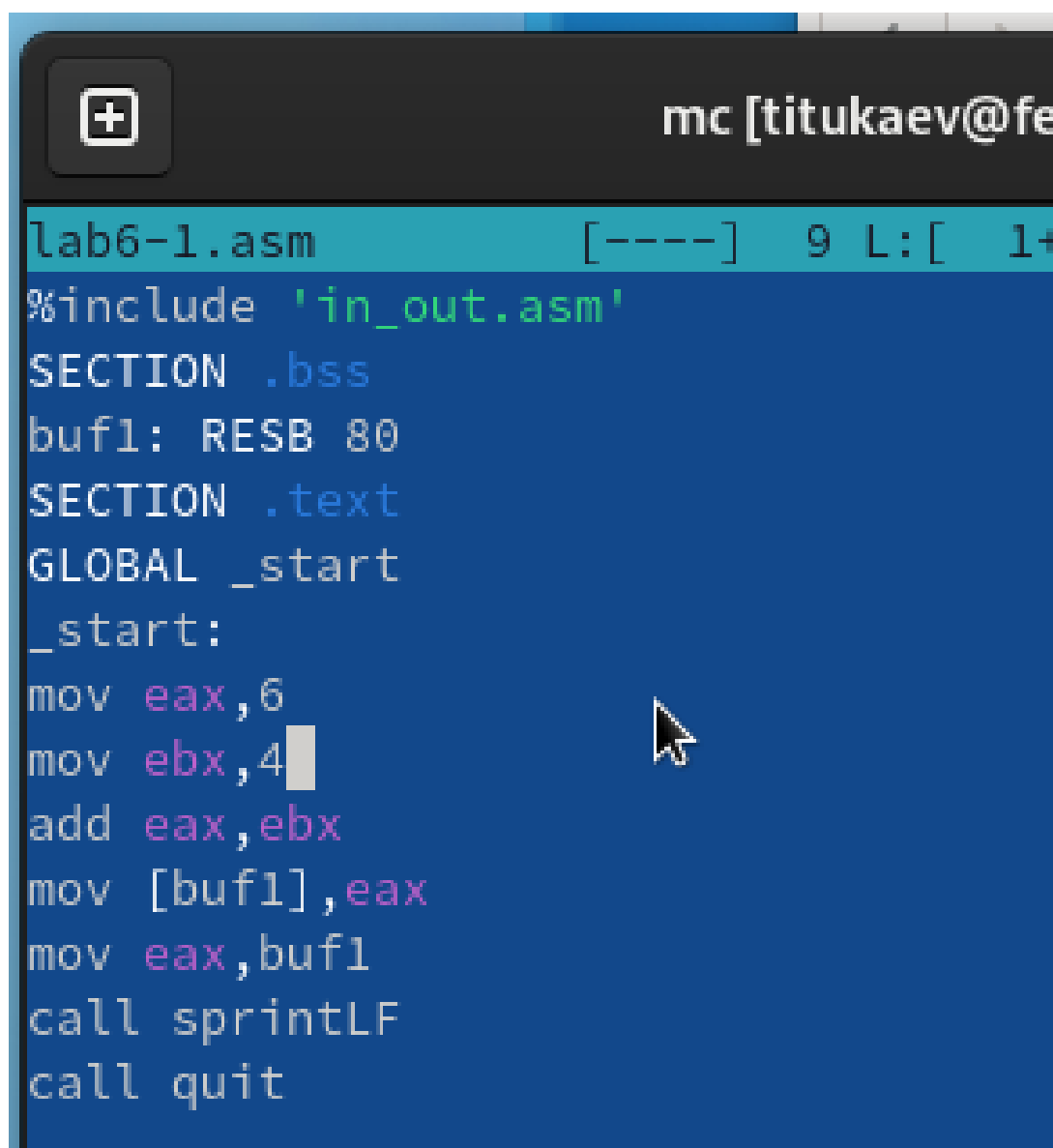
A terminal window with a dark background. The title bar shows the user 'titukaev@fedora' and the path '~/work/arch-pc/lab06'. The terminal contains the following commands and output:

```
[titukaev@fedora lab06]$
[titukaev@fedora lab06]$ nasm -f elf lab6-1.asm
[titukaev@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[titukaev@fedora lab06]$ ./lab6-1
j
[titukaev@fedora lab06]$
```

Рис. 2.2: 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 регистр `eax` сумму кодов – 01101010 (106), что в свою очередь является кодом символа `j`.

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



```
lab6-1.asm [-----] 9 L: [ 1+
#include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintLF
call quit
```

Рис. 2.3: Программа в файле lab6-1.asm

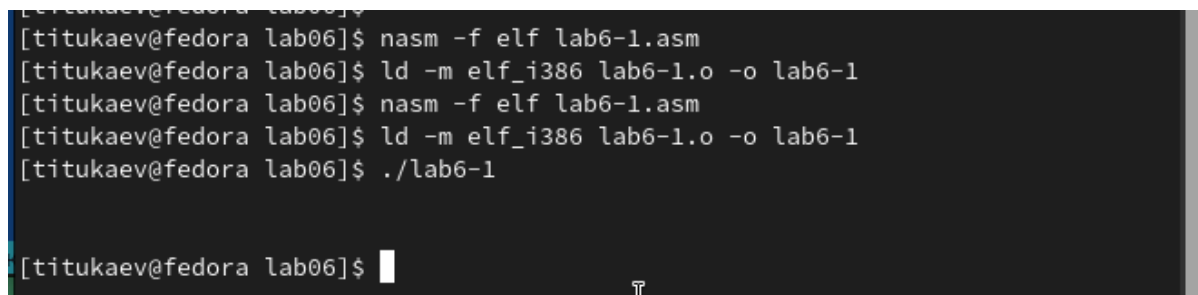
Привожу код программы в отчете

```
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
```

```

_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
mov  [buf1],eax
mov  eax,buf1
call sprintf
call quit

```



```

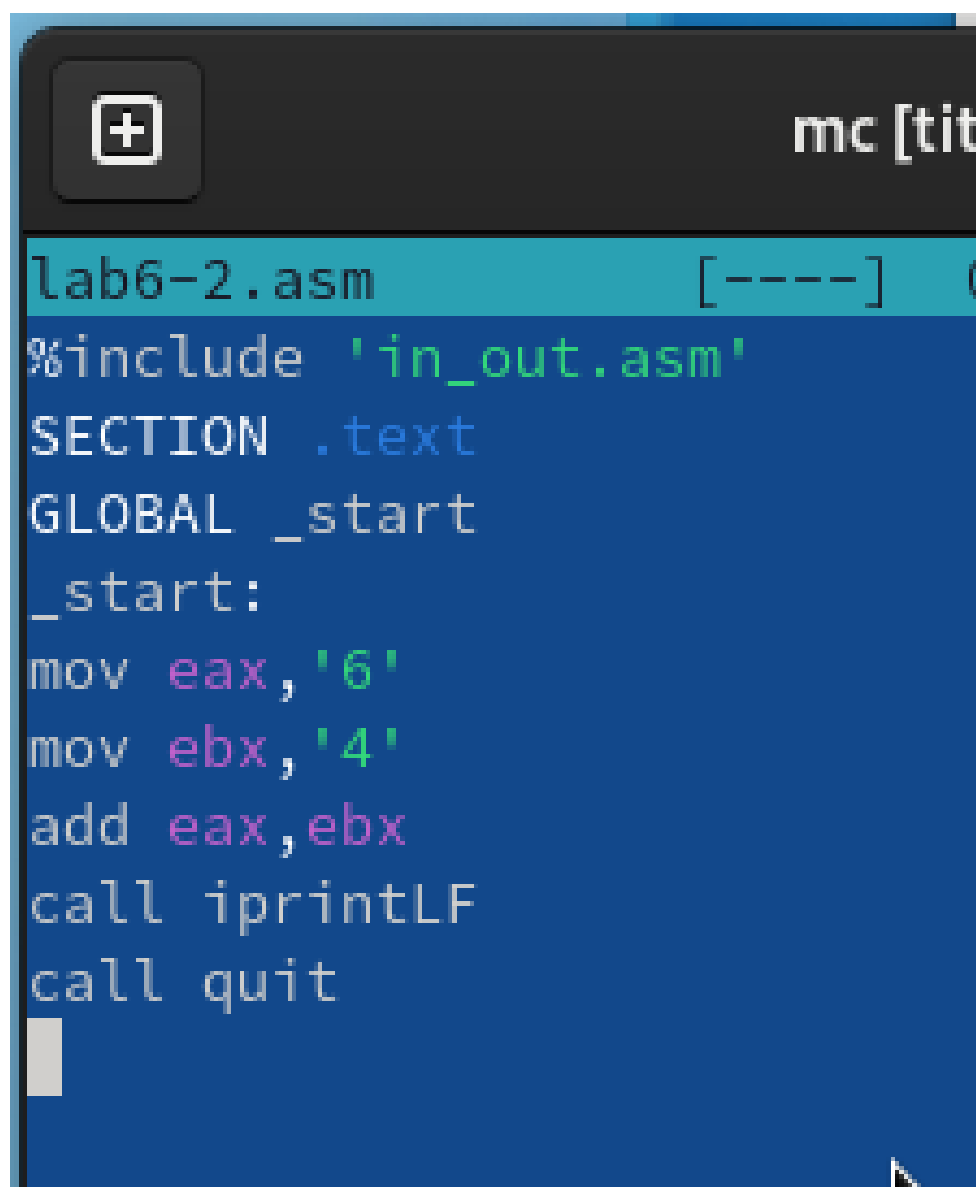
[titukaev@fedora lab06]$ nasm -f elf lab6-1.asm
[titukaev@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[titukaev@fedora lab06]$ nasm -f elf lab6-1.asm
[titukaev@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[titukaev@fedora lab06]$ ./lab6-1
[titukaev@fedora lab06]$

```

Рис. 2.4: 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. Это символ конца строки (возврат каретки). В консоли он не отображается, но добавляет пустую строку.

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.




```
lab6-2.asm [-----]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

Рис. 2.5: Программа в файле lab6-2.asm

Привожу код программы в отчете

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
```

```
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

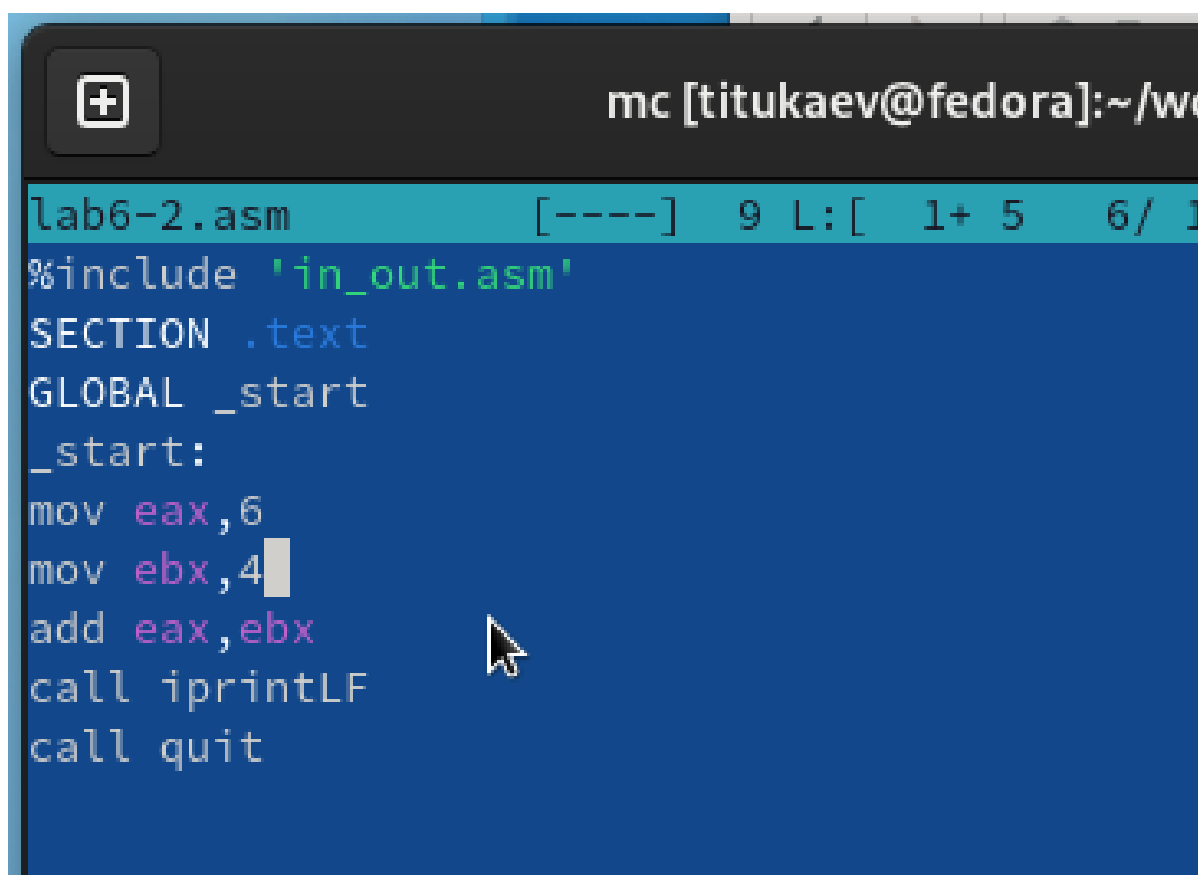


```
[titukaev@fedora lab06]$
[titukaev@fedora lab06]$ nasm -f elf lab6-2.asm
[titukaev@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[titukaev@fedora lab06]$ ./lab6-2
106
[titukaev@fedora lab06]$
```

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.



```
mc [titukaev@fedora]:~/w
lab6-2.asm [-----] 9 L: [ 1+ 5 6/ 1
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 2.7: Программа в файле lab6-2.asm

Привожу код программы в отчете

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Функция iprintLF позволяет вывести число и операндами были числа (а не

коды символов). Поэтому получаем число 10.

```
[titukaev@fedora lab06]$  
[titukaev@fedora lab06]$ nasm -f elf lab6-2.asm  
[titukaev@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2  
[titukaev@fedora lab06]$ ./lab6-2  
10  
[titukaev@fedora lab06]$  
[titukaev@fedora lab06]$
```

Рис. 2.8: Запуск программы lab6-2.asm

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

Привожу код программы в отчете

```
%include 'in_out.asm'  
  
SECTION .text  
  
GLOBAL _start  
  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprint  
call quit
```

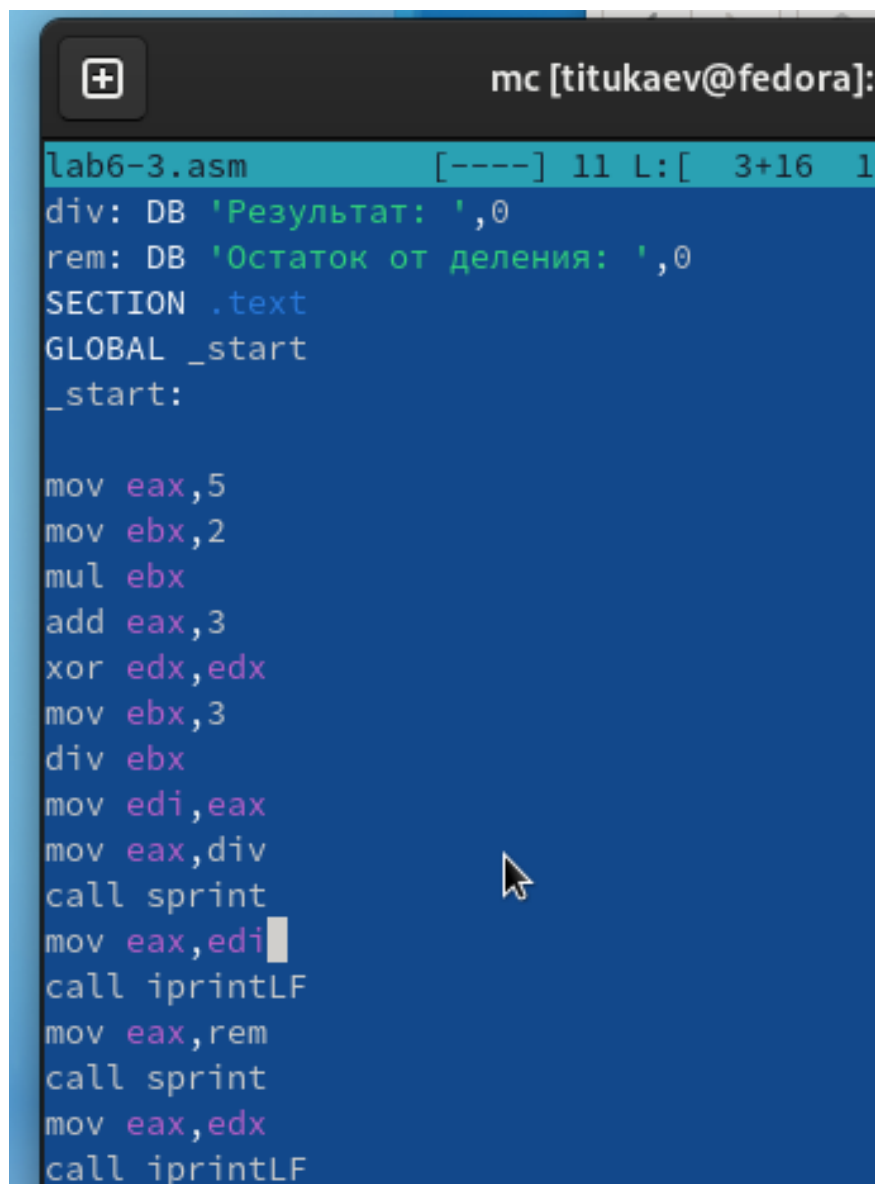
```
[titukaev@fedora lab06]$  
[titukaev@fedora lab06]$ nasm -f elf lab6-2.asm  
[titukaev@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2  
[titukaev@fedora lab06]$ ./lab6-2  
10[titukaev@fedora lab06]$  
[titukaev@fedora lab06]$
```

Рис. 2.9: Запуск программы lab6-2.asm

6. В качестве примера выполнения арифметических операций в NASM приве-

дем программу вычисления арифметического выражения

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



```
mc [titukaev@fedora]:-
lab6-3.asm [----] 11 L: [ 3+16 19
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:

mov eax,5
mov ebx,2
mul ebx
add eax,3
xor edx,edx
mov ebx,3
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
```

Рис. 2.10: Программа в файле lab6-3.asm

Также размещаю код программы в отчете.

```

#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

SECTION .text
GLOBAL _start

_start:

mov eax,5
mov ebx,2
mul ebx
add eax,3
xor edx,edx
mov ebx,3
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit

```



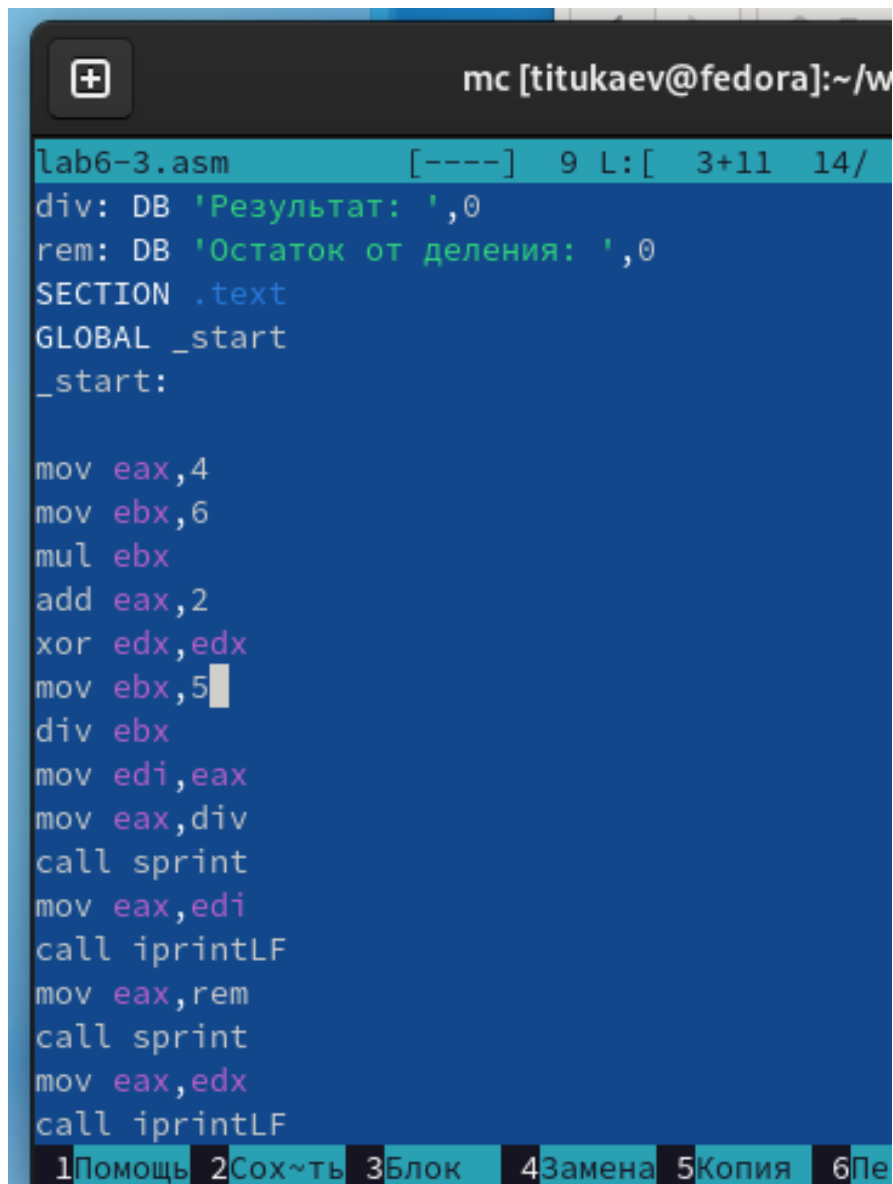
```
[titukaev@fedora lab06]$  
[titukaev@fedora lab06]$ nasm -f elf lab6-3.asm  
[titukaev@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3  
[titukaev@fedora lab06]$ ./lab6-3  
Результат: 4  
Остаток от деления: 1  
[titukaev@fedora lab06]$  
[titukaev@fedora lab06]$
```

Рис. 2.11: Запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения

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

. Создал исполняемый файл и проверил его работу.



```
lab6-3.asm [----] 9 L: [ 3+11 14/
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:

mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перезагрузить

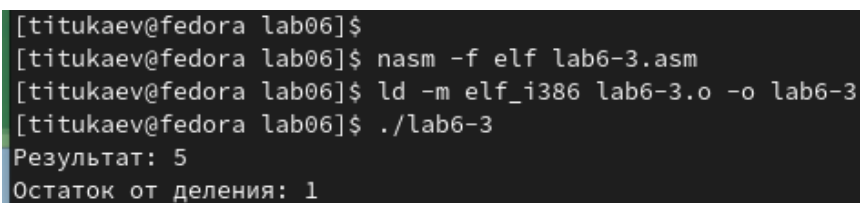
Рис. 2.12: Программа в файле lab6-3.asm

Также размещаю код программы в отчете.

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
```

```
GLOBAL _start
_start:
```

```
mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

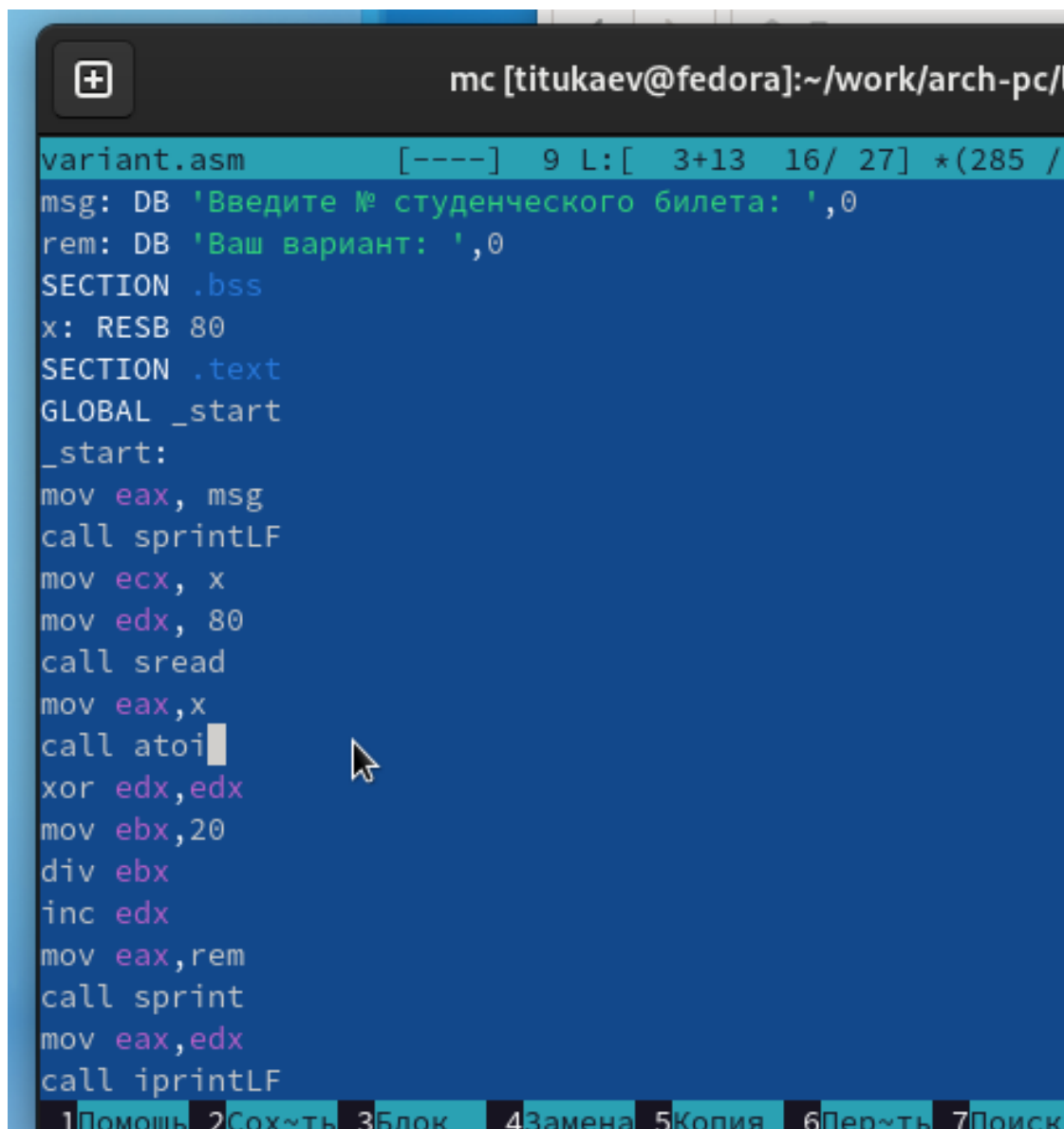


```
[titukaev@fedora lab06]$
[titukaev@fedora lab06]$ nasm -f elf lab6-3.asm
[titukaev@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3
[titukaev@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 2.13: Запуск программы lab6-3.asm

7. 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`.



```
variant.asm [----] 9 L:[ 3+13 16/ 27] *(285 /
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprint
mov eax, edx
call iprintLF
```

Рис. 2.14: Программа в файле `variant.asm`

Также размещаю код программы в отчете.

```

#include 'in_out.asm'

SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi
xor edx,edx
mov ebx,20
div ebx
inc edx
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit

```

```

[titukaev@fedora lab06]$
[titukaev@fedora lab06]$ nasm -f elf variant.asm
[titukaev@fedora lab06]$ ld -m elf_i386 variant.o -o variant
[titukaev@fedora lab06]$ ./variant
Введите № студенческого билета:
1132232884
Ваш вариант: 5
[titukaev@fedora lab06]$

```

Рис. 2.15: Запуск программы variant.asm

ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения ‘Ваш вариант:’?
 - `mov eax,rem` – перекладывает в регистр значение переменной с фразой ‘Ваш вариант:’
 - `call sprint` – вызов подпрограммы вывода строки
2. Для чего используются следующие инструкции?

```

nasm
mov ecx, x
mov edx, 80
call sread

```

Считывает значение студбилета в переменную X из консоли

3. Для чего используется инструкция “call atoi”?

Эта подпрограмма переводит введенные символы в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?

```

xor edx,edx
mov ebx,20
div ebx
inc edx

```

Здесь происходит деление номера студ билета на 20. В регистре edx хранится остаток, к нему прибавляется 1.

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

регистр edx

6. Для чего используется инструкция “inc edx”?

по формуле вычисления варианта нужно прибавить единицу

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

mov eax,edx – результат перекладывается в регистр eax

call iprintLF – вызов подпрограммы вывода

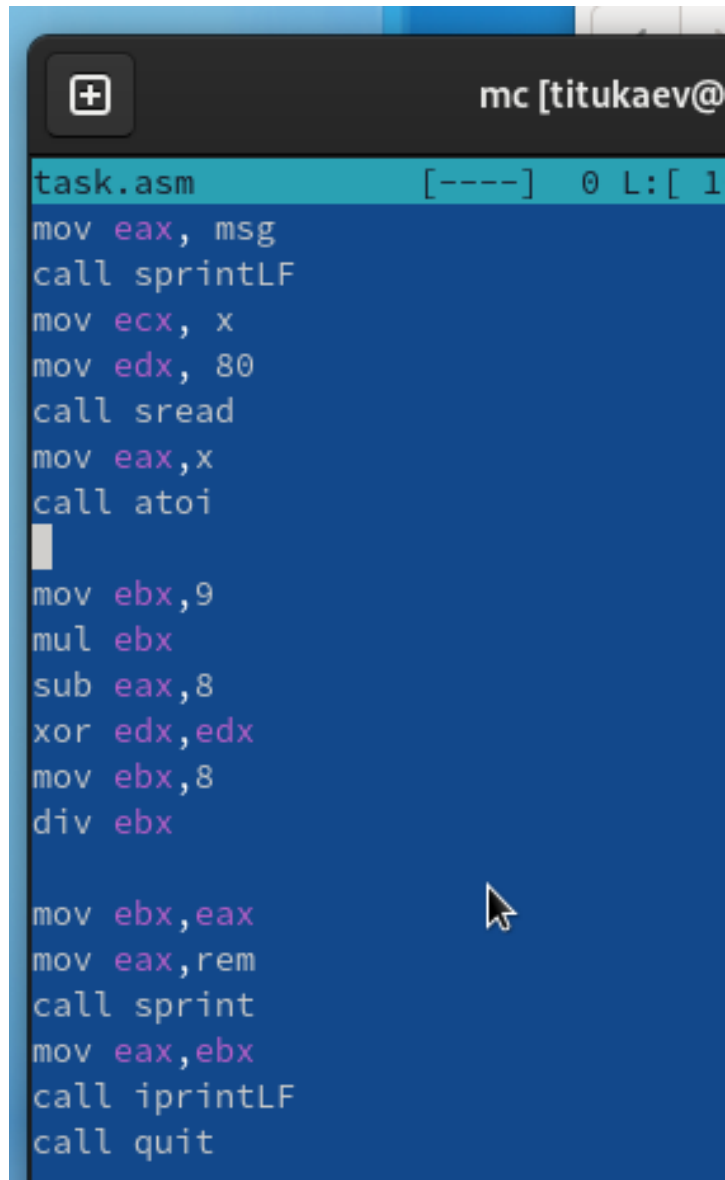
8. Написать программу вычисления выражения $y = f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии 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 5 -

$$(9x - 8)/8$$

для

$$x_1 = 8, x_2 = 64$$



```
mc [titukaev@
task.asm [----] 0 L: [ 1
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov ebx, 9
mul ebx
sub eax, 8
xor edx, edx
mov ebx, 8
div ebx
mov ebx, eax
mov eax, rem
call sprint
mov eax, ebx
call iprintLF
call quit
```

Рис. 2.16: Программа в файле task.asm

Также размещаю код программы в отчете.

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите X ', 0
rem: DB 'выражение = : ', 0
SECTION .bss
```



```

x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi

add eax,10
mov ebx,3
mul ebx
sub eax,20

mov ebx,eax
mov eax,rem
call sprint
mov eax,ebx
call iprintLF
call quit

```

```
[titukaev@fedora lab06]$ nasm -f elf task.asm
[titukaev@fedora lab06]$ ld -m elf_i386 task.o -o task
[titukaev@fedora lab06]$ ./task
Введите X
8
выражение = : 8
[titukaev@fedora lab06]$ ./task
Введите X
64
выражение = : 71
[titukaev@fedora lab06]$
```

Рис. 2.17: Запуск программы task.asm

3 Выводы

Изучили работу с арифметическими операциями.