

Отчёт по лабораторной работе 6

Арифметические операции в NASM.

Уржиндорж Мягмар

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	21

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	8
2.5	Программа lab6-2.asm	9
2.6	Запуск программы lab6-2.asm	9
2.7	Программа lab6-2.asm	10
2.8	Запуск программы lab6-2.asm	11
2.9	Программа lab6-2.asm	11
2.10	Запуск программы lab6-2.asm	12
2.11	Программа lab6-3.asm	13
2.12	Запуск программы lab6-3.asm	13
2.13	Программа lab6-3.asm	14
2.14	Запуск программы lab6-3.asm	15
2.15	Программа variant.asm	16
2.16	Запуск программы variant.asm	17
2.17	Программа prog.asm	19
2.18	Запуск программы prog.asm	20

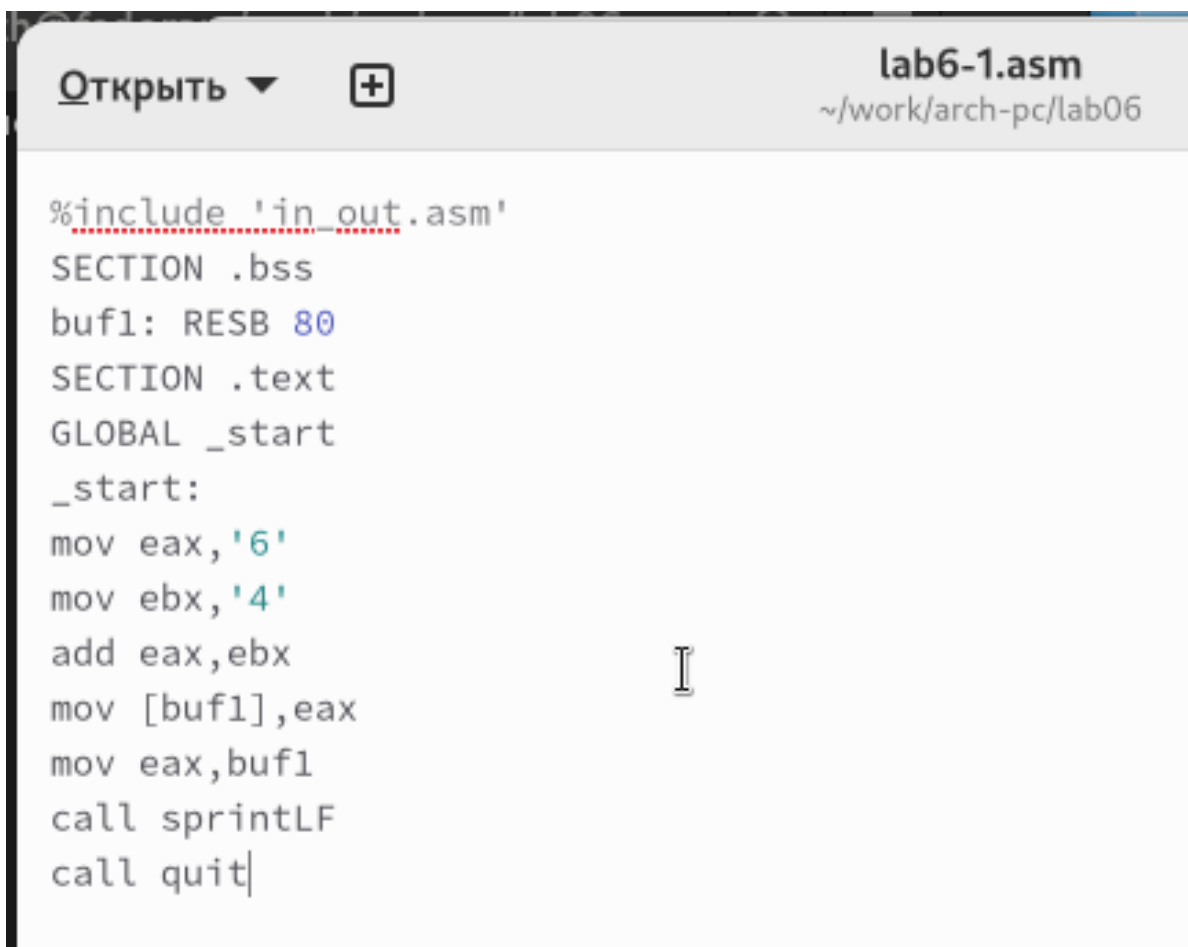
Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

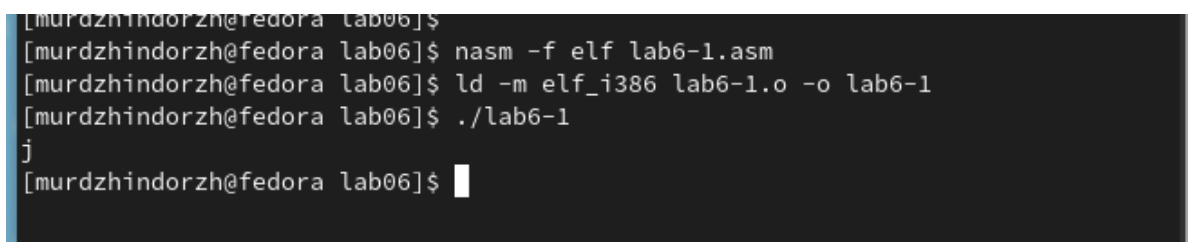
1. Создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab6-1.asm.
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр eax.



```
Открыть ▾ + lab6-1.asm
~/work/arch-pc/lab06

%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|
```

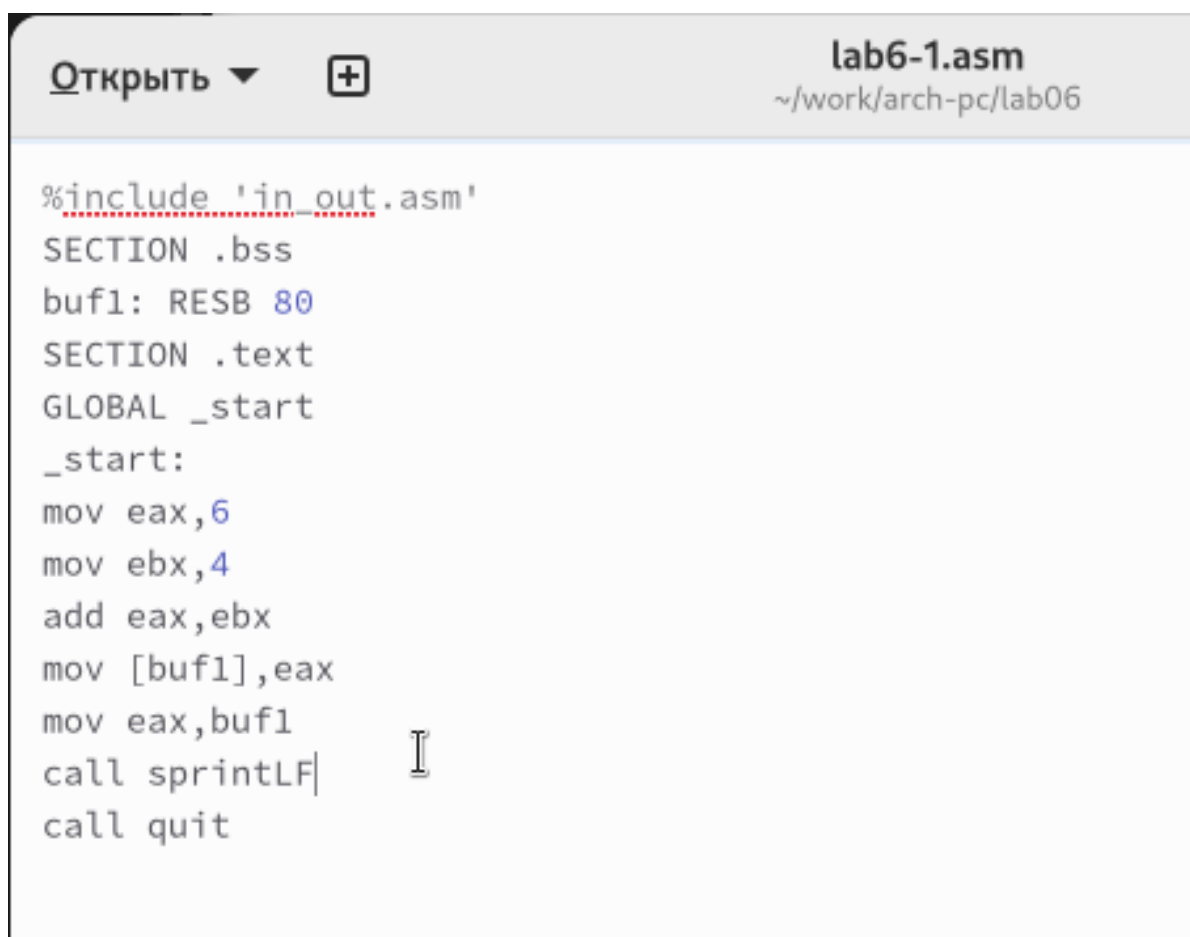
Рис. 2.1: Программа lab6-1.asm



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

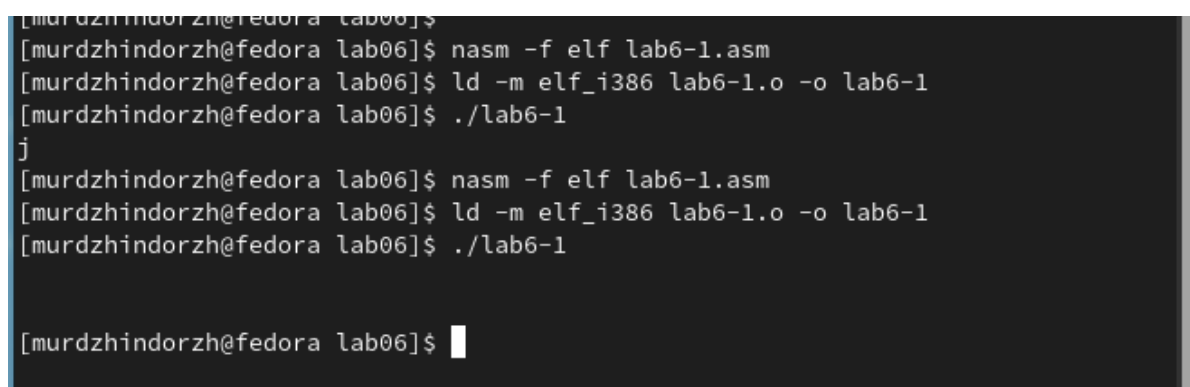
Рис. 2.2: Запуск программы lab6-1.asm

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



```
%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 lab6-1.asm

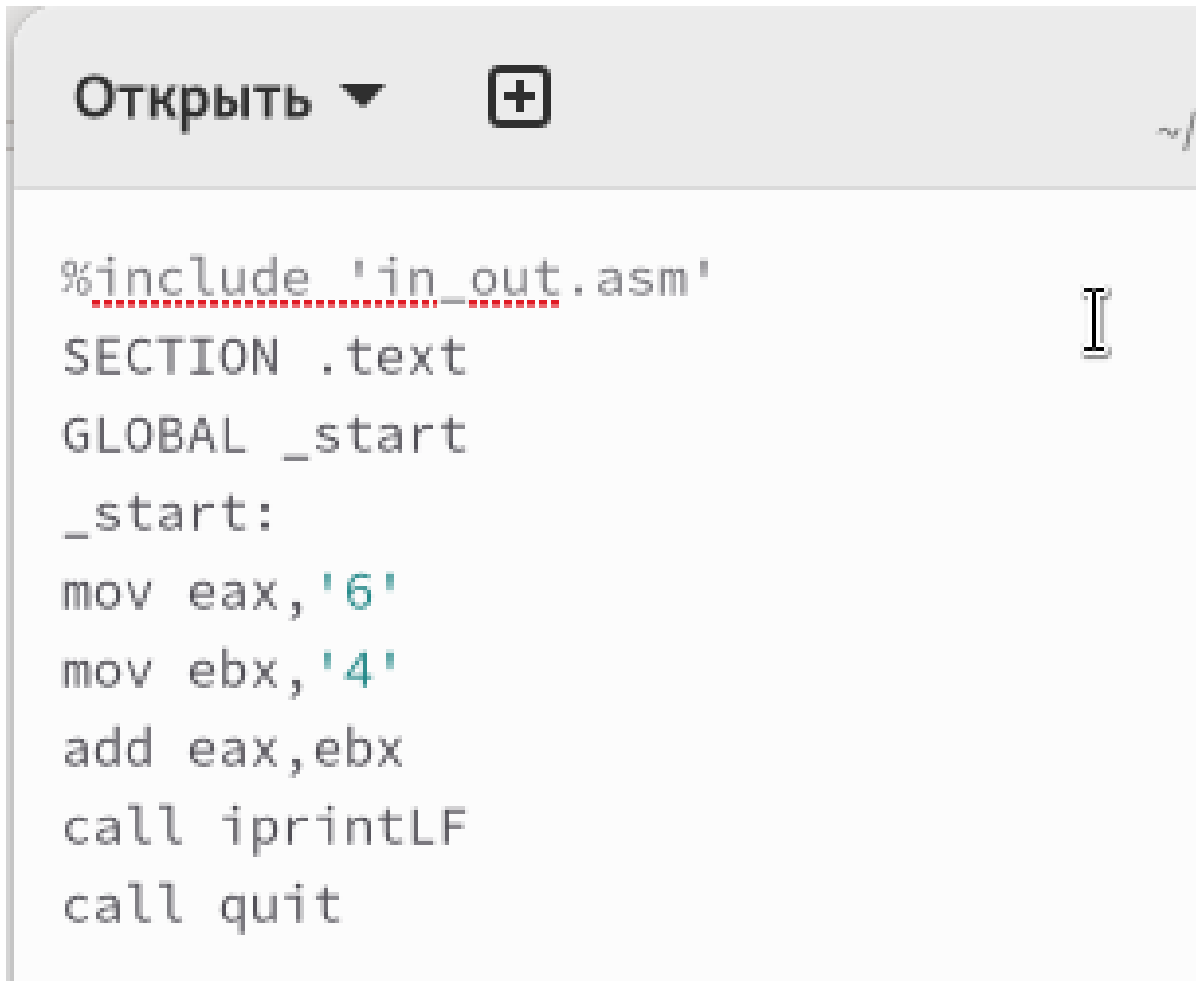


```
[murdzhindorzh@fedora lab06]$
[murdzhindorzh@fedora lab06]$ nasm -f elf lab6-1.asm
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[murdzhindorzh@fedora lab06]$ ./lab6-1
j
[murdzhindorzh@fedora lab06]$ nasm -f elf lab6-1.asm
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 lab6-1.o -o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

Рис. 2.4: Запуск программы lab6-1.asm

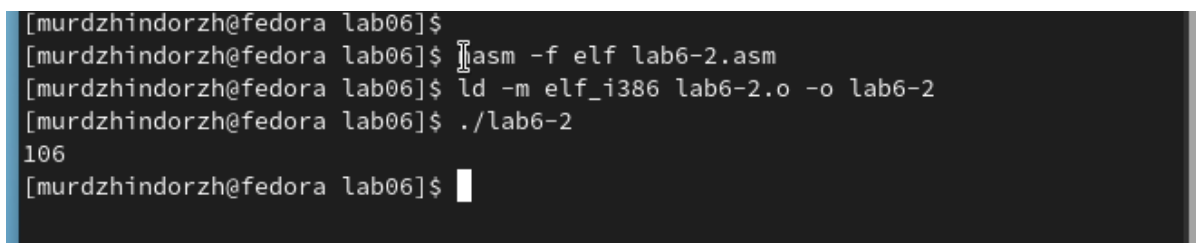
Никакой символ не виден, но он есть. Это возврат каретки LF.

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.



```
Открыть ▼ +  
  
%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 lab6-2.asm

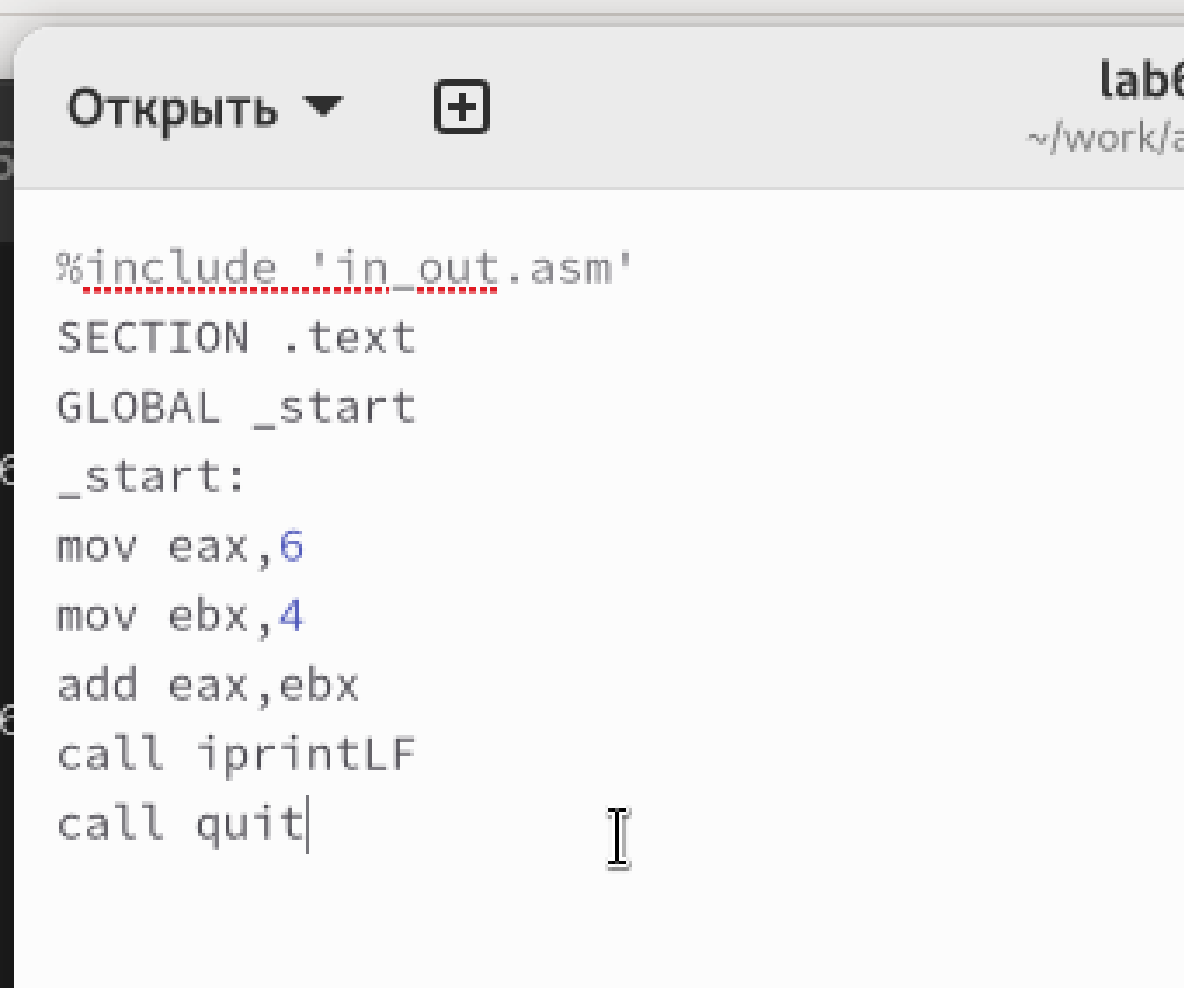


```
[murdzhindorzh@fedora lab06]$  
[murdzhindorzh@fedora lab06]$ nasm -f elf lab6-2.asm  
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2  
[murdzhindorzh@fedora lab06]$ ./lab6-2  
106  
[murdzhindorzh@fedora lab06]$
```

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.



```
Открыть ▼ + lab6
~/work/a

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

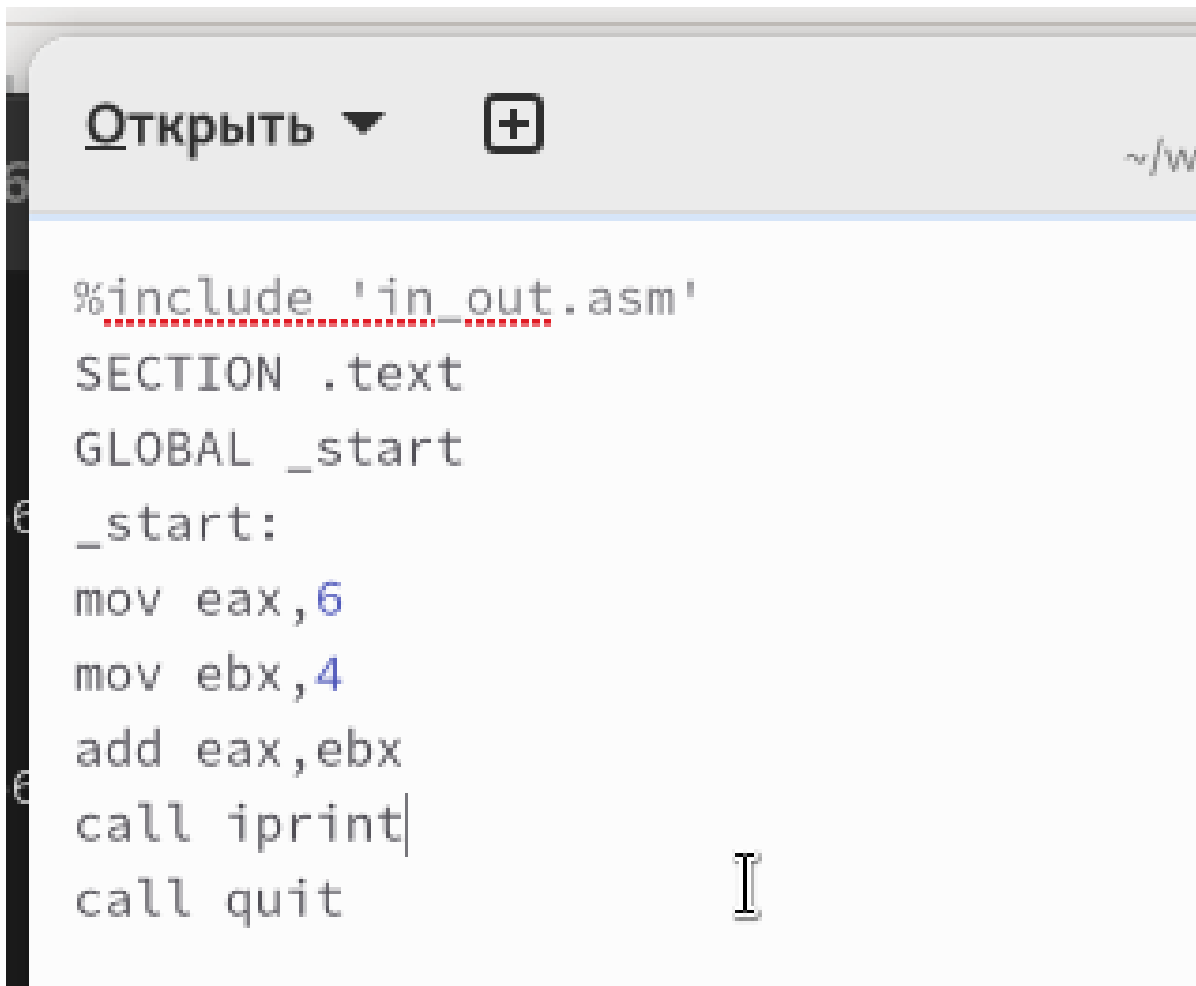
Рис. 2.7: Программа lab6-2.asm

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

```
[murdzhindorzh@fedora lab06]$ nasm -f elf lab6-2.asm
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[murdzhindorzh@fedora lab06]$ ./lab6-2
106
[murdzhindorzh@fedora lab06]$
[murdzhindorzh@fedora lab06]$ nasm -f elf lab6-2.asm
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[murdzhindorzh@fedora lab06]$ ./lab6-2
10
[murdzhindorzh@fedora lab06]$
```

Рис. 2.8: Запуск программы lab6-2.asm

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



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

Рис. 2.9: Программа lab6-2.asm

```

[murdzhindorzh@fedora lab06]$
[murdzhindorzh@fedora lab06]$ nasm -f elf lab6-2.asm
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[murdzhindorzh@fedora lab06]$ ./lab6-2
106
[murdzhindorzh@fedora lab06]$
[murdzhindorzh@fedora lab06]$ nasm -f elf lab6-2.asm
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[murdzhindorzh@fedora lab06]$ ./lab6-2
10
[murdzhindorzh@fedora lab06]$ nasm -f elf lab6-2.asm
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[murdzhindorzh@fedora lab06]$ ./lab6-2
10[murdzhindorzh@fedora lab06]$
[murdzhindorzh@fedora lab06]$

```

Рис. 2.10: Запуск программы lab6-2.asm

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

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

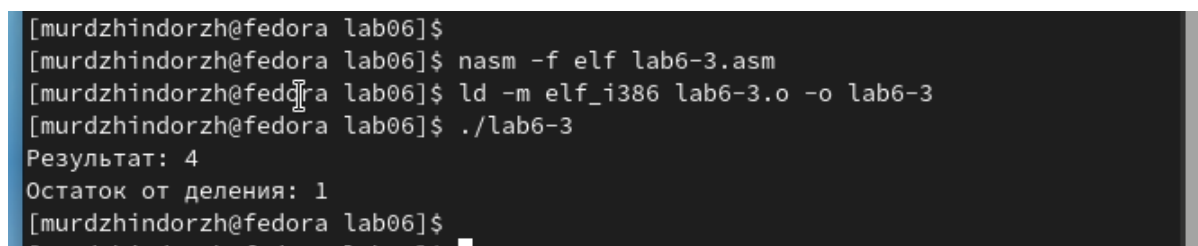
.



```
Открыть ▾ + ~/wc
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
```

Рис. 2.11: Программа lab6-3.asm




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



```
Открыть ▼ + lab6-3.asm
~/work/arch-pc/labC

div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:

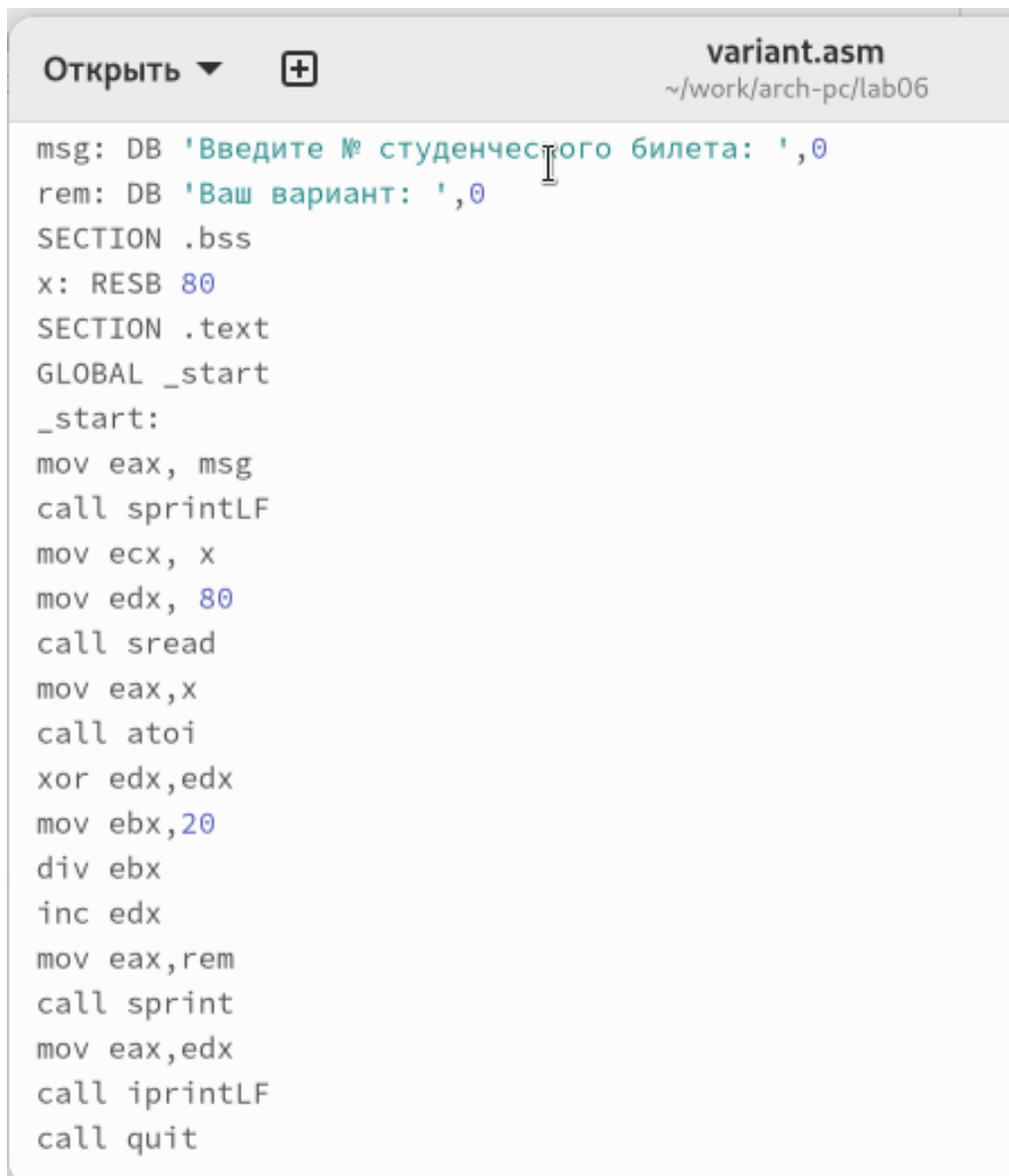
mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit|
```

Рис. 2.13: Программа lab6-3.asm

```
[murdzhindorzh@fedora lab06]$  
[murdzhindorzh@fedora lab06]$ nasm -f elf lab6-3.asm  
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3  
[murdzhindorzh@fedora lab06]$ ./lab6-3  
Результат: 4  
Остаток от деления: 1  
[murdzhindorzh@fedora lab06]$  
[murdzhindorzh@fedora lab06]$ ^C  
[murdzhindorzh@fedora lab06]$ nasm -f elf lab6-3.asm  
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3  
[murdzhindorzh@fedora lab06]$ ./lab6-3  
Результат: 5  
Остаток от деления: 1  
[murdzhindorzh@fedora lab06]$  
[murdzhindorzh@fedora lab06]$
```

Рис. 2.14: Запуск программы lab6-3.asm

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



```
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
```

Рис. 2.15: Программа variant.asm


```
[murdzhindorzh@fedora lab06]$
[murdzhindorzh@fedora lab06]$ nasm -f elf variant.asm
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 variant.o -o variant
[murdzhindorzh@fedora lab06]$ ./variant
Введите № студенческого билета:
1032235131
Ваш вариант: 12
[murdzhindorzh@fedora lab06]$
```

Рис. 2.16: Запуск программы variant.asm

ответы на вопросы

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

В строке `mov eax,ret` значение переменной с фразой 'Ваш вариант:' перекладывается в регистр `eax`.

Строка `call sprint` вызывает подпрограмму для вывода строки.

2. Для чего используются следующие инструкции?

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

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

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

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

3. Для чего используется инструкция "call atoi"?

Эта инструкция вызывает подпрограмму, которая преобразует введенные символы 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` - производит 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 увеличивает значение 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?

mov eax,edx - результат вычислений перекладывается в регистр eax.

call iprintLF - вызывается подпрограмма для вывода результата на экран.

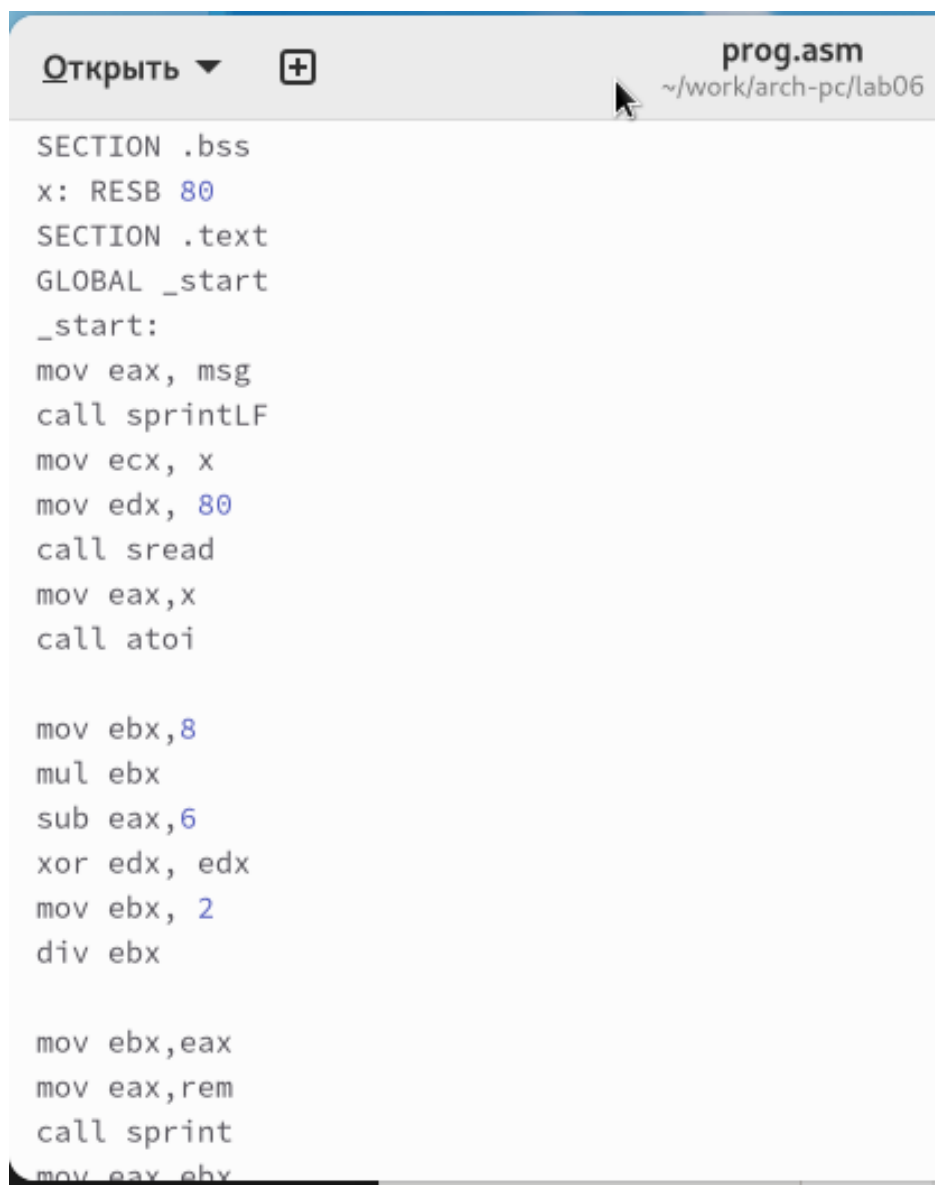
8. Написать программу вычисления выражения $y = f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии 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 12 -

$$(8x - 6)/2$$

для

$$x = 1, x = 5$$



```
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

mov ebx, 8
mul ebx
sub eax, 6
xor edx, edx
mov ebx, 2
div ebx

mov ebx, eax
mov eax, rem
call sprint
mov eax, ebx
```

Рис. 2.17: Программа prog.asm

```
[murdzhindorzh@fedora lab06]$  
[murdzhindorzh@fedora lab06]$ nasm -f elf prog.asm  
[murdzhindorzh@fedora lab06]$ ld -m elf_i386 prog.o -o prog  
[murdzhindorzh@fedora lab06]$ ./prog  
Введите X  
1  
выражение = : 1  
[murdzhindorzh@fedora lab06]$ ./prog  
Введите X  
5  
выражение = : 17  
[murdzhindorzh@fedora lab06]$
```

Рис. 2.18: Запуск программы prog.asm

3 Выводы

Изучили работу с арифметическими операциями.