

Отчёта по лабораторной работе 6

Архитектура компьютеров и операционные системы

Кенан Гашимов НКАБд-02-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	23

Список иллюстраций

2.1	Редактирование файла lab6-1.asm	7
2.2	Компиляция и проверка программы программы lab6-1.asm	8
2.3	Редактирование файла lab6-1.asm	9
2.4	Компиляция и проверка программы программы lab6-1.asm	10
2.5	Редактирование файла lab6-2.asm	11
2.6	Компиляция и проверка программы программы lab6-2.asm	11
2.7	Редактирование файла lab6-2.asm	12
2.8	Компиляция и проверка программы программы lab6-2.asm	12
2.9	Компиляция и проверка программы программы lab6-2.asm	13
2.10	Редактирование файла lab6-3.asm	14
2.11	Компиляция и проверка программы программы lab6-3.asm	15
2.12	Редактирование файла lab6-3.asm	16
2.13	Компиляция и проверка программы программы lab6-3.asm	17
2.14	Редактирование файла variant.asm	18
2.15	Компиляция и проверка программы программы variant.asm	19
2.16	Редактирование файла task.asm	21
2.17	Компиляция и проверка программы программы task.asm	22

Список таблиц

1 Цель работы

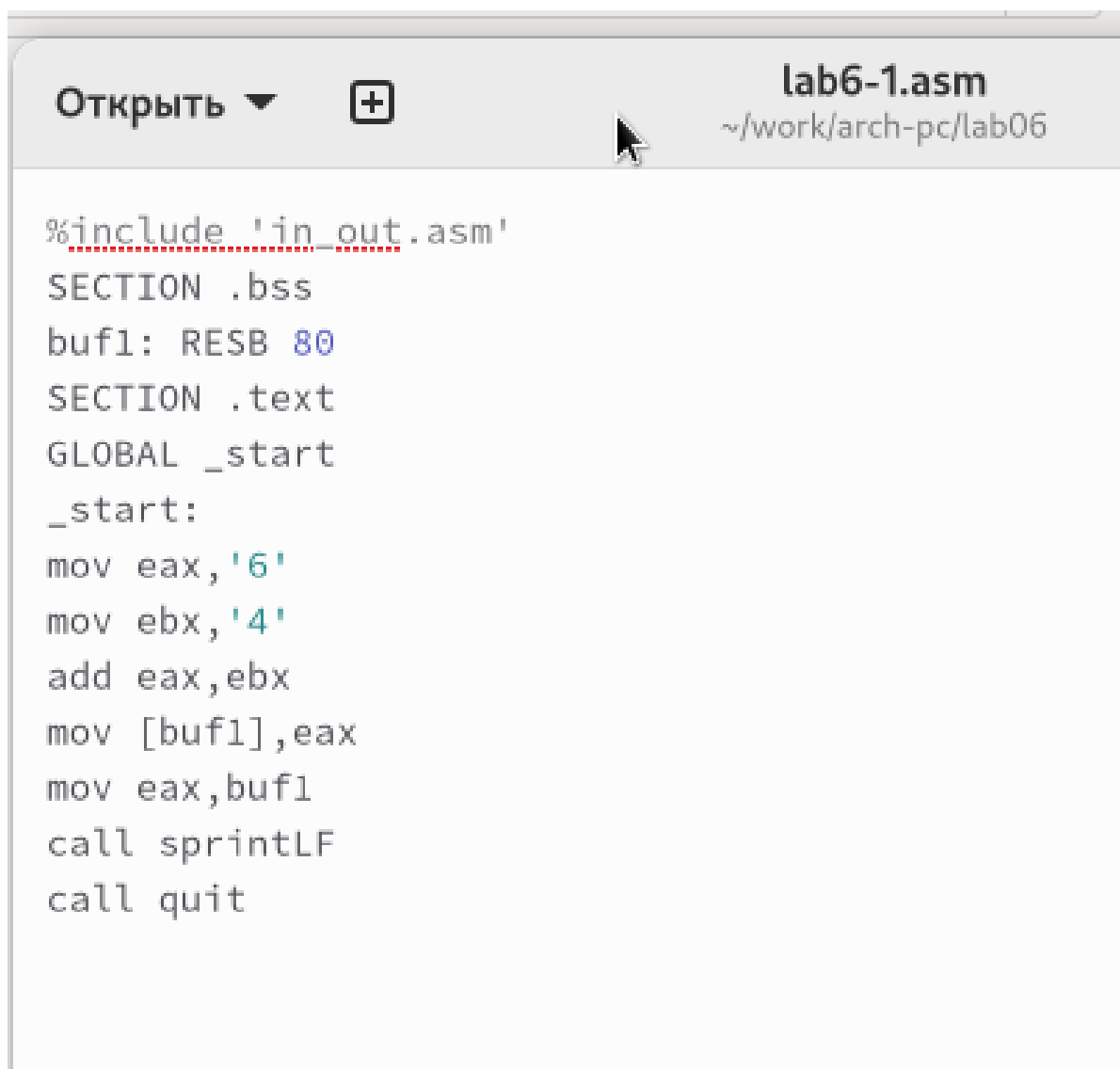
Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

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

Давайте рассмотрим примеры программ, которые выводят символьные и числовые значения. В этих программах значения будут записываться в регистр `eax`.

В данной программе мы записываем символ '6' в регистр `eax` (`mov eax, '6'`), а символ '4' в регистр `ebx` (`mov ebx, '4'`). Затем мы добавляем значение регистра `ebx` к значению в регистре `eax` (`add eax, ebx`, результат сложения записывается в регистр `eax`). После этого мы выводим результат. Однако, для работы функции `sprintf`, необходимо, чтобы в регистре `eax` был записан адрес, поэтому мы используем дополнительную переменную. Мы записываем значение регистра `eax` в переменную `buf1` (`mov [buf1], eax`), а затем записываем адрес переменной `buf1` в регистр `eax` (`mov eax, buf1`) и вызываем функцию `sprintf`.

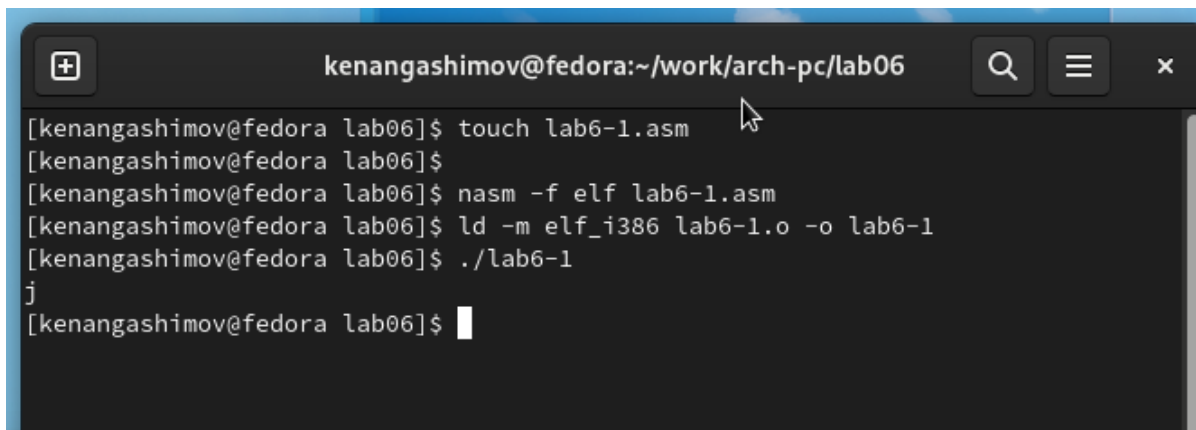


Открыть ▾ +

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: Редактирование файла lab6-1.asm

A terminal window titled 'kenangashimov@fedora:~/work/arch-pc/lab06' with search, menu, and close icons. The terminal shows the following commands and output:

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

Рис. 2.2: Компиляция и проверка программы 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 'б' равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа '4' равен 00110100 (или 52 в десятичном представлении). Когда мы выполняем команду `add eax, ebx`, результатом будет сумма кодов - 01101010 (или 106 в десятичном представлении), который соответствует символу 'j'.

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

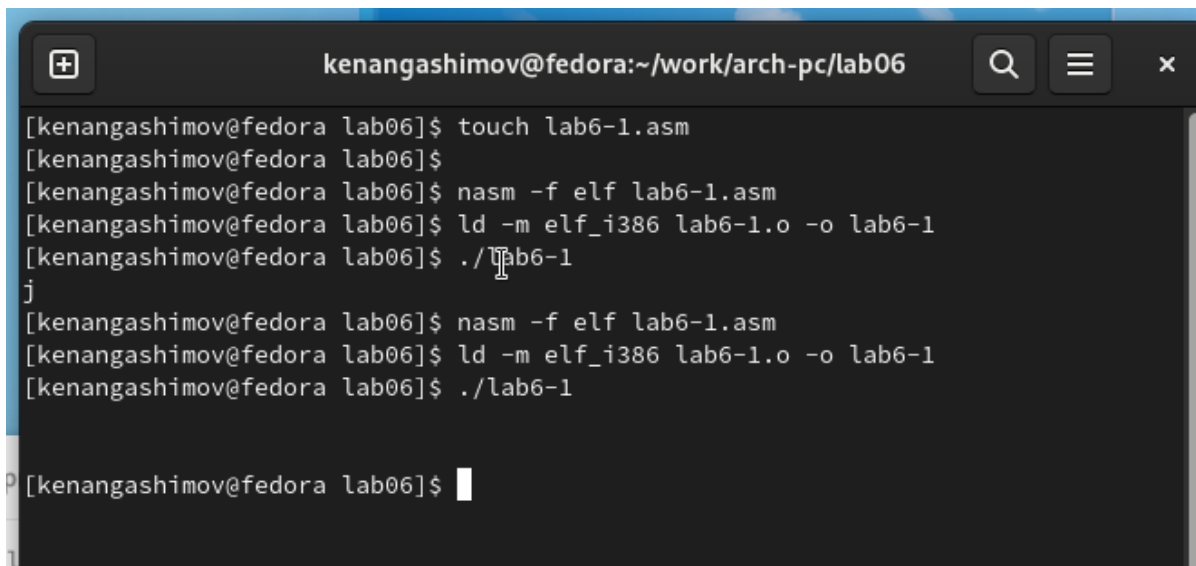
Открыть ▾

+

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 sprintf  
call quit
```

Рис. 2.3: Редактирование файла lab6-1.asm

A terminal window titled 'kenangashimov@fedora:~/work/arch-pc/lab06' with search, menu, and close icons. The terminal shows the following commands and output:

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

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

При изменении текста программы и записи чисел в регистры, мы также не получим число 10 при выполнении программы. Вместо этого будет выведен символ с кодом 10, который представляет собой символ конца строки (возврат каретки). В консоли этот символ не отображается, но он добавляет пустую строку.

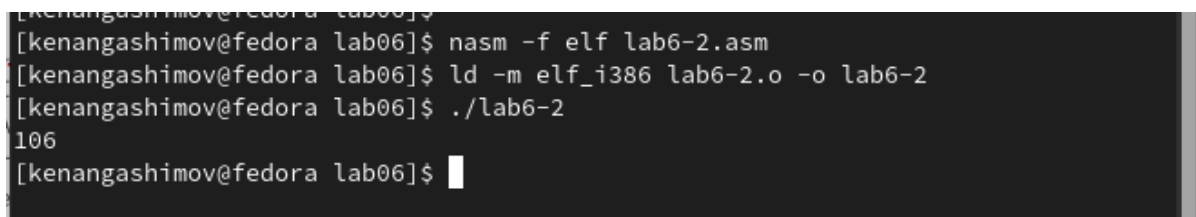
Как уже было отмечено ранее, в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Я преобразовал текст программы, используя эти функции.



```
lab6-2.asm
~/work/arch-pc/lab06

%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: Редактирование файла lab6-2.asm

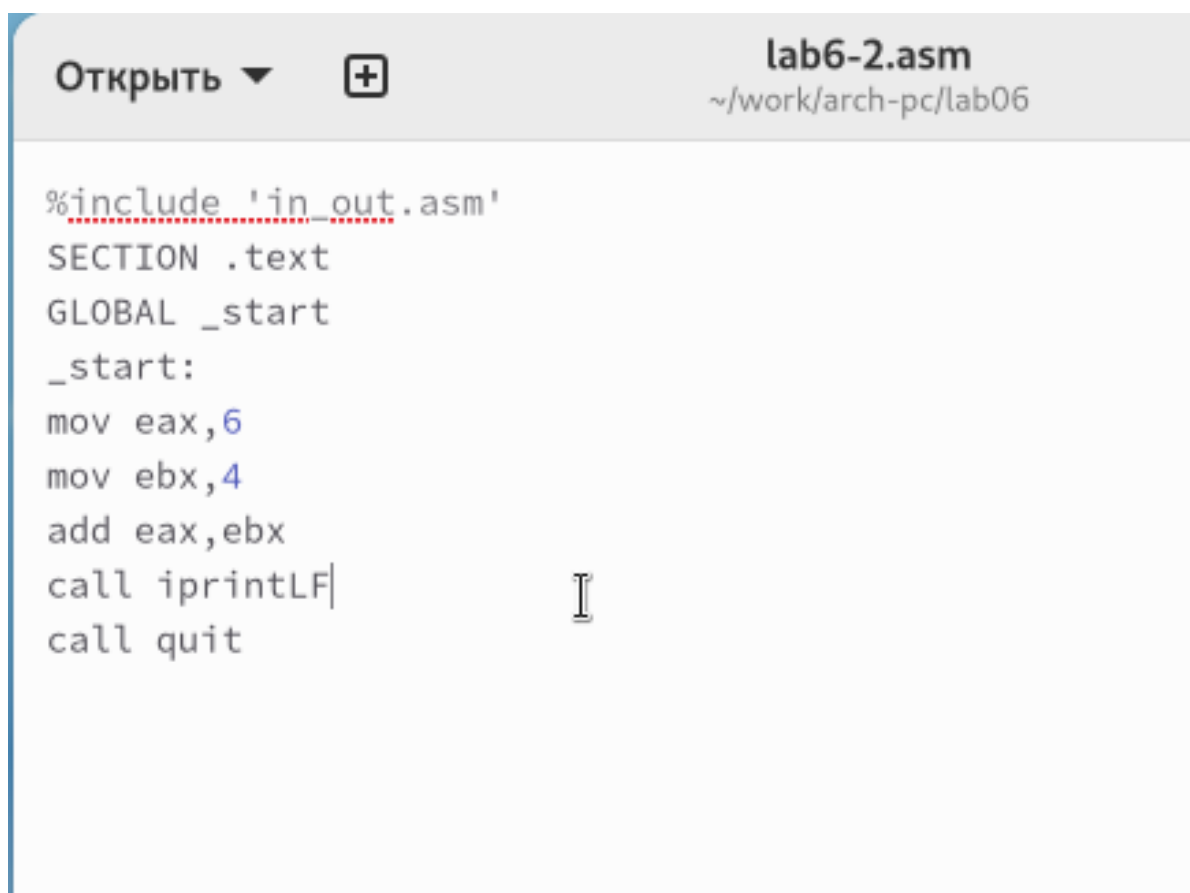


```
[kenangashimov@fedora lab06]$ nasm -f elf lab6-2.asm
[kenangashimov@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[kenangashimov@fedora lab06]$ ./lab6-2
106
[kenangashimov@fedora lab06]$
```

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

В результате выполнения программы мы получим число 106. В данном случае, как и в первом случае, 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 изменим символы на числа.

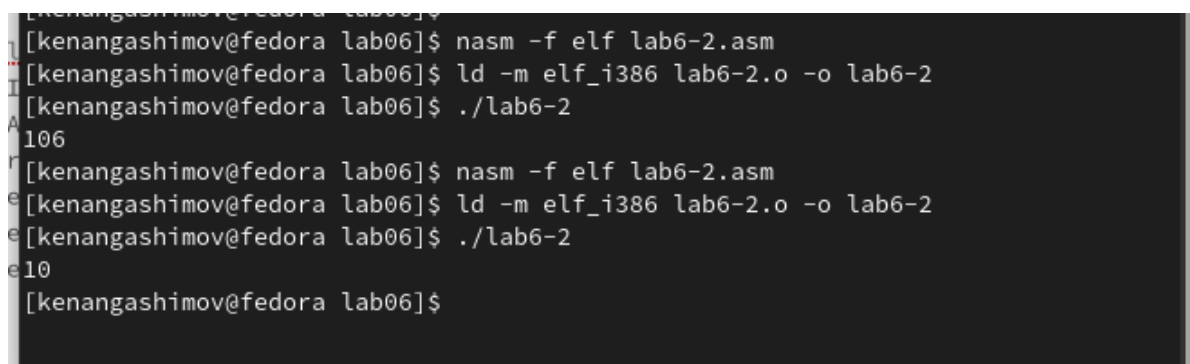


```
Открыть ▾ + lab6-2.asm
~/work/arch-pc/lab06

%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: Редактирование файла lab6-2.asm

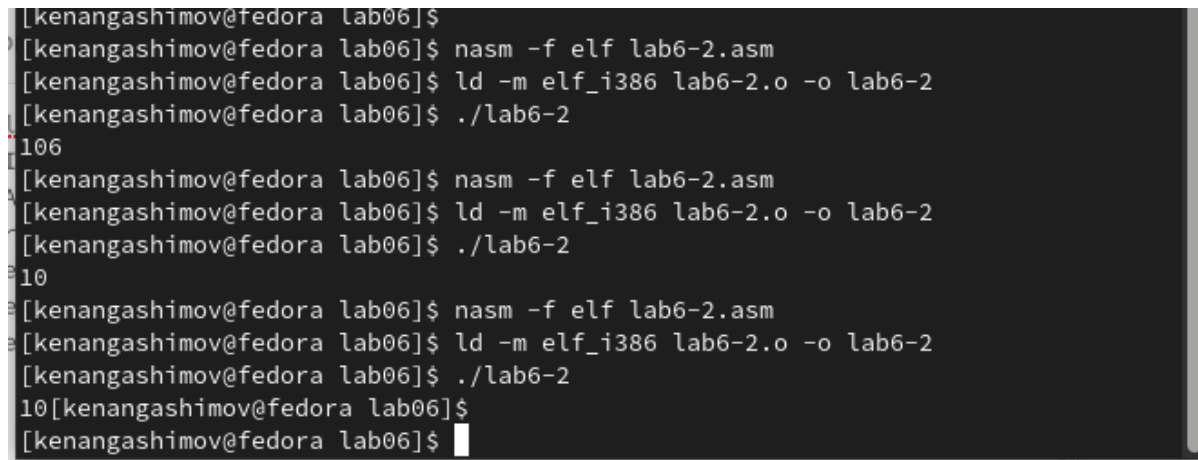
Функция `iprintLF` позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10.



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

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

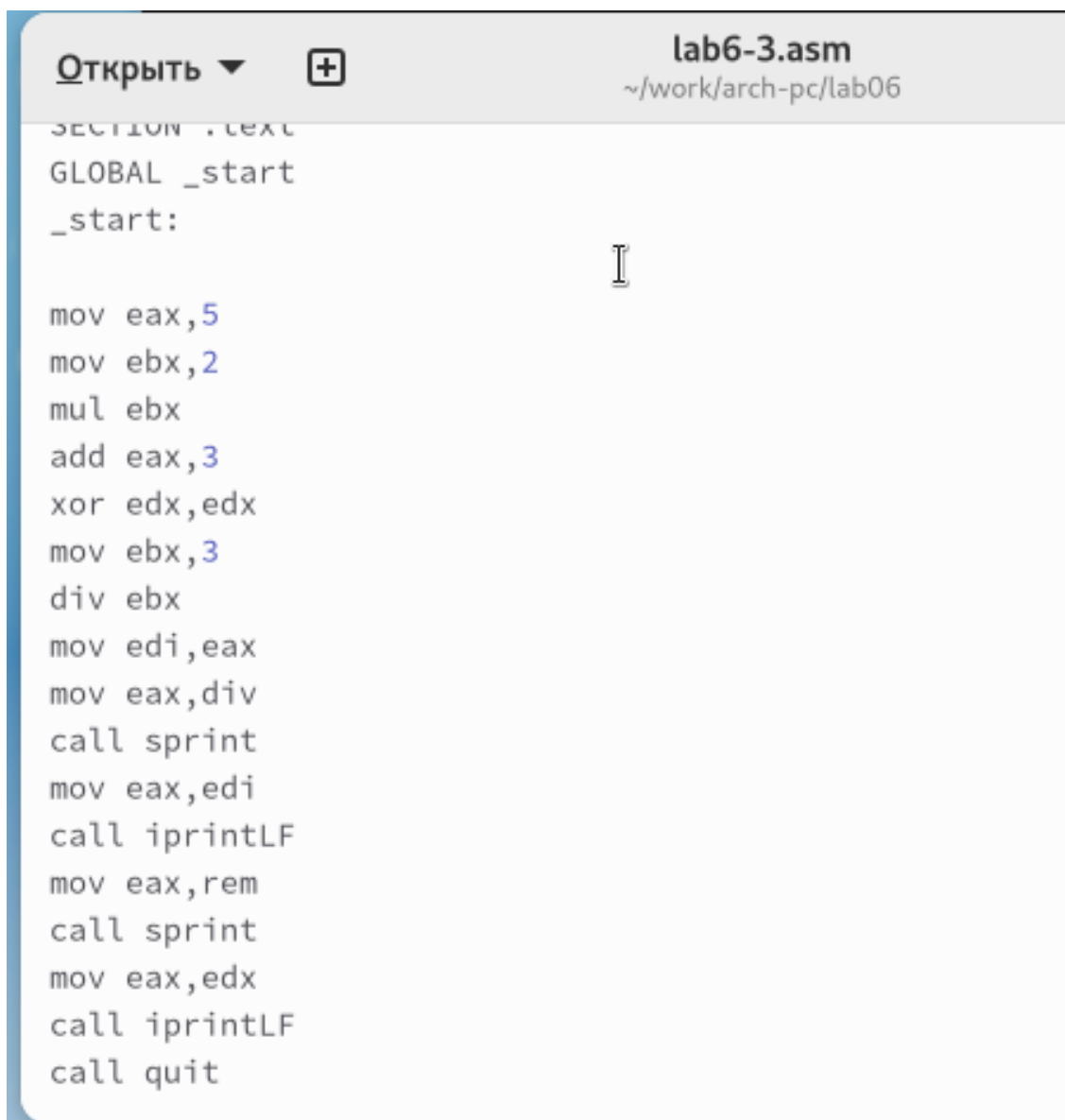
Заменяю функцию `iprintLF` на `iprint`. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки.



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

Рис. 2.9: Компиляция и проверка программы программы `lab6-2.asm`

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3) / 3$.



```
SECTION .text
GLOBAL _start
_start:

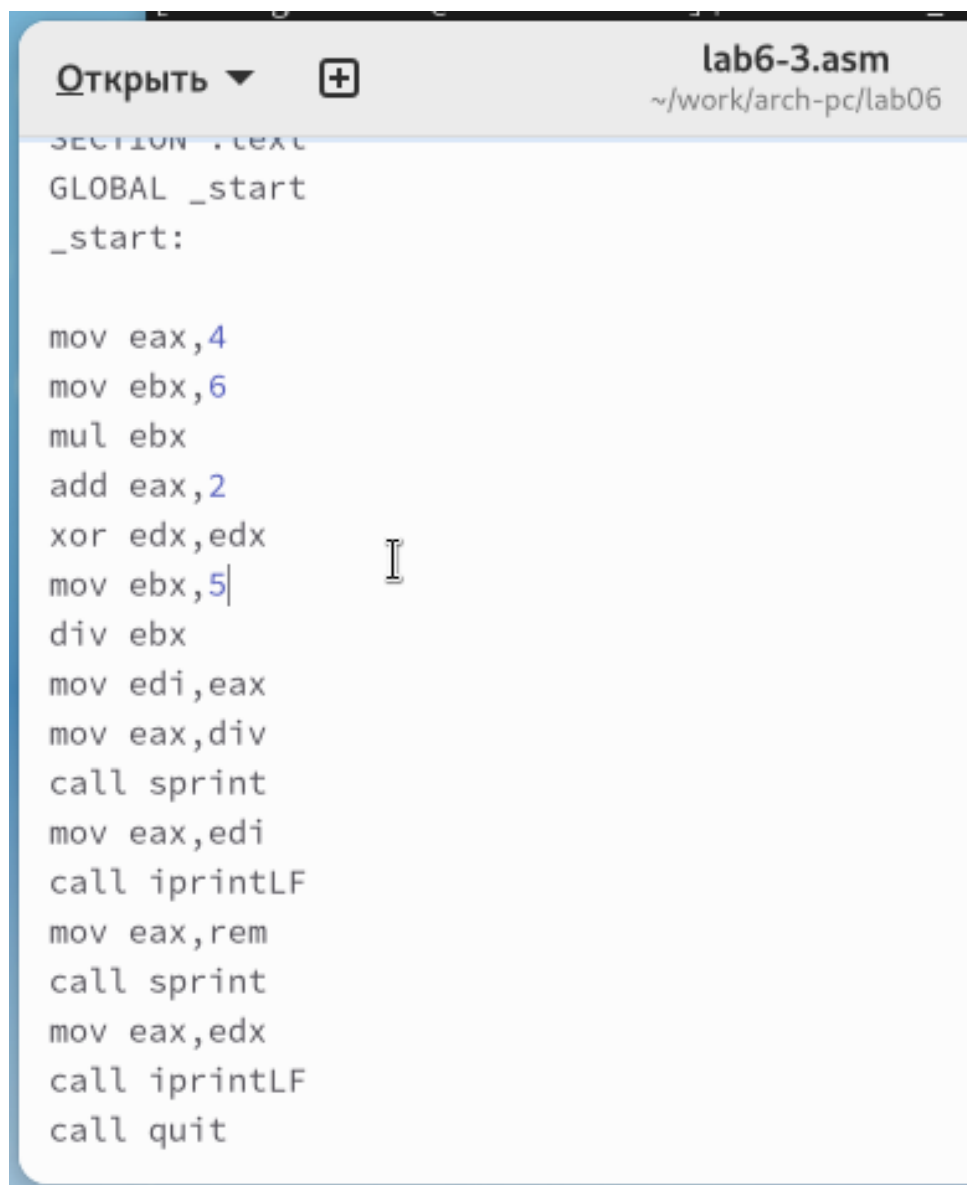
mov eax,5
mov ebx,2
mul ebx
add eax,3
xor edx,edx
mov ebx,3
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Рис. 2.10: Редактирование файла lab6-3.asm

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

Рис. 2.11: Компиляция и проверка программы программы lab6-3.asm

Изменил текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$.
Создал исполняемый файл и проверил его работу.



```
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
```

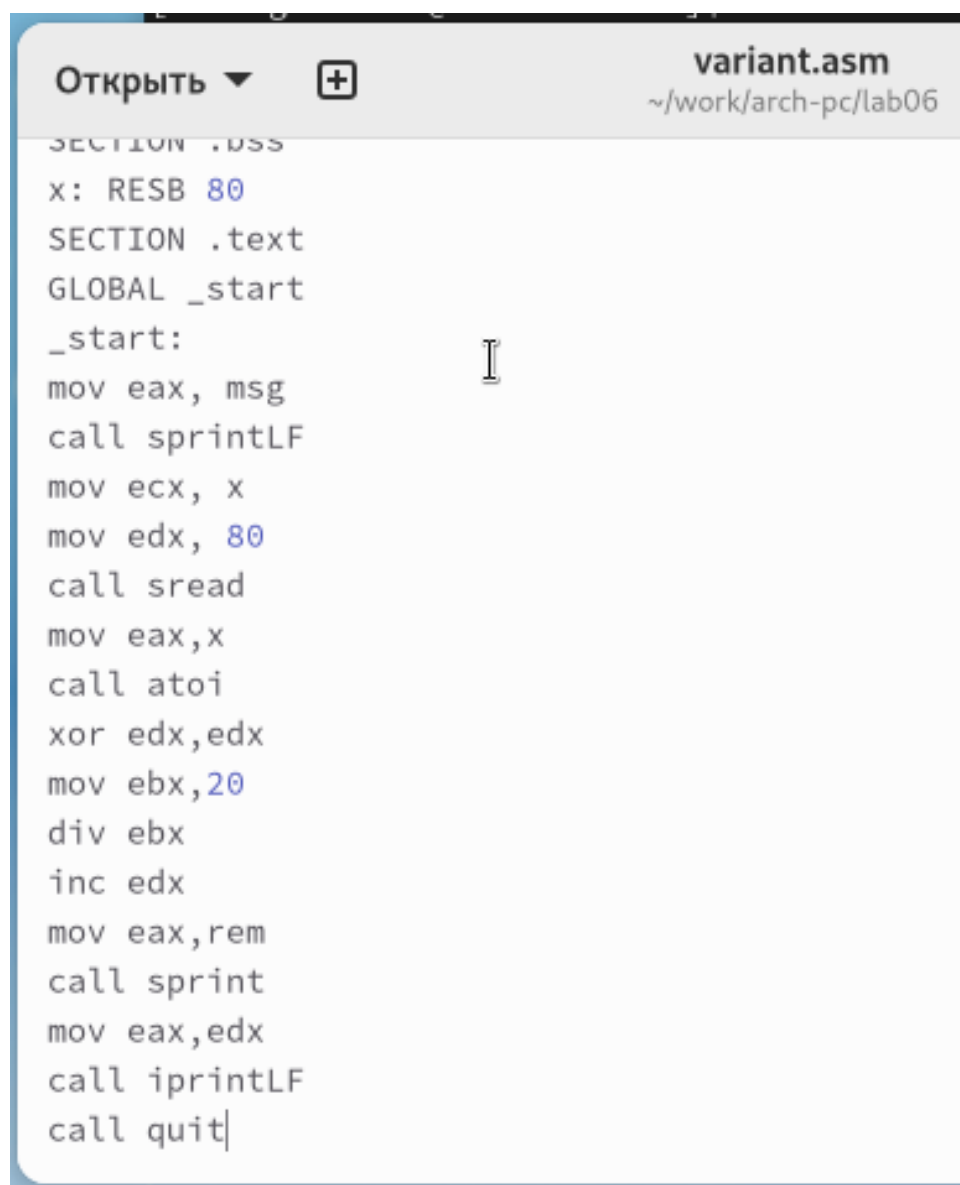
Рис. 2.12: Редактирование файла lab6-3.asm


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

Рис. 2.13: Компиляция и проверка программы программы lab6-3.asm

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

В данном случае число, над которым нужно выполнить арифметические операции, вводится с клавиатуры. Как было отмечено ранее, ввод с клавиатуры осуществляется в символьном виде, и для правильной работы арифметических операций в NASM символы должны быть преобразованы в числа. Для этой цели можно использовать функцию `atoi` из файла `in_out.asm`.



```
variant.asm
~/work/arch-pc/lab06

Открыть ▼ +

SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
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 sprintf
mov eax, edx
call iprintLF
call quit
```

Рис. 2.14: Редактирование файла variant.asm

```
[kenangashimov@fedora lab06]$  
[kenangashimov@fedora lab06]$ nasm -f elf variant.asm  
[kenangashimov@fedora lab06]$ ld -m elf_i386 variant.o -o variant  
[kenangashimov@fedora lab06]$ ./variant  
Введите № студенческого билета:  
1032235820  
Ваш вариант: 1  
[kenangashimov@fedora lab06]$  
[kenangashimov@fedora lab06]$
```

Рис. 2.15: Компиляция и проверка программы программы variant.asm

ответы на вопросы

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

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

Строка “call sprint” вызывает подпрограмму для вывода строки.

2. Для чего используются следующие инструкции?

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

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

Инструкция “call sread” вызывает подпрограмму для считывания значения студенческого билета из консоли.sread

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, согласно формуле вычисления варианта.

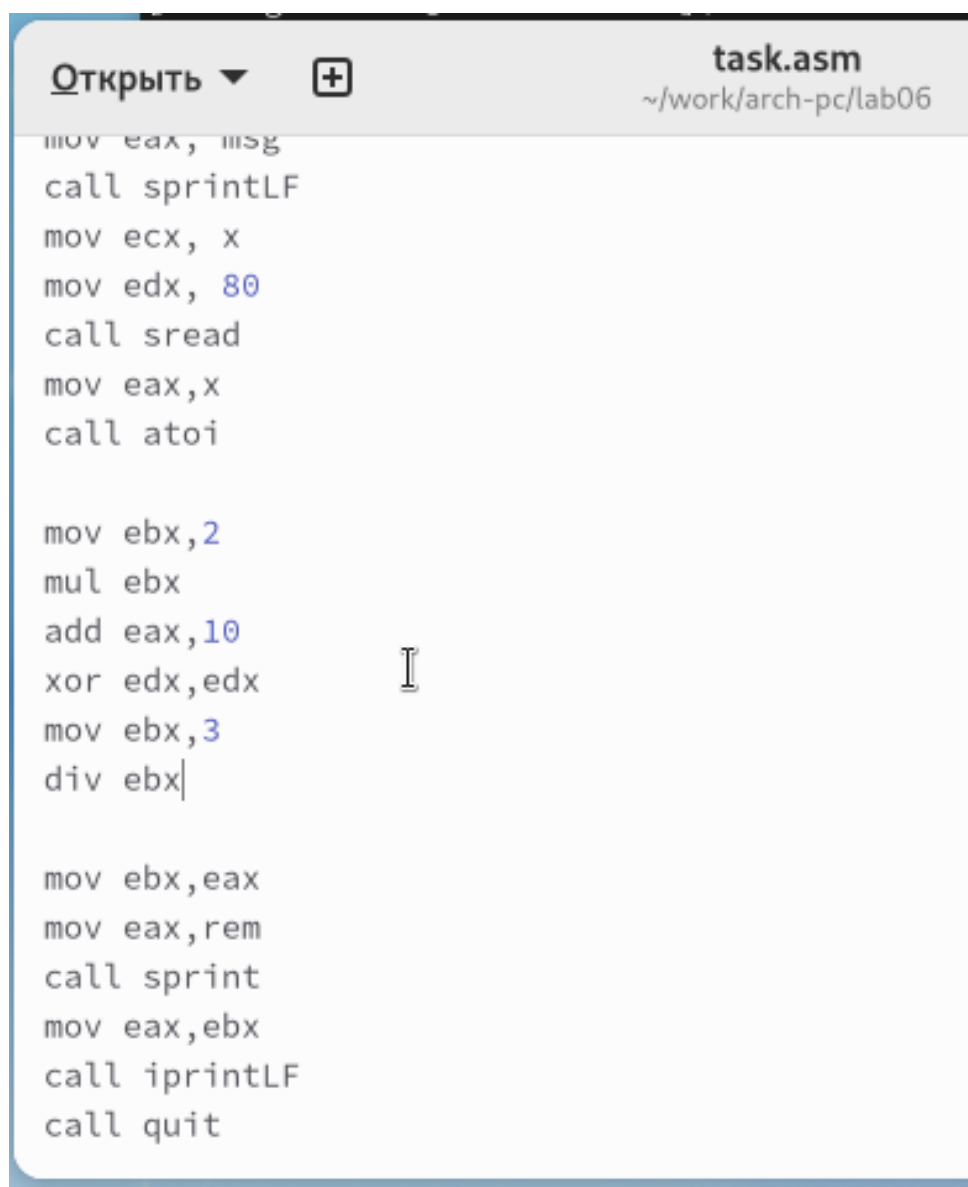
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

Строка “mov eax, edx” перекладывает результат вычислений в регистр eax.

Строка “call iprintLF” вызывает подпрограмму для вывода значения на экран.

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 1 - $(10 + 2x)/3$ для $x = 1, x = 10$



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

mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi

mov ebx, 2
mul ebx
add eax, 10
xor edx, edx
mov ebx, 3
div ebx

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

Рис. 2.16: Редактирование файла task.asm

Также размещаю код программы в отчете.

```
[kenangashimov@fedora lab06]$  
[kenangashimov@fedora lab06]$ nasm -f elf task.asm  
[kenangashimov@fedora lab06]$ ld -m elf_i386 task.o -o task  
[kenangashimov@fedora lab06]$ ./task  
Введите X  
1  
выражение = : 4  
[kenangashimov@fedora lab06]$ ./task  
Введите X  
10  
выражение = : 10  
[kenangashimov@fedora lab06]$
```

Рис. 2.17: Компиляция и проверка программы программы task.asm

3 Выводы

Изучили работу с арифметическими операциями.