

# **Арифметические операции NASM**

**Дисциплина: Архитектура компьютера**

Швед Карина Дмитриевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Ход лабораторной</b>	<b>6</b>
<b>3</b>	<b>Ответы на вопросы</b>	<b>14</b>
<b>4</b>	<b>Задание для самостоятельной работы</b>	<b>16</b>
<b>5</b>	<b>Выводы</b>	<b>18</b>

## Список иллюстраций

2.1	Создание каталога для лабораторной 6 и файла asm . . . . .	6
2.2	код программы lab6-1.asm . . . . .	7
2.3	создание исполняемого файла lab6-1.asm . . . . .	7
2.4	работа программы lab6-1.asm . . . . .	8
2.5	код программы lab6-2.asm . . . . .	9
2.6	запуск программы lab6-2.asm . . . . .	9
2.7	программа в файле lab6-2.asm . . . . .	10
2.8	работа программы lab6-2.asm . . . . .	10
2.9	работа программы lab6-2.asm . . . . .	10
2.10	работа программы lab6-3.asm . . . . .	11
2.11	код программы lab6-3.asm . . . . .	11
2.12	работа программы lab6-3.asm . . . . .	12
2.13	код программы variant.asm . . . . .	13
2.14	работа программы variant.asm . . . . .	13
4.1	код программы task.asm . . . . .	17
4.2	работа программы task.asm . . . . .	17

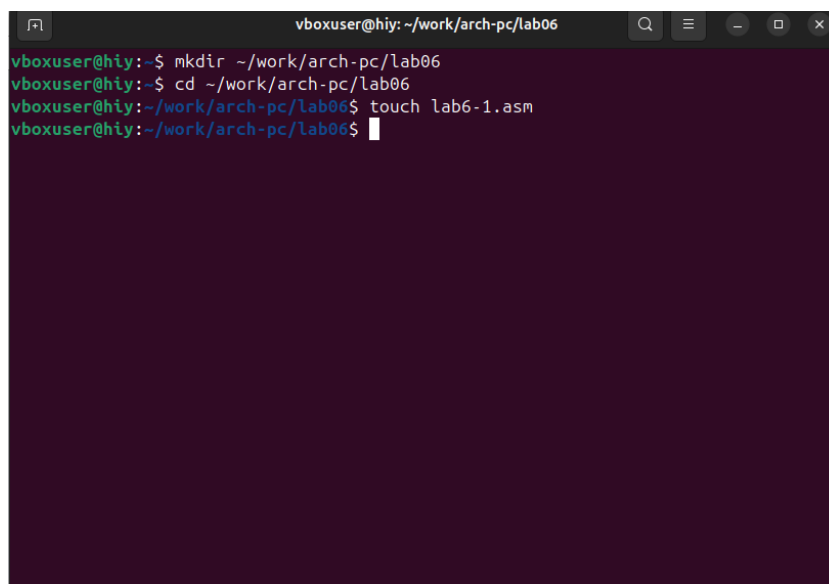
## **Список таблиц**

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

## 2 Ход лабораторной

Я создала каталог для программ лабораторной работы № 6, перешла в него и создала файл lab6-1.asm: `mkdir ~/work/arch-pc/lab06 cd ~/work/arch-pc/lab06 touch lab6-1.asm` (рис. 2.1).

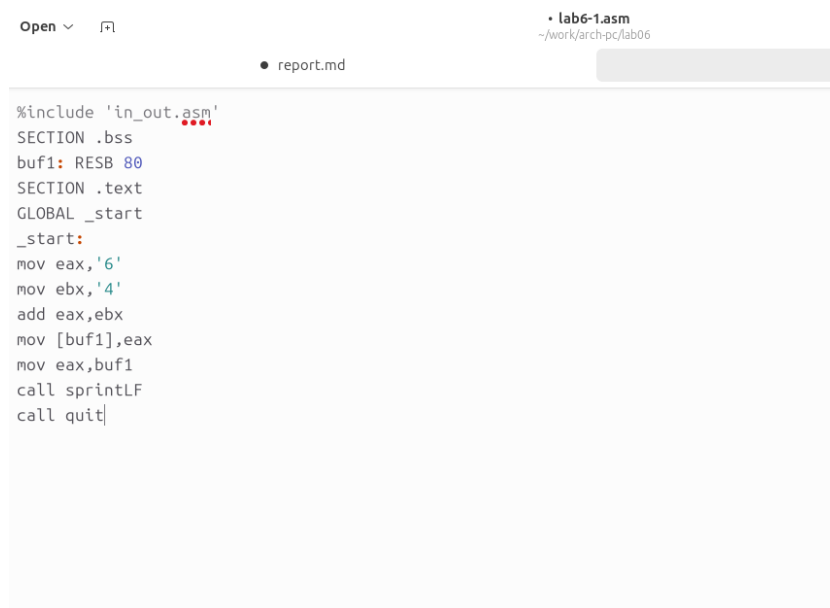
A screenshot of a terminal window with a dark purple background. The window title is 'vboxuser@hiy: ~/work/arch-pc/lab06'. The terminal shows the following commands and their outputs: 

```
vboxuser@hiy:~$ mkdir ~/work/arch-pc/lab06
vboxuser@hiy:~$ cd ~/work/arch-pc/lab06
vboxuser@hiy:~/work/arch-pc/lab06$ touch lab6-1.asm
vboxuser@hiy:~/work/arch-pc/lab06$
```

Рис. 2.1: Создание каталога для лабораторной 6 и файла asm

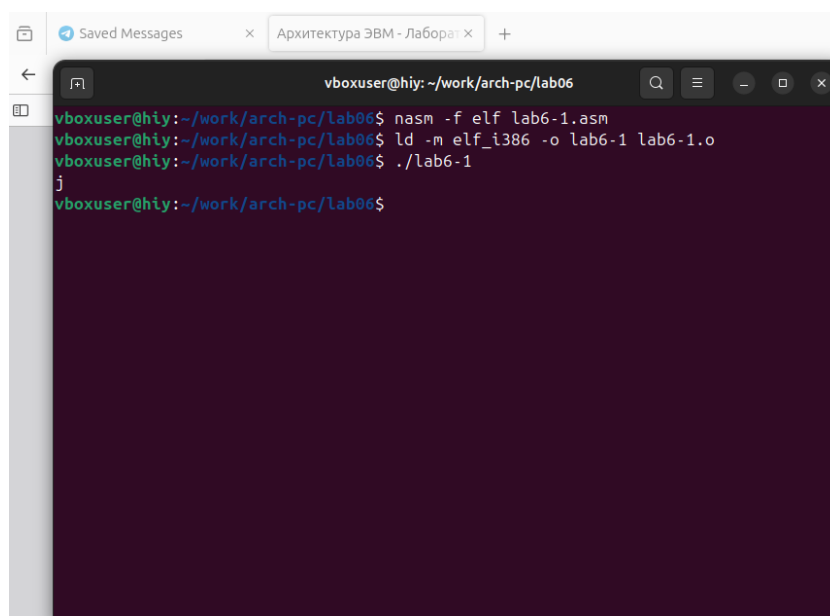
Далее я ввела в файл lab6-1.asm текст программы (рис. 2.2) из листинга 6.1. Перед созданием исполняемого файла я создала копию файла in\_out.asm в каталоге ~/work/arch-pc/lab06. Далее создала исполняемый файл и запустила его. (рис. 2.3)

На экране я увидела символ j, так как код символа 6 равен 00110110 в двоичном представлении, а код символа 4 – . Команда `add eax,ebx` запишет в регистр `eax` сумму кодов – 01101010, что в свою очередь является кодом символа j согласно таблице ASCII.



```
Open ▾  report.md  • 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. 2.2: код программы lab6-1.asm



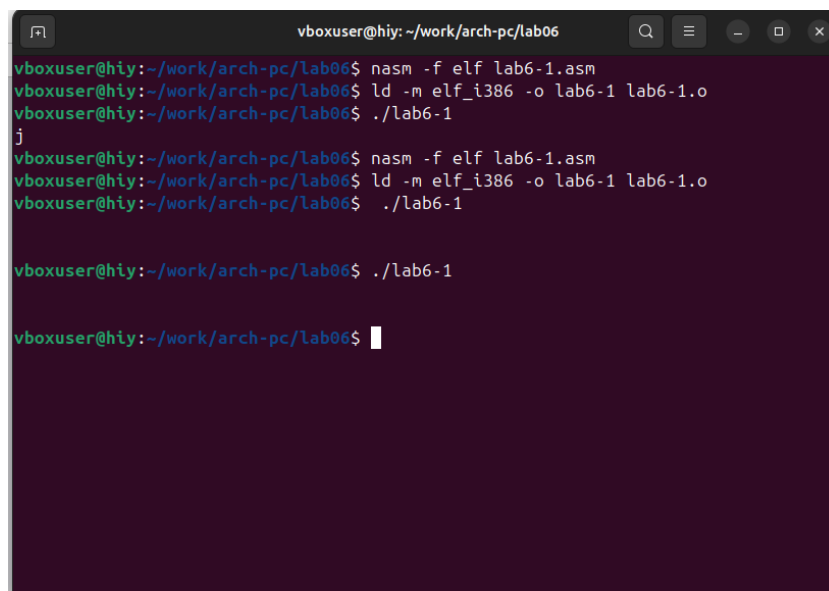
```
 Saved Messages  x  Архитектура ЭВМ - Лаборатория  x  +  
vboxuser@hiy: ~/work/arch-pc/lab06  
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm  
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o  
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-1  
j  
vboxuser@hiy:~/work/arch-pc/lab06$
```

Рис. 2.3: создание исполняемого файла lab6-1.asm

Далее я изменила текст программы и вместо символов записала в регистры числа. Я заменила строки:

mov eax,'6' mov ebx,'4' на строки mov eax,6 mov ebx,4

Создала исполняемый файл и запустила его (рис. 2.4). Как и в предыдущем случае, при выполнении программы я не получила число 10. Вместо этого выводится символ с кодом 10, который представляет собой символ конца строки (возврат каретки). Этот символ не отображается на экране, но он добавляет пустую строку.



```
vboxuser@hiy: ~/work/arch-pc/lab06
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-1
j
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-1

vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-1

vboxuser@hiy:~/work/arch-pc/lab06$
```

Рис. 2.4: работа программы lab6-1.asm

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 из Листинга 6.1 с использованием этих функций. Создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввела в него текст программы из листинга 6.2 (рис. 2.5), создала исполняемый файл и запустила его (рис. 2.6).



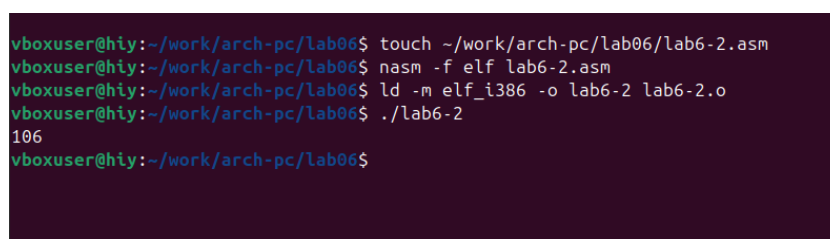


```
lab6-2.asm
~/work/arch-pc/lab06
lab6-1.asm

report.md

#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: код программы lab6-2.asm



```
vboxuser@hiy:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-2
106
vboxuser@hiy:~/work/arch-pc/lab06$
```

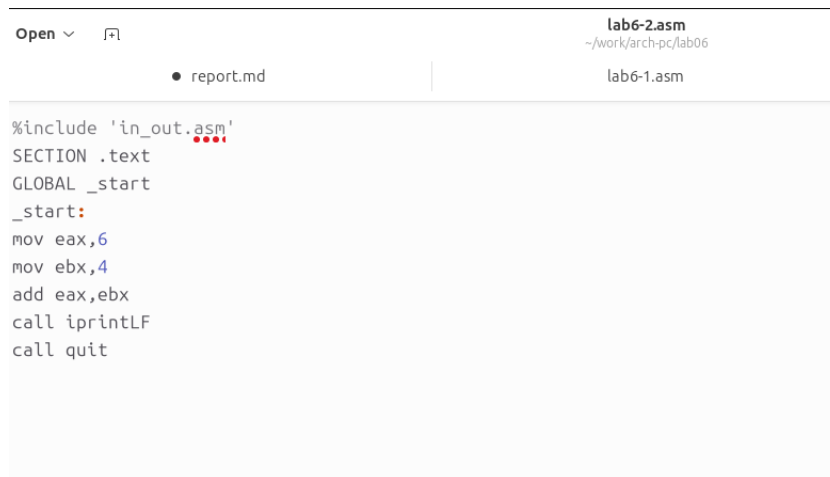
Рис. 2.6: запуск программы lab6-2.asm

В результате работы программы я получила число 106. В данном случае, как и в первом, команда add складывает коды символов '6' и '4' ( $54+52=106$ ). Однако, в отличие от программы из листинга 6.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру я изменила символы на числа. Я заменила строки (рис. 2.7).

```
mov eax, '6' mov ebx, '4' на строки mov eax, 6 mov ebx, 4
```

Создала исполняемый файл и запустила его (рис. 2.8). Функция iprintLF позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10.

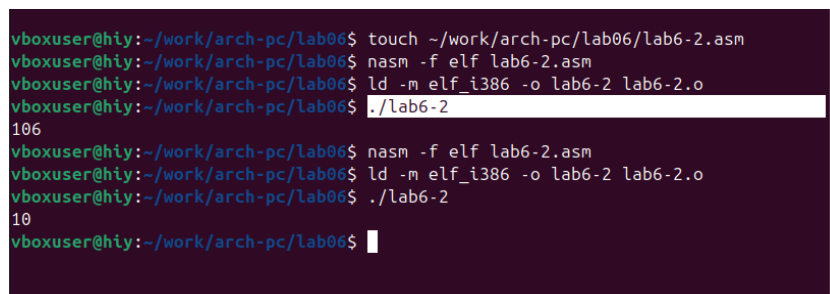


```
lab6-2.asm
~/work/arch-pc/lab06
lab6-1.asm

● report.md

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

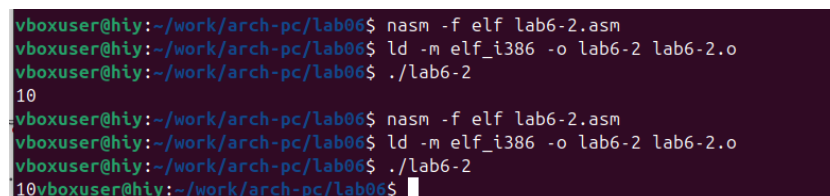
Рис. 2.7: программа в файле lab6-2.asm



```
vboxuser@hiy:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-2
106
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-2
10
vboxuser@hiy:~/work/arch-pc/lab06$
```

Рис. 2.8: работа программы lab6-2.asm

Я заменила функцию `iprintLF` на `iprint`. Создала исполняемый файл и запустила его (рис. 2.9). Вывод отличается тем, что нет переноса строки

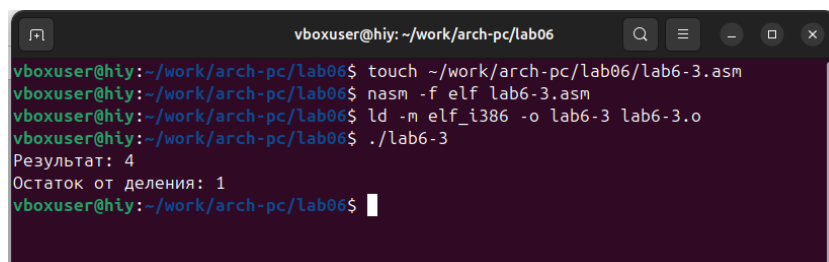


```
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-2
10
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-2
10
vboxuser@hiy:~/work/arch-pc/lab06$
```

Рис. 2.9: работа программы lab6-2.asm

В качестве примера выполнения арифметических операций в NASM приведу программу вычисления арифметического выражения  $\frac{5 \cdot 2 + 3}{3}$ . Я создала файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06`: `touch ~/work/arch-pc/lab06/lab6-3.asm` Внимательно изучила текст программы из листинга 6.3 и ввела в `lab6-`

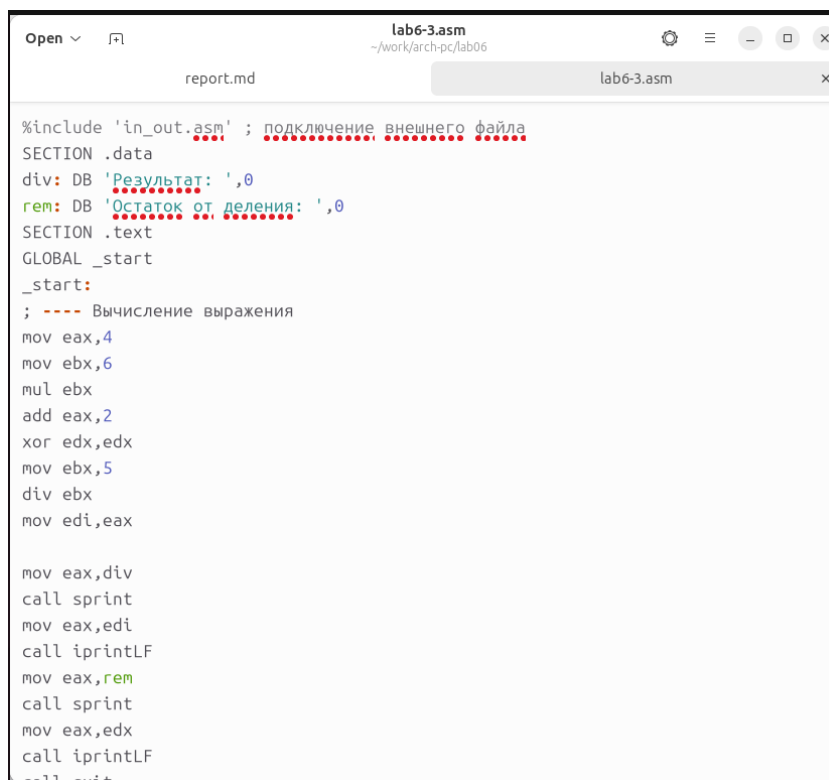
3.asm. Создала исполняемый файл и запустила его. Результат работы программы получился следующим (рис. 2.10).



```
vboxuser@hiy: ~/work/arch-pc/lab06
vboxuser@hiy:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
vboxuser@hiy:~/work/arch-pc/lab06$
```

Рис. 2.10: работа программы lab6-3.asm

Далее я изменила текст программы (рис. 2.11) для вычисления выражения  $f(x) = (4 \cdot 6 + 2)/5$ . Создала исполняемый файл и проверила его работу (рис. 2.12)



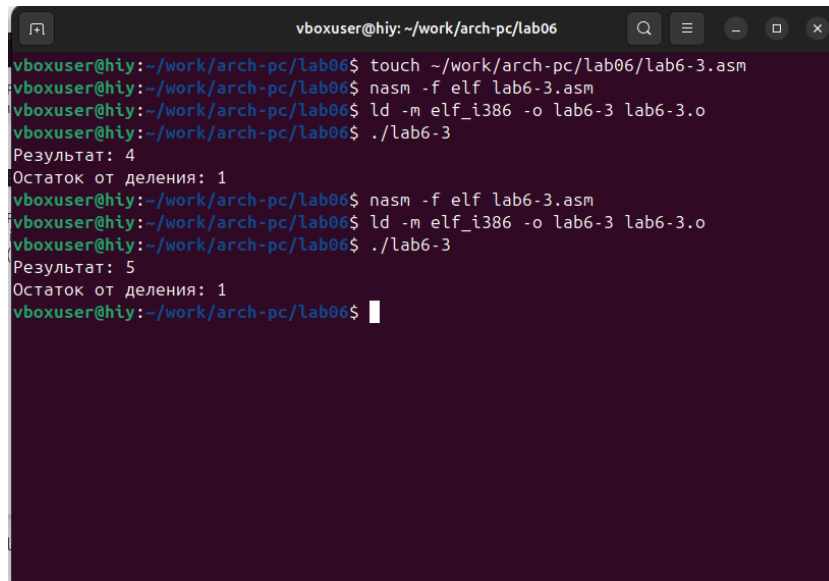
```
lab6-3.asm
~/work/arch-pc/lab06

report.md
lab6-3.asm

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
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. 2.11: код программы lab6-3.asm

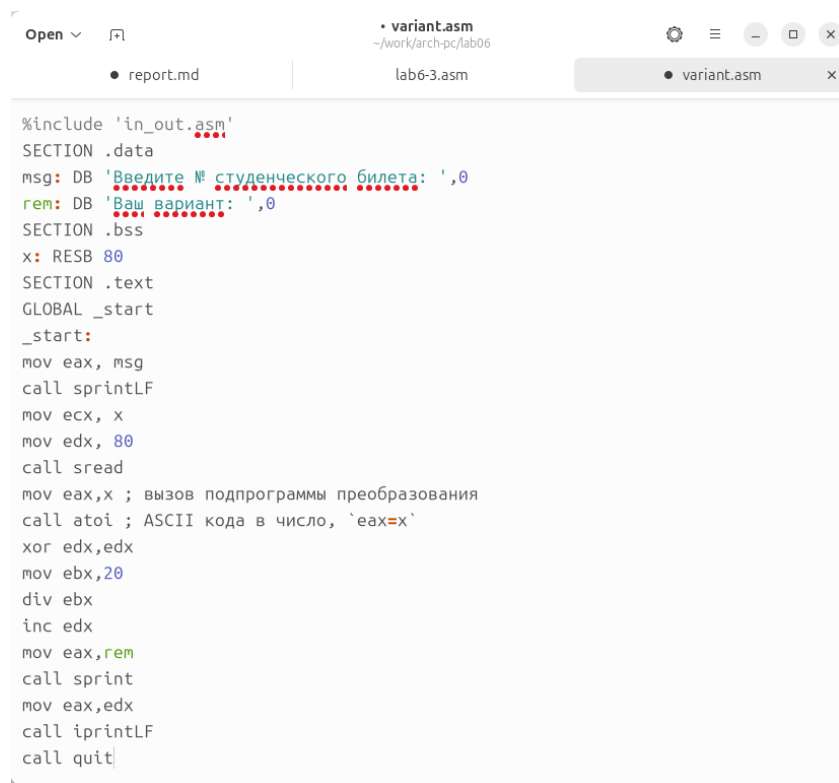


```
vboxuser@hiy: ~/work/arch-pc/lab06
vboxuser@hiy:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
vboxuser@hiy:~/work/arch-pc/lab06$
```

Рис. 2.12: работа программы lab6-3.asm

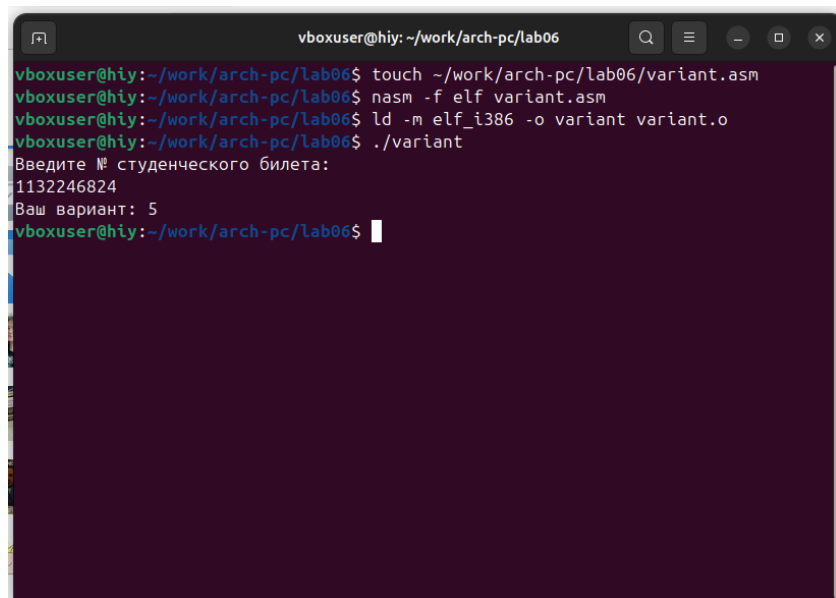
В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция `atoi` из файла `in_out.asm`.

Я создала файл `variant.asm` в каталоге `~/work/arch-pc/lab06`: `touch ~/work/arch-pc/lab06/variant.asm` Внимательно изучила текст программы из листинга 6.4 и ввела в файл `variant.asm`. (рис. 2.13). Создала исполняемый файл и запустила его. Проверила результат работы программы вычислив номер варианта аналитически (рис. 2.14)



```
%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 sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprintf
mov eax, edx
call iprintLF
call quit
```

Рис. 2.13: код программы variant.asm



```
vboxuser@hiy: ~/work/arch-pc/lab06
vboxuser@hiy:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf variant.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246824
Ваш вариант: 5
vboxuser@hiy:~/work/arch-pc/lab06$
```

Рис. 2.14: работа программы variant.asm

### 3 Ответы на вопросы

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? Строка “mov eax, ptr” перекладывает в регистр значение переменной с фразой “Ваш вариант:” Строка “call sprint” вызывает подпрограмму вывода строки
2. Для чего используются следующие инструкции? mov ecx, x Загружает адрес переменной x в регистр ECX. mov edx, 80 Загружает значение 80 в регистр EDX. 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 6.4 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

## 4 Задание для самостоятельной работы

Формулировка задания: 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

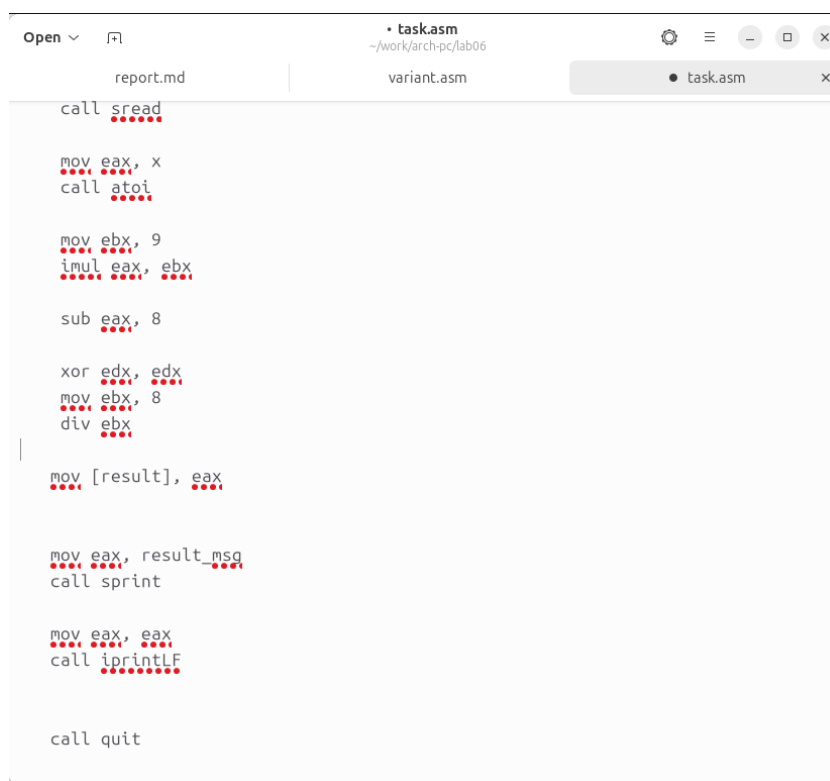
Я получила вариант 5:

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

для  $x_1=8$   $x_2=64$

Сначала я создала отдельный файл `task.asm` и ввела код программы (рис. 4.1), затем получила исполняемый файл и проверила работу программы, введя значения  $x_1$  и  $x_2$  (рис. 4.2)





```
call sread

mov eax, x
call atoi

mov ebx, 9
imul eax, ebx

sub eax, 8

xor edx, edx
mov ebx, 8
div ebx

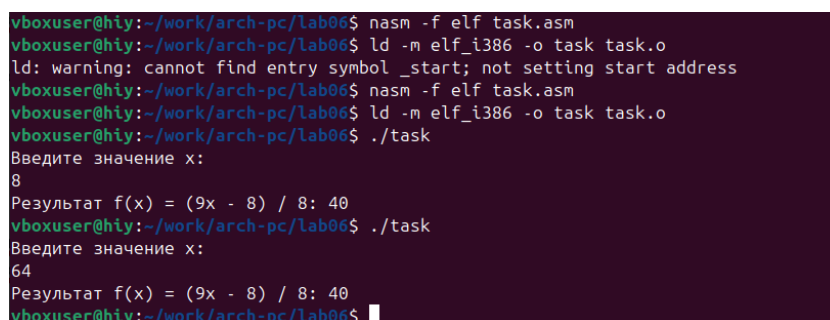
mov [result], eax

mov eax, result_msg
call sprint

mov eax, eax
call iprintLF

call quit
```

Рис. 4.1: код программы task.asm



```
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf task.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o task task.o
ld: warning: cannot find entry symbol _start; not setting start address
vboxuser@hiy:~/work/arch-pc/lab06$ nasm -f elf task.asm
vboxuser@hiy:~/work/arch-pc/lab06$ ld -m elf_i386 -o task task.o
vboxuser@hiy:~/work/arch-pc/lab06$ ./task
Введите значение x:
8
Результат f(x) = (9x - 8) / 8: 40
vboxuser@hiy:~/work/arch-pc/lab06$ ./task
Введите значение x:
64
Результат f(x) = (9x - 8) / 8: 40
vboxuser@hiy:~/work/arch-pc/lab06$
```

Рис. 4.2: работа программы task.asm

## **5 Выводы**

Я освоила арифметические инструкции языка ассемблера NASM