

# **Лабораторная работа №6.**

**Арифметические операции в NASM.**

Казначеев Сергей Ильич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Выводы</b>	<b>18</b>

# Список иллюстраций

2.1	scr1	6
2.2	scr2	7
2.3	scr3	8
2.4	scr4	8
2.5	scr5	9
2.6	scr6	10
2.7	scr7	10
2.8	scr8	11
2.9	scr9	11
2.10	scr10	12
2.11	scr10	12
2.12	scr10	12
2.13	scr10	13
2.14	scr10	14
2.15	scr10	14
2.16	scr10	14
2.17	scr10	15
2.18	scr10	15
2.19	scr11	17
2.20	scr12	17

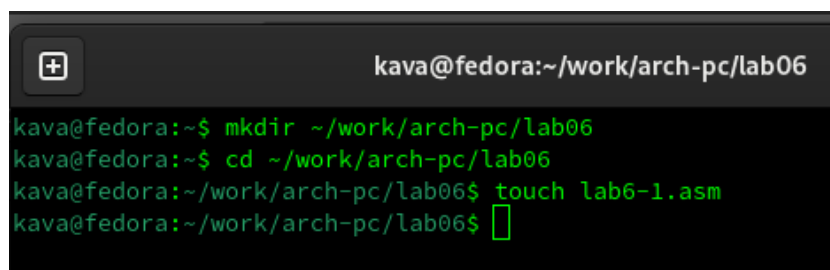
## **Список таблиц**

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

Для начала я создал папку с названием lab06 и файл lab6-1.asm

A terminal window with a dark background. The title bar shows a window icon and the text 'kava@fedora:~/work/arch-pc/lab06'. The terminal contains four lines of text: 'kava@fedora:~\$ mkdir ~/work/arch-pc/lab06', 'kava@fedora:~\$ cd ~/work/arch-pc/lab06', 'kava@fedora:~/work/arch-pc/lab06\$ touch lab6-1.asm', and 'kava@fedora:~/work/arch-pc/lab06\$' followed by a green cursor.

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

Рис. 2.1: scr1

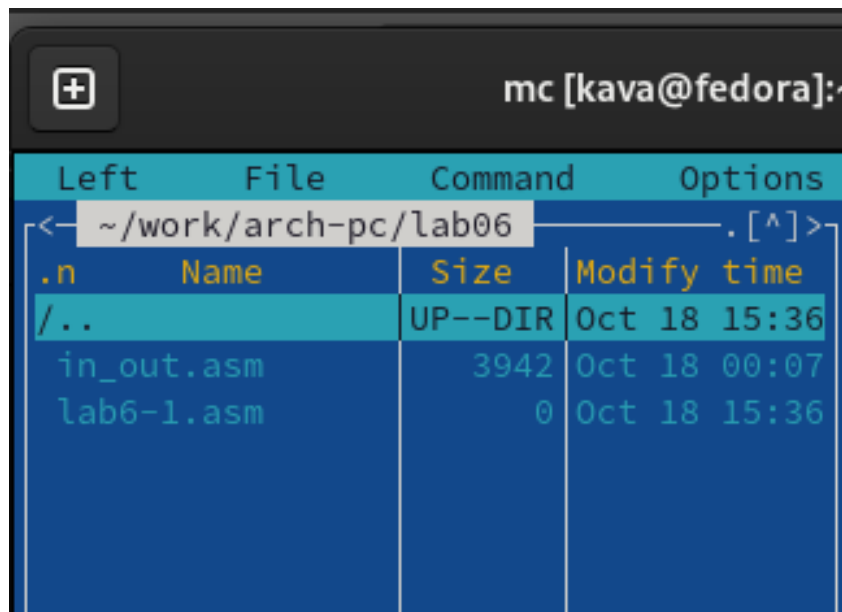
Далее заходим в папку и открываем только что созданный файл и вставляем код из листинга 6.1

```
GNU nano 7.2 /home/  
%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: scr2

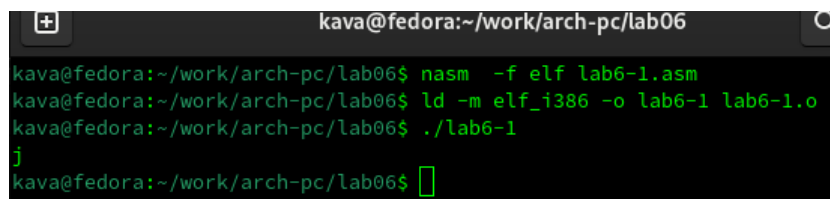
После чего копируем файл in\_out.asm



Left	File	Command	Options
<-	~/work/arch-pc/lab06		. [^]>
.n	Name	Size	Modify time
/..		UP--DIR	Oct 18 15:36
	in_out.asm	3942	Oct 18 00:07
	lab6-1.asm	0	Oct 18 15:36

Рис. 2.3: scr3

Теперь соберем наш файл и запустим его мы увидим что вывелось j а нам нужно вывести сумму 6 и 4, и чтобы вывелось число 10



```

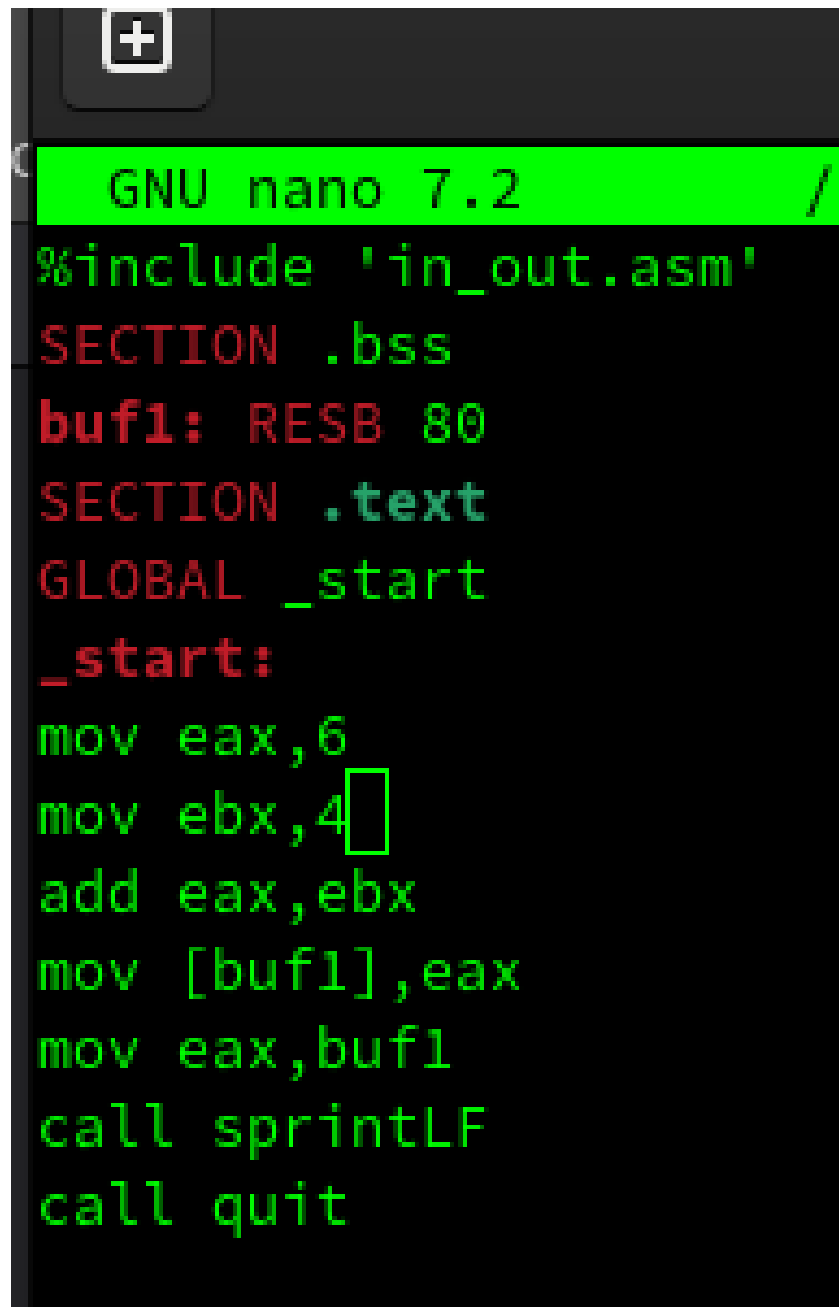
kava@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
kava@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
kava@fedora:~/work/arch-pc/lab06$ ./lab6-1
j
kava@fedora:~/work/arch-pc/lab06$

```

Рис. 2.4: scr4

Чтобы исправить это нам нужно убрать кавычки, теперь мы будем складывать числа, а не символы

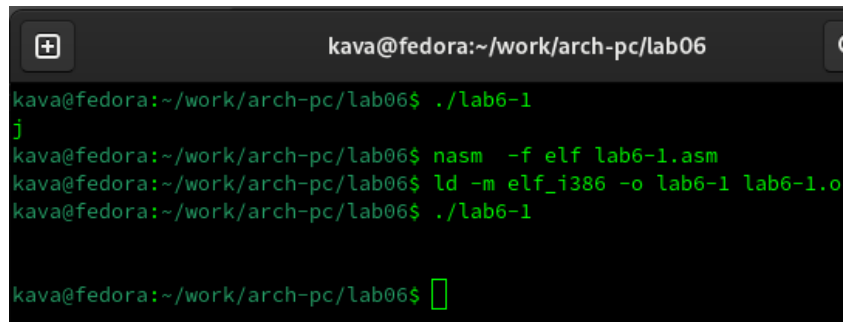




```
GNU nano 7.2 /  
%include 'in_out.asm'  
SECTION .bss  
buf1: RESB 80  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
mov [buf1],eax  
mov eax,buf1  
call sprintLF  
call quit
```

Рис. 2.5: scr5

После исправлений запустим файл. Увидим, что ничего не вывелось. Это произошло из-за того, что мы выводим символы, а не число.

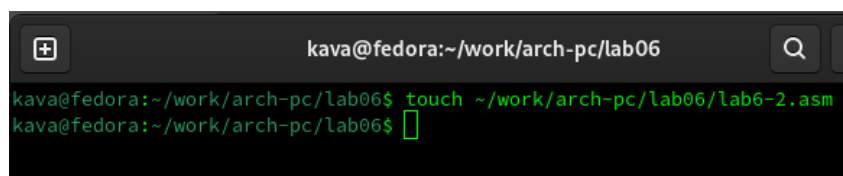
A terminal window with a dark background and green text. The title bar shows 'kava@fedora:~/work/arch-pc/lab06'. The terminal contains the following commands and output:

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

kava@fedora:~/work/arch-pc/lab06$ █
```

Рис. 2.6: scr6

Теперь создадим файл lab6-2.asm

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

```
kava@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
kava@fedora:~/work/arch-pc/lab06$ █
```

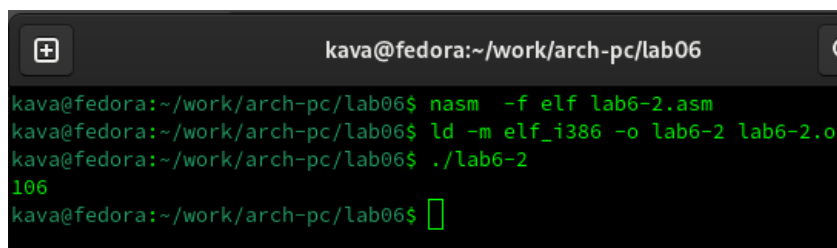
Рис. 2.7: scr7

После вставим в него код из листинга 6.2

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

Рис. 2.8: scr8

Он выведет нам 106 это произойдет, так как у нас числа стоят в кавычках и мы складываем их коды (54+52=106)



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

Рис. 2.9: scr9

Теперь,если мы уберем кавычки то у нас выведется 10

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

Рис. 2.10: scr10

Теперь посмотрим в чем разница между `iprintLF` и `iprint`

```
.../2023-2024/Архитект
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

{#fig:011width=70%}

Собираем программу и запускаем

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

Рис. 2.11: scr10

Мы увидим, что оперция `iprint` не переносит на следующую строку

Теперь создадим третий файл `lab6-3`

```
kava@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
kava@fedora:~/work/arch-pc/lab06$
```

Рис. 2.12: scr10

И вставляем код из файла листинга 6.3

```

GNU nano 7.2 /home/kava/work/arch-pc/lab06/lab
#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,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '

```

Рис. 2.13: scr10

```

kava@fedora:~/work/arch-pc/lab06$ ./scr10
Результат: 4
Остаток от деления: 1
kava@fedora:~/work/arch-pc/lab06$

```

Собираем программу и запускаем, и получаем верный результат

Теперь меняем файл так, чтобы мы могли посчитать значение выражения  $(4*6+2)/5$

```

%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 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения

```

Рис. 2.14: scr10

Собираем программу и запускаем, и получаем верный результат

```

kava@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
kava@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
kava@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
kava@fedora:~/work/arch-pc/lab06$ 

```

Рис. 2.15: scr10

Теперь создаем файл variat.asm

```

kava@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
kava@fedora:~/work/arch-pc/lab06$ 

```

Рис. 2.16: scr10

И вставляем код из файла листинга 6.4

```
kava@fedora:~/work/arch-pc/lab06
GNU nano 7.2 /home/kava/work/arch-pc/lab06/variant.asm
%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 ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx,edx
mov ebx,20
div ebx
inc edx
```

Рис. 2.17: scr10

Соберем и запустим ее

```
kava@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
kava@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
kava@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132240693
Ваш вариант: 14
kava@fedora:~/work/arch-pc/lab06$
```

Рис. 2.18: scr10

И нам выведется число 14,и это действительно так

Ответим на вопросы лабораторной работы

1 Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант:'?

За это отвечает 21 строчка кода call sprint перед которой идёт строка mov eax,rem, которая перемещает строку с фразой в регистр eax ,из которого мы считаем данные для вывода

2 Для чего используются следующие инструкции?

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

Эти инструкции используются для того, чтобы записать данные в переменную  
x

3 Для чего используется инструкция “call atoi”?

Для преобразования ASCII кода в число

4 Какие строки листинга 6.4 отвечают за вычисления варианта?

```
div ebx  
inc edx
```

Первая делит число x в регистре eax на значение ebx регистра, а вторая прибавляет к значению регистра edx единицу

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

В регистр edx

6 Для чего используется инструкция “inc edx”?

Для увеличения значения регистра edx на единицу

7 Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

```
mov eax,edx  
call iprintLF
```

Первая строка переносит значение регистра edx в eax, а вторая вызывает операцию вывода значения регистра eax

Задание для самостоятельной работы

Я написал программу, которая вычисляет пример под номером 10

Предварительно, я создал файл под именем task10.asm и написал следующий код



```

GNU nano 7.2 /home/kava/work/arch-pc/lab06/task10.asm
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov eax, msg2
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
add eax, 18
mov ebx, 5
mul ebx
sub eax, 28
call iprintLF
call quit

```

Рис. 2.19: scr11

И запустил код, в качестве x я указал число 5

```

kava@fedora:~/work/arch-pc/lab06$ nasm -f elf task10.asm
kava@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o task10 task10.o
kava@fedora:~/work/arch-pc/lab06$ ./task10
Выражение для вычисления 5(x+18)-28
5
87

```

Рис. 2.20: scr12

Как видим, программа работает исправно и правильно вычисляет выражения.

## **3 Выводы**

В результате выполнения лабораторной работы, я освоил арифметические операции которые есть в Ассемблере и как они работают. Здесь кратко описываются итоги проделанной работы.