

# **Отчёт по лабораторной работе 6**

**Арифметические операции в NASM.**

Власов Артем Сергеевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Символьные и численные данные в NASM . . . . .	6
3.2	Выполнение арифметических операций в NASM . . . . .	10
3.3	Ответы на вопросы по программе . . . . .	13
3.4	Задание для самостоятельной работы . . . . .	14
<b>4</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

3.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code> . . . . .	6
3.2	Заполняем файл . . . . .	6
3.3	Запускаем файл и смотрим на его работу . . . . .	7
3.4	Изменяем файл . . . . .	7
3.5	Запускаем файл и смотрим на его работу . . . . .	7
3.6	Создаем файл . . . . .	8
3.7	Заполняем файл . . . . .	8
3.8	Смотрим на работу программы . . . . .	8
3.9	Изменяем файл . . . . .	9
3.10	Смотрим на работу программы . . . . .	9
3.11	Изменяем файл . . . . .	9
3.12	Смотрим на работу программы . . . . .	10
3.13	Создаем файл . . . . .	10
3.14	Заполняем файл . . . . .	11
3.15	Смотрим на результат работы программы . . . . .	11
3.16	Редактируем файл . . . . .	12
3.17	Смотрим на результат работы программы . . . . .	12
3.18	Создаем файл . . . . .	12
3.19	Заполняем файл . . . . .	13
3.20	Проверяем результат работы программы . . . . .	13
3.21	Создаем файл . . . . .	14
3.22	Заполняем файл . . . . .	15
3.23	Проверяем работу программы . . . . .	15
3.24	Проверяем работу программы . . . . .	15

# 1 Цель работы

Освоить арифметических инструкций языка ассемблера NASM и написать программы для вычисления арифметических выражений с неизвестной.

## 2 Задание

Написать программы для решения выражений.

## 3 Выполнение лабораторной работы

### 3.1 Символьные и численные данные в NASM

Создаем каталог для программ ЛБ6, и в нем создаем файл (рис. fig. 3.1).

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

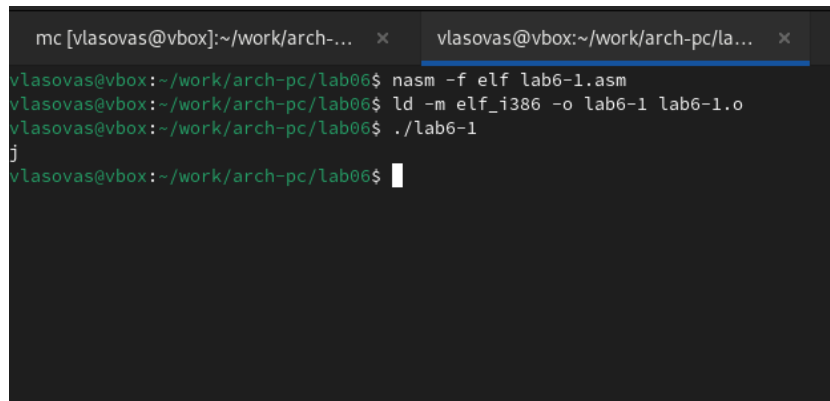
Рис. 3.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 6.1 (рис. fig. 3.2).

```
lab6-1.asm [-M--] 13 L:[ 1+10 11/ 13] *(155 / 181b) 0010 0x00A [*]
%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. 3.2: Заполняем файл

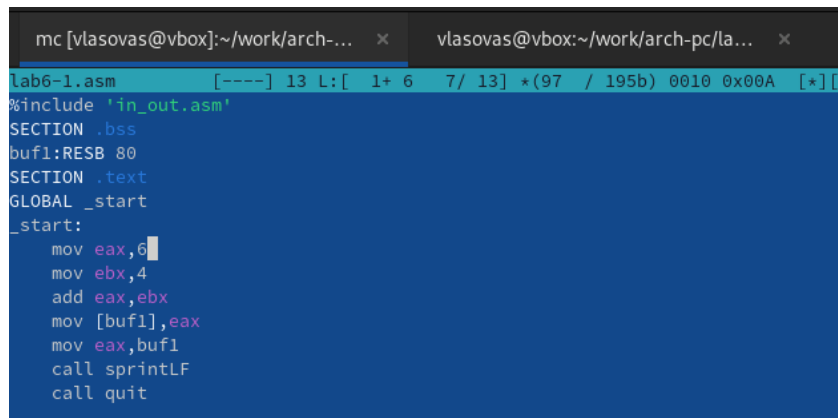
Создаем исполняемый файл и запускаем его (рис. fig. 3.3).



```
mc [vlasovas@vbox]:~/work/arch-... x  vlasovas@vbox:~/work/arch-pc/la... x
vlasovas@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
vlasovas@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
vlasovas@vbox:~/work/arch-pc/lab06$ ./lab6-1
j
vlasovas@vbox:~/work/arch-pc/lab06$
```

Рис. 3.3: Запускаем файл и смотрим на его работу

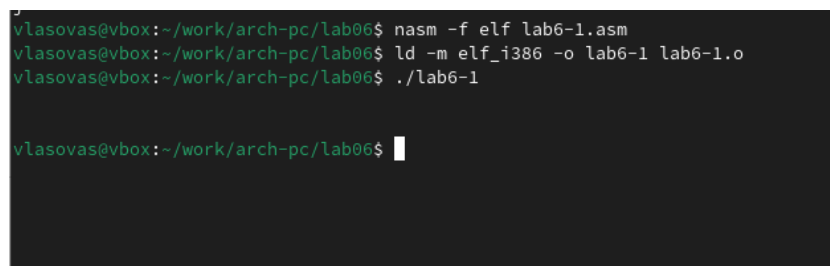
Снова открываем файл для редактирования и убираем кавычки с числовых значений (рис. fig. 3.4).



```
lab6-1.asm  [----] 13 L: [ 1+ 6 7/ 13] *(97 / 195b) 0010 0x00A [*] [
%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. 3.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.5).



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

vlasovas@vbox:~/work/arch-pc/lab06$
```

Рис. 3.5: Запускаем файл и смотрим на его работу

Создаем новый файл в каталоге (рис. fig. 3.6).

```
vlasovas@vbox:~/work/arch-pc/lab06$ touch lab6-2.asm
vlasovas@vbox:~/work/arch-pc/lab06$
```

Рис. 3.6: Создаем файл

Заполняем файл в соответствии с листингом 6.2 (рис. fig. 3.7).

```
lab6-2.asm [----] 12 L: [ 1+ 4 5/ 9] *(70 / 122b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

Рис. 3.7: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.8).

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

Рис. 3.8: Смотрим на работу программы

Снова открываем файл для редактирования и убираем кавычки с числовых значений (рис. fig. 3.9).



```
lab6-2.asm [----] 10 L:[ 1+ 5 6/ 9] *(79 / 118b) 0010 0x00A [*][X]
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 3.9: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.10).

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

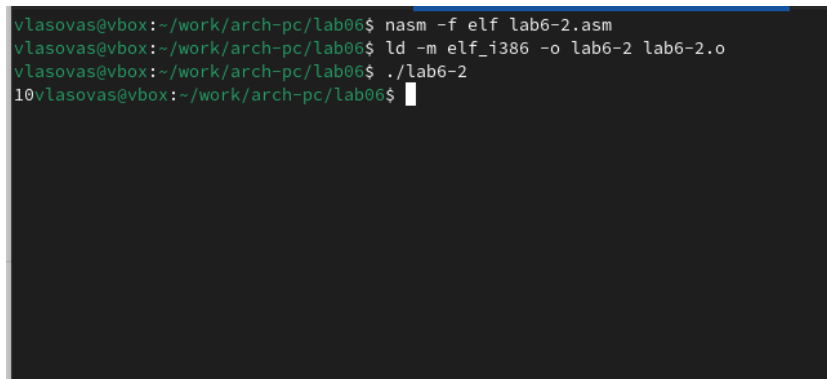
Рис. 3.10: Смотрим на работу программы

Снова открываем файл для редактирования и меняем iprintLF на iprint (рис. fig. 3.11).

```
lab6-2.asm [----] 12 L:[ 1+ 7 8/ 9] *(105 / 116b) 0010 0x00A [*][X]
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 3.11: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.12).



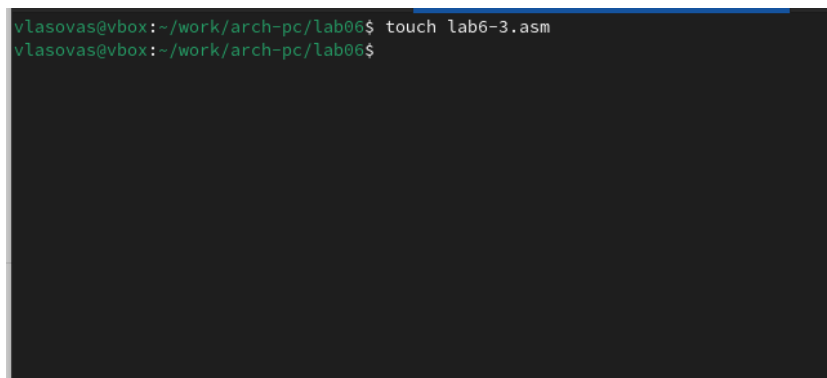
```
vlasovas@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
vlasovas@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
vlasovas@vbox:~/work/arch-pc/lab06$ ./lab6-2
10vlasovas@vbox:~/work/arch-pc/lab06$
```

Рис. 3.12: Смотрим на работу программы

Вывод функций `iprintLF` и `iprint` отличаются только тем, что `LF` переносит на новую строку.

## 3.2 Выполнение арифметических операций в NASM

Создаем новый файл в каталоге (рис. fig. 3.13).



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

Рис. 3.13: Создаем файл

Открываем файл и редактируем в соответствии с листингом 6.3 (рис. fig. 3.14).

```

lab6-3.asm      [-M--]  8 L: [ 1+13 14/ 24] *(228 / 360b) 0010 0x00A  [*][X]
%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

```

Рис. 3.14: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.15).

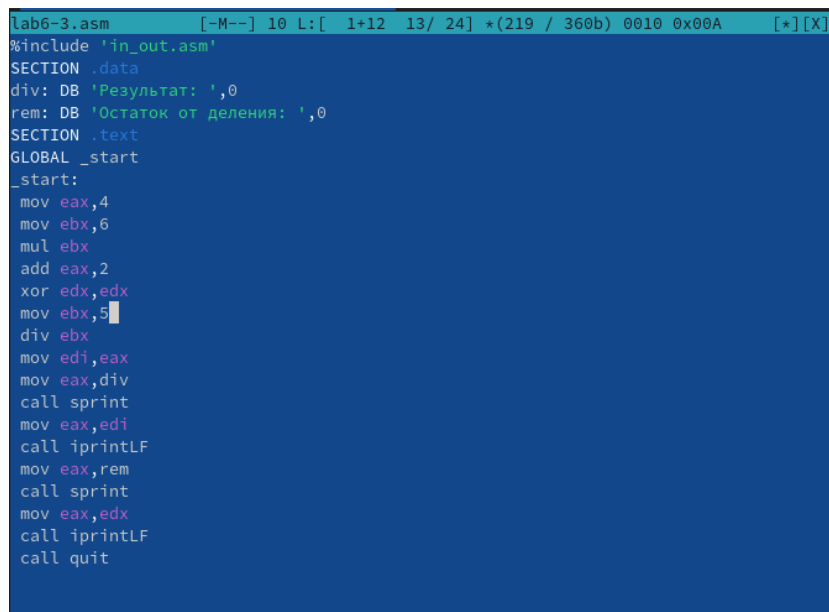
```

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

```

Рис. 3.15: Смотрим на результат работы программы

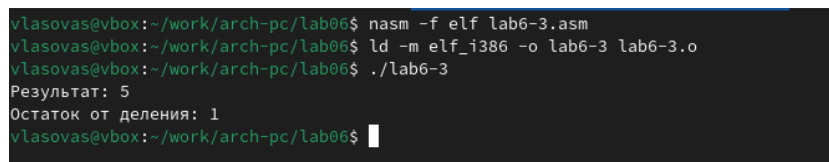
Открываем файл и редактируем его для вычисления выражения  $f(x) = (4 \cdot x + 2)/5$  (рис. fig. 3.16).



```
lab6-3.asm [-M--] 10 L: [ 1+12 13/ 24] *(219 / 360b) 0010 0x00A [*][X]
#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
```

Рис. 3.16: Редактируем файл

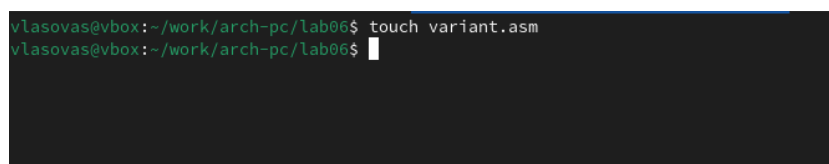
Компилируем файл и запускаем программу (рис. fig. 3.17).



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

Рис. 3.17: Смотрим на результат работы программы

Создаем новый файл в каталоге (рис. fig. 3.18).



```
vlasovas@vbox: ~/work/arch-pc/lab06$ touch variant.asm
vlasovas@vbox: ~/work/arch-pc/lab06$
```

Рис. 3.18: Создаем файл

Открываем файл и редактируем в соответствии с листингом 6.4 (рис. fig. 3.19).

```

variant.asm      [-M--]  8 L:[  1+18  19/ 26] *(325 / 400b) 0010 0x00A  [*][X]
#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
    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. 3.19: Заполняем файл

Компилируем файл и запускаем его (рис. fig. 3.20).

```

vlasovas@vbox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
vlasovas@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
vlasovas@vbox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246841
Ваш вариант: 2
vlasovas@vbox:~/work/arch-pc/lab06$

```

Рис. 3.20: Проверяем результат работы программы

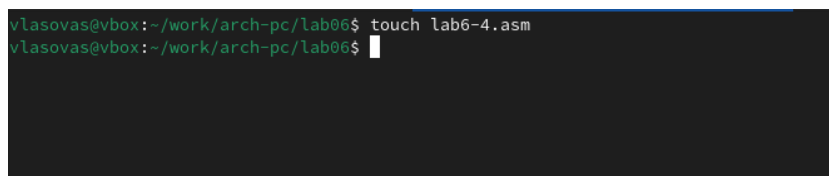
### 3.3 Ответы на вопросы по программе

1. Строка “mov eax,rem” и строка “call sprintf” отвечают за вывод на экран сообщения ‘Ваш вариант:’.
2. Эти инструкции используются для чтения строки с вводом данных пользователя. Начальный адрес строки сохраняется в регистре ecx, а количество символов в строке сохраняется в регистре edx. Затем вызывается процедура sread, которая выполняет чтение строки.

3. Инструкция “call atoi” используется для преобразования строки в целое число.
4. Строка “xor edx,edx” обнуляет регистр edx перед выполнением деления. Строка “mov ebx,20” загружает значение 20 в регистр ebx. Строка “div ebx” выполняет деление eax на ebx сохраняя частное в eax и остаток в edx.
5. Остаток от деления записывается в регистр edx.
6. Инструкция “inc edx” используется для увеличения значения в регистре edx на 1.
7. Строка “mov eax,edx” передает значение остатка от деления в регистр eax. Строка “call iprintLF” вызывает процедуру iprintLF для вывода значения на экран вместе с переводом строки.

### 3.4 Задание для самостоятельной работы

Создаем новый файл в каталоге (рис. fig. 3.21).



```
vlasovas@vbox: ~/work/arch-pc/lab06$ touch lab6-4.asm
vlasovas@vbox: ~/work/arch-pc/lab06$
```

Рис. 3.21: Создаем файл

Открываем его и заполняем, чтобы решалось выражение  $f(x) = (12x + 3) * 5$  (рис. fig. 3.22).

```
lab6-4.asm [----] 11 L: [ 5+26 31/ 31] *(445 / 445b) <EOF> [*][X]
SECTION .bss
rez: RESB 80
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

mov ebx, 12
mul ebx.....
add eax, 3.....
mov ebx, 5
mul ebx

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

Рис. 3.22: Заполняем файл

Компилируем программу и проверяем для  $x=1$  (рис. fig. 3.23).

```
vlasovas@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
vlasovas@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
vlasovas@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите x:
1
Результат: 75
vlasovas@vbox:~/work/arch-pc/lab06$
```

Рис. 3.23: Проверяем работу программы

Компилируем программу и проверяем для  $x=6$  (рис. fig. 3.24).

```
vlasovas@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
vlasovas@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
vlasovas@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите x:
6
Результат: 375
vlasovas@vbox:~/work/arch-pc/lab06$
```

Рис. 3.24: Проверяем работу программы

## 4 Выводы

Мы приобрели навыки создания исполнительных файлов для решения выражений и освоили арифметические инструкции в NASM.