

# **Лабораторная работа №6. Арифметические операции в NASM.**

**Архитектура компьютера**

Сущенко Алина Николаевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>22</b>
	<b>Список литературы</b>	<b>23</b>

## Список иллюстраций

3.1	Создание директория и проверка наличия . . . . .	7
3.2	Создание директория и проверка наличия 2 . . . . .	7
3.3	Код . . . . .	8
3.4	Запуск и конечный результат работы файла . . . . .	8
3.5	Замена части кода . . . . .	9
3.6	Создание файла и конечный результат . . . . .	10
3.7	Проверка наличия созданного файла . . . . .	10
3.8	Текст кода . . . . .	11
3.9	Создание исполняемого файла и реультат . . . . .	11
3.10	Замена части кода . . . . .	12
3.11	Запуск программы и кнечный результат . . . . .	12
3.12	Замена части кода . . . . .	13
3.13	Запуск программы и конечный результат . . . . .	13
3.14	Текст программы . . . . .	14
3.15	Запуск программы и конечный результат . . . . .	15
3.16	Замена текста программы . . . . .	16
3.17	Запуск программы и конечный результат . . . . .	17
3.18	Текст кода . . . . .	18
3.19	Текст кода . . . . .	18
3.20	Текст кода . . . . .	20
3.21	Проверка файлов . . . . .	21

## **Список таблиц**

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

1. Выводим значение регистра.
2. Выводим другое значение регистра с ответом на вопросы.
3. С помощью команды вычисляем значение выражения.
4. Выводим с помощью программы номер студенческого билета с ответом на вопрос.
5. Решаем заданный вариант с помощью изученных программ.

### 3 Выполнение лабораторной работы

subtitle: “Символьные и численные данные NASM”

1. Создаём директорию и файл типа .asm в нём (рис.1 [fig:001 width=70%]), (рис.2 [fig:002 width=70%]).

```
ansuthenko@dk8n59 ~ $ mkdir ~/work/arch-pc/lab06
ansuthenko@dk8n59 ~ $ cd ~/work/arch-pc/lab06
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ touch lab6-1.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ mc

ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ls
lab6-1.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $
```

Рис. 3.1: Создание директория и проверка наличия

2. Перенос in-out.asm в наш директорию.

```
in_out.asm
lab6-1.asm
```

Рис. 3.2: Создание директория и проверка наличия 2

3. Вводим текст команды для вывода символа (рис.3 [fig:003 width=70%])

```

lab6-1.asm [-----]
#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 sprintfLF
call quit

```

Рис. 3.3: Код

4. Создаём исполняемый файл и запускаем работу (рис.4 [fig:004 width=70%]).

```

ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ./lab6
bash: ./lab6: Нет такого файла или каталога
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ./lab6-1
j

```

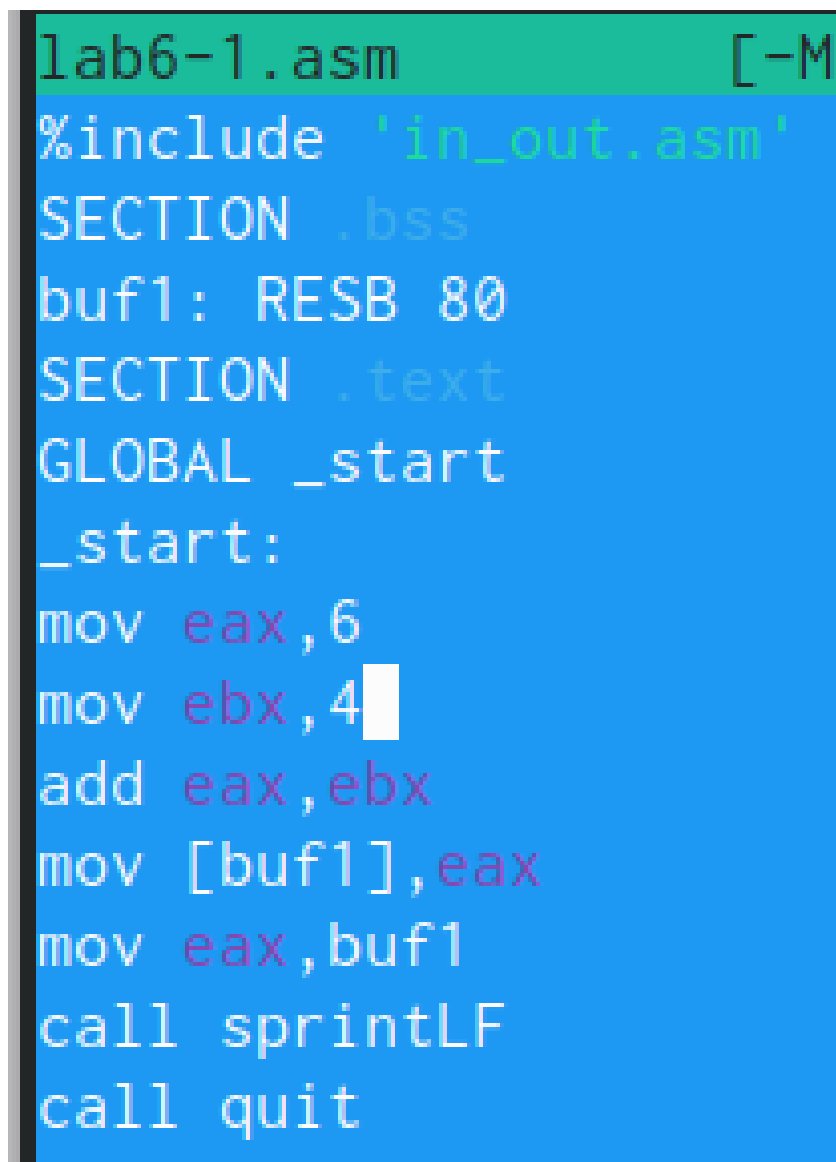
Рис. 3.4: Запуск и конечный результат работы файла

Программа вывела символ “j” потому что в соответствии с системой ASCII мы



вывели числа 4 и 6 и сложили.

5. Чуть изменим текст программы и запишем регистры вместо символов.  
(рис.5 [fig:005 width=70%])



```
lab6-1.asm [-M-]
#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. 3.5: Замена части кода

6. Создадим исполняемый файл и запустим его работу (рис.6 [fig:006 width=70%])

```

ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ./lab6-1

ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ █

```

Рис. 3.6: Создание файла и конечный результат

На данном этапе выводится символ с кодом 10, а именно символ переноса строки. Он не отображается при выводе на экран.

7. Создадим ещё один файл .asm (рис.7 [fig:007 width=70%])

```

ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ touch lab6-2.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ █

```

Рис. 3.7: Проверка наличия созданного файла

8. Введем код программы для вывода значения eax (рис.8 [fig:008 width=70%])

```
lab6-2.asm [-
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

Рис. 3.8: Текст кода

9. Создаём исполняемый файл и запускаем работу. (рис.9 [fig:009 width=70%])

```
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ./lab6-2~
bash: ./lab6-2~: Нет такого файла или каталога
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 3.9: Создание исполняемого файла и результат

В результате работы получаем число 106. Команда сложила коды символов 6 и 4, а именно 54+52. Однако в отличие от программы, которую мы используем в 6.1, функция iprintLF выводит число, а не символ закодированный этим числом.

10. Заменяем кусочек кода в исходно программе. (рис.10 [fig:010 width=70%]).

```

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

```

Рис. 3.10: Замена части кода

11. Запускаем исполняемый файл. (рис.11 [fig:011 width=70%])

```

ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ./lab6-2
10
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $

```

Рис. 3.11: Запуск программы и конечный результат

Программа сложила просто числа 4 и 6, поэтому и вывело 10.

12. Заменяем `iprintLf` на `iprint` (рис.12 [fig:012 width=70%]).

```

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

```

Рис. 3.12: Замена части кода

13. Запускаем исполняемый файл (рис.13 [fig:013 width=70%]).

```

ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ./lab6-2
10ansuthenko@dk8n59 ~/work/arch-pc/lab06 $

```

Рис. 3.13: Запуск программы и конечный результат

В результате не было изменения, кроме отстусвия переноса строки.

subtitle:“Выполнение арифметических операций в NASM”

1. Создаём файл lab6-3.asm и вводим в созданный файл текст программы для вычисления значений выражения  $f(x) = (5 \times 2 + 3)/3$  (рис.14 [fig:014 width=70%])

```
lab6-3.asm [----] 13 L:[
#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: Текст программы

2. Запускаем исполняемый файл. (рис.15 [fig:015 width=70%])

```

ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1

```

Рис. 3.15: Запуск программы и конечный результат

3. Заменяем программу так, чтобы оно решало выражение  $f(x) = (4x + 2)/5$   
(рис.16 [fig:016 width=70%])

```
lab6-3.asm      [-M--]  9  L:[
#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: Замена текста программы



4. Создаём файл и запускаем его работу. (рис.17 [fig:017 width=70%])

```
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $
```

Рис. 3.17: Запуск программы и конечный результат

5. Создадим файл variant.asm и вычислим номер варианта (рис.18 [fig:018 width=70%])

```

variant.asm      [-M--]  9 L:[  1+24  25/ 25
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите No 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. 3.18: Текст кода

## 6. Запуск исполняемого файла (рис.19 [fig:019 width=70%])

```

ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o\
>
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132231439
Ваш вариант: 20
ansuthenko@dk8n59 ~/work/arch-pc/lab06 $

```

Рис. 3.19: Текст кода

subtitle: Ответы на вопросы.

1. Строки листинга 6.4 отвечающие за вывод строки кода: `mov eax,rem` `call sprint`
2. Инструкция `mov ecx, x` используется для прокладывания пути к вводимой строке `mov ecx, 80` используется для записи в регистр `edx`(длина строки). `call sread` используется для вызова подпрограммы внешнего файла, чтобы вывести сообщение с клавиатуры.
3. Инструкция `call atoi` нужна для вызова подпрограммы из внешнего файла для считывания кода ASCII и перевода его в числовое значение, после чего идёт запись в регистр `eax`
4. За вычисления варианта отвечают строки : `xor edx,edx` ;обнуление `edx` для корректной работы `div` `mov ebx,2` ;`ebx=2` `div ebx` ;`eax=eax/2` ,`edx` -остаток от деления `inc edx` ;`edx=edx+1`
5. При выполнении функции `div ebx` остаток от деления идёт в регистр `edx`.
6. `inc edx` увеличивает значение `edx` на 1.
7. Отвечают за вывод на экран результата вычислений строки: `mov eax,edx` `call inprintLF`

subtitle: Выполнение задания для самостоятельной работы.

1. Создаём файл `lab6-4.asm` и пишем код программы.(рис.20 [fig:020 width=70%])

```

lab6-4.asm      [----] 10 L:[ 1+1
%include 'in_out.asm'
SECTION .data
task: DB 'f(x)=x**3/3+21',0
vvod: DB 'Введите x ',0
ans: DB 'Ответ: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:

mov eax,task
call sprintf
mov eax, vvod
call sprintf
mov ecx,x
mov edx,80
call sread

mov eax,x
call atoi
mov ebx,eax
mul ebx
mul ebx
xor edx, edx
mov ebx,3
div ebx
add eax,21
mov ebx, eax
mov eax, ans
call sprintf
mov eax,ebx
call iprintLF
call quit

```

Рис. 3.20: Текст кода

## 2. Проверяем работу программы (рис.20 [fig:020 width=70%])

```
ansuthenko@dk8n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
ansuthenko@dk8n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
ansuthenko@dk8n55 ~/work/arch-pc/lab06 $ ./lab6-4
f(x)=x**3/3+21
Введите x 3
Ответ: 30
ansuthenko@dk8n55 ~/work/arch-pc/lab06 $ ./lab6-4
f(x)=x**3/3+21
Введите x 1
Ответ: 21
ansuthenko@dk8n55 ~/work/arch-pc/lab06 $
```

Рис. 3.21: Проверка файлов

## 4 Выводы

Выполняя эту работу мы освоили арифметические инструкции языка ассемблер NASM.

## **Список литературы**