

# Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Козина Дарья Александровна

## Содержание

Цель работы .....	4
Задание.....	5
Выполнение лабораторной работы.....	6
Символьные и численные данные в NASM .....	6
Выполнение арифметических операций в NASM .....	11
Ответы на вопросы.....	15
Выполнение заданий для самостоятельной работы .....	16
Выводы.....	18

## Список иллюстраций

Создание каталога и файла .....	6
Копирование файла.....	6
Редактирование файла.....	7
Создание и запуск файла .....	7
Изменение программы .....	8
Создание и запуск файла .....	8
Создание файла .....	8
Редактирование файла.....	9
Создание и запуск файла .....	9
Изменение программы .....	10
Создание и запуск файла .....	10
Замена функции .....	11
Создание и запуск файла .....	11
Создание файла .....	11
Редактирование файла.....	12
Создание и запуск файла .....	12
Редактирование программы.....	13
Создание и запуск файла .....	13
Создание файла .....	13
Редактирование файла.....	14
Создание и запуск файла .....	14
Создание файла .....	16
Редактирование файла.....	16
Создание и запуск файла .....	17

## Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

## Задание

1. Символьные и численные данные в NASM;
2. Выполнение арифметических операций в NASM;
3. Выполнение заданий для самостоятельной работы.

# Выполнение лабораторной работы

## Символьные и численные данные в NASM

С помощью команды `mkdir` создадим каталог для выполнения лабораторной работы № 6. С помощью команды `touch` в созданном каталоге создадим файл `lab6-1.asm` (рис. [-@fig:001]).

---

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

### *Создание каталога и файла*

Для того, чтобы программы корректно работали, копирую файл `in_out.asm` из каталога, созданного при выполнении прошлой лабораторной работы в только что созданный каталог (рис. [-@fig:002]).

```
dakozina@vbox:~/work/arch-pc/lab06$ cp ~/work/arch-pc/lab05/in_out.asm in_out.asm
dakozina@vbox:~/work/arch-pc/lab06$ █
```

### *Копирование файла*

Введем программу в файл `lab6-1.asm` с помощью редактора `mcedit` (рис. [-@fig:003]).

```

lab6-1.asm [----] 11
#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

```

### Редактирование файла

Создадим исполняемый файл и запустим его (рис. [-@fig:004]).

```

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

```

### Создание и запуск файла

Изменим текст программы. В регистр вместо символов '6' и '4' запишем числа 6 и 4 (рис. [-@fig:005]).

```

lab6-1.asm [-f
%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

```

### *Изменение программы*

Создадим исполняемый файл и запустим его (рис. [-@fig:006]). Теперь выводится символ с кодом 10. Этот символ не вывелся на экран.

```

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

```

### *Создание и запуск файла*

С помощью команды touch создадим файл lab6-2.asm (рис. [-@fig:007]).

```

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

```

### *Создание файла*

Введем текст программы в созданный файл с помощью редактора mcedit (рис. [-@fig:008]).



```

lab6-2.asm      [-M--] 1
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:
* *
mov  eax, '6'
mov  ebx, '4'
add  eax, ebx
call iprintLF
* *
call quit

```

### *Редактирование файла*

Создадим исполняемый файл и запустим его (рис. [-@fig:009]).

```

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

```

### *Создание и запуск файла*

Изменим текст программы. В регистр вместо символов '6' и '4' запишем числа 6 и 4 (рис. [-@fig:010]).

```

lab6-2.asm [-M-
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:
..
mov eax, 6
mov ebx, 4
add eax, ebx
call iprintLF
..
call quit

```

### *Изменение программы*

Создадим исполняемый файл и запустим его (рис. [-@fig:011]). Выводится число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит сложение символов '6' и '4'.

```

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

```

### *Создание и запуск файла*

Заменим функцию iprintLF на iprint (рис. [-@fig:012]).

```

%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:
..
    mov eax, 6
    mov ebx, 4
    add eax, ebx
    call iprint
..
    call quit

```

### *Замена функции*

Создадим исполняемый файл и запустим его (рис. [-@fig:013]). Вывод не изменился. Символ переноса строки не отображался, когда программа исполнялась с функцией `iprintLF`, а `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`.

```

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

```

### *Создание и запуск файла*

## Выполнение арифметических операций в NASM

С помощью команды `touch` создадим файл `lab6-3.asm` (рис. [-@fig:014]).

```

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

```

### *Создание файла*

Введем текст программы вычисления выражения  $f(x) = (5 \cdot 2 + 3)/3$  в созданный файл с помощью редактора `mcedit` (рис. [-@fig:015]).

```

lab6-3.asm      [-M--]  0 L:[ 1+14 15/ 41] *(301 /1395b) 0010 0x00A
;
; Программа вычисления выражения
;
-----
%include 'in_out.asm'.

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

```

### Редактирование файла

Создадим исполняемый файл и запустим его (рис. [-@fig:016]).

```

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

```

### Создание и запуск файла

Изменим текст программы в редакторе mcedit таким образом, чтобы она вычислила  $f(x) = (4*6 + 2)/5$  (рис. [-@fig:017]).

```

lab6-3.asm      [-M--] 22 L:[ 1+23 24/ 41] *(571 /1395b) 0044 0x0
;
; -----
; Программа вычисления выражения
; -----
;
%include 'in_out.asm'.

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

```

### Редактирование программы

Создадим исполняемый файл и запустим его (рис. [-@fig:018]).

```

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

```

### Создание и запуск файла

С помощью команды touch создадим файл variant.asm (рис. [-@fig:019]).

```

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

```

### Создание файла

Введем текст программы вычисления варианта задания по номеру студенческого билета в созданный файл с помощью редактора mcedit (рис. [-@fig:020]).

```

variant.asm      [-M--]  0 L:[ 1+22 23/ 39] *(417 / 694b) 0032 0x0
;-----
; Программа вычисления варианта
;-----

%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

```

### *Редактирование файла*

Создадим исполняемый файл и запустим его (рис. [-@fig:021]). Программа вывела, что мой вариант - 12.

```

variant.asm      [-M--]  0 L:[ 1+22 23/ 39] *(417 / 694b) 0032 0x0
;-----
; Программа вычисления варианта
;-----

%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

```

### *Создание и запуск файла*

## Ответы на вопросы

```
mov eax,rem  
call sprint
```

2. Инструкция `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx` `mov edx, 80` - запись в регистр `edx` длины вводимой строки `call sread` - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры;
3. `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`;

```
xor edx,edx ; обнуление edx для корректной работы div  
mov ebx,20 ; ebx = 20  
div ebx ; eax = eax/20, edx - остаток от деления  
inc edx ; edx = edx + 1
```

5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`;
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1;

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

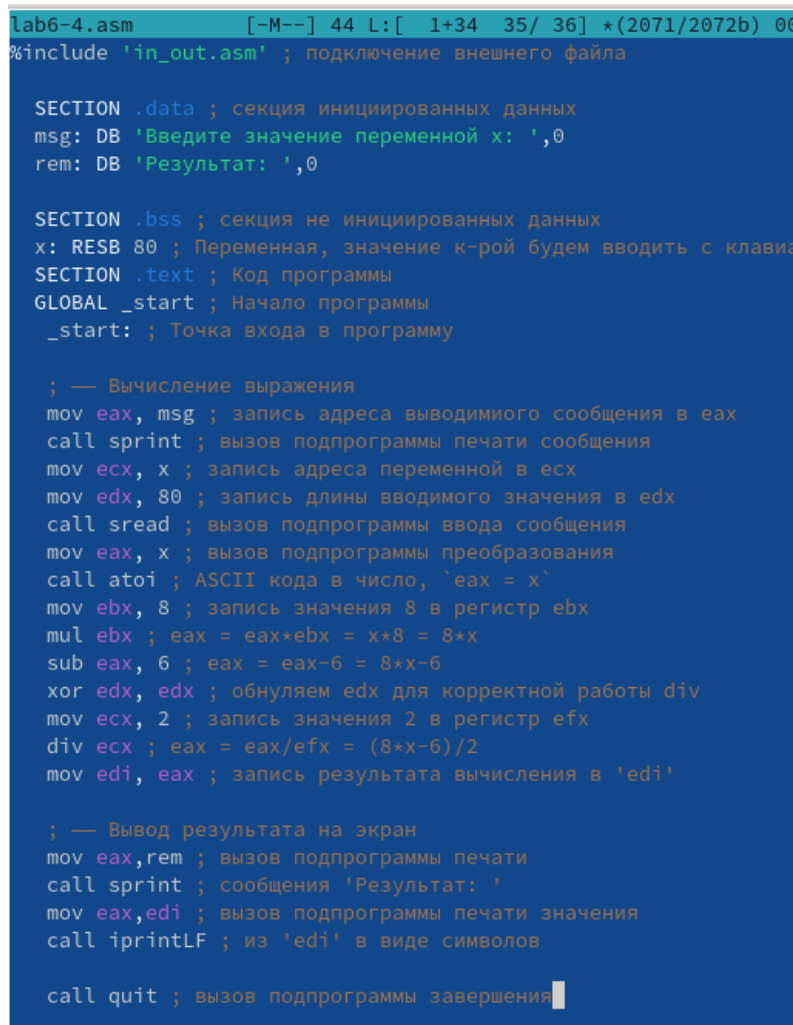
# Выполнение заданий для самостоятельной работы

С помощью команды touch создадим файл lab6-4.asm (рис. [-@fig:022]).

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

## Создание файла

Введем текст программы вычисления выражения, которое соответствует варианту по номеру моего студенческого билета, а именно  $f(x) = (8 \cdot x - 6)/2$  с помощью редактора mcedit (рис. [-@fig:023]).



```
lab6-4.asm      [-M--] 44 L:[ 1+34 35/ 36] *(2071/2072b) 00
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data ; секция инициированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0

SECTION .bss ; секция не инициированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

; — Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax = x`
mov ebx, 8 ; запись значения 8 в регистр ebx
mul ebx ; eax = eax*ebx = x*8 = 8*x
sub eax, 6 ; eax = eax-6 = 8*x-6
xor edx, edx ; обнуляем edx для корректной работы div
mov ecx, 2 ; запись значения 2 в регистр ecx
div ecx ; eax = eax/ecx = (8*x-6)/2
mov edi, eax ; запись результата вычисления в 'edi'

; — Вывод результата на экран
mov eax, rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

call quit ; вызов подпрограммы завершения
```

## Редактирование файла

Создадим исполняемый файл и запустим его (рис. [-@fig:024]).



```

dakozi@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
dakozi@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
dakozi@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 1
dakozi@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
dakozi@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
dakozi@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 5
Результат: 17

```

## Создание и запуск файла

### Листинг программы

`%include 'in_out.asm'` ; подключение внешнего файла

```

SECTION .data ; секция инициализированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0

```

```

SECTION .bss ; секция не инициализированных данных
x: RESB 80 ; Переменная, значение которой будем вводить с клавиатуры, выделенный
размер - 80 байт

```

```

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

```

```

; — Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax = x`
mov ebx, 8 ; запись значения 8 в регистр ebx
mul ebx ; eax = eax*ebx = x*8 = 8*x
sub eax, 6 ; eax = eax-6 = 8*x-6
xor edx, edx ; обнуляем edx для корректной работы div
mov ecx, 2 ; запись значения 2 в регистр ecx
div ecx ; eax = eax/ecx = (8*x-6)/2
mov edi, eax ; запись результата вычисления в 'edi'

```

```

; — Вывод результата на экран
mov eax, rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

```

```

call quit ; вызов подпрограммы завершения

```

## Выводы

В ходе лабораторной работы мы освоили арифметические инструкции языка ассемблера NASM.