

# **Лабораторная работа №6**

**Арифметические операции в NASM**

Бунин Арсений Викторович

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Самостоятельная работа	16
6	Выводы	18
	Список литературы	19

## Список иллюстраций

4.1	Вызов Midnight Commander . . . . .	8
4.2	Каталог курса в Midnight Commander . . . . .	9
4.3	Окно создания папки . . . . .	10
4.4	Окно создания файла . . . . .	10
4.5	Файл в редакторе mcedit . . . . .	11
4.6	Файл в режиме просмотра . . . . .	11
4.7	Линковка и компоновка . . . . .	12
4.8	Работа первой программы . . . . .	12
4.9	Окно замены файла . . . . .	13
4.10	Код второй программы . . . . .	13
4.11	Вторая программа . . . . .	14
4.12	Измененная вторая программа . . . . .	15
4.13	Измененная вторая программа . . . . .	15
5.1	Код программы . . . . .	17
5.2	Работа программы . . . . .	17

## Список таблиц

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

## 2 Задание

1. Создать файл на языке Ассемблер, выводящий значения регистра
2. Создать файл на языке Ассемблер, выполняющий арифметические операции
3. Выполнить индивидуальное задание по написанию программы на Ассемблере
4. Загрузить файлы на github

### 3 Теоретическое введение

Схема команды целочисленного сложения `add` (от англ. *addition* - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда `add` работает как с числами со знаком, так и без знака. Команда целочисленного вычитания `sub` (от англ. *subtraction* – вычитание) работает аналогично команде `add`. Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: `inc` (от англ. *increment*) и `dec` (от англ. *decrement*), которые увеличивают и уменьшают на 1 свой операнд. Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производятся по-разному, поэтому существуют различные команды. Для беззнакового умножения используется команда `mul`. Для знакового умножения используется команда `imul`. Для деления, как и для умножения, существует 2 команды `div` и `idiv`.

## 4 Выполнение лабораторной работы

Создаем исполняемый файл(рис. 4.1 и рис. 4.2)

```
[arsenii@fedora ~]$ mkdir /home/arsenii/work/study/2023-2024/"Архитектура компью  
тера"/arch-pc/labs/lab06  
[arsenii@fedora ~]$ cd /home/arsenii/work/study/2023-2024/"Архитектура компьюте  
ра"/arch-pc/labs/lab06  
[arsenii@fedora lab06]$ touch lab6-1.asm  
[arsenii@fedora lab06]$
```

Рис. 4.1: Вызов Midnight Commander



```
%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. 4.2: Каталог курса в Midnight Commander

Результат работы программы (рис. 4.3).

```
[arsenii@fedora lab06]$ ./lab6-1  
j  
[arsenii@fedora lab06]$
```

Рис. 4.3: Окно создания папки

Заменяем символы на цифры в программе (рис. 4.4).

```
%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. 4.4: Окно создания файла

Программа с подключенными внешними функциями (рис. 4.5)

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

Рис. 4.5: Файл в редакторе mcedit

Результат работы программы (рис. 4.6)

```
[arsenii@fedora lab06]$ ./lab6-2
106
[arsenii@fedora lab06]$
```

Рис. 4.6: Файл в режиме просмотра

Заменяем символы на цифры в программе (рис. 4.7)

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

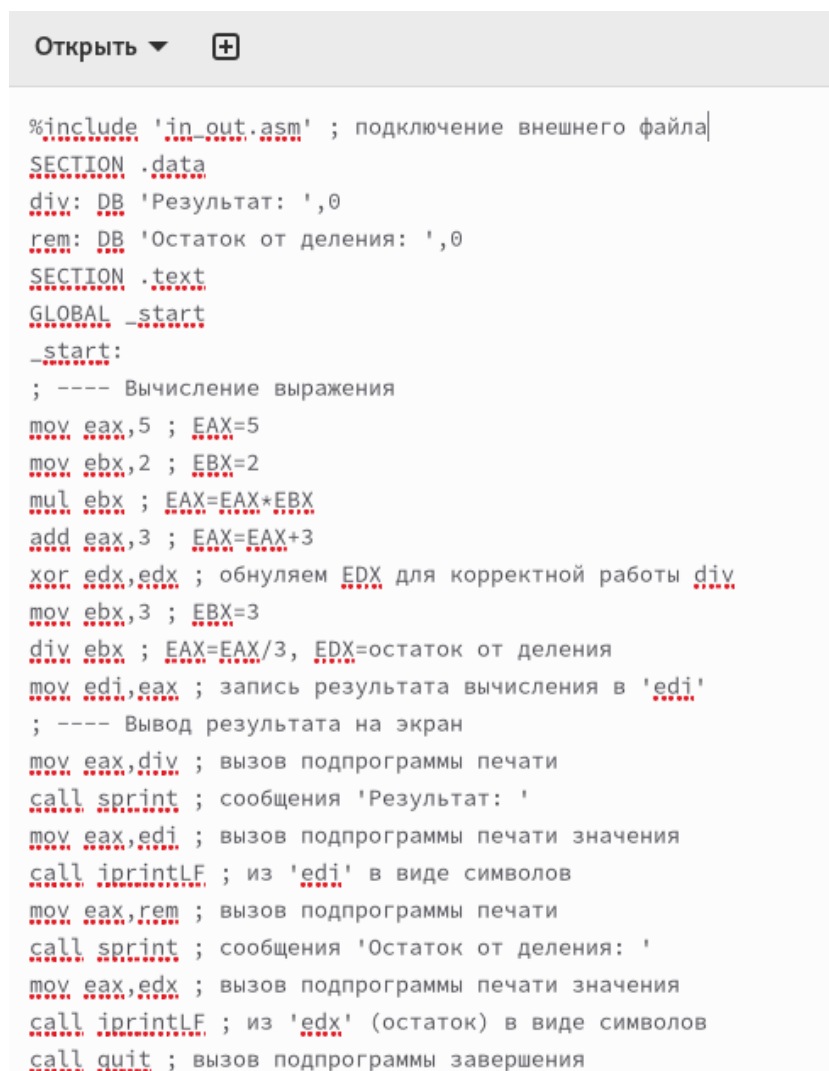
Рис. 4.7: Линковка и компоновка

Результат работы программы(рис. 4.8)

```
[arsenii@fedora lab06]$ ./lab6-2
10
[arsenii@fedora lab06]$
```

Рис. 4.8: Работа первой программы

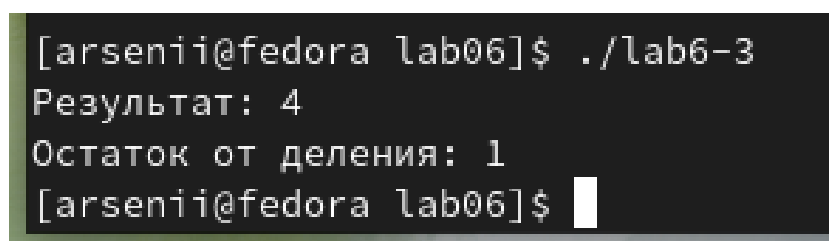
Код программы, считающей значение выражения  $(5*2 + 3)/3$ (рис. 4.9)



```
%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: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.9: Окно замены файла

Результат работы программы(рис. 4.10)



```
[arsenii@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1
[arsenii@fedora lab06]$
```

Рис. 4.10: Код второй программы

Код программы, считающей значение выражения  $(4*6 + 2)/5$  (рис. 4.11)

```
%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=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: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.11: Вторая программа

Результат работы программы (рис. 4.12)

```
[arsenii@fedora lab06]$ ./lab6-3  
Результат: 5  
Остаток от деления: 1  
[arsenii@fedora lab06]$
```

Рис. 4.12: Измененная вторая программа

Результат работы программы, выводящей номер варианта по номеру студенческого билета (рис. 4.13)

```
[arsenii@fedora lab06]$ ./lab6-4  
Введите № студенческого билета:  
1132236118  
Ваш вариант: 19  
[arsenii@fedora lab06]$
```

Рис. 4.13: Измененная вторая программа

## 5 Самостоятельная работа

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  $f(x)=(x/3+5)*7$  Создайте исполняемый файл и проверьте его работу для значений 3 и 9

Код программы(рис. 5.1)



```

%include 'in_out.asm'
SECTION .data
msg: DB 'Введите число: ',0
rem: DB 'Результат: ',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, 3
div ebx
add eax, 5 ; EAX=EAX+5
mov ebx, 7 ; EBX=7
mul ebx ; EAX=EAX*7
mov edi, eax ; запись результата вычисления в 'edi'
mov eax, rem
call sprint
mov eax, edi
call iprintLF
call quit

```

Рис. 5.1: Код программы

Результат работы программы (рис. 5.2)

```

[arsenii@fedora lab06]$ ./lab6-6
Введите число:
3
Результат: 42
[arsenii@fedora lab06]$ ./lab6-6
Введите число:
9
Результат: 56
[arsenii@fedora lab06]$

```

Рис. 5.2: Работа программы

## 6 Выводы

Освоили арифметические инструкции языка ассемблера NASM

## **Список литературы**