

# **Лабораторная работа №6**

**Арифметические операции в NASM**

Чекмарев Александр Дмитриевич | группа: НПИбд 02-23

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Символьные и численные данные в NASM . . . . .	6
2.2	Выполнение арифметических операций в NASM . . . . .	12
2.3	Вопросы . . . . .	18
<b>3</b>	<b>Самостоятельная работа</b>	<b>20</b>
<b>4</b>	<b>Выводы</b>	<b>23</b>

## Список иллюстраций

2.1	Рис 2.1.1: Создание каталога и файла .asm . . . . .	6
2.2	Рис 2.1.2: Демонстрация текста программы в файле . . . . .	7
2.3	Рис 2.1.3: Копирование файла для программы . . . . .	7
2.4	Рис 2.1.4: Создание исполняемого файла и проверка работы . . .	7
2.5	Рис 2.1.5: Демонстрация изменения программы . . . . .	9
2.6	Рис 2.1.6: Создание исполняемого файла и проверка работы . . .	9
2.7	Рис 2.1.7: Создание файла . . . . .	10
2.8	Рис 2.1.8: Демонстрация программы . . . . .	10
2.9	Рис 2.1.9: Создание исполняемого файла и проверка работы . . .	10
2.10	Рис 2.1.10: Демонстрация измененной программы . . . . .	11
2.11	Рис 2.1.11: Создание исполняемого файла и проверка работы . . .	12
2.12	Рис 2.1.12: Создание исполняемого файла и проверка работы . . .	12
2.13	Рис 2.2.1: Создание файла . . . . .	12
2.14	Рис 2.2.2: Демонстрация программы . . . . .	13
2.15	Рис 2.2.3: Создание исполняемого файла и проверка работы . . .	14
2.16	Рис 2.2.4: Демонстрация измененной программы . . . . .	15
2.17	Рис 2.2.5: Создание исполняемого файла и проверка работы . . .	16
2.18	Рис 2.2.6: Создание файла . . . . .	16
2.19	Рис 2.2.7: Демонстрация программы . . . . .	17
2.20	Рис 2.2.8: Создание исполняемого файла и проверка работы . . .	18
3.1	Рис 3.1.1: Создание файла . . . . .	20
3.2	Рис 3.1.2: Демонстрация программы . . . . .	21
3.3	Рис 3.1.3: Проверка программы . . . . .	22

## Список таблиц

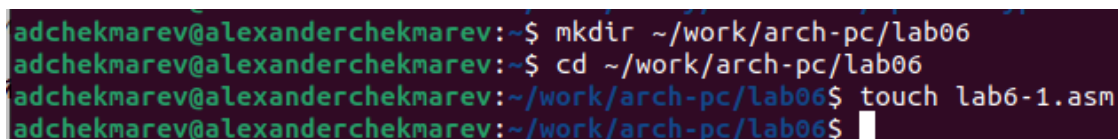
# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

### 2.1 Символьные и численные данные в NASM

Создадим каталог для программ лабораторной работы № 6, перейдем в него и создадим файл *lab6-1.asm*:



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

Рис. 2.1: Рис 2.1.1: Создание каталога и файла .asm

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр *eax*.

Введем в файле *lab6-1.asm* текст программы из листинга 6.1. В данной программе в регистр *eax* записывается символ 6 (*mov eax, '6'*), в регистр *ebx* символ 4 (*mov ebx, '4'*). Далее к значению в регистре *eax* прибавляем значение регистра *ebx* (*add eax, ebx*, результат сложения запишется в регистр *eax*). Далее выводим результат. Так как для работы функции *sprintf* в регистр *eax* должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра *eax* в переменную *buf1* (*mov [buf1], eax*), а затем запишем адрес переменной *buf1* в регистр *eax* (*mov eax, buf1*) и вызовем функцию *sprintf*.

```

1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1: RESB 80
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov eax, '6'
11 mov ebx, '4'
12 add eax, ebx
13 mov [buf1], eax
14 mov eax, buf1
15 call sprintf
16
17 call quit

```

Рис. 2.2: Рис 2.1.2: Демонстрация текста программы в файле

Скопируем *in\_out.asm* из каталога *lab05* в *lab06*

```

adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$ cp in_out.asm ~/work/arch-pc/lab06
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab05$

```

Рис. 2.3: Рис 2.1.3: Копирование файла для программы

Создадим исполняемый файл и запустим его

```

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

```

Рис. 2.4: Рис 2.1.4: Создание исполняемого файла и проверка работы

В данном случае при выводе значения регистра *eax* мы ожидаем увидеть число

10. Однако результатом будет символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда `add eax,ebx` запишет в регистр `eax` сумму кодов – 01101010 (106), что в свою очередь является кодом символа j (см. таблицу ASCII в приложении).

Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы следующим образом:

- `mov eax,'6'`
- `mov ebx,'4'`

на строки

- `mov eax,6`
- `mov ebx,4`



```

1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1: RESB 80
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov eax,6
11 mov ebx,4
12 add eax,ebx
13 mov [buf1],eax
14 mov eax,buf1
15 call sprintf
16
17 call quit

```

Рис. 2.5: Рис 2.1.5: Демонстрация изменения программы

Создадим исполняемый файл и запустим его.

```

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

```

Рис. 2.6: Рис 2.1.6: Создание исполняемого файла и проверка работы

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определим какому символу соответствует код 10, это *LF*, *\n*. Который не отображается на выводе. Как отмечалось выше, для работы с числами в файле *in\_out.asm* реализованы подпрограммы для преобразования ASCII символов в числа и об-

ратно. Преобразуем текст программы из Листинга 6.1 с использованием этих функций.

Создадим файл *lab6-2.asm* в каталоге *~/work/arch-pc/lab06* и введем в него текст программы из листинга 6.2.

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

Рис. 2.7: Рис 2.1.7: Создание файла

```
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 call iprintLF
11
12 call quit
```

Рис. 2.8: Рис 2.1.8: Демонстрация программы

Создадим исполняемый файл и запустим его.

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

Рис. 2.9: Рис 2.1.9: Создание исполняемого файла и проверка работы

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда `add` складывает коды символов '6' и '4' ( $54+52=106$ ). Однако, в отличие от программы из листинга 6.1, функция *iprintLF* позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа. Заменяем строки

- `mov eax,'6'`
- `mov ebx,'4'`

на строки

- `mov eax,6`
- `mov ebx,4`

```
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprintLF
11
12 call quit
```

Рис. 2.10: Рис 2.1.10: Демонстрация измененной программы

Создадим исполняемый файл и запустим его.

```

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

```

Рис. 2.11: Рис 2.1.11: Создание исполняемого файла и проверка работы

В результате мы получили число 10

Заменим функцию *iprintLF* на *iprint*. Создадим исполняемый файл и запустим его.

```

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

```

Рис. 2.12: Рис 2.1.12: Создание исполняемого файла и проверка работы

В результате также мы получили число 10, но в случае *iprint* число выведено без красной строки

## 2.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения  $f(x) = (5 * 2 + 3) / 3$ .

Создадим файл *lab6-3.asm* в каталоге *~/work/arch-pc/lab06*:

```

10adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$ touch lab6-3.asm
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$

```

Рис. 2.13: Рис 2.2.1: Создание файла

Внимательно изучим текст программы из листинга 6.3 и введем в *lab6-3.asm*.

```

1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4
5 %include 'in_out.asm' ; подключение внешнего файла
6
7 SECTION .data
8
9 div: DB 'Результат: ',0
10 rem: DB 'Остаток от деления: ',0
11
12 SECTION .text
13 GLOBAL _start
14 _start:
15
16 ; ---- Вычисление выражения
17 mov eax,5 ; EAX=5
18 mov ebx,2 ; EBX=2
19 mul ebx ; EAX=EAX*EBX
20 add eax,3 ; EAX=EAX+3
21 xor edx,edx ; обнуляем EDX для корректной работы div
22 mov ebx,3 ; EBX=3
23 div ebx ; EAX=EAX/3, EDX=остаток от деления
24
25 mov edi,eax ; запись результата вычисления в 'edi'
26
27 ; ---- Вывод результата на экран
28
29 mov eax,div ; вызов подпрограммы печати
30 call sprint ; сообщения 'Результат: '
31 mov eax,edi ; вызов подпрограммы печати значения
32 call iprintLF ; из 'edi' в виде символов
33
34 mov eax,rem ; вызов подпрограммы печати
35 call sprint ; сообщения 'Остаток от деления: '
36 mov eax,edx ; вызов подпрограммы печати значения
37 call iprintLF ; из 'edx' (остаток) в виде символов
38
39 call quit ; вызов подпрограммы завершения

```

Рис. 2.14: Рис 2.2.2: Демонстрация программы

Создадим исполняемый файл и запустим его.

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

Рис. 2.15: Рис 2.2.3: Создание исполняемого файла и проверка работы

Изменим текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$ .

```

1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4
5 %include 'in_out.asm' ; подключение внешнего файла
6
7 SECTION .data
8
9 div: DB 'Результат: ',0
10 rem: DB 'Остаток от деления: ',0
11
12 SECTION .text
13 GLOBAL _start
14 _start:
15
16 ; ---- Вычисление выражения
17 mov eax,4 ; EAX=4
18 mov ebx,6 ; EBX=6
19 mul ebx ; EAX=EAX*EBX
20 add eax,2 ; EAX=EAX+2
21 xor edx,edx ; обнуляем EDX для корректной работы div
22 mov ebx,5 ; EBX=5
23 div ebx ; EAX=EAX/5, EDX=остаток от деления
24
25 mov edi,eax ; запись результата вычисления в 'edi'
26
27 ; ---- Вывод результата на экран
28
29 mov eax,div ; вызов подпрограммы печати
30 call sprint ; сообщения 'Результат: '
31 mov eax,edi ; вызов подпрограммы печати значения
32 call iprintLF ; из 'edi' в виде символов
33
34 mov eax,rem ; вызов подпрограммы печати
35 call sprint ; сообщения 'Остаток от деления: '
36 mov eax,edx ; вызов подпрограммы печати значения
37 call iprintLF ; из 'edx' (остаток) в виде символов
38
39 call quit ; вызов подпрограммы завершения

```

Рис. 2.16: Рис 2.2.4: Демонстрация измененной программы

Создадим исполняемый файл и проверим его работу.

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

Рис. 2.17: Рис 2.2.5: Создание исполняемого файла и проверка работы

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

- вывести запрос на введение  $N^{\circ}$  студенческого билета
- вычислить номер варианта по формуле:  $(S_n \bmod 20) + 1$ , где  $S_n$  – номер студенческого билета (В данном случае  $a \bmod b$  – это остаток от деления  $a$  на  $b$ ).
- вывести на экран номер варианта.

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция `atoi` из файла `in_out.asm`.

Создадим файл *variant.asm* в каталоге `~/work/arch-pc/lab06`:

```
adchekmarev@alexandercchekmarev:~/work/arch-pc/lab06$ touch variant.asm
adchekmarev@alexandercchekmarev:~/work/arch-pc/lab06$
```

Рис. 2.18: Рис 2.2.6: Создание файла

Внимательно изучим текст программы из листинга 6.4 и введем в файл *variant.asm*.



```

1 ;-----
2 ; Программа вычисления варианта
3 ;-----
4
5 %include 'in_out.asm'
6
7 SECTION .data
8 msg: DB 'Введите № студенческого билета: ',0
9 rem: DB 'Ваш вариант: ',0
10
11 SECTION .bss
12 x: RESB 80
13
14 SECTION .text
15 GLOBAL _start
16 _start:
17
18 mov eax, msg
19 call sprintf
20
21 mov ecx, x
22 mov edx, 80
23 call sread
24
25 mov eax,x ; вызов подпрограммы преобразования
26 call atoi ; ASCII кода в число, `eax=x`
27 xor edx,edx
28 mov ebx,20
29 div ebx
30 inc edx
31
32 mov eax,rem
33 call sprintf
34 mov eax,edx
35 call iprintLF
36
37 call quit

```

Рис. 2.19: Рис 2.2.7: Демонстрация программы

Создадим исполняемый файл и запустим его.

```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$ nasm -f elf variant.asm
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132230303
Ваш вариант: 4
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$
```

Рис. 2.20: Рис 2.2.8: Создание исполняемого файла и проверка работы

## 2.3 Вопросы

Включите в отчет по выполнению лабораторной работы ответы на следующие вопросы:

1) Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

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

2) Для чего используются следующие инструкции?

- `mov ecx, x` - перемещает адрес вводимой строки в `ecx`
- `mov edx, 80` - записывает длину строки в регистр `edx`
- `call sread` - вызывает подпрограммы, которые обеспечивают ввод сообщения с помощью клавиатуры

3) Для чего используется инструкция “`call atoi`”?

Она используется для вызова подпрограммы, которая преобразует ASCII код символа в целое число, записывая его в результат регистра `eax`

4) Какие строки листинга 6.4 отвечают за вычисления варианта?

- `xor edx, edx` ; обнуление `edx` для `div`
- `mov ebx, 10` ; `ebx=10`
- `div ebx` ; `eax = eax/10`, `edx` - остаток от деления
- `inc edx` ; `edx=edx+1`

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

При `div ebx` остаток от деления записывается в `edx`

6) Для чего используется инструкция “`inc edx`”?

`inc edx` увеличивает значение регистра `edx` на +1

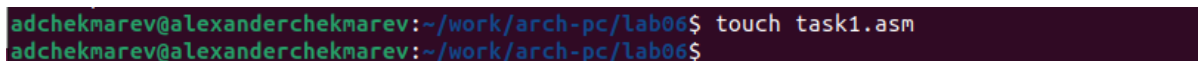
7) Какие строки листинга 6.4 отвечают за вывод на экран результата вычисления?

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

### 3 Самостоятельная работа

**Задание№1** 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  $f(x)$  выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений  $x_1$  и  $x_2$  из 6.3. При выполнении задания преобразовывать (упрощать) выражения для  $f(x)$  нельзя. При выполнении деления в качестве результата можно использовать только целую часть от деления и не учитывать остаток (т.е.  $5 : 2 = 2$ )

Создадим новый файл для задания и напишем программу для  $f(x)=4/3*(x-1)+5$



```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$ touch task1.asm
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$
```

Рис. 3.1: Рис 3.1.1: Создание файла

```

1 %include 'in_out.asm'
2
3 SECTION .data
4     msg: db 'Введите значение x: ', 0
5     rem: db 'Результат выражения: ', 0
6 SECTION .bss
7 x: RESB 80
8 SECTION .text
9 GLOBAL _start
10
11 _start:
12
13     mov eax, msg
14     call sprintf
15
16
17     mov ecx, x
18     mov edx, 80
19     call sread
20     mov eax, x
21     call atoi
22     xor edx, edx
23
24
25     dec eax
26     mov ebx, 4
27     mul ebx
28     mov ebx, 3
29     div ebx
30     add eax, 5
31     mov edi, eax
32
33
34
35     mov eax, rem
36     call sprintf
37     mov eax, edi
38     call iprintf
39
40
41     call quit
42

```

Рис. 3.2: Рис 3.1.2: Демонстрация программы

Проверим программу

```
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$ nasm -f elf task1.asm
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$ ld -m elf_i386 -o task1 task1.o
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$ ./task1
Введите значение x:
4
Результат выражения: 9
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$ ./task1
Введите значение x:
10
Результат выражения: 17
adchekmarev@alexanderchekmarev:~/work/arch-pc/lab06$
```

Рис. 3.3: Рис 3.1.3: Проверка программы

Загрузим все файлы на github по окончании лаб. работы

## 4 Выводы

Я освоил арифметические инструкции языка ассемблера NASM и приобрел практические навыки по работе с арифметикой в NASM.