

# Отчёт по лабораторной работе 7

## Арифметические операции в NASM.

Львов Сергей НПИбд-02-22

### Содержание

1	Цель работы:.....	1
2	Порядок выполнения лабораторной работы:.....	1
3	Порядок выполнения самостоятельной работы:.....	9
4	Вывод:.....	11

### 1 Цель работы:

Освоение арифметических инструкций языка ассемблера NASM.

### 2 Порядок выполнения лабораторной работы:

#### Символьные и численные данные в NASM.

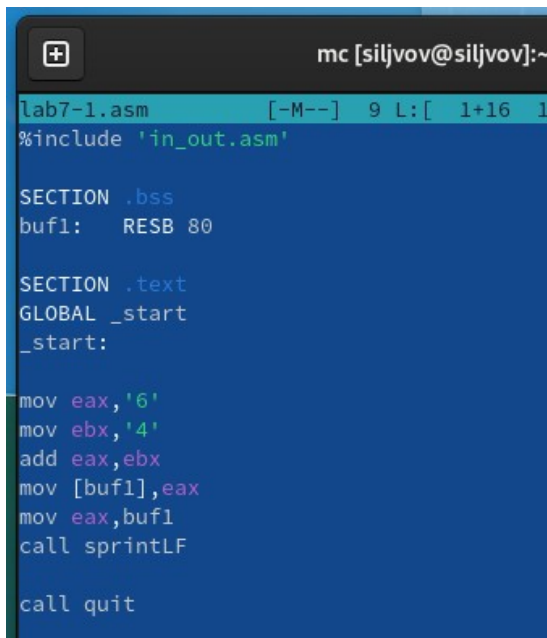
Создадим каталог для программ лабораторной работы №7, перейдем в него и создадим файл lab7-1.asm (рис. 1).

```
[siljvov@siljvov ~]$ mkdir ~/work/arch-pc/lab07
[siljvov@siljvov ~]$ cd ~/work/arch-pc/lab07
[siljvov@siljvov lab07]$ touch lab7-1.asm
[siljvov@siljvov lab07]$
```

Рис. 1. Создание каталога и файла lab7-1.asm

Затем рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр еах.

Введем в файл lab7-1.asm текст программы (рис. 2). В данной программе в регистр еах записывается символ 6 (mov еах,'6'), в регистр ебх символ 4 (mov ебх,'4'). Далее к значению в регистре еах прибавляем значение регистра ебх (add еах,ебх, результат сложения запишется в регистр еах). Далее выводим результат. Так как для работы функции sprintfLF в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра еах в переменную buf1 (mov [buf1],еах), а затем запишем адрес переменной buf1 в регистр еах (mov еах,buf1) и вызовем функцию sprintfLF.

A screenshot of a code editor window titled 'mc [siljvov@siljvov]:~'. The editor shows the assembly file 'lab7-1.asm' with the following code:

```
lab7-1.asm      [-M--]  9 L:[ 1+16 1
%include 'in_out.asm'

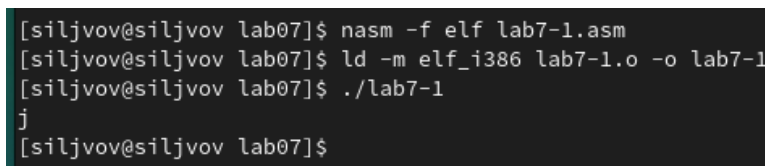
SECTION .bss
buf1:  RESB 80

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. 2. Код программы lab7-1

Затем создадим исполняемый файл и запустим его (рис. 3).

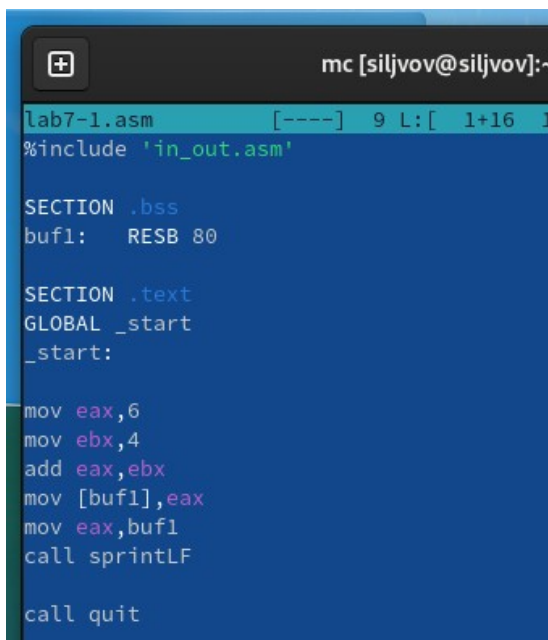
A screenshot of a terminal window showing the following commands and output:

```
[siljvov@siljvov lab07]$ nasm -f elf lab7-1.asm
[siljvov@siljvov lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[siljvov@siljvov lab07]$ ./lab7-1
j
[siljvov@siljvov lab07]$
```

Рис. 3. Результат работы программы lab7-1

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).

Далее изменим текст программы и вместо символов, запишем в регистры числа (рис. 4).

A screenshot of a code editor window titled 'mc [siljvov@siljvov]:-'. The editor shows the assembly file 'lab7-1.asm'. The code includes a header '%include \'in\_out.asm\'', followed by a '.bss' section with a buffer 'buf1' of size 80. Then, a '.text' section is defined with a global '\_start' symbol. The main code starts with 'mov eax,6', 'mov ebx,4', 'add eax,ebx', 'mov [buf1],eax', 'mov eax,buf1', 'call sprintf', and 'call quit'.

```
lab7-1.asm [-----] 9 L: [ 1+16 ]
%include 'in_out.asm'

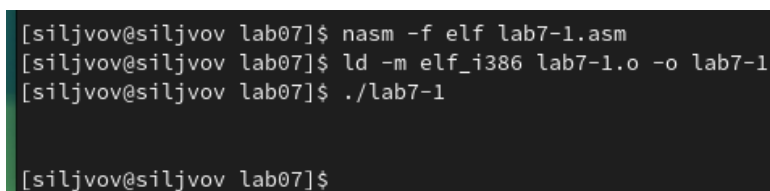
SECTION .bss
buf1:  RESB 80

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. Исправленный код программы lab7-1

Создадим исполняемый файл и запустим его (рис. 5).

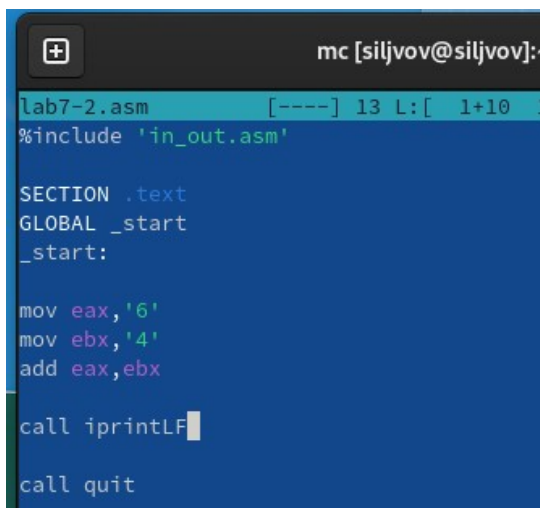
A screenshot of a terminal window showing the compilation and execution of the assembly program. The commands executed are: 'nasm -f elf lab7-1.asm', 'ld -m elf\_i386 lab7-1.o -o lab7-1', and './lab7-1'. The prompt returns to the shell after each command.

```
[siljvov@siljvov lab07]$ nasm -f elf lab7-1.asm
[siljvov@siljvov lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[siljvov@siljvov lab07]$ ./lab7-1
[siljvov@siljvov lab07]$
```

Рис. 5. Результат работы программы lab7-1

Как и в предыдущем случае при исполнении программы мы не получим число 10, в данном случае выводится символ с кодом 10, этому коду соответствует управляющий символ перевода строки.

Для работы с числами в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы с использованием этих функций. Создадим файл lab7-2.asm и введем в него следующий текст (рис. 6).



```
mc [siljvov@siljvov]:-
lab7-2.asm [----] 13 L: [ 1+10
#include 'in_out.asm'

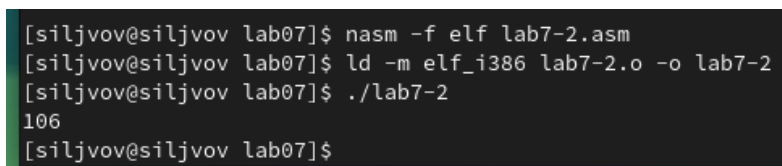
SECTION .text
GLOBAL _start
_start:

mov eax, '6'
mov ebx, '4'
add eax, ebx

call iprintLF
call quit
```

Рис. 6. Код программы lab7-2

Создадим исполняемый файл и запустим его (рис. 7).



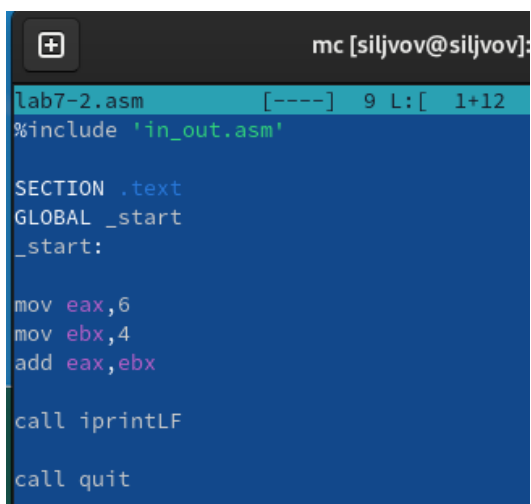
```
[siljvov@siljvov lab07]$ nasm -f elf lab7-2.asm
[siljvov@siljvov lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[siljvov@siljvov lab07]$ ./lab7-2
106
[siljvov@siljvov lab07]$
```

Рис. 7. Результат работы программы

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 7.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа (рис. 8).



```
mc [siljvov@siljvov]:-
lab7-2.asm [----] 9 L: [ 1+12
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

mov eax, 6
mov ebx, 4
add eax, ebx

call iprintLF
call quit
```

Рис. 8. Изменение кода программы lab7-2

В итоге при выполнении программы получился следующий результат (рис. 9).

```
[siljvov@siljvov lab07]$ nasm -f elf lab7-2.asm
[siljvov@siljvov lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[siljvov@siljvov lab07]$ ./lab7-2
10
[siljvov@siljvov lab07]$
```

*Рис. 9. Результат работы программы lab7-2*

Заменяем функцию `iprintLF` из рис. 8 на `iprint` (рис. 10).

```
[siljvov@siljvov lab07]$ nasm -f elf lab7-2.asm
[siljvov@siljvov lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2
[siljvov@siljvov lab07]$ ./lab7-2
10[siljvov@siljvov lab07]$
```

*Рис. 10. Результат работы программы lab7-2 с `iprint`*

Отличие команды `iprint` от `iprintLF` заключается в том, что команда `iprint` не переводит строку.

### **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 `lab7-3.asm` в каталоге `~/work/arch-pc/lab07` и введем в него следующий текст (рис. 11).

```
mc [siljvov@siljvov]:-
lab7-3.asm [----] 0 L:[ 1+ 0
%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток 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. 11. Код программы lab7-3

Создадим исполняемый файл и запустим программу (рис. 12).

```
[siljvov@siljvov lab07]$ nasm -f elf lab7-3.asm
[siljvov@siljvov lab07]$ ld -m elf_i386 lab7-3.o -o lab7-3
[siljvov@siljvov lab07]$ ./lab7-3
Результат: 4
Остаток деления: 1
[siljvov@siljvov lab07]$
```

Рис. 12. Результат работы программы lab7-3

Изменим текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$  (рис. 13). Затем создадим файл и проверим его работу (рис. 14).

```

mc [siljvov@siljvov]:~/
lab7-3.asm [----] 9 L: [ 1+31 32
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток 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. 13. Код новой программы lab7-3

```

[siljvov@siljvov lab07]$ nasm -f elf lab7-3.asm
[siljvov@siljvov lab07]$ ld -m elf_i386 lab7-3.o -o lab7-3
[siljvov@siljvov lab07]$ ./lab7-3
Результат: 5
Остаток деления: 1
[siljvov@siljvov lab07]$

```

Рис. 14. Результат работы программы lab7-3

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

- вывести запрос на введение № студенческого билета
- вычислить номер варианта по формуле:  $(Sn \bmod 20) + 1$ , где  $Sn$  – 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 используется функция `atoi` из файла `in_out.asm`.

Создадим файл `variant.asm` в каталоге `~/work/arch-pc/lab07` и напишем в нем код программы (рис. 15).



```
mc [siljvov@siljvov]:~/work/
variant.asm [----] 9 L: [ 1+33 34/ 34]
#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 ebx

mov eax, rem
call sprintf
mov eax, edx
call iprintLF

call quit
```

Рис. 15. Код программы *variant*



```

[siljvov@siljvov lab07]$ nasm -f elf variant.asm
[siljvov@siljvov lab07]$ ld -m elf_i386 variant.o -o variant
[siljvov@siljvov lab07]$ ./variant
Введите № студенческого билета:
1132221554
Ваш вариант: 14
[siljvov@siljvov lab07]$

```

*Рис. 16. Результат работы программы variant*

В моем случае вариант для всех следующих заданий будет №14.

Ответы на вопросы:

1. За вывод на экран сообщения 'Ваш вариант:' отвечают следующие строки:

```
rem: DB 'Ваш вариант:',0
```

```
mov eax,rem
```

```
call sprint
```

2. Инструкция `nasm` используется для преобразования текста программы в объектный код; инструкция `mov ecx, x` используется для записи адреса под вводимую строку; инструкция `mov edx, 80` используется для определения длины вводимой строки; инструкция `call sread` используется для ввода сообщения с клавиатуры.
3. Инструкция `call atoi` используется для преобразования `ascii`-кода символа в целое число и записывает результат в регистр `eax`.
4. За вычисление варианта отвечают следующие строки кода:

```
xor edx,edx
```

```
mov ebx,20
```

```
div ebx
```

```
inc edx
```

5. Остаток от деления при выполнении инструкции `div ebx` записывается в регистр `edx`.
6. Инструкция `inc edx` используется для увеличения значения регистра `edx` на 1.
7. За вывод на экран результата вычислений отвечают следующие строки кода:

```
mov eax,edx
```

```
call iprintLF
```

### 3 Порядок выполнения самостоятельной работы:

Напишем программу вычисления выражения, в соответствии с вариантом, полученным в предыдущем задании - вариант № 14. Выражение будет следующим:  $(8x + 6) * 10$ . Создадим файл `function.asm` и напомним код (рис. 17).



```
mc [siljvov@siljvov]:-
function.asm [----] 10 L:[ 2+22
SECTION .data
msg DB 'Введите x: ', 0
ans DB 'Ответ: ', 0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintf

mov ecx, x
mov edx, 80
call read

mov eax, x
call atoi

mov edx, 0
mov ecx, 2
div ecx
add eax, 8
mov ebx, 3
mul ebx

mov ebx, eax
mov eax, ans
call sprintf
mov eax, ebx
call iprintLF

call quit
```

Рис. 17. Код программы `function`

Затем создадим исполняемый файл, запустим программу и проверим его для значений  $x_1 = 1$ ;  $x_2 = 4$  (рис. 18).

```
[siljvov@siljvov lab07]$ nasm -f elf function.asm
[siljvov@siljvov lab07]$ ld -m elf_i386 function.o -o function
[siljvov@siljvov lab07]$ ./function
Введите x:
1
Ответ: 24
[siljvov@siljvov lab07]$ ./function
Введите x:
4
Ответ: 30
[siljvov@siljvov lab07]$
```

*Рис. 18. Результат работы программы function*

#### **4 Вывод:**

Во время выполнения лабораторной работы были освоены арифметические инструкции языка ассемблера NASM: add – сложение, sub – вычитание, mul – умножение, div – деление нацело, inc – увеличение на 1, dec – уменьшение на 1, neg – изменение знака числа.