

# **Отчёт по лабораторной работе №7**

**Арифметические операции в NASM**

Мулин Иван Владимирович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Ход работы</b>	<b>5</b>
2.1	Выполнение лабораторной работы . . . . .	5
2.1.1	Анализ программы variant . . . . .	7
2.2	Выполнение заданий для самостоятельной работы . . . . .	8
<b>3</b>	<b>Листинги написанных программ</b>	<b>9</b>
<b>4</b>	<b>Заключение</b>	<b>15</b>

## Список иллюстраций

2.1	Запуск программы 1, дубль 1 . . . . .	5
2.2	Запуск программы 1, дубль 2 . . . . .	5
2.3	Запуск программы 2, дубль 1 . . . . .	6
2.4	Запуск программы 2, дубль 2 . . . . .	6
2.5	Значение выражения $(5 * 2 + 3)/3$ . . . . .	6
2.6	Значение выражения $(4 * 6 + 2)/5$ . . . . .	6
2.7	Вычисление номера варианта . . . . .	7
2.8	Запуск программы для самостоятельной работы . . . . .	8

# 1 Цель работы

Цель выполнения лабораторной работы № 7 - изучить арифметические операции в языке ассемблера NASM. Репозиторий github можно найти по адресу [https://github.com/ivmulin/study\\_2022-2023\\_arch-pc](https://github.com/ivmulin/study_2022-2023_arch-pc).

## 2 Ход работы

### 2.1 Выполнение лабораторной работы

Вычислим значение выражения

$$6 + 4.$$

В рабочей директории напомним программу, складывающую коды символов в двоичном представлении и проверяем её работу:

```
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-1
j
```

Рис. 2.1: Запуск программы 1, дубль 1

В изменённой версии в консоль выводится символ с кодом  $6 + 4 = 10$  - символ перевода строки в таблице ASCII:

```
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-1
```

Рис. 2.2: Запуск программы 1, дубль 2

Программа lab7-2 выводит ожидаемый нами результат:

```
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-2
10
```

Рис. 2.3: Запуск программы 2, дубль 1

Заменим в программе функцию `iprintLF` на `iprint`, которая выводит число без перевода строки:

```
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-2
10ivmulin@dk6n52 ~/work/arch-pc/lab07 $
```

Рис. 2.4: Запуск программы 2, дубль 2

Программа `lab7-3` вычисляет значение выражения

$$\frac{5 \cdot 2 + 3}{3} :$$

```
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-3
Результат: 4
Остаток от деления: 1
```

Рис. 2.5: Значение выражения  $(5 * 2 + 3)/3$

Изменённая версия этой же программы должна вычислять значение числа  $\frac{4 \cdot 6 + 2}{5}$ .

```
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-3
Результат: 5
Остаток от деления: 1
```

Рис. 2.6: Значение выражения  $(4 * 6 + 2)/5$

Программа `variant` вычисляет номер варианта самостоятельной работы по формуле

$$V = (S_n \bmod 20) + 1.$$

Очевидно, программа работает корректно:

```
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf variant.asm
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o variant variant.o
ivmulin@dk6n52 ~/work/arch-pc/lab07 $ ./variant
Введите No студенческого билета:
1132226470
Ваш вариант: 11
```

Рис. 2.7: Вычисление номера варианта

### 2.1.1 Анализ программы `variant`

В программе `variant` строки

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

отвечают за вывод в консоль надписи “Ваш вариант:”.

Строки

```
mov ecx, x
mov edx, 80
call sread
```

отвечают за ввод значения переменной `x`.

Инструкция `call atoi` преобразует значение регистра `eax` из кода ASCII в число.

Непосредственно номер варианта вычисляют следующие строки:

```
mov ebx, 20
div ebx
inc edx
```

Остаток от целочисленного деления помещается в регистр `edx`.

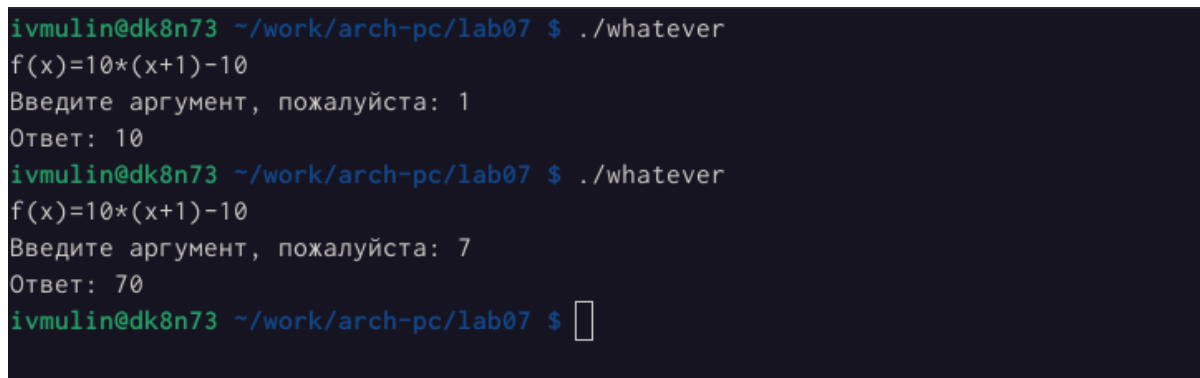
Команда `inc edx` увеличивает на единицу значение в соответствующем регистре.

Результат вычислений осуществляется посредством строк

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

## 2.2 Выполнение заданий для самостоятельной работы

В ходе самостоятельной работы необходимо написать программу, вычисляющую значение многочлена  $f(x) = 10 \cdot (x + 1) - 10$ , причём значение аргумента нужно получить от пользователя. В качестве входных значений используем сначала 1, затем 7:



```
ivmulin@dk8n73 ~/work/arch-pc/lab07 $ ./whatever
f(x)=10*(x+1)-10
Введите аргумент, пожалуйста: 1
Ответ: 10
ivmulin@dk8n73 ~/work/arch-pc/lab07 $ ./whatever
f(x)=10*(x+1)-10
Введите аргумент, пожалуйста: 7
Ответ: 70
ivmulin@dk8n73 ~/work/arch-pc/lab07 $
```

Рис. 2.8: Запуск программы для самостоятельной работы

Как видно, программа работает корректно.



### 3 Листинги написанных программ

1. lab7-1.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 sprintLF
```

```
    call quit
```

2. lab7-2.asm

```
%include 'in_out.asm'
```

```
section .text
    global _start
```

```
_start:
    mov eax, 6
    mov ebx, 4
    add eax, ebx
    call iprint

    call quit
```

### 3. lab7-3.asm

```
%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
mov eax, div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprintfLF ; 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 iprintfLF ; из 'edx' (остаток) в виде символов

call quit ; вызов подпрограммы завершения

```

#### 4. variant.asm

```
%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 ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx, edx
mov ebx, 20
div ebx
inc edx

mov eax, rem
call sprint

mov eax, edx
call iprintLF

call quit

```

5. whatever.asm

```
; f(x)=10*(x+1)-10, 1, 7
```

```
%include 'in_out.asm'
```

```
section .data
```

```

function: db 'f(x)=10*(x+1)-10', 0
msg: db 'Введите аргумент, пожалуйста: ', 0
ans: db 'Ответ: ', 0

```

```
section .bss
```

```
x: resb 80
```

```

section .text
    global _start

_start:
    mov eax, function
    call printf

    mov eax, msg
    call printf

    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi ; eax=x

    inc eax
    mov ebx, 10
    mul ebx
    sub eax, 10
    mov edx, eax

    mov eax, ans
    call printf

    mov eax, edx
    call iprintf

```

`call quit`

## 4 Заключение

Цель данной лабораторной работы была достигнута, потому как были изучены арифметические операции (сложение, вычитание, умножение, деление, домножение на -1) на языке NASM.