

# **Отчёт по лабораторной работе №6**

**Дисциплина: архитектура компьютера**

Булыгин Николай Александрович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Символьные и численные данные в NASM . . . . .	7
3.2	Выполнение арифметических операций в NASM . . . . .	10
3.3	Выполнение самостоятельной работы . . . . .	16
<b>4</b>	<b>Выводы</b>	<b>19</b>

## Список иллюстраций

3.1	Подготовка . . . . .	7
3.2	Программа . . . . .	8
3.3	Запуск . . . . .	9
3.4	Символ с кодом 10 . . . . .	9
3.5	Код . . . . .	9
3.6	Запуск . . . . .	10
3.7	Сумма . . . . .	10
3.8	Программа . . . . .	11
3.9	Успешное выполнение . . . . .	12
3.10	Изменения . . . . .	12
3.11	Проверка . . . . .	13
3.12	Вычислитель варианта . . . . .	14
3.13	Мой вариант . . . . .	15
3.14	Работа . . . . .	18

## **Список таблиц**

# 1 Цель работы

Целью данной лабораторной работы является освоение арифметических инструкций языка ассмблера NASM.

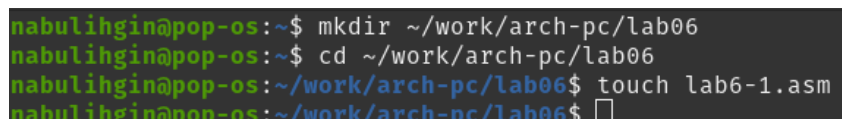
## **2 Задание**

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

## 3 Выполнение лабораторной работы

### 3.1 Символьные и численные данные в NASM

Открываю терминал, создаю рабочий каталог и в нём создаю файл lab6-1.asm (рис. 3.1).



```
nabulihgin@pop-os:~$ mkdir ~/work/arch-pc/lab06
nabulihgin@pop-os:~$ cd ~/work/arch-pc/lab06
nabulihgin@pop-os:~/work/arch-pc/lab06$ touch lab6-1.asm
nabulihgin@pop-os:~/work/arch-pc/lab06$
```

Рис. 3.1: Подготовка

Ввожу в созданный файл данный текст программы (рис. 3.2).

```

*lab6-1.asm
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. 3.2: Программа

Создаю исполняемый файл и запускаю его, предварительно поместив in\_out.asm в этот каталог для корректной работы программы. Она выводит символ j, так как в программе мы складываем не числа, а коды их символов (рис. 3.3).



```
nabulihgin@pop-os:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
nabulihgin@pop-os:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
nabulihgin@pop-os:~/work/arch-pc/lab06$ ./lab6-1
j
nabulihgin@pop-os:~/work/arch-pc/lab06$ █
```

Рис. 3.3: Запуск

Меняю символы '6' и '4' на числа 6 и 4 в коде программы и запускаю обновлённую программу. Теперь выводится символ с кодом 10, чем является символ перехода на следующую строку (рис. 3.4).

```
nabulihgin@pop-os:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
nabulihgin@pop-os:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
nabulihgin@pop-os:~/work/arch-pc/lab06$ ./lab6-1

nabulihgin@pop-os:~/work/arch-pc/lab06$ █
```

Рис. 3.4: Символ с кодом 10

Создаю новый файл lab6-2.asm и ввожу в него данную программу (рис. 3.5).

```
lab6-2.asm

%include 'in_out.asm'

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

Рис. 3.5: Код

Запускаю программу и вижу, что она выводит число 106, соответствующее сумме кодов символов '4' и '6' (рис. 3.6).

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

Рис. 3.6: Запуск

Как и в прошлый раз, меняю символы '4' и '6' на числа 4 и 6. Теперь программа выводит их сумму (рис. 3.7).

```
nabulihgin@pop-os:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
nabulihgin@pop-os:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
nabulihgin@pop-os:~/work/arch-pc/lab06$ ./lab6-2
10
nabulihgin@pop-os:~/work/arch-pc/lab06$
```

Рис. 3.7: Сумма

## 3.2 Выполнение арифметических операций в NASM

Создаю новый файл lab6-3.asm и ввожу в него данную программу вычисления выражения (рис. 3.8).

### lab6-3.asm

```
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от 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. 3.8: Программа

Запускаю программу, результат соответствует данному выражению (рис. 3.9).

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

Рис. 3.9: Успешное выполнение

Меняю текст программы для вычисления нового выражения (рис. 3.10).

```
10
11 mov eax,4
12 mov ebx,6
13 mul ebx
14 add eax,2
15 xor edx,edx
16 mov ebx,5
17 div ebx
18 mov edi,eax
19
```

Рис. 3.10: Изменения

Запускаю программу и проверяю правильность вывода. Результат равен 5, остаток равен 1, значит программа сработала верно (рис. 3.11).

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

Рис. 3.11: Проверка

Теперь создаю файл `variant.asm` и ввожу туда данную программу вычисления варианта (рис. 3.12).

```

variant.asm
7 SECTION .bss
8 x: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13
14 mov eax,msg
15 call sprintf
16
17 mov ecx,x
18 mov edx,80
19 call sread
20
21 mov eax,x
22 call atoi
23
24 xor edx,edx
25 mov ebx,20
26 div ebx
27 inc edx
28
29 mov eax,rem
30 call sprintf
31 mov eax,edx
32 call iprintf
33
34 call quit|

```

Рис. 3.12: Вычислитель варианта

Запускаю программу. Вариант вычислен верно, так как сумма последней цифры моего билета и единицы равна двойке (рис. 3.13).

```
nabulhgin@pop-os:~/work/arch-pc/lab06$ nasm -f elf variant.asm
nabulhgin@pop-os:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
nabulhgin@pop-os:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246801
Ваш вариант: 2
nabulhgin@pop-os:~/work/arch-pc/lab06$
```

Рис. 3.13: Мой вариант

Ответы на вопросы лабораторной работы:

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

Эти строки:

```
mov eax, msg
call printf
```

- 2) Для чего используются следующие инструкции?

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

Для считывания ввода из терминала. Первая строка поместит считанное значение в x, вторая строка задаёт максимальный объем данных для считывания, третья - вызывает функцию считывания строки из терминала.

- 3) Для чего используется инструкция "call atoi"?

Она преобразует регистр eax в число.

- 4) Какие строки листинга 6.4 отвечают за вычисления варианта?

Эти строки:

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

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

edx.

6) Для чего используется инструкция “inc edx”?

Для увеличения значения в edx на 1.

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

Эти строки:

```
mov eax,rem
call sprint
mov eax,edx
call iprintLF
```

### 3.3 Выполнение самостоятельной работы

Для второго варианта выражение  $(12x+3)^5$  для  $x=1$  и  $x=6$ .

Копирую файл lab6-3.asm, переименовываю его на task.asm и изменяю его для вычисления данного выражения:

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

```
SECTION .data
```

```
msg: DB 'Enter x',0
```



```
div: DB 'Result: ',0
```

```
SECTION .bss
```

```
x: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
;Stuff from variant.asm
```

```
mov eax,msg
```

```
call sprintf
```

```
mov ecx,x
```

```
mov edx,80
```

```
call sread
```

```
mov eax,x
```

```
call atoi
```

```
mov ebx,12
```

```
mul ebx
```

```
add eax,3
```

```
xor edx,edx
```

```
mov ebx,5
```

```
div ebx
```

```
mov edi,eax
```

```
mov eax,div
```

```
call sprintf
```

```
mov eax,edi
```

```
call iprintLF
```

```
call quit
```

Запускаю написанную программу и проверяю верность вычислений. Программа даёт верные ответы рис. 3.14).

```
nabulihgin@pop-os:~/work/arch-pc/lab06$ nasm -f elf task.asm
nabulihgin@pop-os:~/work/arch-pc/lab06$ ld -m elf_i386 -o task task.o
nabulihgin@pop-os:~/work/arch-pc/lab06$ ./task
Enter x
1
Result: 3
nabulihgin@pop-os:~/work/arch-pc/lab06$ ./task
Enter x
6
Result: 15
nabulihgin@pop-os:~/work/arch-pc/lab06$ █
```

Рис. 3.14: Работа

## 4 Выводы

Я освоил арифметические инструкции языка ассемблера NASM.