

# **Лабораторная работа №6**

**Архитектура вычислительных систем**

Зеленко Ирина Юрьевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
<b>5</b>	<b>Ответы на вопросы:</b>	<b>16</b>
<b>6</b>	<b>Выводы</b>	<b>17</b>
	<b>Список литературы</b>	<b>18</b>

# Список иллюстраций

4.1	1.png	. . . . .	9
4.2	2.png	. . . . .	10
4.3	3.png	. . . . .	10
4.4	4.png	. . . . .	11
4.5	5.png	. . . . .	11
4.6	6.png	. . . . .	12
4.7	7.png	. . . . .	12
4.8	8.png	. . . . .	13
4.9	9.png	. . . . .	13
4.10	10.png	. . . . .	14
4.11	11.png	. . . . .	14
4.12	12.png	. . . . .	15

## **Список таблиц**

# 1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

## 2 Задание

Написать программу вычисления выражения. 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 выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создать исполняемый файл и проверить его работу для значений из 6.3.

### 3 Теоретическое введение

1. Адресация в NASM Существует три основных способа адресации: • Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`. • Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.
2. Арифметические операции в NASM 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.
3. Целочисленное вычитание `sub` Команда целочисленного вычитания `sub` (от англ. subtraction – вычитание) работает аналогично команде `add`.
4. Команды инкремента и декремента 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.
5. Команда изменения знака операнда `neg` Команда рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.

6. Команды умножения `mul` и `imul` 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` (от англ. `multiply` – умножение). Для знакового умножения используется команда `imul`.
7. Команды деления `div` и `idiv` Для деления, как и для умножения, существует 2 команды `div` (от англ. `divide` – деление) и `idiv`. Для беззнакового умножения используется команда `div`. Для знакового умножения используется команда `idiv`.



## 4 Выполнение лабораторной работы

1. Создаём каталог для программ лабораторной работы No 7, перейдём в него и создаём файл lab6-1.asm

```
iyzelenko@dk8n60 ~ $ mkdir ~/work/arch-pc/lab06
mkdir: невозможно создать каталог «/afs/.dk.sci.pfu.edu.ru/home/i/y/iyzele
pc/lab06»: Файл существует
iyzelenko@dk8n60 ~ $ cd ~/work/arch-pc/lab06
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ nano lab6-1.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 4.1: 1.png

2. Введем в файл lab6-1.asm текст программы из листинга 6.1.

```

/afs/.dk.sci.pfu.edu.ru/home/i/y/iyzelenko/work/a
%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: 2.png

3. Создаём копию файла in\_out.asm в каталоге.

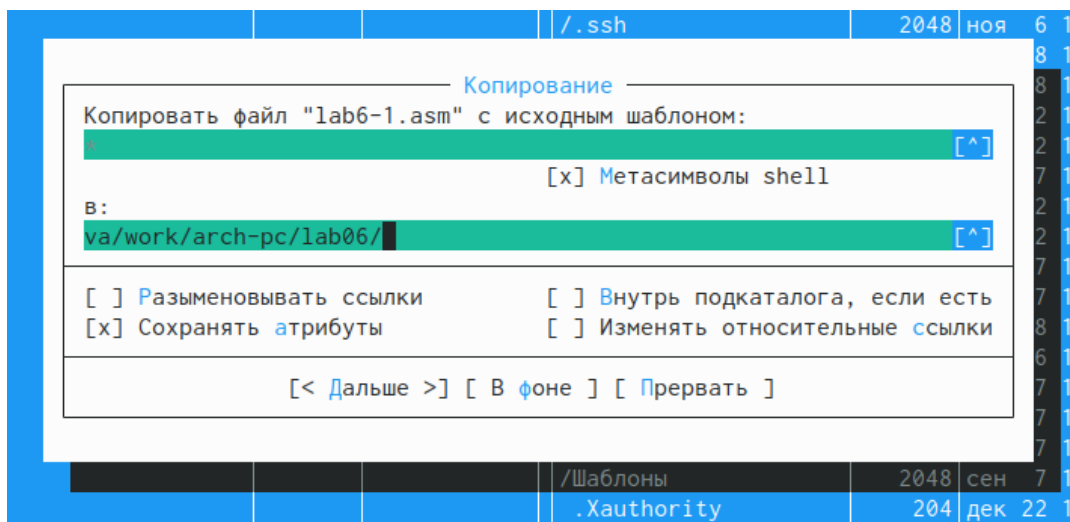


Рис. 4.3: 3.png

4. Создадим исполняемый файл и запустим его.

```
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-1
j
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 4.4: 4.png

5. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы.

```
/afs/.dk.sci.pfu.edu.ru/home/iyzelenko/
%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. 4.5: 5.png

6. Создадим исполняемый файл и запустим его (6-1).

```
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ nano lab6-1.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 4.6: 6.png

```
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ touch lab6-2.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ nano lab6-2.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 4.7: 7.png

7. Создадим файл lab6-2.asm в каталоге. Введем в него текст программы из листинга 6.2 и запустим его.

```
GNU nano 6.4
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

Рис. 4.8: 8.png

8. Изменим символы на числа в lab6-2. Создадим исполняемый файл и запустим его.

```
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ nano lab6-2.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 4.9: 9.png

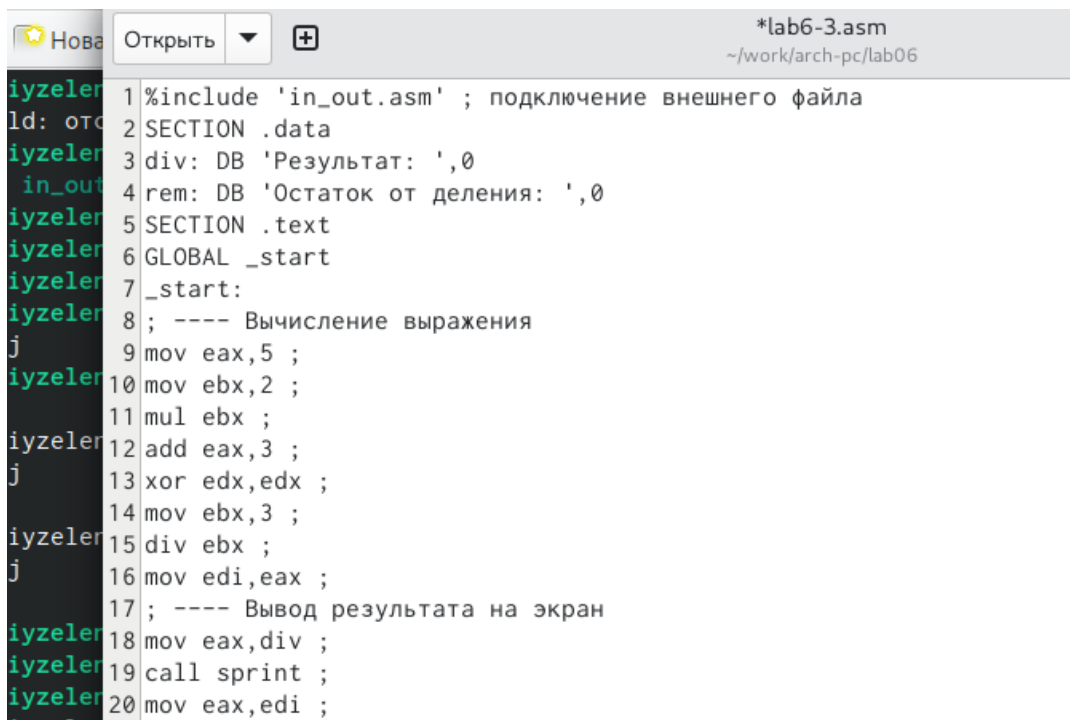
9. Создадим файл lab6-3.asm в каталоге. Введем в файл lab6-3.asm текст про-

граммы из листинга 6.3

```
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ touch lab6-3.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.10: 10.png

10. Введем в файл lab6-3 программу вычисления выражения .



```
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,5 ;
10 mov ebx,2 ;
11 mul ebx ;
12 add eax,3 ;
13 xor edx,edx ;
14 mov ebx,3 ;
15 div ebx ;
16 mov edi,eax ;
17 ; ---- Вывод результата на экран
18 mov eax,div ;
19 call sprint ;
20 mov eax,edi ;
```

Рис. 4.11: 11.png

11. Создадим исполняемый файл и запустим его для вычисления выражения.

```
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
iyzelenko@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 4.12: 12.png

## 5 Ответы на вопросы:

1. строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант:  
mov eax и rem call sprint;
2. mov ecx,x - запись входной переменной в регистр ecx; mov edx, 80 - запись  
размера переменной в регистр edx; call sread - вызов процедуры чтения  
данных;
3. call atoi - функция преобразующая ASCII код символа в целое число и за-  
писывающая результат в регистр eax;
4. xor edx, edx mov ebx, 20 div ebx, inc edx;
5. div ebx - ebx;
6. inc - используется для увеличения операнда на единицу;
7. Следующие строки листинга отвечают за вывод на экран результата вычис-  
лений mov eax, rem call sprint mov eax, edx call iprintLF.



## 6 Выводы

В ходе выполнения данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

## **Список литературы**