

# **Отчет по лабораторной работе №6**

**Дисциплина: архитектура компьютера**

Романова Елизавета Романовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Символьные и численные данные в NASM . . . . .	8
4.2	Выполнение арифметических операций в NASM . . . . .	9
4.2.1	Ответы на вопросы по программе . . . . .	10
4.3	Выполнение заданий для самостоятельной работы . . . . .	11
<b>5</b>	<b>Выводы</b>	<b>13</b>
<b>6</b>	<b>Список литературы</b>	<b>14</b>

## **Список иллюстраций**

# 1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

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

### 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - 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.

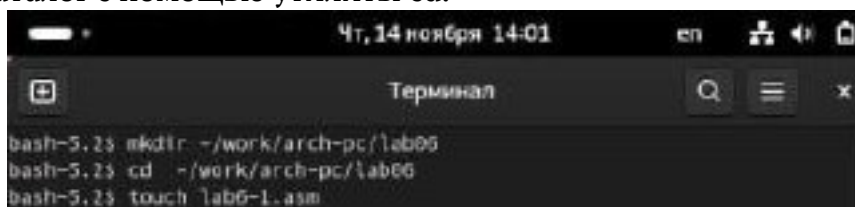
Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного

результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно

## 4 Выполнение лабораторной работы


### 4.1 Символьные и численные данные в NASM

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. ??). Перехожу в созданный каталог с помощью утилиты `cd`.



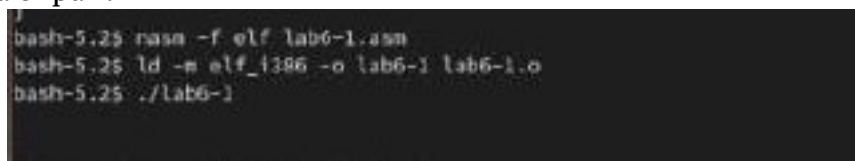
```
bash-5.2$ mkdir -p /work/arch-pc/lab06
bash-5.2$ cd -/work/arch-pc/lab06
bash-5.2$ touch lab6-1.asm
```

Создаю исполняемый файл программы и запускаю его (рис. ??). Вывод программы: символ `j`, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.



```
bash 5.2$ cd -/work/arch-pc/lab06
bash-5.2$ nasm -f elf lab6-1.asm
bash-5.2$ ld -m elf_i386 -o lab6-1 lab6-1.o
bash-5.2$ ./lab6-1
j
bash-5.2$
```

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4. Создаю новый исполняемый файл программы и запускаю его (рис. ??). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.



```
bash-5.2$ nasm -f elf lab6-1.asm
bash-5.2$ ld -m elf_i386 -o lab6-1 lab6-1.o
bash-5.2$ ./lab6-1
```

Создаю новый файл `lab6-2.asm` с помощью утилиты `touch`. Ввожу в файл текст



другой программы для вывода значения регистра eax. Создаю и запускаю исполняемый файл lab6-2 (рис. ??). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

```
bash-5.2$ cd -/work/arch-pc/lab06
bash-5.2$ touch lab6-2.asm
bash-5.2$ cd -/work/arch-pc/lab06
bash-5.2$ nasm -f elf lab6-2.asm
bash-5.2$ ld -m elf_i386 -o lab6-2 lab6-2.o
bash-5.2$ ./lab6-2
106
```

?? width=70% }

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4. Создаю и запускаю новый исполняемый файл (рис. ??).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```
bash-5.2$ nasm -f elf lab6-2.asm
bash-5.2$ ld -m elf_i386 -o lab6-2 lab6-2.o
bash-5.2$ ./lab6-2
10
```

} { #fig:012 width=70% }

## 4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch.

Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 * 2 + 3)/3$ . Создаю исполняемый файл и запускаю его (рис. ??).

```
bash-5.2$ touch -/work/arch-pc/lab06/lab6-3.asm
bash-5.2$ nasm -f elf lab6-3.asm
bash-5.2$ ld -m elf_i386 -o lab6-3 lab6-3.o
bash-5.2$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

} { ?? width=70% }

Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4 * 6 + 2)/5$ . Создаю и запускаю новый исполняемый файл.??

```

bash-5.2$ nasm -f elf lab6-3.asm
bash-5.2$ ld -m elf_i386 -o lab6-3 lab6-3.o
bash-5.2$ ./lab6-3
Результат: 5
Остаток от деления: 1

```

{ ?? width=70% }

Создаю файл variant.asm с помощью утилиты touch Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета. Создаю и запускаю исполняемый файл (рис. ??). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 3.

```

bash 5.2$ nasm -f elf variant.asm
bash-5.2$ ld -m elf_i386 -o variant variant.o
bash-5.2$ ./variant
Введите № студенческого билета:
1132246762
Ваш вариант: 3

```

{ ?? width=70% }

## 4.2.1 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```

mov eax, rem
call sprint

```

2. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
4. За вычисления варианта отвечают строки:

```

xor edx, edx ; обнуление edx для корректной работы div
mov ebx, 20 ; ebx = 20
div ebx ; eax = eax/20, edx - остаток от деления
inc edx ; edx = edx + 1

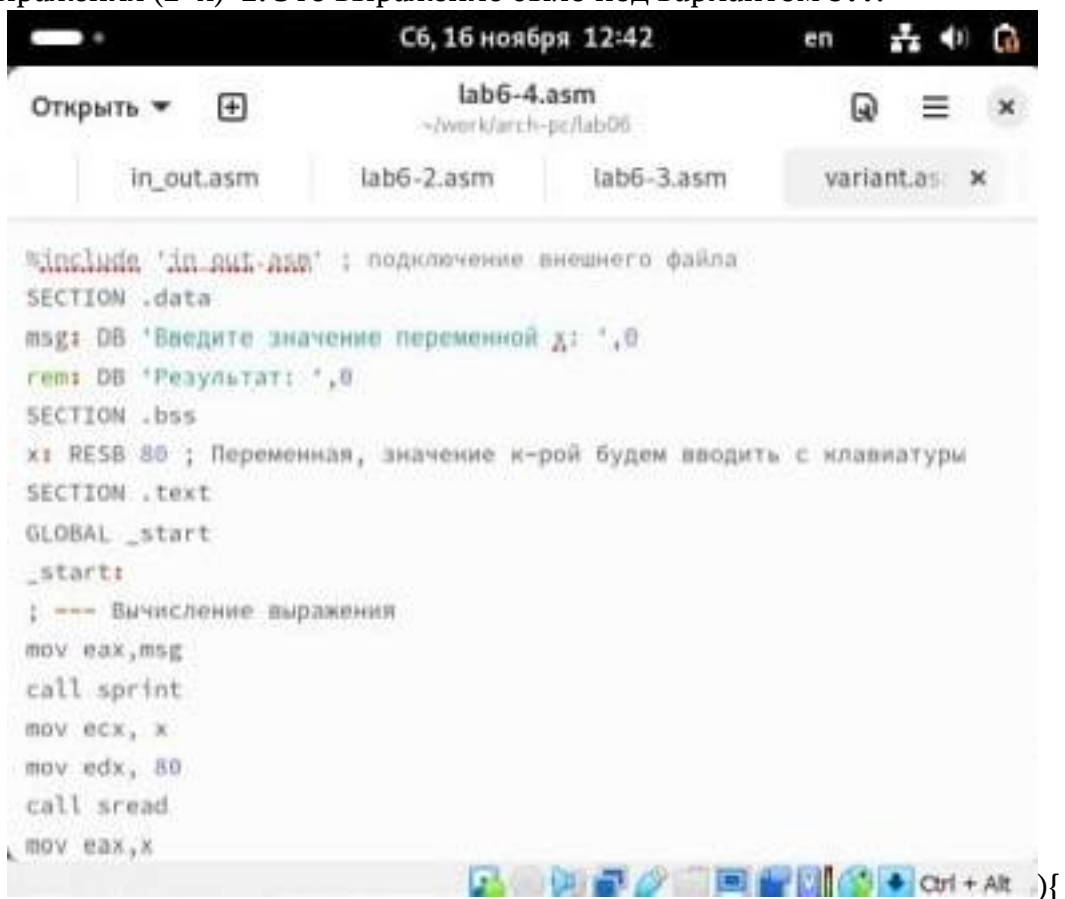
```

5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1
7. За вывод на экран результатов вычислений отвечают строки:

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

## 4.3 Выполнение заданий для самостоятельной работы

Создаю файл `lab6-4.asm` с помощью утилиты `touch`. Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения  $(2+x)^2$ . Это выражение было под вариантом 3??.



```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры
SECTION .text
GLOBAL _start
_start:
; --- Вычисление выражения
mov eax,msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x
```

width=70% }

Создаю и запускаю исполняемый файл (рис. ??).

```
bash 5.2$ nasm -f elf variant.asm
bash-5.2$ ld -m elf_i386 -o variant variant.o
bash-5.2$ ./variant
Введите № студенческого билета:
1132246762
Ваш вариант: 3
```

**Листинг 4.1. Программа для вычисления значения выражения  $(11 + x) * 2$**

– 6.

%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data msg: DB 'Введите значение переменной x:',0  
rem: DB 'Результат:',0  
SECTION .bss x: RESB 80  
; Переменная, значение которой будем вводить с клавиатуры  
SECTION .text GLOBAL \_start \_start: ; — Вычисление выражения  
mov eax,msg call sprint mov ecx,x mov edx,80  
call sread mov eax,x call atoi add eax,2 mov ebx,eax  
mul ebx mov edi,eax ; запись результата вычисления в 'edi'  
; — Вывод результата на экран  
mov eax,rem ; вызов подпрограммы печати  
call sprint ; сообщения 'Результат:'  
mov eax,edi ; вызов подпрограммы печати значения  
call iprintLF ; из 'edi' в виде символов  
call quit ; вызов подпрограммы завершения

## **5 Выводы**

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

## **6 Список литературы**

1. Лабораторная работа №6
2. Таблица ASCII