

Отчет по лабораторной работе 6

Исаев Рамазан Курбанович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Символьные и численные данные в NASM	9
8.1	Выполнение арифметических операций в NASM	18
8.2	Задание для самостоятельной работы	19
9	Выводы	20
10	Контрольные вопросы	21

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

Символьные и численные данные в NASM Выполнение арифметических операций в NASM 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

Создаю каталог для программ лабораторной работы №6 и перехожу в него, создаю там файл

A terminal window titled 'rkisaev@dk3n17 - arch-pc' with search, menu, and window control icons. It shows two lines of command execution: 'rkisaev@dk3n17 ~/work/arch-pc \$ touch lab6-1.asm' and 'rkisaev@dk3n17 ~/work/arch-pc \$' followed by a cursor.

```
rkisaev@dk3n17 ~/work/arch-pc $ touch lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc $
```

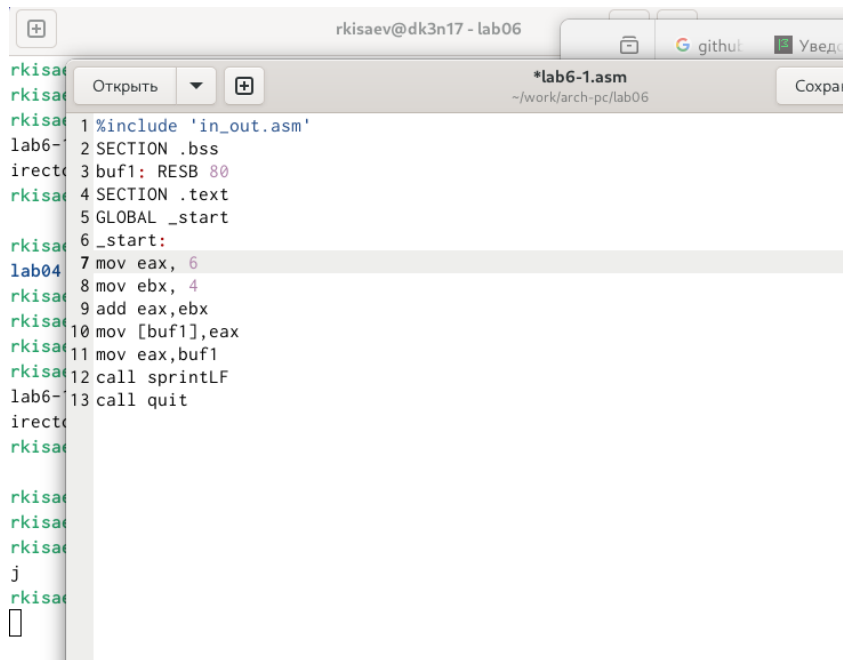
Создаю исполняемый файл и запускаю его, вывод программы отличается от предполагаемого изначально, ибо коды символов в сумме дают символ j по таблице ASCII.

```
rkisaev@dk3n17 - lab06
rkisaev@dk3n17 ~/work/arch-pc $ touch lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc $ gedit lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc $ nasm -f elf lab6-1.asm
lab6-1.asm:1: error: unable to open include file 'in_out.asm': No such file or d
irectory
rkisaev@dk3n17 ~/work/arch-pc $ mc

rkisaev@dk3n17 ~/work/arch-pc $ ls
lab04 lab05 lab06
rkisaev@dk3n17 ~/work/arch-pc $ cd lab06
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ touch lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
lab6-1.asm:1: error: unable to open include file 'in_out.asm': No such file or d
irectory
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ mc

rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-1
j
rkisaev@dk3n17 ~/work/arch-pc/lab06 $
```

5



```
rkisae~ 1 %include 'in_out.asm'
rkisae~ 2 SECTION .bss
lab6~ 3 buf1: RESB 80
irect~ 4 SECTION .text
rkisae~ 5 GLOBAL _start
rkisae~ 6 _start:
rkisae~ 7 mov eax, 6
lab04~ 8 mov ebx, 4
rkisae~ 9 add eax,ebx
rkisae~ 10 mov [buf1],eax
rkisae~ 11 mov eax,buf1
rkisae~ 12 call printf
lab6~ 13 call _exit
irect~
rkisae~
rkisae~
rkisae~
j
rkisae~
█
```

На этот раз программа выдала пустую строчку, это связано с тем, что символ 10 означает переход на новую строку

```
rkisaev@dk3n17 - lab06
irectory
rkisaev@dk3n17 ~/work/arch-pc $ mc

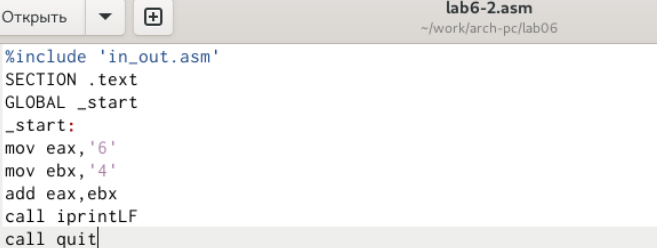
rkisaev@dk3n17 ~/work/arch-pc $ ls
lab04  lab05  lab06
rkisaev@dk3n17 ~/work/arch-pc $ cd lab06
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ touch lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
lab6-1.asm:1: error: unable to open include file 'in_out.asm': No such file or d
irectory
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ mc

rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-1
j
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-1

rkisaev@dk3n17 ~/work/arch-pc/lab06 $
```

6

Создаю новый файл для будущей программы и записываю в нее код из листинга



The screenshot shows a code editor window titled "lab6-2.asm" with the path "~/work/arch-pc/lab06". The editor contains the following assembly code:

```

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5     mov eax, '6'
6     mov ebx, '4'
7     add eax, ebx
8     call iprintLF
9     call quit

```

The code is displayed in a monospaced font with syntax highlighting. The window's title bar includes standard OS controls (minimize, maximize, close) and a search icon. The editor's toolbar shows a button labeled "Открыть" (Open) and a dropdown menu.

Создаю исполняемый файл и запускаю его, теперь отображается результат 106, программа, как и в первый раз, сложила коды символов, но вывела само число, а не его символ, благодаря замене функции вывода на `iprintLF`

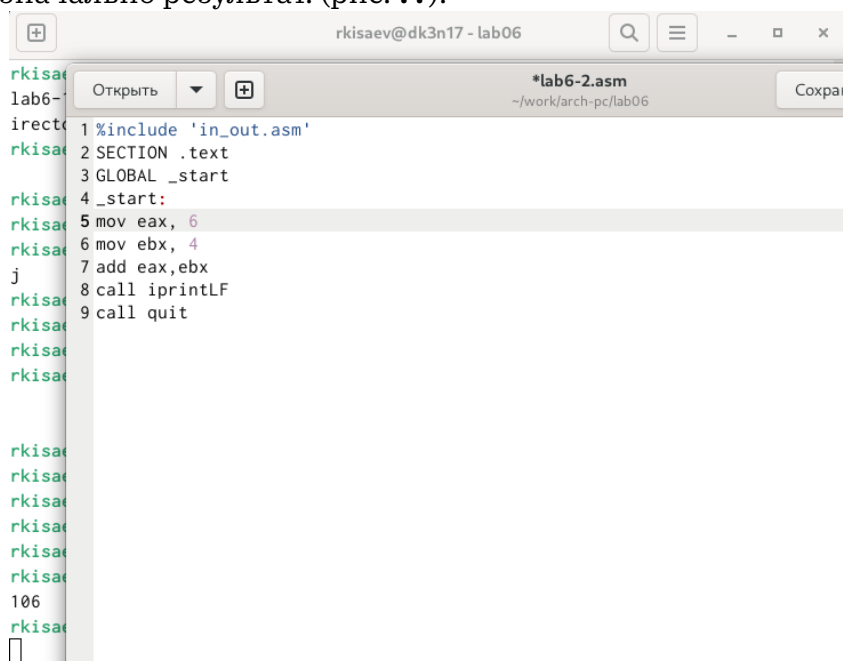
```
rkisaev@dk3n17 - lab06
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
lab6-1.asm:1: error: unable to open include file 'in_out.asm': No such file or d
irectory
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ mc

rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-1
j
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-1

rkisaev@dk3n17 ~/work/arch-pc/lab06 $ touch lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-2
106
rkisaev@dk3n17 ~/work/arch-pc/lab06 $
```

7

Убрав кавычки в программе, я снова ее запускаю и получаю предполагаемый изначально результат. (рис. ??).



The screenshot shows a code editor window titled "rkisaev@dk3n17 - lab06". The editor displays the following assembly code:

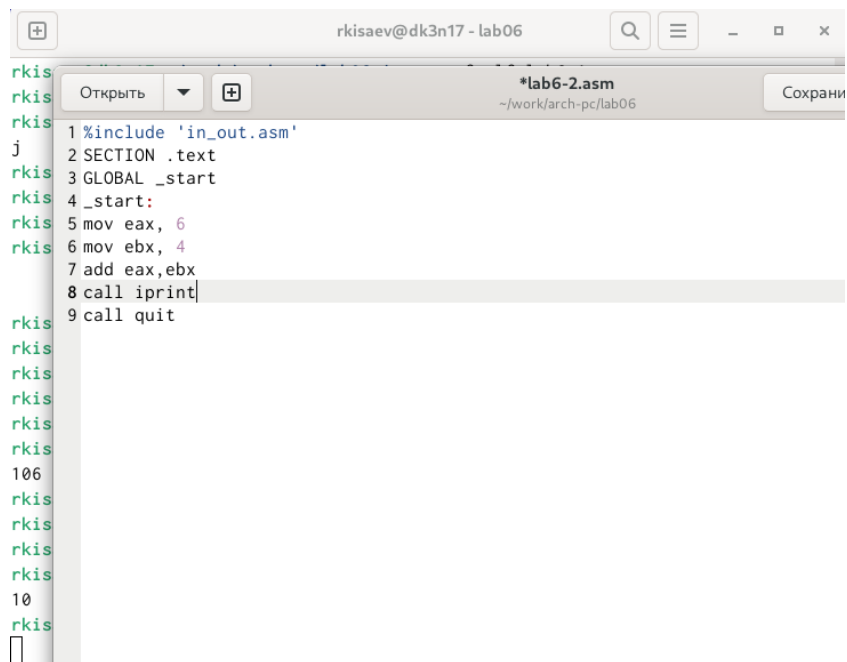
```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, 6
6 mov ebx, 4
7 add eax, ebx
8 call iprintLF
9 call quit
```

The code is highlighted with a light blue background. The editor also shows a file explorer on the left with a tree view containing "lab6-2.asm" and "in_out.asm". The status bar at the bottom indicates the file path as "~/work/arch-pc/lab06".

```
rkisaev@dk3n17 - lab06
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-1
j
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-1

rkisaev@dk3n17 ~/work/arch-pc/lab06 $ touch lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-2
106
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-2
10
rkisaev@dk3n17 ~/work/arch-pc/lab06 $
```


8



The screenshot shows a code editor window with the title bar "rkisaev@dk3n17 - lab06". The editor contains the following assembly code:

```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, 6
6 mov ebx, 4
7 add eax, ebx
8 call iprint
9 call quit
```

The code is displayed in a monospaced font with syntax highlighting. The line numbers 1 through 9 are visible on the left side of the code. The editor has a toolbar with buttons for "Открыть" (Open), "Сохранить" (Save), and a search icon. The file name "lab6-2.asm" is shown in the title bar.

```
rkisaev@dk3n17 - lab06
j
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-1

rkisaev@dk3n17 ~/work/arch-pc/lab06 $ touch lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-2
106
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-2
10
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-2
10rkisaev@dk3n17 ~/work/arch-pc/lab06 $
```

8.1 Выполнение арифметических операций в NASM

Программа выполняет арифметические вычисления, на вывод идет результирующее выражения и его остаток от деления

```
rkisaev@dk3n17 - lab06
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ touch lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-2
106
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-2
10
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-2
10rkisaev@dk3n17 ~/work/arch-pc/lab06 $ touch lab6-3.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
rkisaev@dk3n17 ~/work/arch-pc/lab06 $
```

Запуск программы дает корректный результат

```
rkisaev@dk3n17 - lab06
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета: 1132236010 Ваш вариант: 9

Ваш вариант: 9
1
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета: 1132236010 Ваш вариант: 9
9
Ваш вариант: 9 9
10
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit variant.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
variant.asm:3: warning: character constant too long [-w+other]
variant.asm:3: error: comma expected after operand, got '\0'
variant.asm:4: warning: character constant too long [-w+other]
variant.asm:4: error: comma expected after operand, got '\0'
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit variant.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132236010
Ваш вариант: 11
rkisaev@dk3n17 ~/work/arch-pc/lab06 $
```

8.2 Задание для самостоятельной работы

В соответствии с выбранным вариантом, я реализую программу для подсчета функции $f(x) = 10(x + 1) - 10$, проверка на нескольких переменных показывает корректное выполнение программы

```
rkisaev@dk3n17 - lab06
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit variant.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 1
Результат: 360Ошибка сегментирования (образ памяти сброшен на диск)
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 7
Результат: 2220Ошибка сегментирования (образ памяти сброшен на диск)
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-4.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit variant.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ gedit lab6-4.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x:
1
Результат: 2
rkisaev@dk3n17 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x:
7
Результат: 8
rkisaev@dk3n17 ~/work/arch-pc/lab06 $
```

9 Выводы

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

10 Контрольные вопросы

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`