**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

# ОТЧЕТ
# ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

***Дисциплина*: Архитектура компьютеров**

Студент: Болдырева Дельгир

Группа: НКАбд-01-25

Москва

2025 г.

## Оглавление

# 1 Цель работы

Цель данной лабораторной работы - освоение арифметческих 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. 4.1). Перехожу в созданный каталог с помощью утилиты cd. С помощью утилиты touch создаю файл lab6-1.asm.

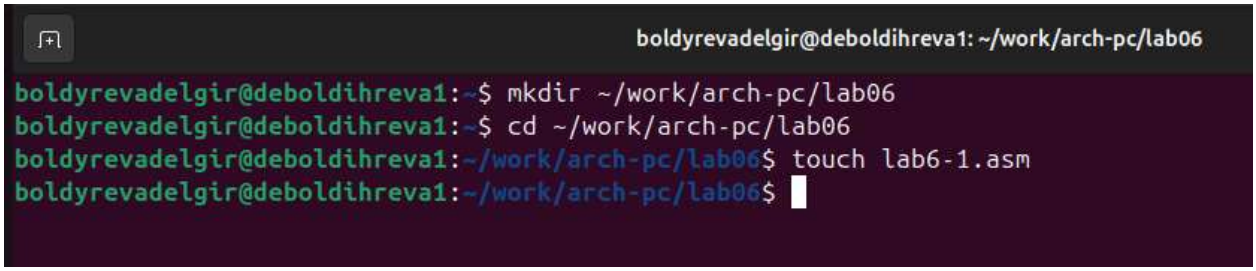


Рис.4.1

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. 4.2).


Рис.4.2

Создаю исполняемый файл программы и запускаю его (рис. 4.3). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

```
j
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ./lab6-1
j
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$
```

Рис.4.3

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис.4.4)

Открыть

• lab6-1.asm
~/work/arch-pc/lab06

lab5-1.asm | lab5-2.asm | • lab6-1.asm × | in_out.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
```

Рис 4.4

Создаю новый исполняемый файл программы и запускаю его (рис. 4.5). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

```
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ./lab6-1


boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$
```

Рис.4.5

Создаю файл lab6-2.asm с помощью утилиты touch (рис. 4.6)

```
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$
```

Рис.4.6

Ввожу в файл текст другой программы для вывода значения регистра eax (рис.4.7)

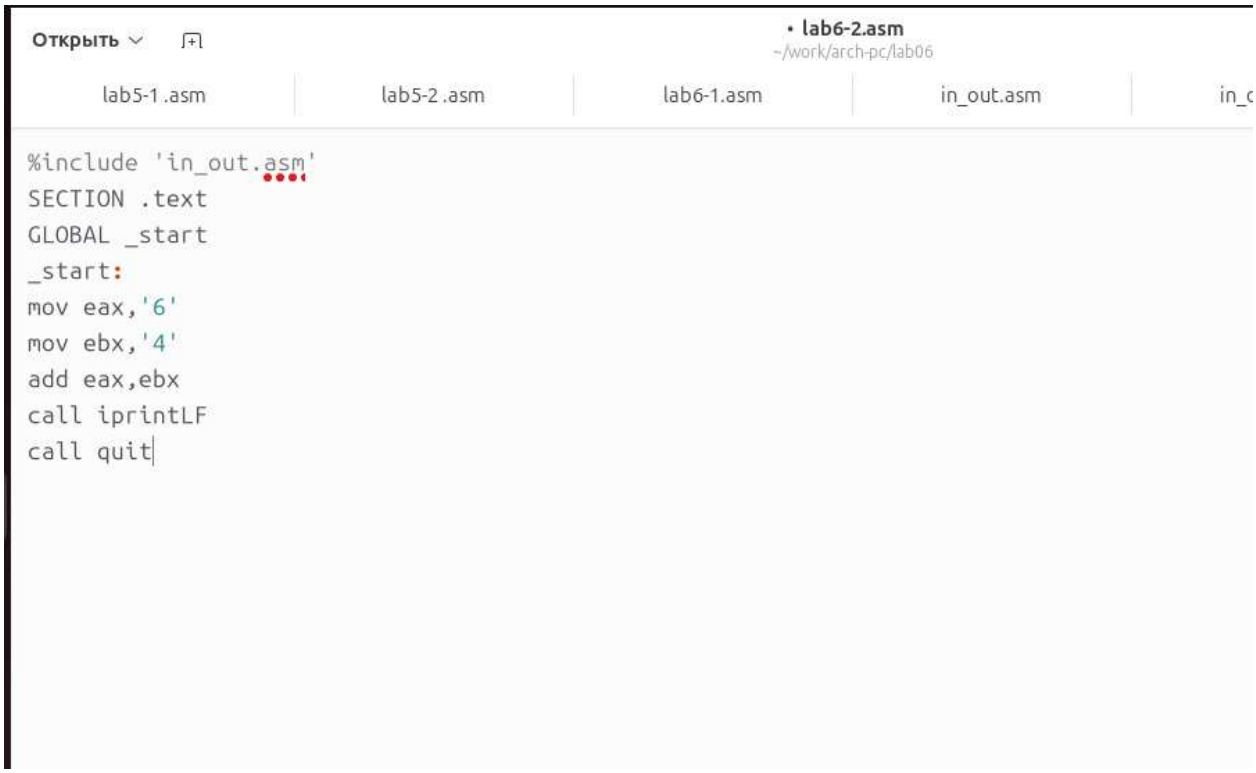


Рис.4.7

Создаю и запускаю исполняемый файл lab6-2 (рис. 4.8). 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”

```
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ./lab6-2
106
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$
```

Рис.4.8

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис.4.9)


Рис.4.9

Создаю и запускаю новый исполняемый файл (рис. 4.10).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.


Рис.4.10

Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.11).

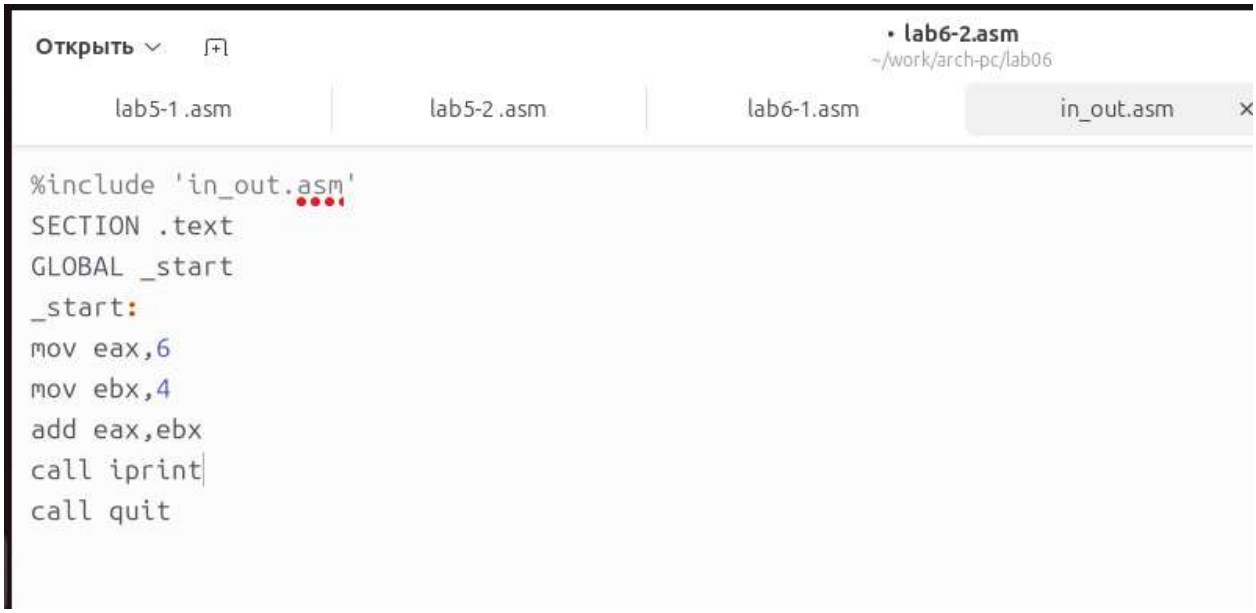


Рис.4.11

Создаю и запускаю новый исполняемый файл (рис. 4.12). Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

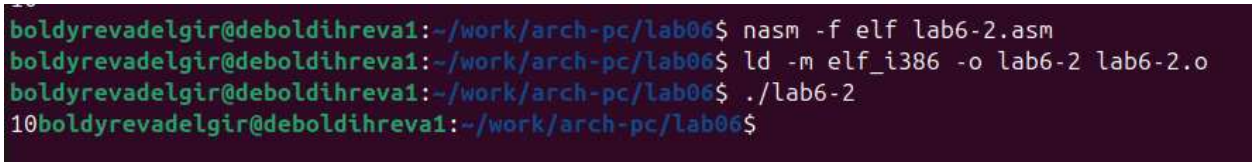


Рис.4.12

## 4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. 4.13).

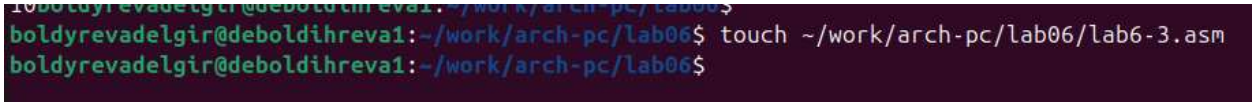


Рис.4.13

Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 * 2 + 3)/3 (рис. 4.14).

Открыть • lab6-3.asm ~/work/arch-pc/lab06

lab5-1.asm | lab5-2.asm | lab6-1.asm | in_out.asm | in_out.asm

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, 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 iprintLF ; 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 iprintLF ; из 'edx' (остаток) в виде символов
```

Рис.4.14

Создаю исполняемый файл и запускаю его (рис. 4.15).

```
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$
```

Рис.4.15

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 * 6 + 2)/5 (рис. 4.16).

Открыть

• lab6-3.asm
~/work/arch-pc/lab06

lab5-1.asm | lab5-2.asm | lab6-1.asm | in_out.asm | in_out.asm | lab6-2.as

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (4 * 6 + 2) / 5
mov eax,4    ; EAX=4
mov ebx,6    ; EBX=6
mul ebx      ; EAX=EAX*EBX = 4*6 = 24
add eax,2    ; EAX=EAX+2 = 24+2 = 26
xor edx,edx  ; обнуляем EDX для корректной работы div
mov ebx,5    ; EBX=5
div ebx      ; EAX=EAX/5 = 26/5 = 5, EDX=остаток 1
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 iprintLF ; из 'edi' в виде символов

mov eax,rem  ; вызов подпрограммы печати
call sprint  ; сообщения 'Остаток от деления: '
```

Рис.4.16

Создаю и запускаю новый исполняемый файл (рис. 4.17). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$
```

Рис.4.17

Создаю файл variant.asm с помощью утилиты touch (рис. 4.18)

```
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$
```

Рис.4.18

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.19)

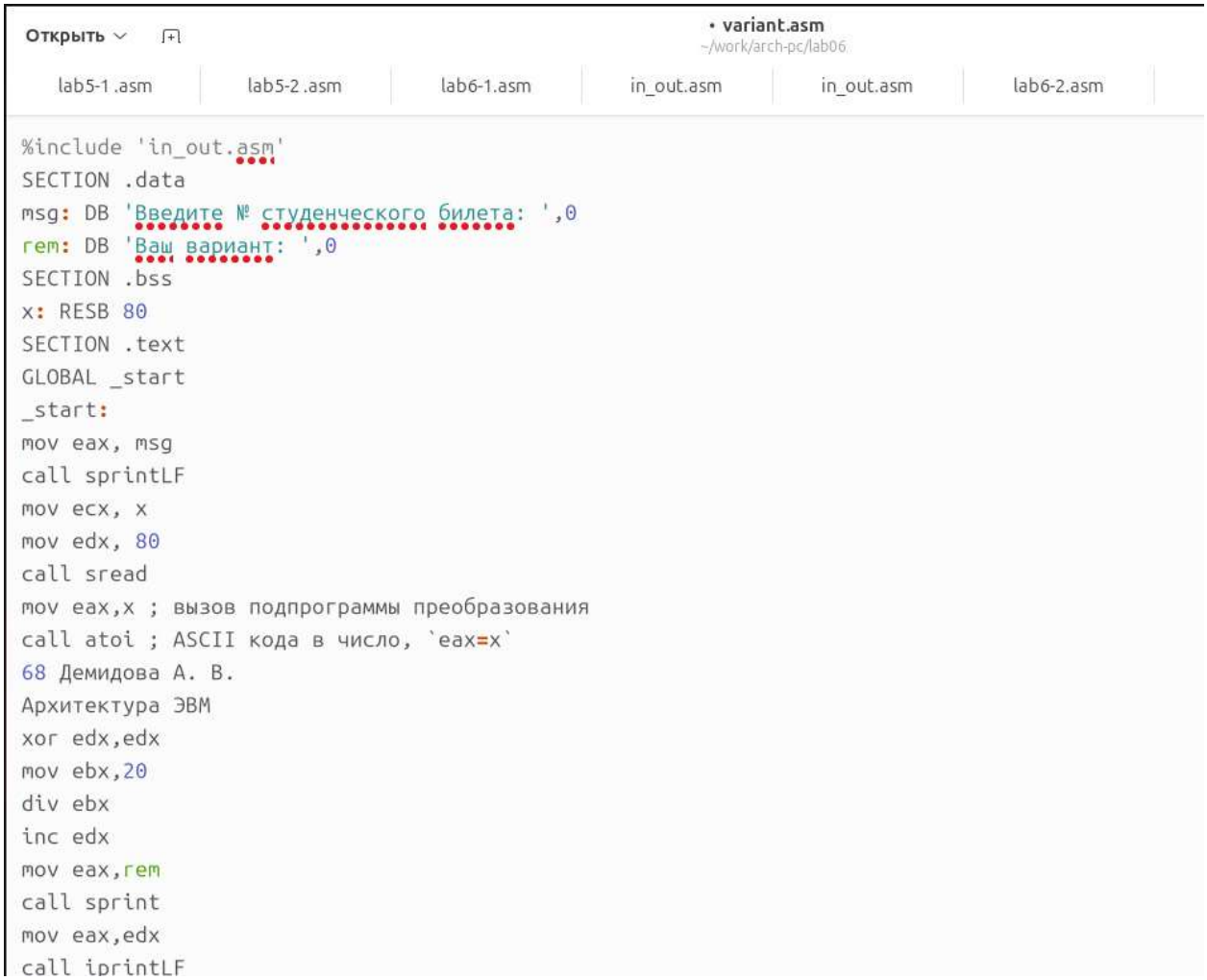


Рис.4.19

Создаю и запускаю исполняемый файл (рис. 4.20). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 3.


Рис.4.20

# 5 Выполнение заданий для самостоятельной работы

Создаю файл lab6-sam.asm с помощью утилиты touch (рис. 4.21).

```
boldyrevadelgir@deboldihreva1: ~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-sam.asm
boldyrevadelgir@deboldihreva1: ~/work/arch-pc/lab06$
```

Рис.4.21

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения (2+x)^2 (рис. 4.22). Это выражение было под вариантом 3.

Открыть • lab6-sam.asm ~/work/arch-pc/lab06

lab5-1.asm | lab5-2.asm | lab6-1.asm | in_out.asm | in_out.asm | lab6-2.asm | lab6-3.asm

```
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data ; секция инициированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0

SECTION .bss ; секция не инициированных данных
x: RESB 80 ; Переменная, значение которой будем вводить с клавиатуры

SECTION .text ; Код программы
GLOBAL _start ; Начало программы

_start: ; Точка входа в программу
mov eax, msg ; запись адреса выводимого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения

mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения

mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, eax = x

; ---- Вычисление выражения (2 + x)^2 ----
add eax, 2    ; eax = 2 + x
```

Рис.4.22

Создаю и запускаю исполняемый файл (рис. 4.23). При вводе значения 2, вывод – 16, при вводе значения 8, вывод-100.


boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-sam.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ nasm -f elf lab6-sam.asm
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-sam lab6-sam.o
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ./lab6-sam
Введите значение переменной x: 2
Результат: 16
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$ ./lab6-sam
Введите значение переменной x: 8
Результат: 100
boldyrevadelgir@deboldihreva1:~/work/arch-pc/lab06$

Рис.4.23

# 6 Выводы

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

## Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.

2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.

3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander.

org/.

4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.

5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. —

354 c. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.