

Отчёт по лабораторной работе № 6

Дисциплина: Архитектура компьютера

Кижваткина Анна Юрьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	17

Список иллюстраций

4.1	Создание каталога	8
4.2	Перемещение в каталог	8
4.3	Создание файла	8
4.4	Изменение файла	9
4.5	Перенос файла	9
4.6	Создание исполняемого файла и его запуск	9
4.7	Изменение программы	10
4.8	Создание исполняемого файла и его запуск	10
4.9	Создание файла	10
4.10	Ввод программы	11
4.11	Создание исполняемого файла и его запуск	11
4.12	Изменение программы	11
4.13	Создание исполняемого файла и его запуск	12
4.14	Изменение программы	12
4.15	Создание исполняемого файла и его запуск	12
4.16	Создание файла	12
4.17	Ввод программы	13
4.18	Создание исполняемого файла и его запуск	13
4.19	Изменение программы	13
4.20	Создание исполняемого файла и его запуск	14
4.21	Создание файла	14
4.22	Ввод программы	14
4.23	Создание исполняемого файла и его запуск	14
4.24	Создание файла	15
4.25	Ввод программы	16
4.26	Создание исполняемого файла и его запуск	16

Список таблиц

1 Цель работы

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

2 Задание

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

3 Теоретическое введение

Большинство указаний на языке ассемблера требуют обработки операндов. Адрес операнда указывает на место, где хранятся данные, которые необходимо обработать. Эти данные могут быть хранящимися в регистре или ячейке памяти.

Адресация по регистрам – данные хранятся в регистрах, и в команде используются имена этих регистров, например: `mov ax, bx`.

Непосредственная адресация – значение операнда указывается непосредственно в команде, например: `mov ax, 2`.

Адресация памяти – операнд указывает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой нужно выполнить операцию.

Ввод информации с клавиатуры и вывод ее на экран происходит в символьном виде в соответствии с таблицей символов ASCII.

Согласно стандарту ASCII, каждый символ кодируется одним байтом. Среди инструкций NASM отсутствует инструкция для вывода чисел (не в символьном виде). Поэтому для вывода чисел нужно сначала преобразовать цифры в ASCII-коды и выводить на экран эти коды, а не сами числа.

Подобная ситуация возникает и при вводе данных с клавиатуры – введенные данные будут представлять символы, что затрудняет арифметические операции с ними. Для решения этой проблемы необходимо преобразовать ASCII символы в числа и наоборот.

4 Выполнение лабораторной работы

1. Символьные и численные данные в NASM.

Создаем каталог для программ лабораторной работы №6 (рис. 4.1).

```
aykizhvatkina@dkln22 ~ $ mkdir ~/work/arch-pc/lab06
```

Рис. 4.1: Создание каталога

Переходим в созданный каталог. (рис. 4.2).

```
aykizhvatkina@dkln22 ~ $ cd ~/work/arch-pc/lab06
```

Рис. 4.2: Перемещение в каталог

Создаем файл lab6-1.asm. (рис. 4.3).

```
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

Рис. 4.3: Создание файла

Рассмотрим примеры программ вывода символьных и численных значений.
Вводим в файле программу с листинга 6.1 (рис. 4.4).

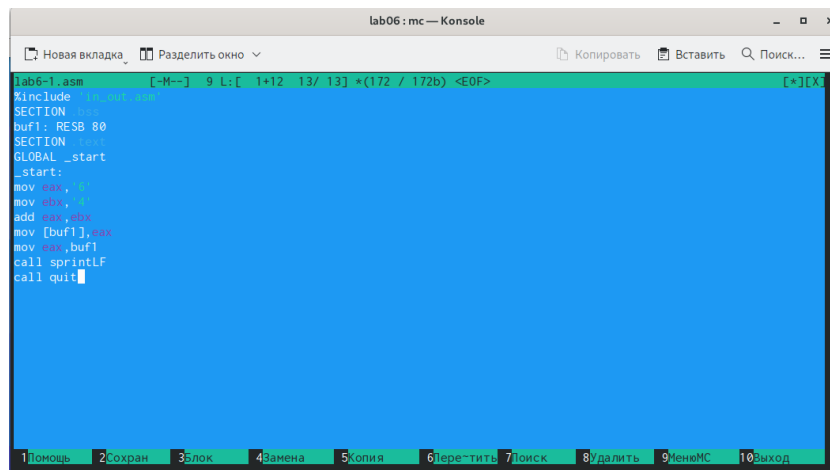


Рис. 4.4: Изменение файла

Скачиваем и переносим файл `in_out.asm` в каталог лабораторной работы 6. (рис. 4.5).

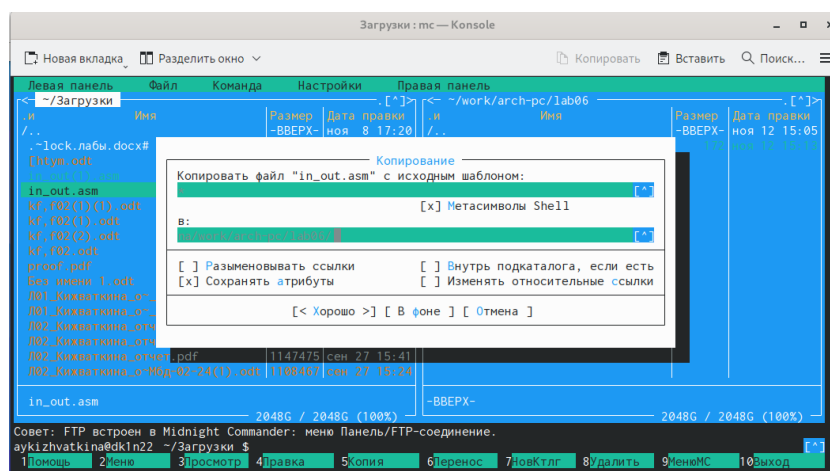


Рис. 4.5: Перенос файла

Создаем исполняемый файл и запускаем его. Программа выводит символ `j` так как программа выводит символ, соответствующий сумме двоичных кодов 6 и 4. (рис. 4.6).

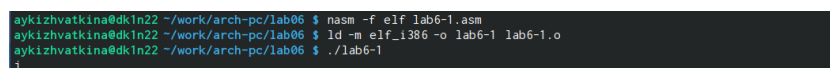


Рис. 4.6: Создание исполняемого файла и его запуск

Изменяем в тексте программы символы '6' и '4' на 6 и 4. (рис. 4.7).

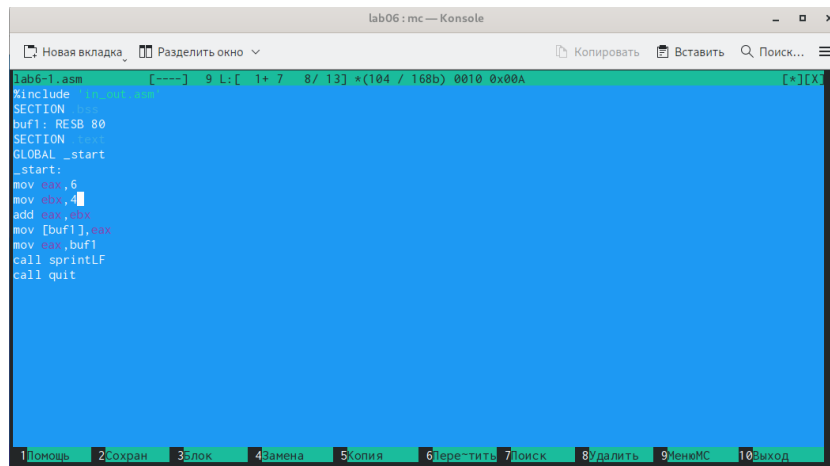


Рис. 4.7: Изменение программы

Создаем исполняемый файл и запускаем его. Теперь вывелся символ с кодом 10. Это символ перевода строки, он не отображается на экране. (рис. 4.8).

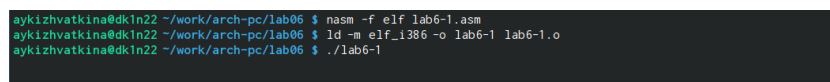


Рис. 4.8: Создание исполняемого файла и его запуск

Создаем файл lab6-2.asm. (рис. 4.9).

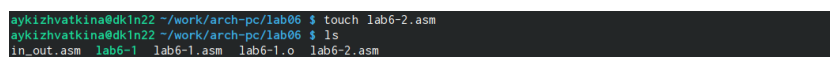
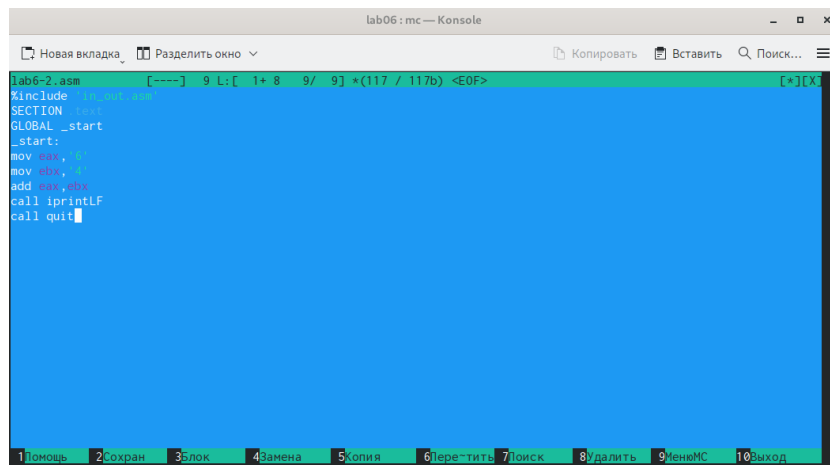


Рис. 4.9: Создание файла

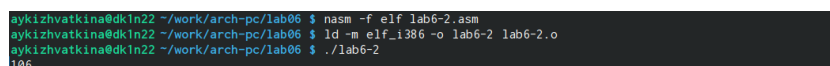
Вводим в файл текст программы из листинга 6.2. (рис. 4.10).



```
lab6-2.asm [----] 9 L: [ 1+ 8 9/ 9] *(117 / 117b) <EOF> [*][X]
#include <stdio.h>
SECTION .text
GLOBAL _start
_start:
mov eax, 1
mov ebx, 8
add eax, ebx
call iprintLF
call quit
```

Рис. 4.10: Ввод программы

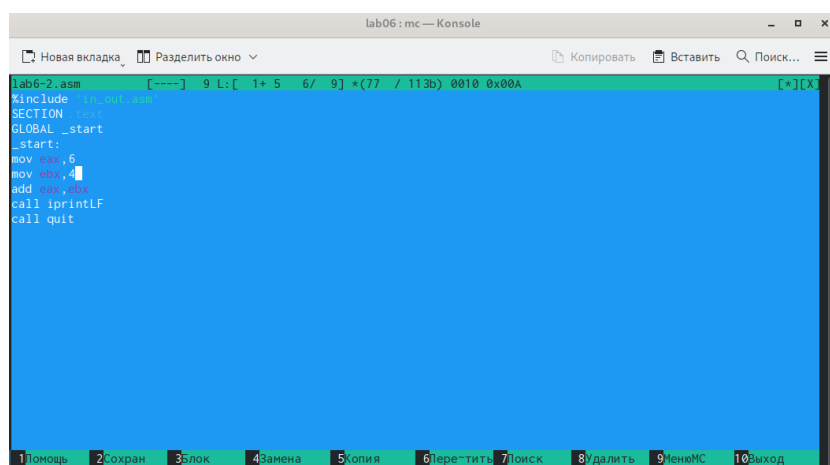
Создаем исполняемый файл и запускаем его. В результате работы программы мы получим число 106 т.к команда add складывает коды символов, но функция iprintLF позволяет вывести число, а не символ, кодом которого является это число. (рис. 4.11).



```
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 4.11: Создание исполняемого файла и его запуск

Изменяем в тексте программы символы '6' и '4' на 6 и 4. (рис. 4.12).



```
lab6-2.asm [----] 9 L: [ 1+ 5 6/ 9] *(77 / 113b) 0010 0x00A [*][X]
#include <stdio.h>
SECTION .text
GLOBAL _start
_start:
mov eax, 6
mov ebx, 4
add eax, ebx
call iprintLF
call quit
```

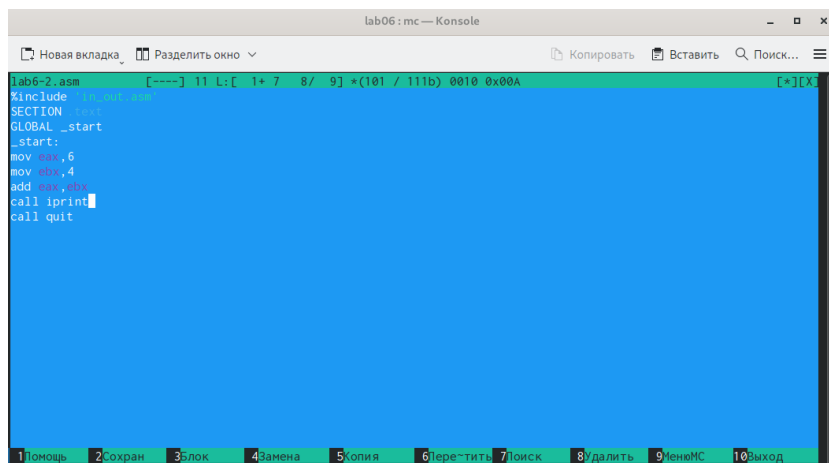
Рис. 4.12: Изменение программы

Создаем исполняемый файл и запускаем его. Теперь программа складывает не коды, а сами числа, поэтому результатом является число 10. (рис. 4.13).

```
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 4.13: Создание исполняемого файла и его запуск

Заменяем в тексте программы функцию `iprintLF` на `iprint`. (рис. 4.14).



```
lab6-2.asm [-----] 11 L: [ 1+ 7 8/ 9] *(101 / 111b) 0010 0x00A [*][X]
%include "lab6-2.asm"
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 4.14: Изменение программы

Создаем исполняемый файл и запускаем его. Вывод не изменился, но символ переноса строки не отображается. (рис. 4.15).

```
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ ./lab6-2
10aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $
```

Рис. 4.15: Создание исполняемого файла и его запуск

2. Выполнение арифметических операций в NASM.

Создаем файл `lab6-3.asm`. (рис. 4.16).

```
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ touch lab6-3.asm
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.o lab6-3.asm
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $
```

Рис. 4.16: Создание файла

Вводим в созданный файл текст программы из листинга 6.3. (рис. 4.17).

```

lab6-3.asm [----] 32 L: [ 1+ 0 1/ 29] *(32 /1365b) 0045 0x02D [*][X]
; Программа вычисления выражения
;
#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 iprintf ; из 'edi' в виде символов

```

Рис. 4.17: Ввод программы

Создаем исполняемый файл и запускаем его. (рис. 4.18).

```

aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
aykizhvatkina@dkln22 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1

```

Рис. 4.18: Создание исполняемого файла и его запуск

Изменяем текст программы для вычисления выражения. (рис. 4.19).

```

~/afsf/.dk.sci.pfu.edu.ru/home/a/y/aykizhvatkina/work/arch-pc/lab06/lab6-3.asm 1009/1366 73%
; Программа вычисления выражения
;
#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,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, 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 iprintf ; из 'edi' в виде символов

```

Рис. 4.19: Изменение программы

Создаем исполняемый файл и проверяем его работу. (рис. 4.20).

```
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
```

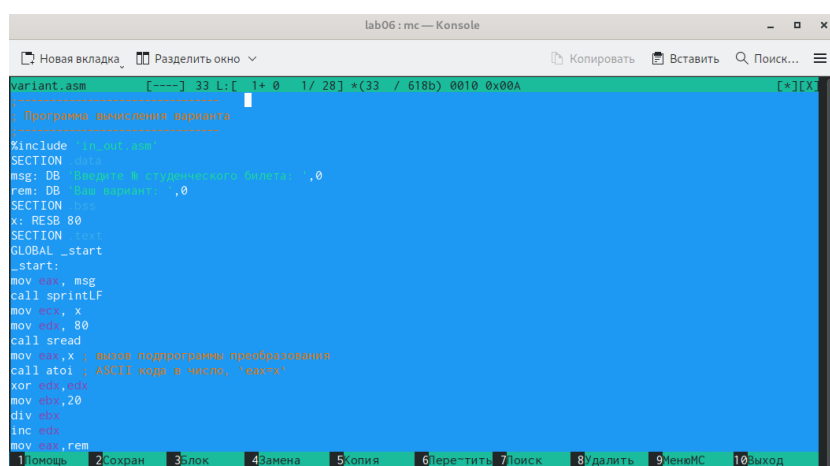
Рис. 4.20: Создание исполняемого файла и его запуск

Создаем файл variant.asm с помощью touch. (рис. 4.21).

```
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ touch variant.asm
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.o lab6-3 lab6-3.asm lab6-3.o variant.asm
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $
```

Рис. 4.21: Создание файла

Вводим в файл текст программы из листинга 6.4. (рис. 4.22).



```
variant.asm [---] 33 L: [1+ 0 1/ 20] *(33 / 618b) 0010 0x00A [*][X]
; Программа вычисления варианта
%include "io.inc.asm"
SECTION .data
msg: DB "Введите № студенческого билета: ",0
rem: DB "Ваш вариант: ",0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call read
mov ebx, x ; вывод подпрограммы преобразования
call atoi ; ASCII код в число, 'eax:hex'
xor edx,edx
mov ebx,20
div ebx
inc edx
mov ecx,rem
```

Рис. 4.22: Ввод программы

Создаем и запускаем исполняемый файл. Вводим номер своего студенческого билета проверяем выведенный ответ аналитически. (рис. 4.23).

```
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132243104
Ваш вариант: 5
```

Рис. 4.23: Создание исполняемого файла и его запуск

Ответы на вопросы:

1. За вывод сообщения «Ваш вариант» отвечает данная строка:

```
mov eax,rem  
call sprint
```

2. Инструкция `mov ecx,x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx` `mov edx,80` – запись регистр `edx` длины вводимой строки `call spread` – 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 вычисление варианта отвечают данные строки:

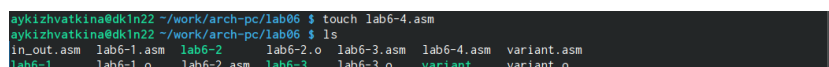
```
xor edx,edx  
mov ebx,20  
div ebx  
inc edx
```

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 результата на экран отвечают данные строки:

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

3. Выполнение самостоятельной работы.

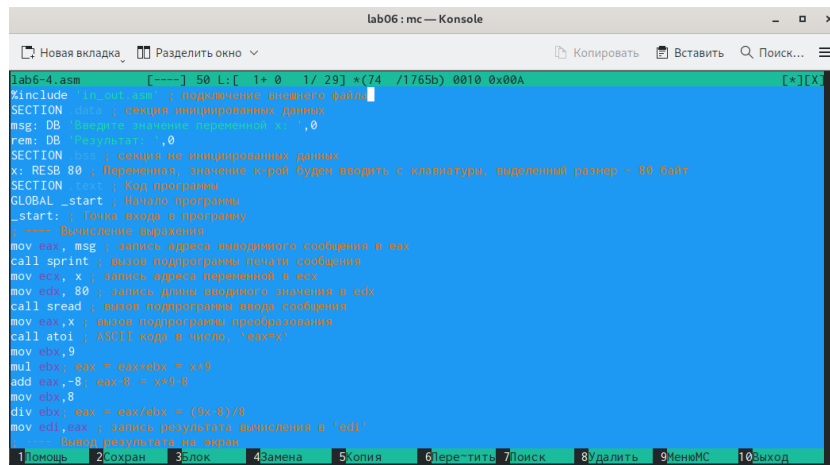
Создаем файл `lab6-4` с помощью `touch`. (рис. 4.24).



```
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ touch lab6-4.asm  
aykizhvatkina@dkin22 ~/work/arch-pc/lab06 $ ls  
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  lab6-4.asm  variant.asm  
lab6-1      lab6-1.o    lab6-2.asm  lab6-3    lab6-3.o    variant     variant.o
```

Рис. 4.24: Создание файла

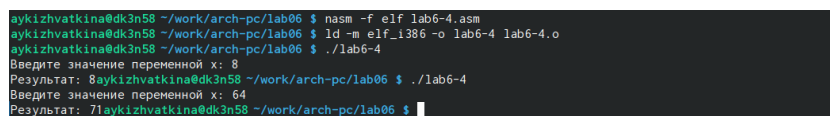
Вводим в файл текст программы для вычисления выражения под номером 18.
(рис. 4.25).



```
lab6-4.asm      [----] 50 L: [ 1+ 0 1/ 29] *(74 /1765b) 0010 0x00a      [*][X]
#include <stdio.h> ; подключение внешнего файла
SECTION .data ; секция инициализированных данных
msg: DB "Введите значение переменной x: ",0
rem: DB "Результат: ",0
SECTION .bss ; секция не инициализированных данных
x: RESB 80 ; переменная, значение которой будет вводиться с клавиатуры, выделенный размер - 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
    ; Вычисление выражения
    mov ecx, msg ; запись адреса выходного сообщения в ecx
    call printf ; вызов подпрограммы печати сообщения
    mov ecx, x ; запись адреса переменной в ecx
    mov edx, 80 ; запись длины входного значения в edx
    call scanf ; вызов подпрограммы ввода сообщения
    mov ecx, x ; вызов подпрограммы преобразования
    call atoi ; ASCII код в число, 'ecx-х'
    mov ebx, 9
    mul ebx; ebx = ecx*ebx = x*9
    add ecx, -8; ecx-8 = x*9-8
    mov ebx, 8
    div ebx; ebx = ecx/ebx = (x*9)/8
    mov edi, ecx ; запись результата вычисления в 'edi'
    ; Вывод результата на экран
```

Рис. 4.25: Ввод программы

Создаем и запускаем файл. Проверяем оба значения. (рис. 4.26).



```
aykizhvatkina@dk3n58 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
aykizhvatkina@dk3n58 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
aykizhvatkina@dk3n58 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 8
Результат: 8aykizhvatkina@dk3n58 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 64
Результат: 71aykizhvatkina@dk3n58 ~/work/arch-pc/lab06 $
```

Рис. 4.26: Создание исполняемого файла и его запуск

5 Выводы

С помощью данной лабораторной работы мы освоили арифметические инструкции языка ассемблера NASM.