

Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютера

Барбакова Алиса Саяновна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	15
4.2.1	Ответы на вопросы по программе	19
4.3	Выполнение заданий для самостоятельной работы	21
5	Выводы	25
	Список литературы	26

Список иллюстраций

4.1	Создание директории и файла	8
4.2	Создание копии файла	9
4.3	Редактирование файла	10
4.4	Запуск исполняемого файла	10
4.5	Редактирование файла	11
4.6	Запуск исполняемого файла	11
4.7	Создание файла	12
4.8	Редактирование файла	12
4.9	Запуск исполняемого файла	13
4.10	Редактирование файла	13
4.11	Запуск исполняемого файла	14
4.12	Редактирование файла	14
4.13	Запуск исполняемого файла	15
4.14	Создание файла	16
4.15	Редактирование файла	16
4.16	Запуск исполняемого файла	17
4.17	Изменение программы	17
4.18	Запуск исполняемого файла	18
4.19	Создание файла	18
4.20	Редактирование файла	19
4.21	Запуск исполняемого файла	19
4.22	Написание программы	21
4.23	Создание исполняемого файла	22
4.24	Запуск исполняемого файла	22
4.25	Запуск исполняемого файла	22

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.

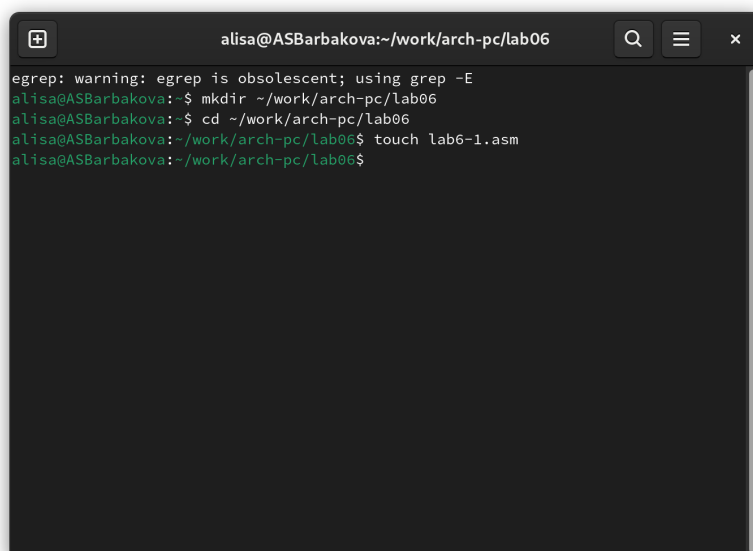
Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно 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 вы-

водить 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.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

С помощью команды `mkdir` создаю директорию, в которой буду создавать файлы с программами. Перехожу в созданный каталог с помощью утилиты `cd`. С помощью утилиты `touch` создаю файл `lab6-1.asm` (рис. 4.1).



```
alisa@ASBarbakova:~/work/arch-pc/lab06
egrep: warning: egrep is obsolescent; using grep -E
alisa@ASBarbakova:~$ mkdir ~/work/arch-pc/lab06
alisa@ASBarbakova:~$ cd ~/work/arch-pc/lab06
alisa@ASBarbakova:~/work/arch-pc/lab06$ touch lab6-1.asm
alisa@ASBarbakova:~/work/arch-pc/lab06$
```

Рис. 4.1: Создание директории и файла

Копирую в текущий каталог файл in_out.asm (рис. 4.2).

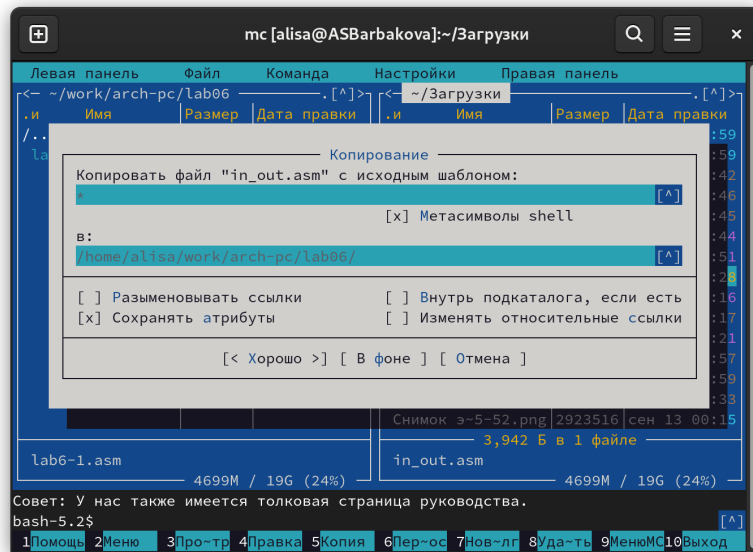


Рис. 4.2: Создание копии файла

Вставляю в созданный файл программу вывода значения регистра eax (рис. 4.3).

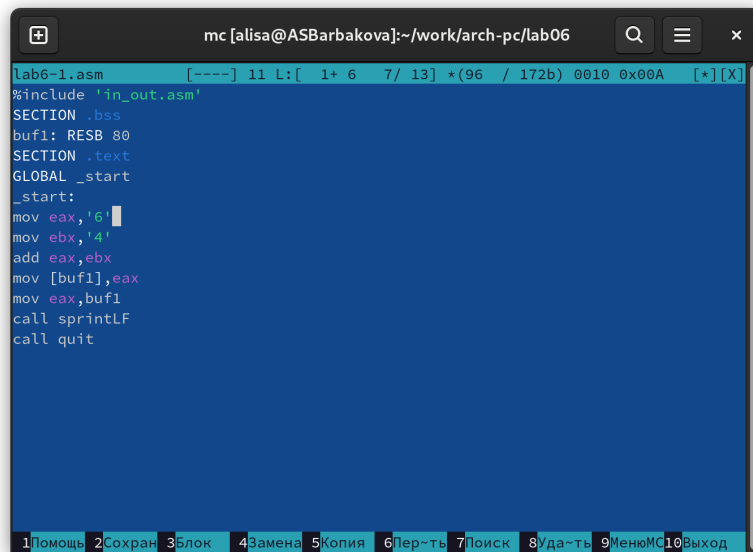


Рис. 4.3: Редактирование файла

Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j (рис. 4.4).

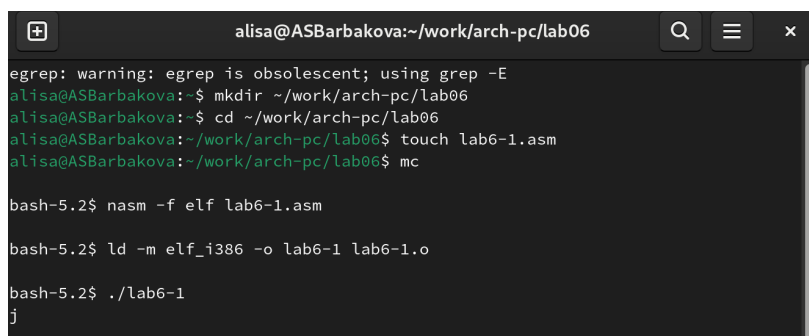


Рис. 4.4: Запуск исполняемого файла

Изменяю в тексте файла lab6-1.asm символы “6” и “4” на цифры 6 и 4 (рис. 4.5).

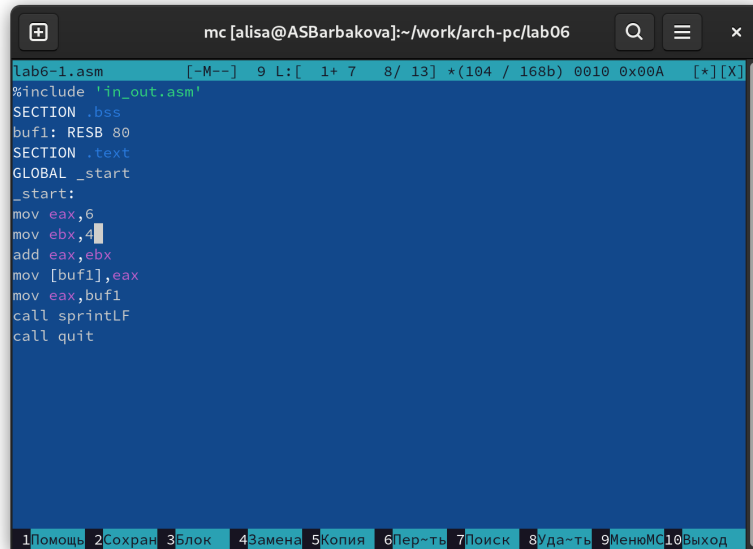


Рис. 4.5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его. Этот символ не отображается при выводе на экран, так как является символом перевода строки (рис. 4.6).

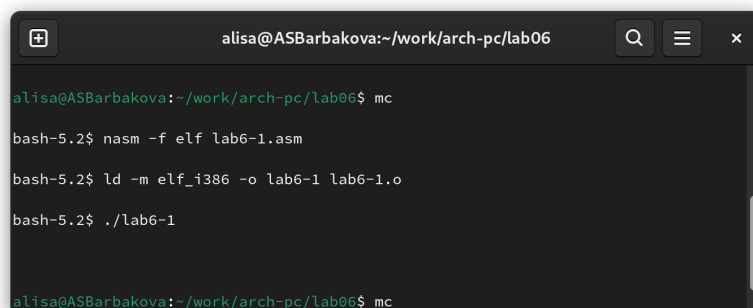


Рис. 4.6: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью touch (рис. 4.7).

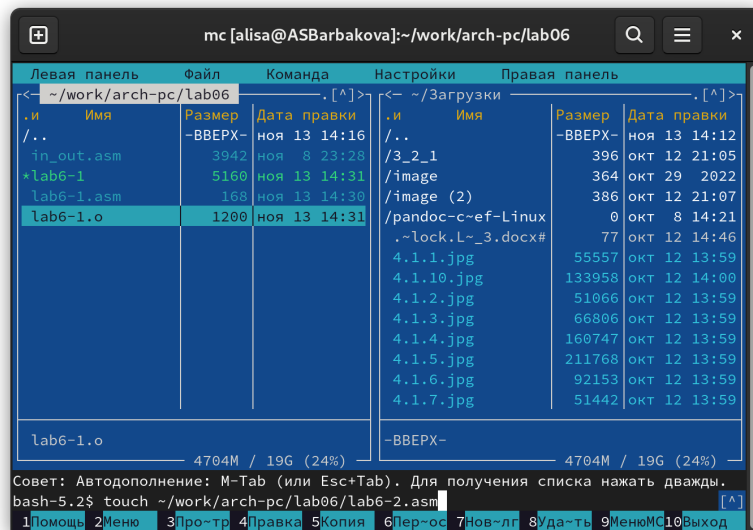


Рис. 4.7: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра `eax` (рис. 4.8).

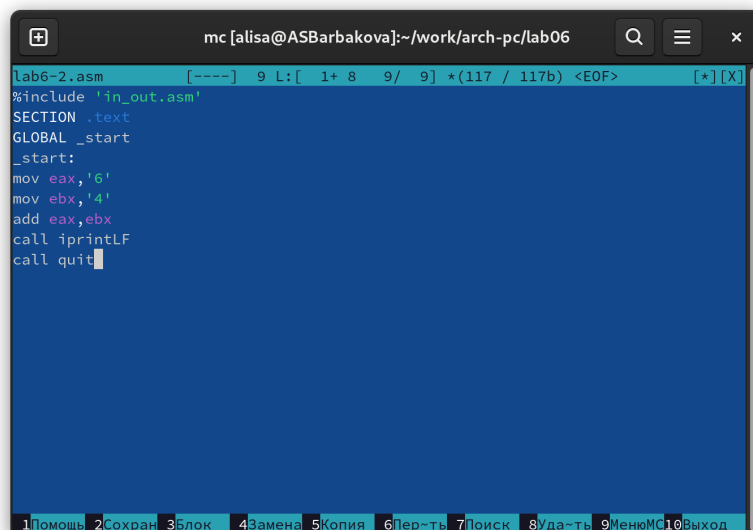
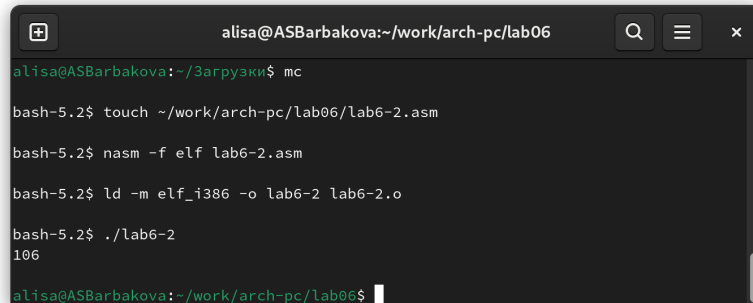


Рис. 4.8: Редактирование файла

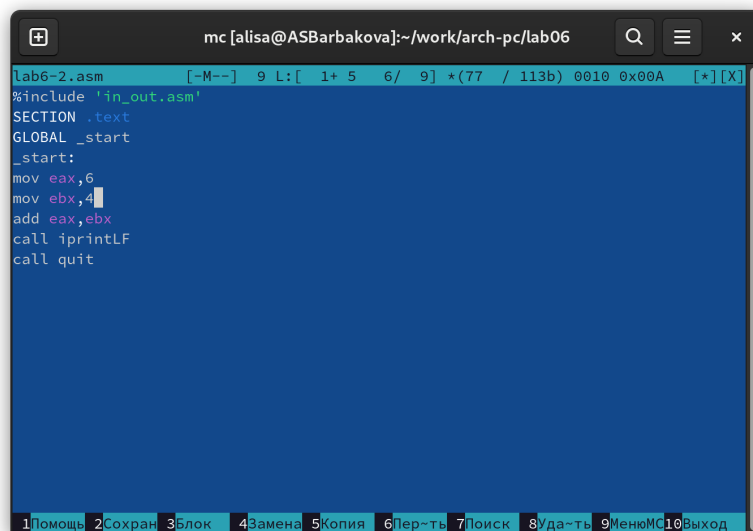
Создаю и запускаю исполняемый файл lab6-2. Выводится число 106 (рис. 4.9).



```
alisa@ASBarbakova:~/work/arch-pc/lab06
alisa@ASBarbakova:~/Заргрузки$ mc
bash-5.2$ touch ~/work/arch-pc/lab06/lab6-2.asm
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
alisa@ASBarbakova:~/work/arch-pc/lab06$
```

Рис. 4.9: Запуск исполняемого файла

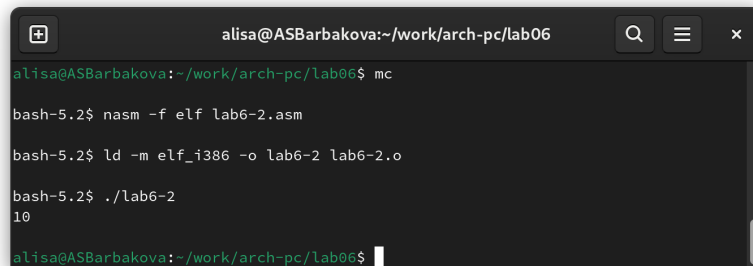
Заменяю в тексте программы в файле символы “6” и “4” на числа 6 и 4 (рис. 4.10).



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

Рис. 4.10: Редактирование файла

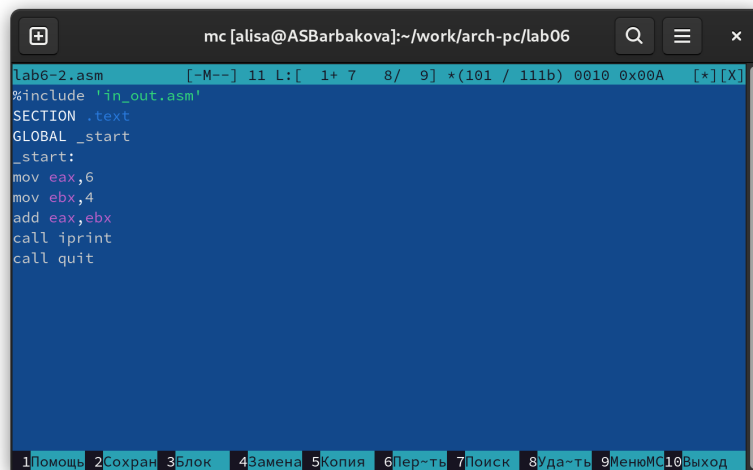
Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. 4.11).



```
alisa@ASBarbakova:~/work/arch-pc/lab06$ mc
alisa@ASBarbakova:~/work/arch-pc/lab06$ 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
alisa@ASBarbakova:~/work/arch-pc/lab06$
```

Рис. 4.11: Запуск исполняемого файла

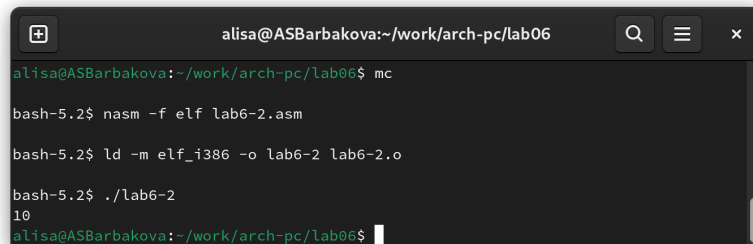
Заменяю в тексте программы файла lab6-2.asm функцию `iprintLF` на `iprint` (рис. 4.12).



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

Рис. 4.12: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.13). 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.

A terminal window with a dark background. The title bar shows 'alisa@ASBarbakova:~/work/arch-pc/lab06'. The terminal content shows the following commands and output:

```
alisa@ASBarbakova:~/work/arch-pc/lab06$ mc
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
alisa@ASBarbakova:~/work/arch-pc/lab06$
```

Рис. 4.13: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch` (рис. 4.14).

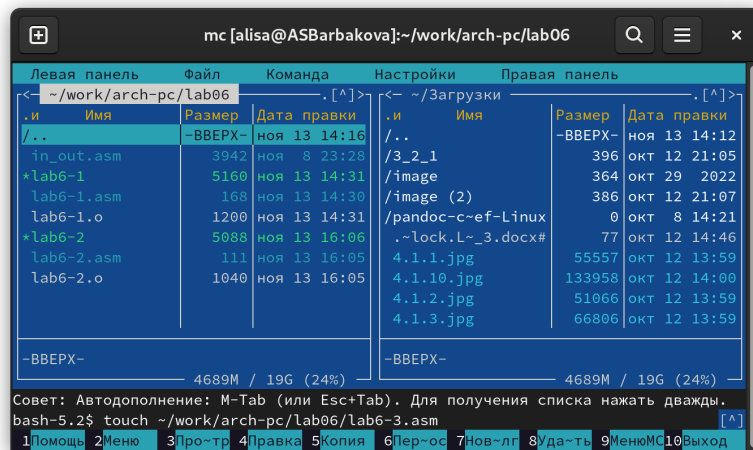


Рис. 4.14: Создание файла

Ввожу в созданный файл текст программы для вычисления $f(x) = (5 * 2 + 3)/3$ (рис. 4.15).

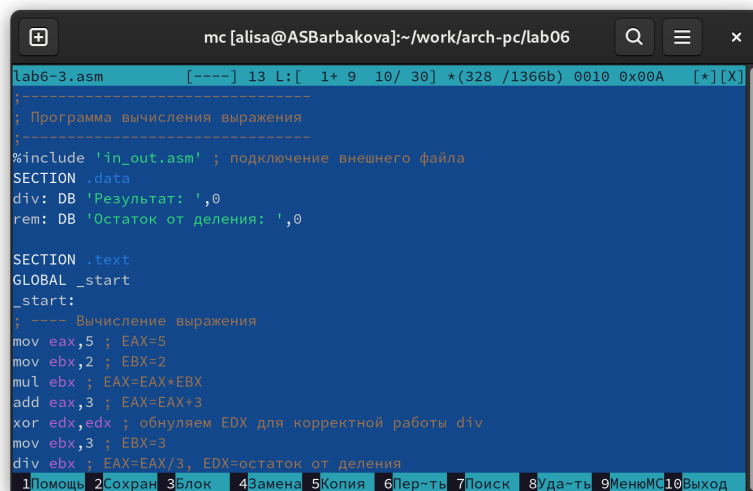
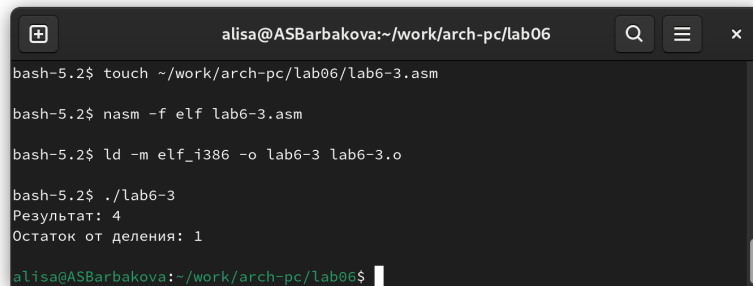


Рис. 4.15: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.16). Решение про-

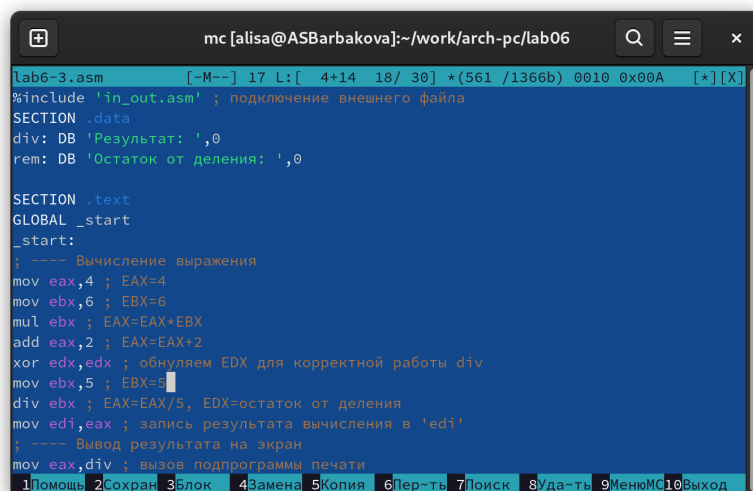
граммы совпадает с ответом.



```
alisa@ASBarbakova:~/work/arch-pc/lab06
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
alisa@ASBarbakova:~/work/arch-pc/lab06$
```

Рис. 4.16: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.17).



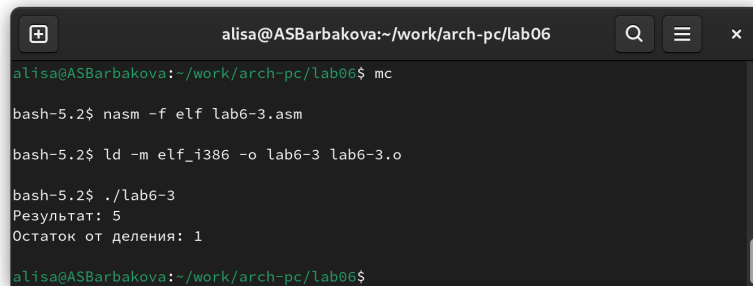
```
lab6-3.asm [-M--] 17 L: [ 4+14 18/ 30] *(561 /1366b) 0010 0x00A [*][X]
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 4.17: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 4.18). Программа

сработала верно.



```
alisa@ASBarbakova:~/work/arch-pc/lab06$ mc
alisa@ASBarbakova:~/work/arch-pc/lab06$ mc
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
alisa@ASBarbakova:~/work/arch-pc/lab06$
```

Рис. 4.18: Запуск исполняемого файла

Создаю файл variant.asm с помощью команды touch (рис. 4.19).

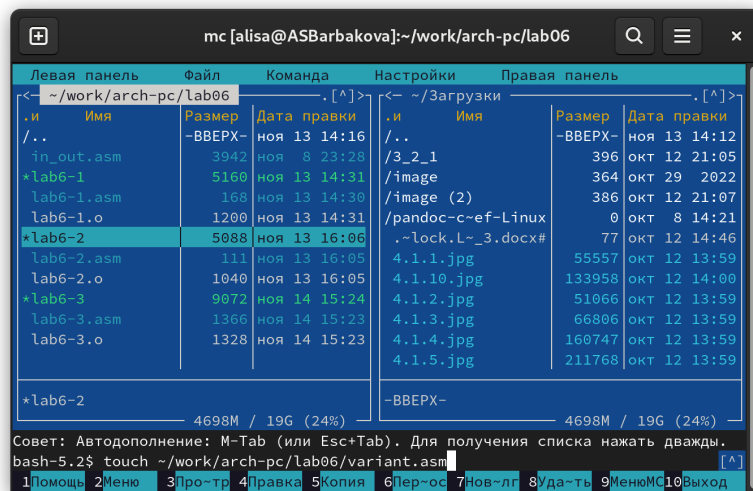
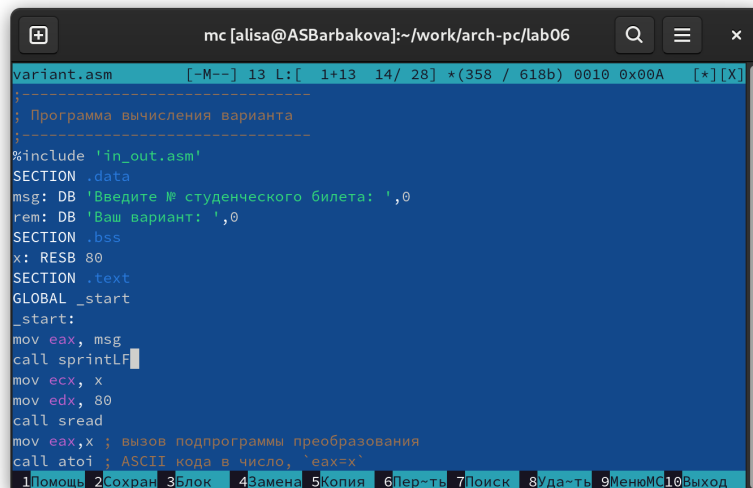


Рис. 4.19: Создание файла

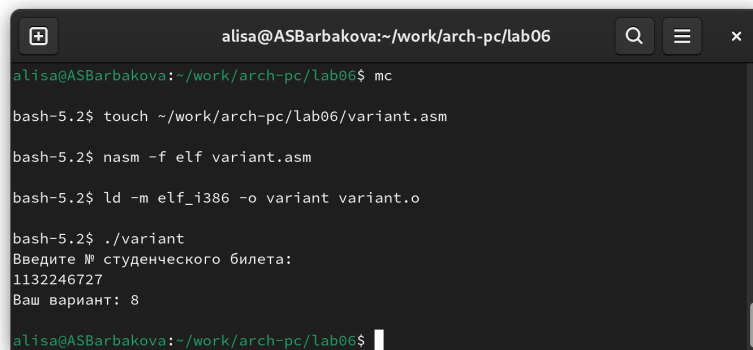
Ввожу в файл текст программы для вычисления варианта задания по номеру моего студенческого билета (рис. 4.20).



```
variant.asm [-M--] 13 L: [ 1+13 14/ 28] *(358 / 618b) 0010 0x00A [*] [X]
; Программа вычисления варианта
-----
%include 'in_out.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 sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
```

Рис. 4.20: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.21). Ввожу номер своего студ. билета с клавиатуры. Программа выводит, что мой вариант - 8.



```
alisa@ASBarbakova:~/work/arch-pc/lab06
alisa@ASBarbakova:~/work/arch-pc/lab06$ mc
bash-5.2$ touch ~/work/arch-pc/lab06/variant.asm
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:
1132246727
Ваш вариант: 8
alisa@ASBarbakova:~/work/arch-pc/lab06$
```

Рис. 4.21: Запуск исполняемого файла

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. Инструкция `call atoi` вызывает подпрограмму `atoi` из внешнего файла, которая преобразует строку ASCII, находящуюся в `x`, в целое число и записывает его в регистр `eax`.
4. За вычисления варианта отвечают строки:

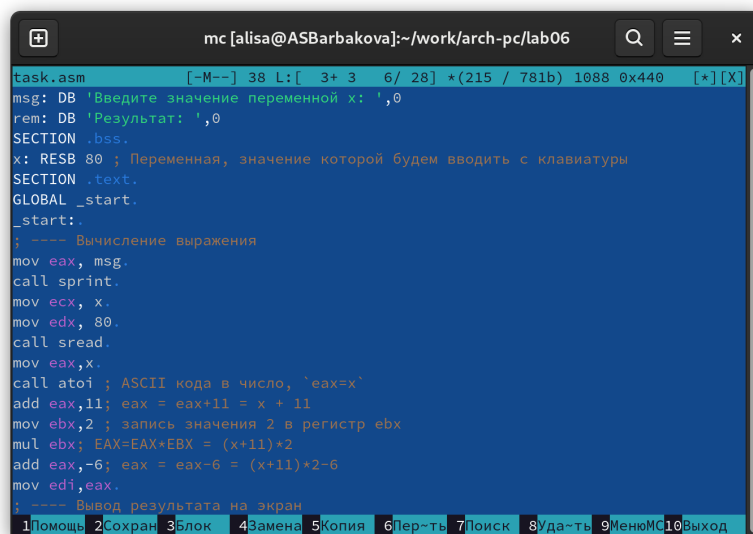
```
xor edx,edx  
mov ebx,20  
div ebx  
inc edx
```

5. Остаток от деления записывается в регистр `edx` при `div ebx`.
6. Инструкция `inc edx` увеличивает значение в регистре `edx` на 1.
7. За вывод на экран результатов вычислений отвечают строки:

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

4.3 Выполнение заданий для самостоятельной работы

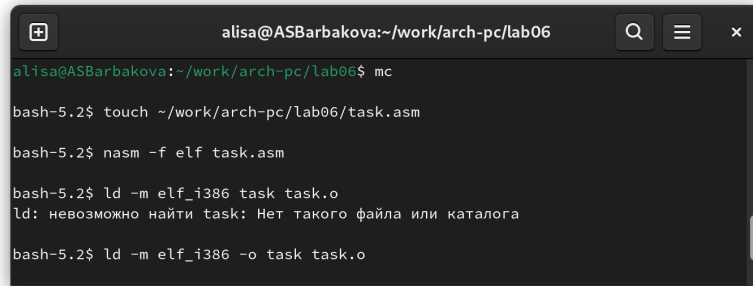
Создаю файл task.asm с помощью утилиты touch. Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления выражение под вариантом 8: $(11 + x) * 2 - 6$ (рис. 4.22).



```
task.asm      [-M--] 38 L:[ 3+ 3 6/ 28] *(215 / 781b) 1088 0x440 [*][X]
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 ; ASCII кода в число, 'eax=x'
add eax, 11; eax = eax+11 = x + 11
mov ebx, 2 ; запись значения 2 в регистр ebx
mul ebx; EAX=EAX*EBX = (x+11)*2
add eax, -6; eax = eax-6 = (x+11)*2-6
mov edi, eax.
; ---- Вывод результата на экран
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9Меню10Выход
```

Рис. 4.22: Написание программы

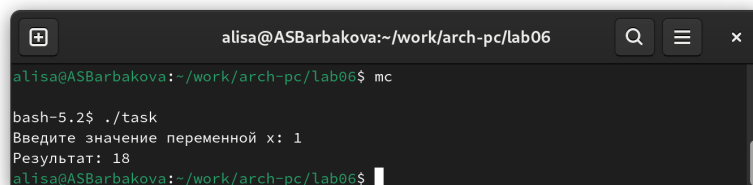
Создаю исполняемый файл (рис. 4.23).



```
alisa@ASBarbakova:~/work/arch-pc/lab06$ mc
bash-5.2$ touch ~/work/arch-pc/lab06/task.asm
bash-5.2$ nasm -f elf task.asm
bash-5.2$ ld -m elf_i386 task task.o
ld: невозможно найти task: Нет такого файла или каталога
bash-5.2$ ld -m elf_i386 -o task task.o
```

Рис. 4.23: Создание исполняемого файла

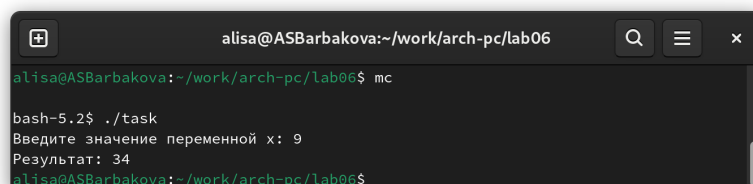
Запускаю исполняемый файл, ввожу $x1=1$ (рис. 4.24).



```
alisa@ASBarbakova:~/work/arch-pc/lab06$ mc
bash-5.2$ ./task
Введите значение переменной x: 1
Результат: 18
alisa@ASBarbakova:~/work/arch-pc/lab06$
```

Рис. 4.24: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла, ввожу $x2=9$ (рис. 4.25).
Программа сработала верно.



```
alisa@ASBarbakova:~/work/arch-pc/lab06$ mc
bash-5.2$ ./task
Введите значение переменной x: 9
Результат: 34
alisa@ASBarbakova:~/work/arch-pc/lab06$
```

Рис. 4.25: Запуск исполняемого файла

Программа для вычисления значения выражения $(11 + x) * 2 - 6$.

```
%include 'in_out.asm'

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 ; ASCII кода в число, `eax=x`
add eax, 11;  $eax = eax + 11 = x + 11$ 
mov ebx, 2 ; запись значения 2 в регистр ebx
mul ebx;  $EAX = EAX * EBX = (x + 11) * 2$ 
add eax, -6;  $eax = eax - 6 = (x + 11) * 2 - 6$ 
mov edi, eax
; ---- Вывод результата на экран
mov eax, rem
call sprint
```

```
mov eax,edi  
call iprint  
call quit
```


5 Выводы

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

Список литературы

1. Лабораторная работа №6