

Отчет по лабораторной работе № 6

Дисциплина: архитектура компьютера

Выслоух Алиса Александровна

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
4.2	Выполнение арифметических операций в NASM	14
4.3	Выполнение заданий для самостоятельной работы	17
5	Выводы	19

Список иллюстраций

4.1	Создание папки.	9
4.2	Создание файла.	9
4.3	Открытие файла и вставка.	10
4.4	Создание и запуск файла.	10
4.5	Изменение файла.	11
4.6	Создание и запуск файла.	11
4.7	Создание файла.	12
4.8	Ввод текста в файл.	12
4.9	Создание и запуск файла.	12
4.10	Изменение файла.	13
4.11	Создание и запуск файла.	13
4.12	Создание файла.	14
4.13	Ввод текста.	14
4.14	Создание и запуск файла.	15
4.15	Изменение программы.	15
4.16	Создание и запуск файла.	16
4.17	Создание файла.	16
4.18	Ввод текста.	16
4.19	Создание и запуск файла.	17
4.20	Создание файла.	17
4.21	Ввод текста программы.	17
4.22	Создание и запуск исполняемого файла.	18

Список таблиц

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, и адрес операнда указывает на место хранения данных, подлежащих обработке. Эти данные могут находиться как в регистрах, так и в ячейках памяти. Регистровая адресация подразумевает, что операнды хранятся в регистрах, а в команде используются их имена. Например: `mov ax, bx`. В случае непосредственной адресации значение операнда задается прямо в команде, как в примере: `mov ax, 2`. Адресация памяти предполагает, что операнд указывает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, с которой нужно выполнять операцию. Ввод информации с клавиатуры и вывод на экран производятся в символьном формате. Кодирование этих данных осуществляется на основе таблицы символов ASCII. ASCII (American Standard Code for Information Interchange) представляет собой стандарт, по которому каждый символ кодируется одним байтом. Важно отметить, что в NASM нет инструкции, которая могла бы непосредственно выводить числа (не в символьном представлении). Поэтому, чтобы отобразить число на экране, его нужно сначала преобразовать в соответствующие коды ASCII. Если попытаться вывести число напрямую, экран воспримет его не как число, а как последовательность ASCII-символов; каждый байт числа будет интерпретирован как отдельный ASCII-символ. Аналогичная ситуация наблюдается при вводе данных с клавиатуры: введенные символы будут интерпретированы как текст, что затруднит выполнение арифметических операций с ними. Для решения этой проблемы необходимо осуществлять преобразование ASCII-символов в числа и

наоборот.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

Создаю папку lab6 и перехожу в нее. (рис. 4.1).

```
aavihsloukh@dk5n51 ~ $ mkdir ~/work/arch-pc/lab06  
aavihsloukh@dk5n51 ~ $ cd ~/work/arch-pc/lab06
```

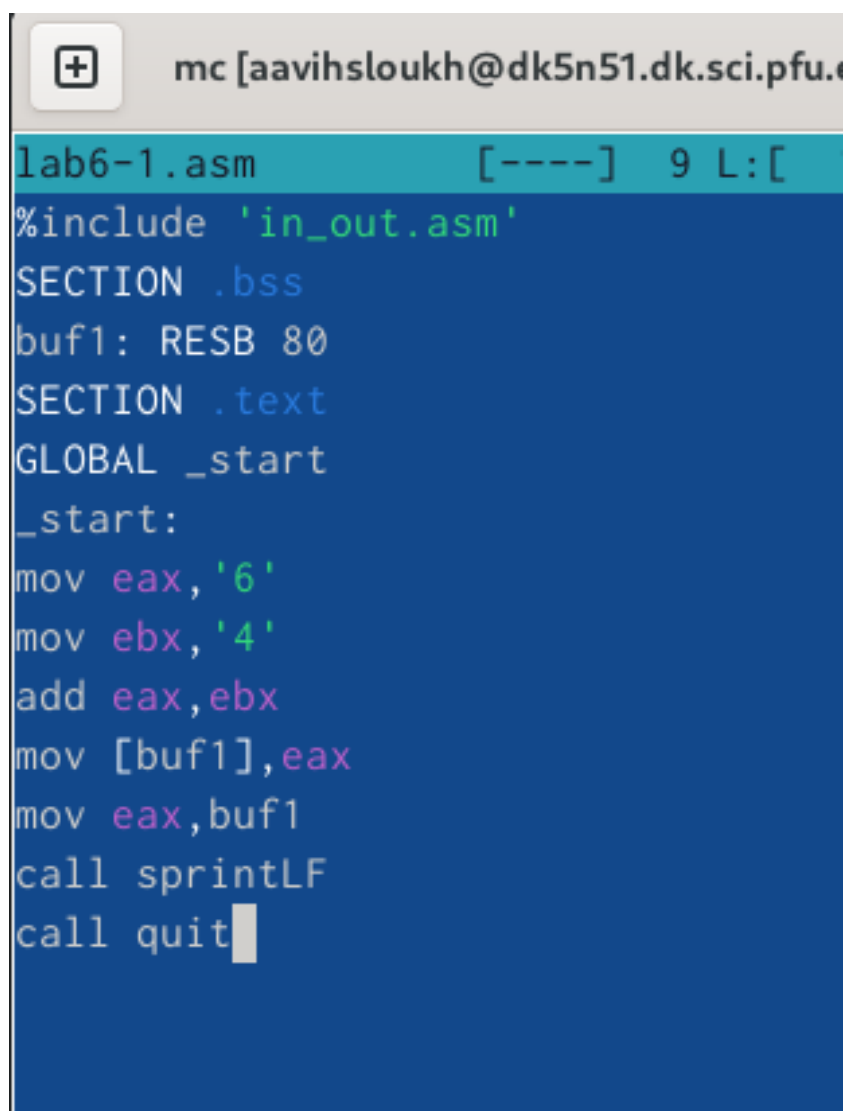
Рис. 4.1: Создание папки.

С помощью утилиты touch создаю файл lab6-1.asm (рис. 4.2).

```
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

Рис. 4.2: Создание файла.

Открываю созданный файл lab7-1.asm, вставляю в него программу вывода значения регистра eax (рис. 4.3).



```
lab6-1.asm [----] 9 L:[
#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 sprintf
call quit
```

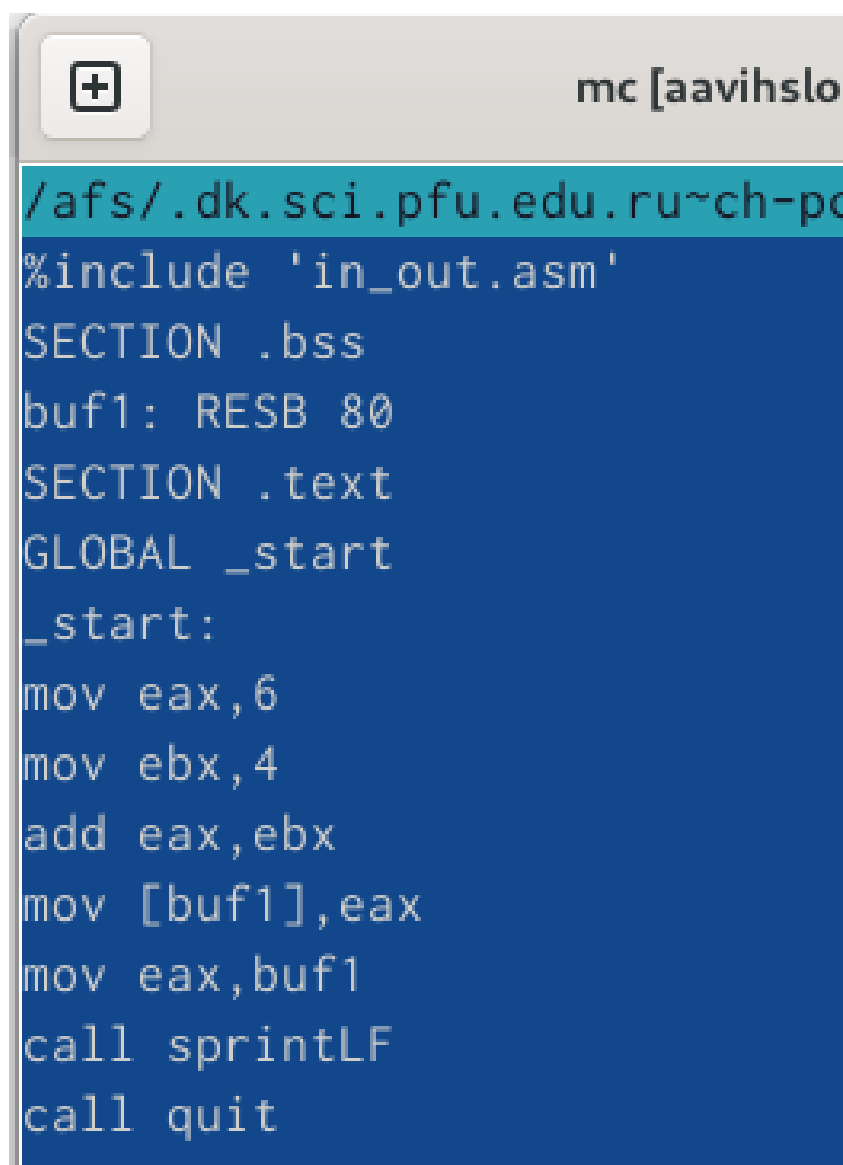
Рис. 4.3: Открытие файла и вставка.

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

```
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ./lab6-1
j
-
```

Рис. 4.4: Создание и запуск файла.

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.5).

A screenshot of a terminal window with a light gray title bar. On the left is a square button with a plus sign. On the right is the text 'mc [aavihslo'. The terminal content is on a dark blue background with white text. The first line is a path: '/afs/.dk.sci.pfu.edu.ru~ch-pc'. The following lines are assembly code: '%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 sprintf', and 'call quit'.

```
mc [aavihslo
/afs/.dk.sci.pfu.edu.ru~ch-pc
%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 sprintf
call quit
```

Рис. 4.5: Изменение файла.

Создаю новый исполняемый файл программы и запускаю его (рис. 4.6). Теперь вывелся символ перевода строки.

```
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ./lab6-1
```

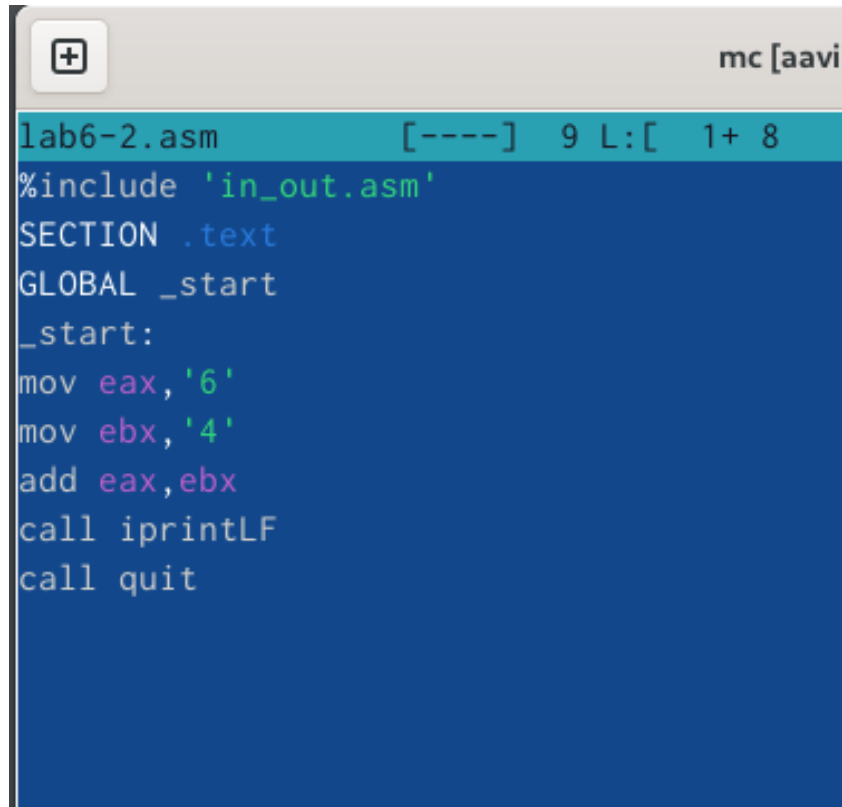
Рис. 4.6: Создание и запуск файла.

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

```
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
```

Рис. 4.7: Создание файла.

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



```
lab6-2.asm [----] 9 L:[ 1+ 8
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

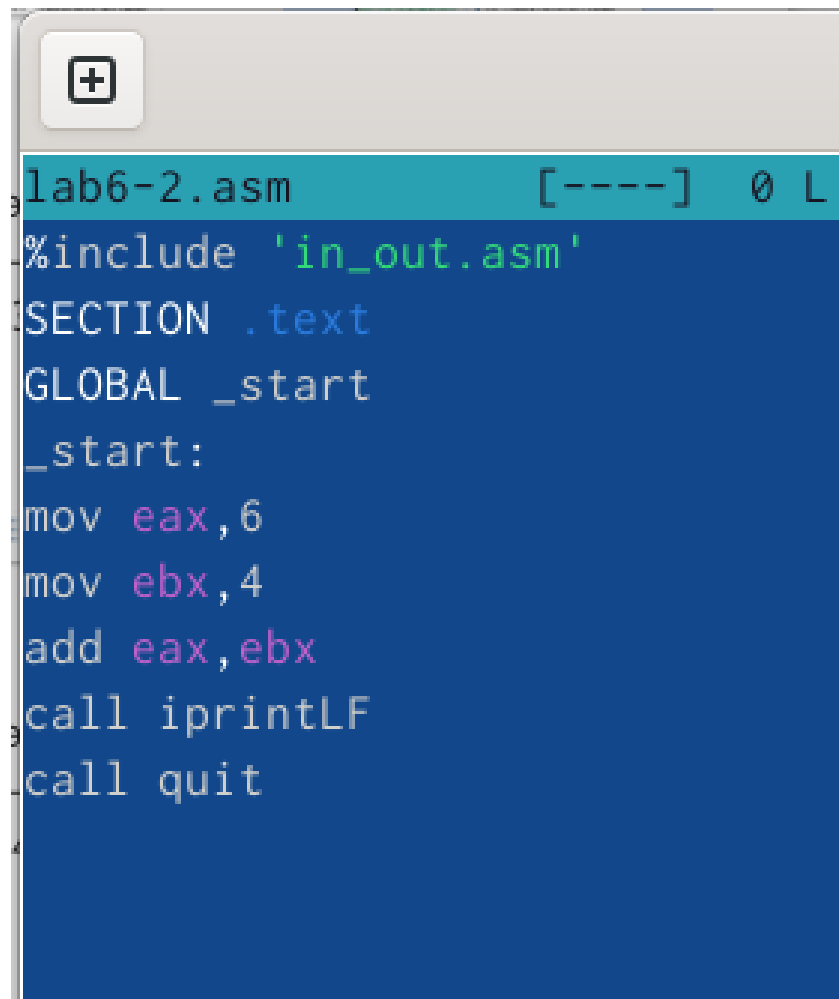
Рис. 4.8: Ввод текста в файл.

Создаю и запускаю исполняемый файл lab7-2 (рис. 4.9). Теперь вывод число 106.

```
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 4.9: Создание и запуск файла.

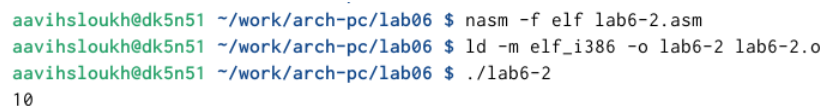
Заменяю в тексте программы в файле lab7-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.10).



```
lab6-2.asm [-----] 0 L:
%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: Изменение файла.

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



```
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 4.11: Создание и запуск файла.

Заменяю в тексте программы функцию iprintLF на iprint. 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`.

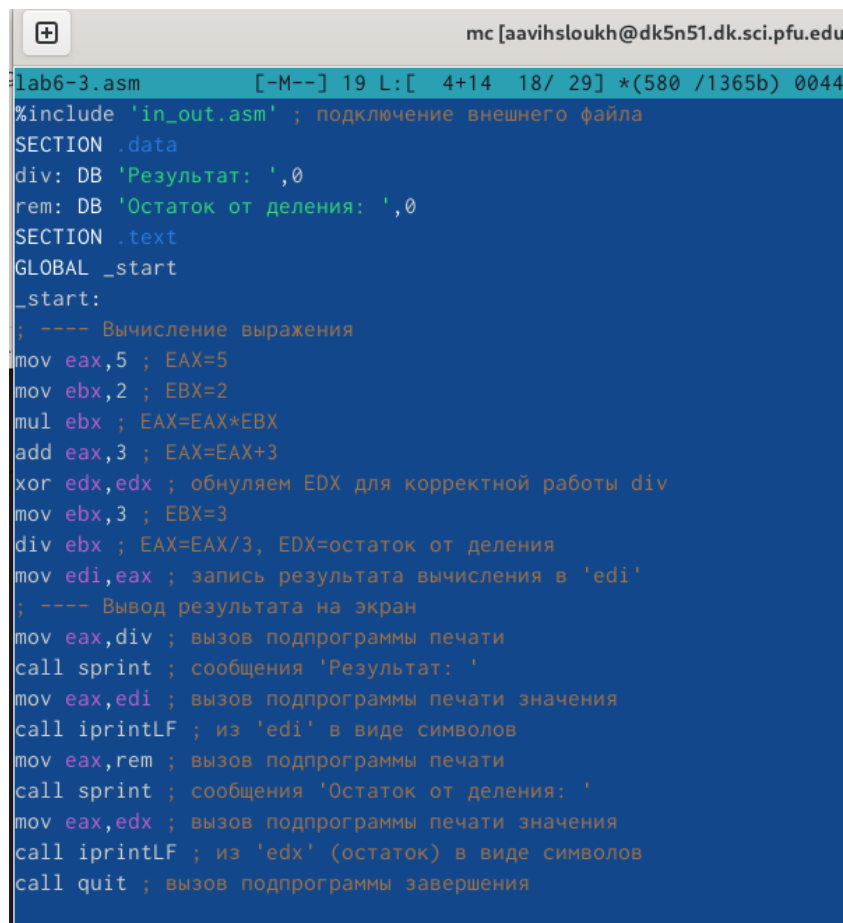
4.2 Выполнение арифметических операций в NASM

Создаю файл `lab7-3.asm` с помощью утилиты `touch` (рис. 4.12).

```
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm
```

Рис. 4.12: Создание файла.

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.13).



```
lab6-3.asm [-M--] 19 L:[ 4+14 18/ 29] *(580 /1365b) 0044
#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
call quit ; вызов подпрограммы завершения
```

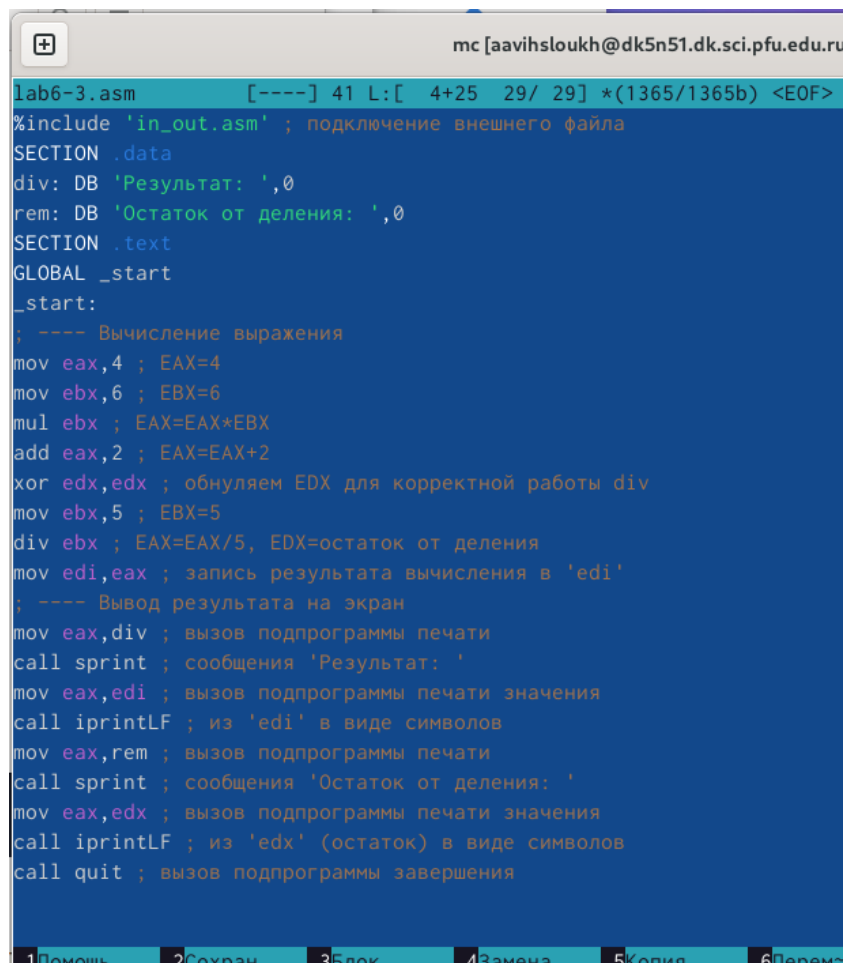
Рис. 4.13: Ввод текста.

Создаю исполняемый файл и запускаю его (рис. 4.14).

```
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.14: Создание и запуск файла.

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.15).



```
lab6-3.asm [----] 41 L: [ 4+25 29/ 29] *(1365/1365b) <EOF>
#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 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
call quit ; вызов подпрограммы завершения
```

Рис. 4.15: Изменение программы.

Создаю и запускаю новый исполняемый файл (рис. 4.16). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```

aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рис. 4.16: Создание и запуск файла.

Создаю файл variant.asm с помощью утилиты touch (рис. 4.17).

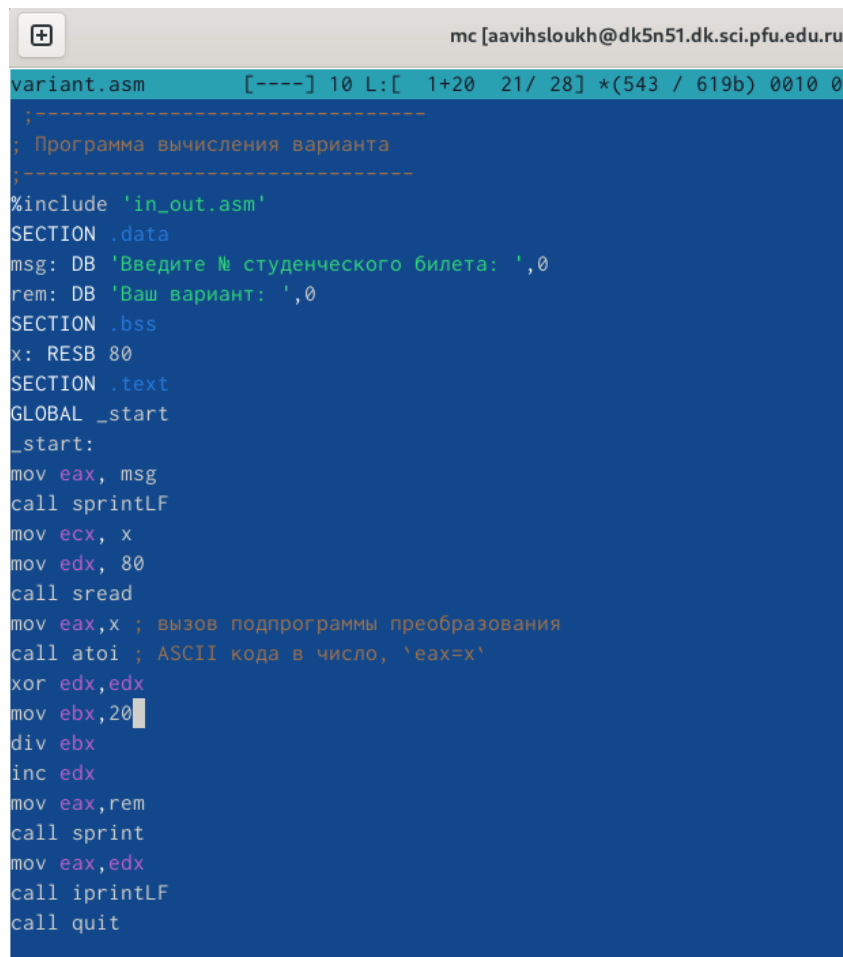
```

aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm

```

Рис. 4.17: Создание файла.

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.18).



```

mc [aavihsloukh@dk5n51.dk.sci.pfu.edu.ru]
variant.asm  [----] 10 L:[ 1+20  21/ 28] *(543 / 619b) 0010 0
;-----
; 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 sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprint
mov eax, edx
call iprintLF
call quit

```

Рис. 4.18: Ввод текста.

Создаю и запускаю исполняемый файл (рис. 4.19). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 19.

```
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant.asm variant.o
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ ./variant.asm
Введите № студенческого билета:
1132246798
Ваш вариант: 19
```

Рис. 4.19: Создание и запуск файла.

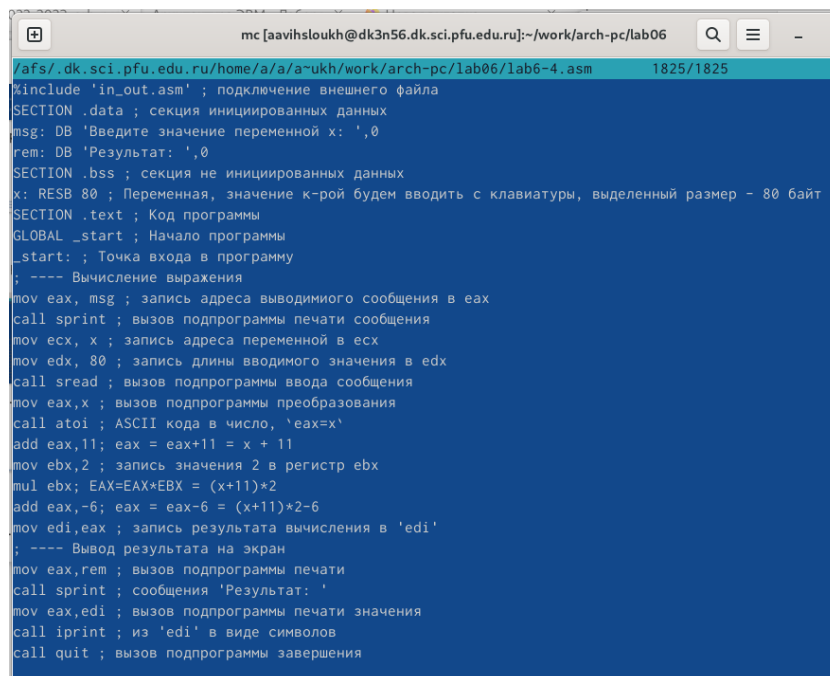
4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab7-4.asm с помощью утилиты touch (рис. 4.20).

```
aavihsloukh@dk5n51 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

Рис. 4.20: Создание файла.

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(11 + x) * 2 - 6$ (рис. 4.21). Это выражение было под вариантом 8.



```
mc [aavihsloukh@dk3n56.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab06 1825/1825
/afs/.dk.sci.pfu.edu.ru/home/a/a-a-ukh/work/arch-pc/lab06/lab6-4.asm
%include 'in_out.asm' ; 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 к-рой будем вводить с клавиатуры, выделенный размер - 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; ---- Вычисление выражения
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'
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 ; 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 iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.21: Ввод текста программы.

Создаю и запускаю исполняемый файл (рис. 4.22). При вводе значения 2, вывод 20.

```
aavihsloukh@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
aavihsloukh@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
aavihsloukh@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 2
Результат: 20aavihsloukh@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 4.22: Создание и запуск исполняемого файла.

5 Выводы

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