

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Рахматова Жылдыз Талантбековна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Символьные и численные данные в NASM	7
3.2	Выполнение арифметических операций в NASM	12
3.2.1	Ответы на вопросы по программе	14
3.3	Выполнение заданий для самостоятельной работы	15
4	Выводы	19

Список иллюстраций

3.1	Создание директории	7
3.2	Создание файла	7
3.3	Создание копии файла	7
3.4	Редактирование файла	8
3.5	Запуск исполняемого файла	8
3.6	Редактирование файла	9
3.7	Запуск исполняемого файла	9
3.8	Создание файла	9
3.9	Редактирование файла	10
3.10	Запуск исполняемого файла	10
3.11	Редактирование файла	10
3.12	Запуск исполняемого файла	11
3.13	Редактирование файла	11
3.14	Запуск исполняемого файла	11
3.15	Создание файла	12
3.16	Редактирование файла	12
3.17	Запуск исполняемого файла	13
3.18	Изменение программы	13
3.19	Запуск исполняемого файла	13
3.20	Создание файла	14
3.21	Редактирование файла	14
3.22	Запуск исполняемого файла	14
3.23	Создание файла	15
3.24	Написание программы	16
3.25	Запуск исполняемого файла	16
3.26	Запуск исполняемого файла	16

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 Выполнение лабораторной работы

3.1 Символьные и численные данные в NASM

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. 3.1). Перехожу в созданный каталог с помощью утилиты `cd`.

```
ztrakhmatova@dk8n72 ~ $ mkdir ~/work/arch-pc/lab06
ztrakhmatova@dk8n72 ~ $ cd ~/work/arch-pc/lab06
```

Рис. 3.1: Создание директории

С помощью утилиты `touch` создаю файл `lab6-1.asm` (рис. 3.2).

```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ touch lab6-1.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ls
lab6-1.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ █
```

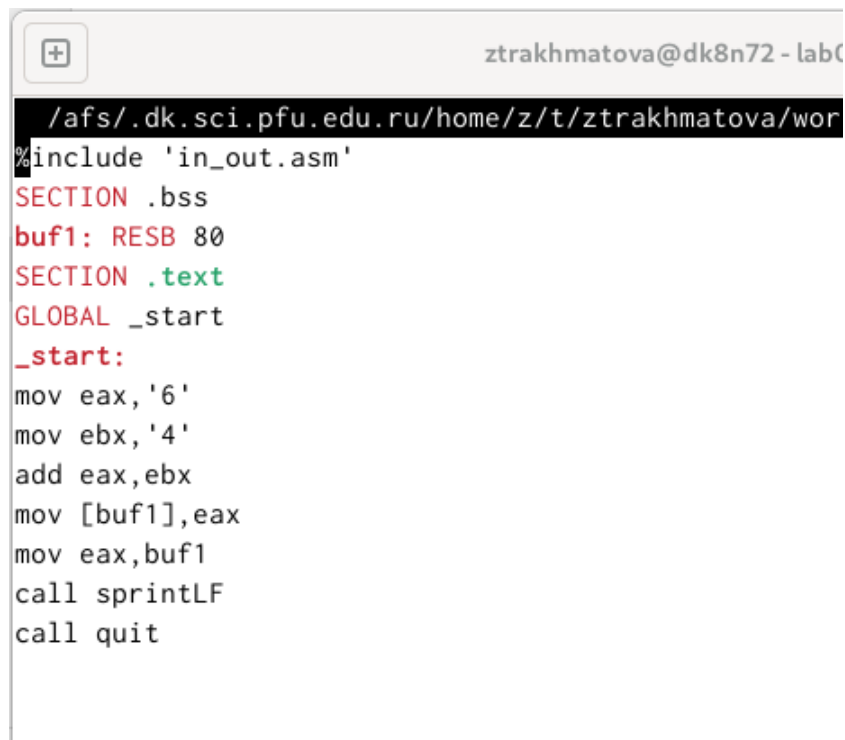
Рис. 3.2: Создание файла

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах (рис. 3.3).

```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ cp ~/Зарпязки/in_out.asm in_out.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ █
```

Рис. 3.3: Создание копии файла

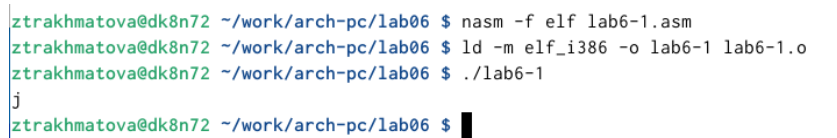
Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 3.4).

A terminal window with a title bar showing a plus icon and the username 'ztrakhmatova@dk8n72 - lab06'. The terminal displays assembly code being edited. The code includes a directive to include 'in_out.asm', defines a .bss section with a buffer 'buf1' of 80 bytes, and a .text section with a global '_start' symbol. The '_start' symbol contains assembly instructions: 'mov eax, '6'', 'mov ebx, '4'', 'add eax, ebx', 'mov [buf1], eax', 'mov eax, buf1', 'call sprintLF', and 'call quit'.

```
ztrakhmatova@dk8n72 - lab06
/afs/.dk.sci.pfu.edu.ru/home/z/t/ztrakhmatova/work
%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. 3.4: Редактирование файла

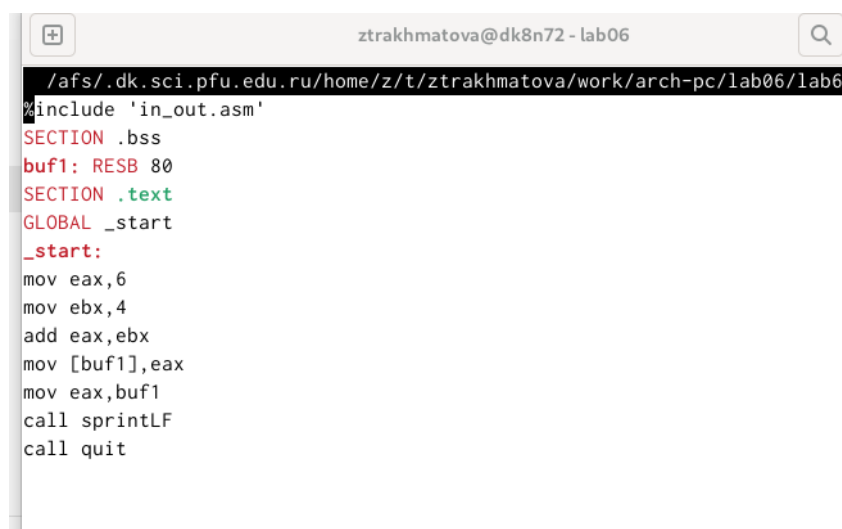
Создаю исполняемый файл программы и запускаю его (рис. 3.5). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

A terminal window showing the compilation and execution of an assembly program. The user runs 'nasm -f elf lab6-1.asm', then 'ld -m elf_i386 -o lab6-1 lab6-1.o', and finally './lab6-1'. The output of the program is the character 'j'.

```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ./lab6-1
j
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $
```

Рис. 3.5: Запуск исполняемого файла

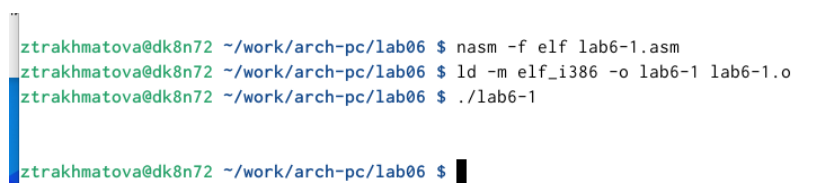
Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 3.6).



```
ztrakhmatova@dk8n72 - lab06
/afs/.dk.sci.pfu.edu.ru/home/z/t/ztrakhmatova/work/arch-pc/lab06/lab6
#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. 3.6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 3.7). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

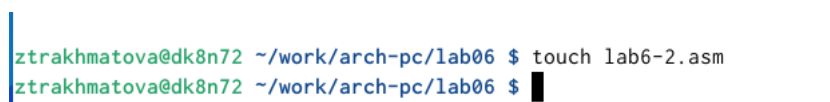


```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ./lab6-1

ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $
```

Рис. 3.7: Запуск исполняемого файла

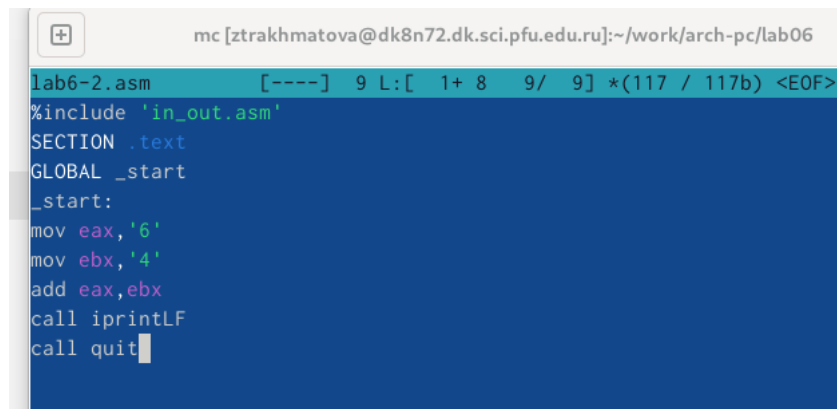
Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 3.8).



```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ touch lab6-2.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $
```

Рис. 3.8: Создание файла

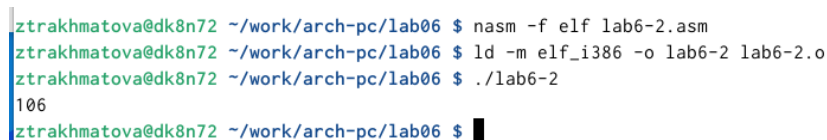
Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 3.9).



```
lab6-2.asm [----] 9 L: [ 1+ 8 9/ 9] *(117 / 117b) <EOF>
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

Рис. 3.9: Редактирование файла

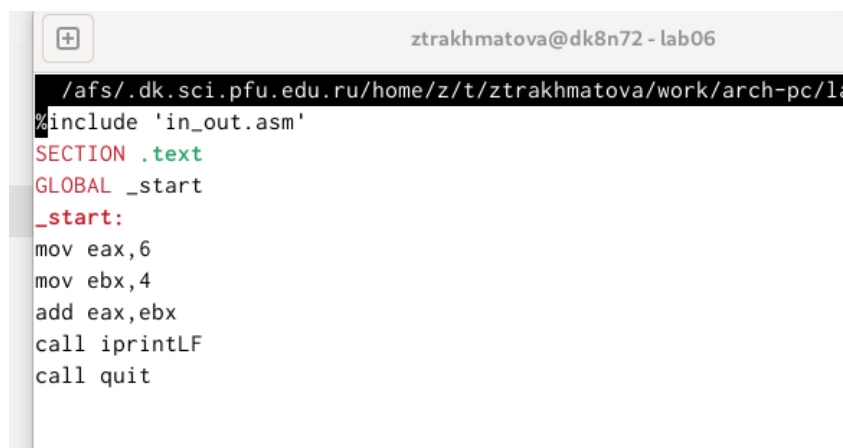
Создаю и запускаю исполняемый файл lab6-2 (рис. 3.10). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.



```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ./lab6-2
106
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $
```

Рис. 3.10: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 3.11).



```
/afs/.dk.sci.pfu.edu.ru/home/z/t/ztrakhmatova/work/arch-pc/lab06
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, 6
mov ebx, 4
add eax, ebx
call iprintLF
call quit
```

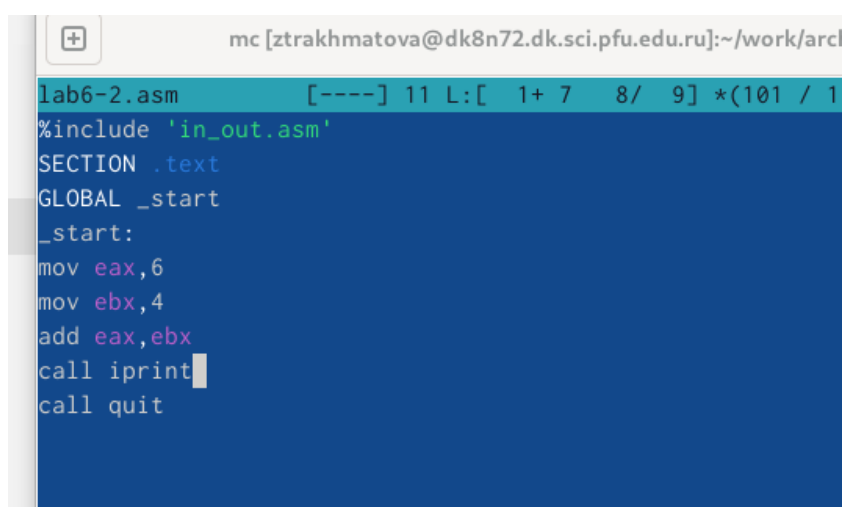
Рис. 3.11: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 3.12).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ./lab6-2
10
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $
```

Рис. 3.12: Запуск исполняемого файла

Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 3.13).



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

Рис. 3.13: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 3.14). 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`.

```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ./lab6-2
10ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $
```

Рис. 3.14: Запуск исполняемого файла

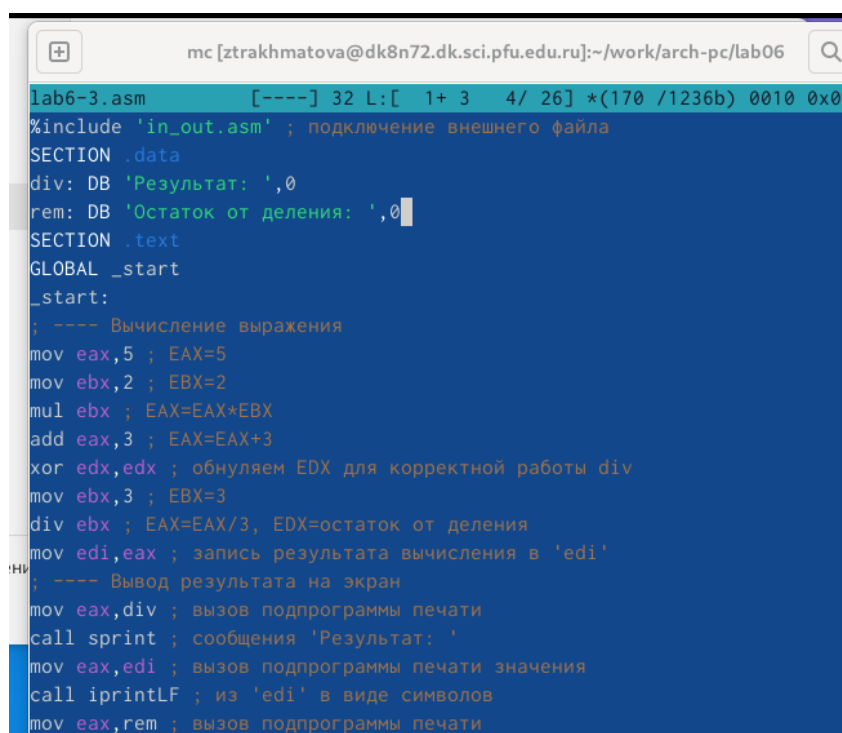
3.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. 3.15).

```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ touch lab6-3.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o
lab6-1      lab6-1.o    lab6-2.asm  lab6-3.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $
```

Рис. 3.15: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 3.16).



```
lab6-3.asm [----] 32 L: [ 1+ 3 4/ 26] *(170 /1236b) 0010 0x0
%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
```

Рис. 3.16: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 3.17).

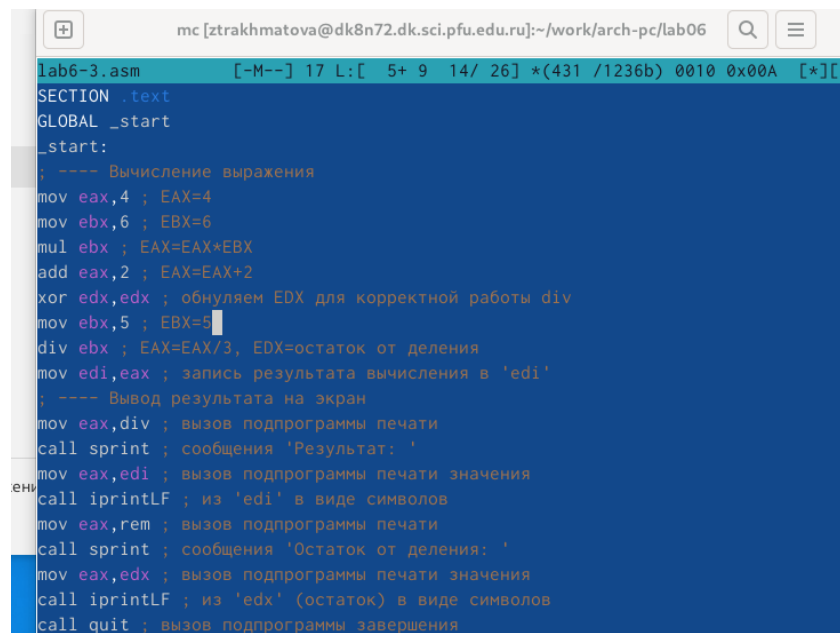
```

ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-3.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ █

```

Рис. 3.17: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 3.18).



```

lab6-3.asm      [-M--] 17 L: [ 5+ 9 14/ 26] *(431 /1236b) 0010 0x00A [*]
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/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,edx ; вызов подпрограммы печати значения
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. 3.18: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 3.19). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```

ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ █

```

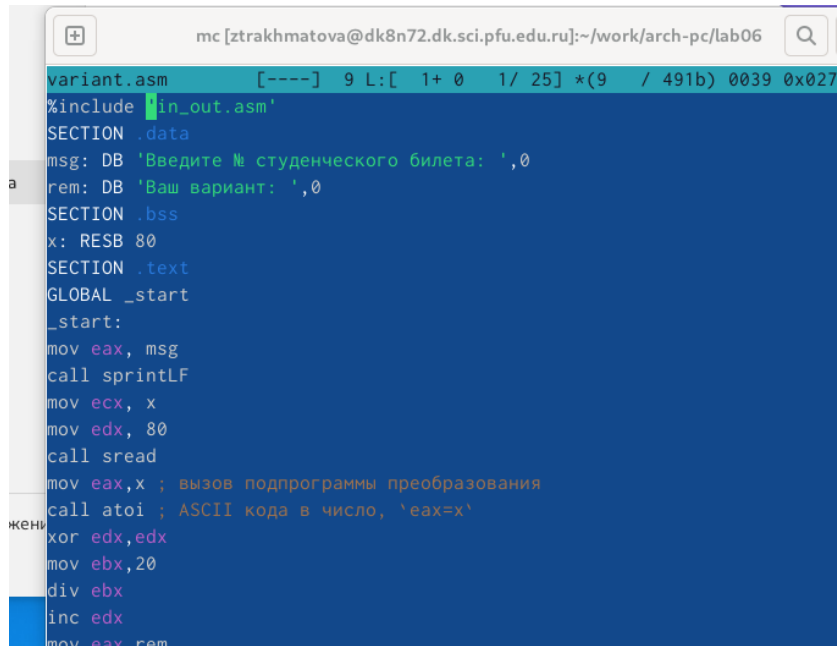
Рис. 3.19: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 3.20).

```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ touch variant.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ mv variant.asm
```

Рис. 3.20: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 3.21).



```
variant.asm [-----] 9 L: [ 1+ 0 1/ 25] *(9 / 491b) 0039 0x027
#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'
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
```

Рис. 3.21: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 3.22). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 8.

Запуск исполняемого файла

Рис. 3.22: Запуск исполняемого файла

3.2.1 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. `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
```

3.3 Выполнение заданий для самостоятельной работы

Создаю файл `lab6-4.asm` с помощью утилиты `touch` (рис. 3.23).

```
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ touch lab6-4.asm
```

Рис. 3.23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(11 + x) * 2 - 6$ (рис. 3.24). Это выражение было под вариантом 8.

```

lab6-4.asm      [-M--] 41 L: [ 7+21  28/ 28] *(1825/1825b) <EOF>  [*]
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

```

Рис. 3.24: Написание программы

Создаю и запускаю исполняемый файл (рис. 3.25). При вводе значения 3, вывод - 22.

```

ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 3
Результат: 22ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $

```

Рис. 3.25: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 3.26). Программа отработала верно.

```

Результат: 22ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 1
Результат: 18ztrakhmatova@dk8n72 ~/work/arch-pc/lab06 $

```

Рис. 3.26: Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения $(11 + x) * 2$
– 6.

```
%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
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*

4 Выводы

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