

Отчет по лабораторной работе №5

Архитектура компьютера

Николенко Анна Николаевна

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
5	Ответы на вопросы	19
6	Выполнение заданий для самостоятельной работы	20
7	Листинги программ	23
8	Выводы	25
	Список литературы	26

Список иллюстраций

4.1	Создание каталога, переход в этот каталог и создание файла . . .	9
4.2	Копирование файла	9
4.3	Ввод программы файл (вывода значения регистра eax)	10
4.4	Создание исполняемого файла и его запуск	10
4.5	Редактирование файла	11
4.6	Редактирование файла	11
4.7	Создание файла	11
4.8	Редактирование файла	12
4.9	Запуск исполняемого файла	12
4.10	Редактирование файла	13
4.11	Запуск исполняемого файла	13
4.12	Запуск исполняемого файла	13
4.13	Редактирование файла	14
4.14	Запуск исполняемого файла	14
4.15	Создание файла	14
4.16	Редактирование файла	15
4.17	Запуск исполняемого файла	15
4.18	Редактирование файла	16
4.19	Запуск исполняемого файла	16
4.20	Создание файла	16
4.21	Редактирование файла	17
4.22	Запуск исполняемого файла	18
6.1	Создание файла	20
6.2	Написание программы	21
6.3	Запуск исполняемого файла	21
6.4	Запуск исполняемого файла	22

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. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти.

Основные способы адресации: * 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.

Арифметические операции в NASM: * Целочисленное сложение - `add`. * Целочисленное вычитание - `sub`. * Команды инкремент(`inc`)- прибавление единицы к операнду и декремент(`dec`)- вычитание единицы. Они выгодны тем, что они занимают меньше места, чем соответствующие команды сложения и вычитания. * Команда изменения знака операнда - `neg`. * Команды умножения - `mul` (для беззнакового умножения) и `imul` (для знакового умножения). * Команды деления `div` и `idiv`.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. 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

- `iprint` – вывод на экран чисел в формате ASCII.
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

Создаю каталог для программ лабораторной работы No 6, перехожу в него и создаю файл lab6-1.asm (рис. [4.1]).

```
annikolenko@dk8n57 ~ $ mkdir ~/work/arch-pc/lab06
annikolenko@dk8n57 ~ $ cd ~/work/arch-pc/lab06
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

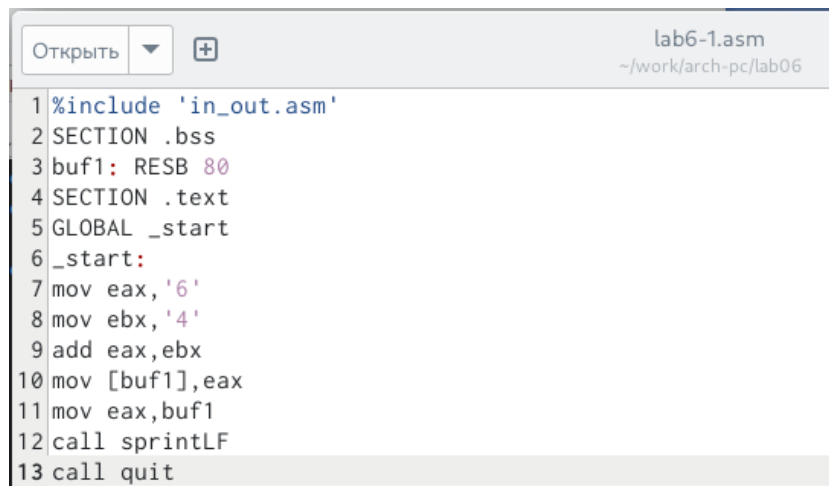
Рис. 4.1: Создание каталога, переход в этот каталог и создание файла

Копирую в текущий каталог файл in_out.asm с помощью команды cp, т.к. он будет использоваться в других программах (рис. [4.2]).

```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ cp ~/Загрузки/in_out.asm in_out.asm
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm
```

Рис. 4.2: Копирование файла

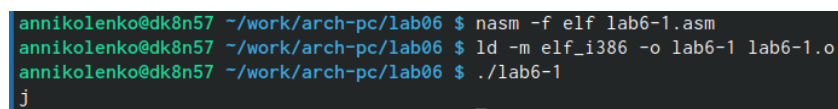
Открываю файл lab6-1.asm и ввожу в него программу вывода значения регистра eax (рис. [4.3]).



```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintLF
13 call quit
```

Рис. 4.3: Ввод программы файл (вывода значения регистра eax)

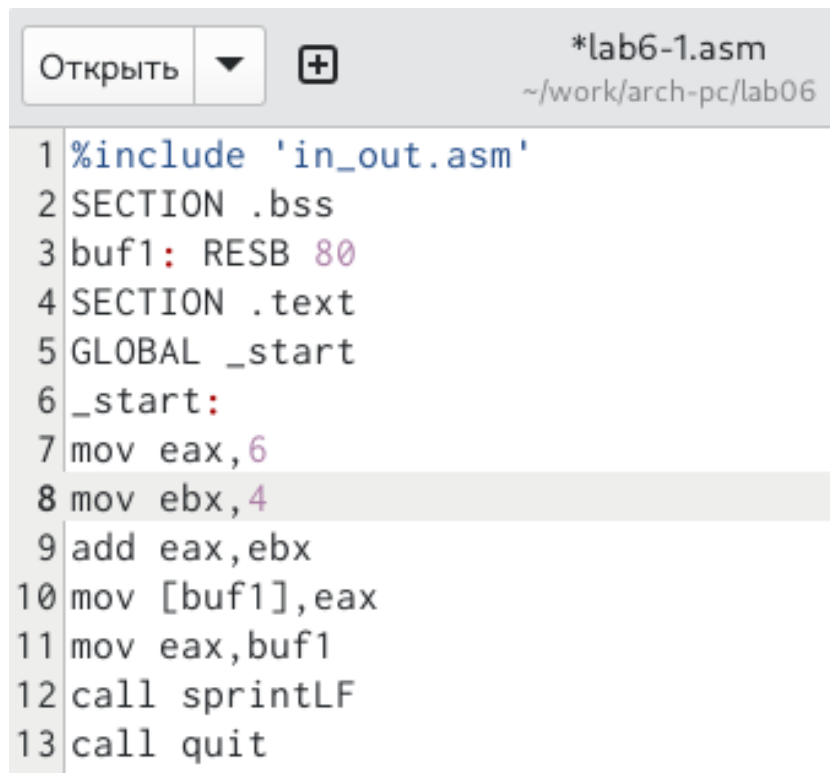
Создайте исполняемый файл и запустите его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6 (рис. [4.4]).



```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ./lab6-1
j
```

Рис. 4.4: Создание исполняемого файла и его запуск

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

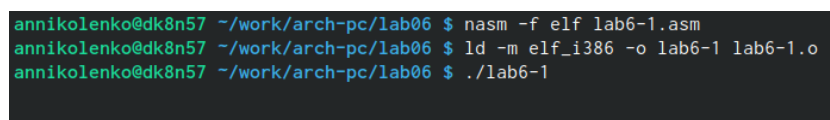


```
Открыть ▼ + *lab6-1.asm
~/work/arch-pc/lab06

1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, 6
8 mov ebx, 4
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintLF
13 call quit
```

Рис. 4.5: Редактирование файла

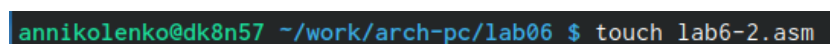
Создаю новый исполняемый файл программы и запускаю его. Выводится символ с кодом 10, это символ перевода строки. Этот символ не отображается при выводе на экран (рис. [4.6]).



```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 4.6: Редактирование файла

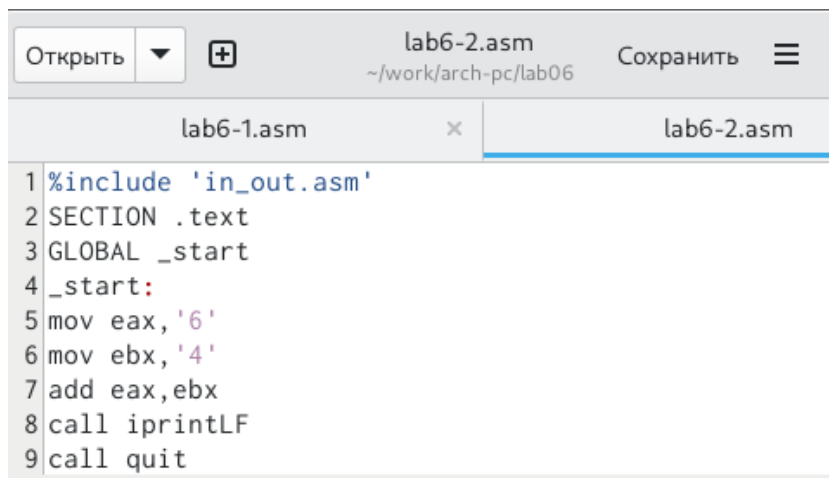
Создаю новый файл lab6-2.asm с помощью команды touch (рис. [4.7]).



```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ touch lab6-2.asm
```

Рис. 4.7: Создание файла

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



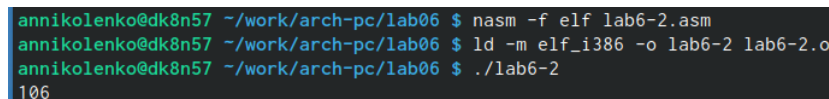
```
lab6-2.asm
~/work/arch-pc/lab06
Сохранить

lab6-1.asm x lab6-2.asm

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, '6'
6 mov ebx, '4'
7 add eax, ebx
8 call iprintLF
9 call quit
```

Рис. 4.8: Редактирование файла

Создаю и запускаю исполняемый файл `lab6-2`. 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” (рис. [4.9]).



```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 4.9: Запуск исполняемого файла

Заменяю в тексте программы в файле `lab6-2.asm` символы “6” и “4” на числа 6 и 4 (рис. [4.10]).

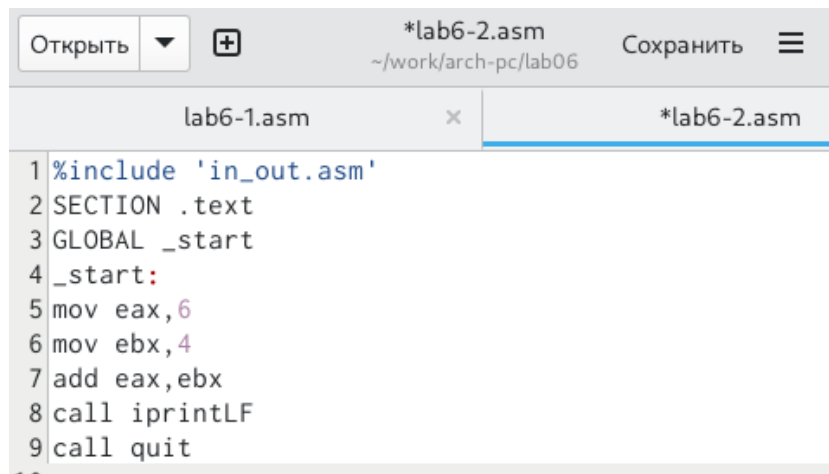


Рис. 4.10: Редактирование файла

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. [4.12]).

```

annikolenko@dk8n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ./lab6-2
10

```

Рис. 4.11: Запуск исполняемого файла

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. [4.12]).

```

annikolenko@dk8n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ./lab6-2
10

```

Рис. 4.12: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. [4.13]).

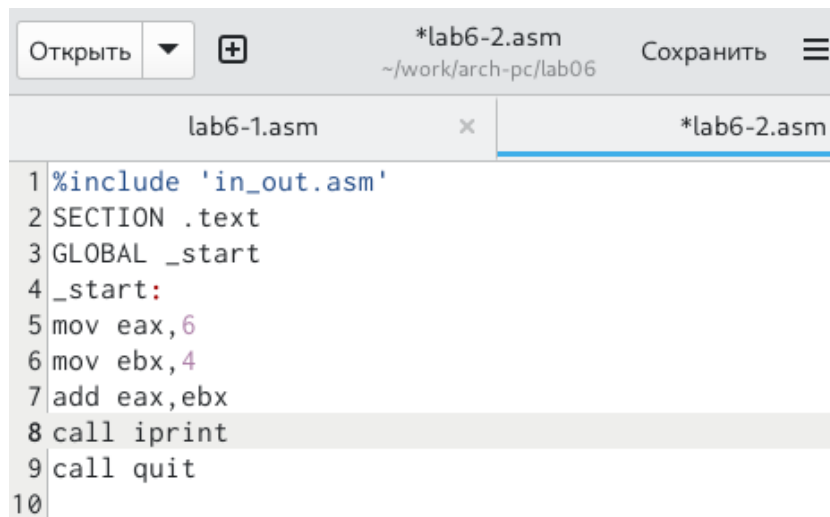


Рис. 4.13: Редактирование файла

Создаю и запускаю новый исполняемый файл. 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. [4.14]).

```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ./lab6-2
10annikolenko@dk8n57 ~/work/arch-pc/lab06 $
```

Рис. 4.14: Запуск исполняемого файла

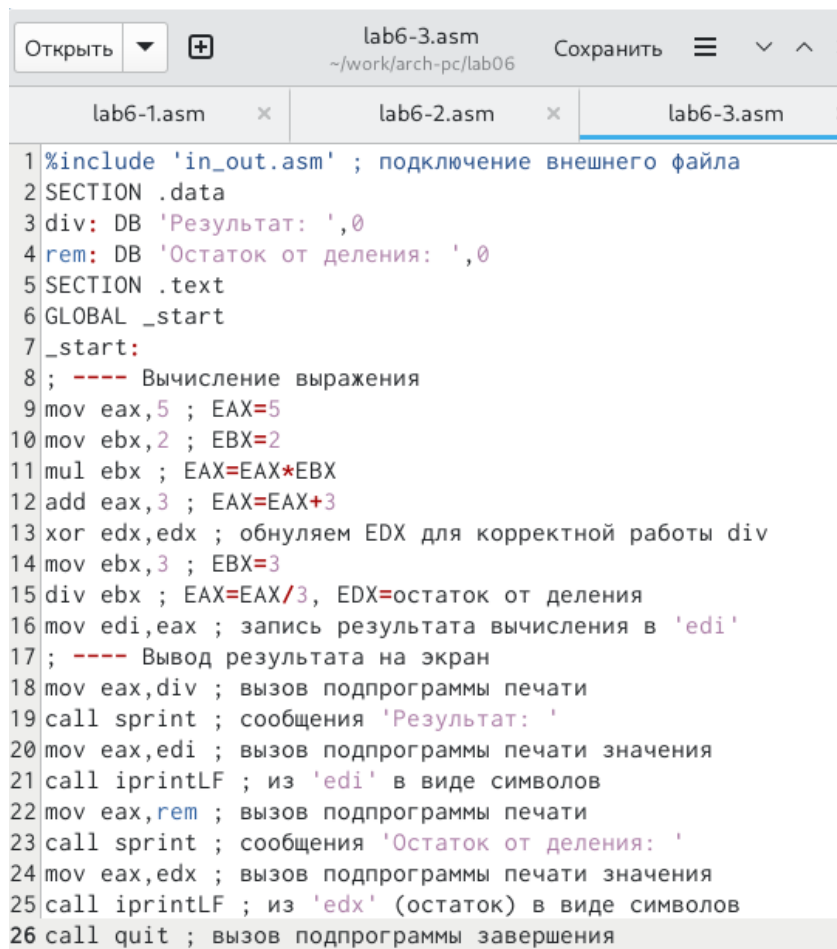
4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch` (рис. [4.15]).

```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ touch lab6-3.asm
```

Рис. 4.15: Создание файла

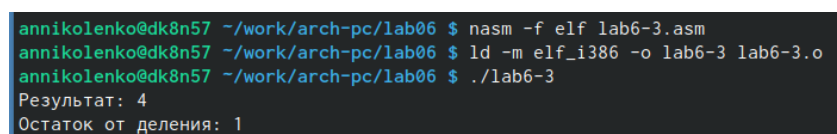
Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. [4.16]).



```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,5 ; EAX=5
10 mov ebx,2 ; EBX=2
11 mul ebx ; EAX=EAX*EBX
12 add eax,3 ; EAX=EAX+3
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,3 ; EBX=3
15 div ebx ; EAX=EAX/3, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
```

Рис. 4.16: Редактирование файла

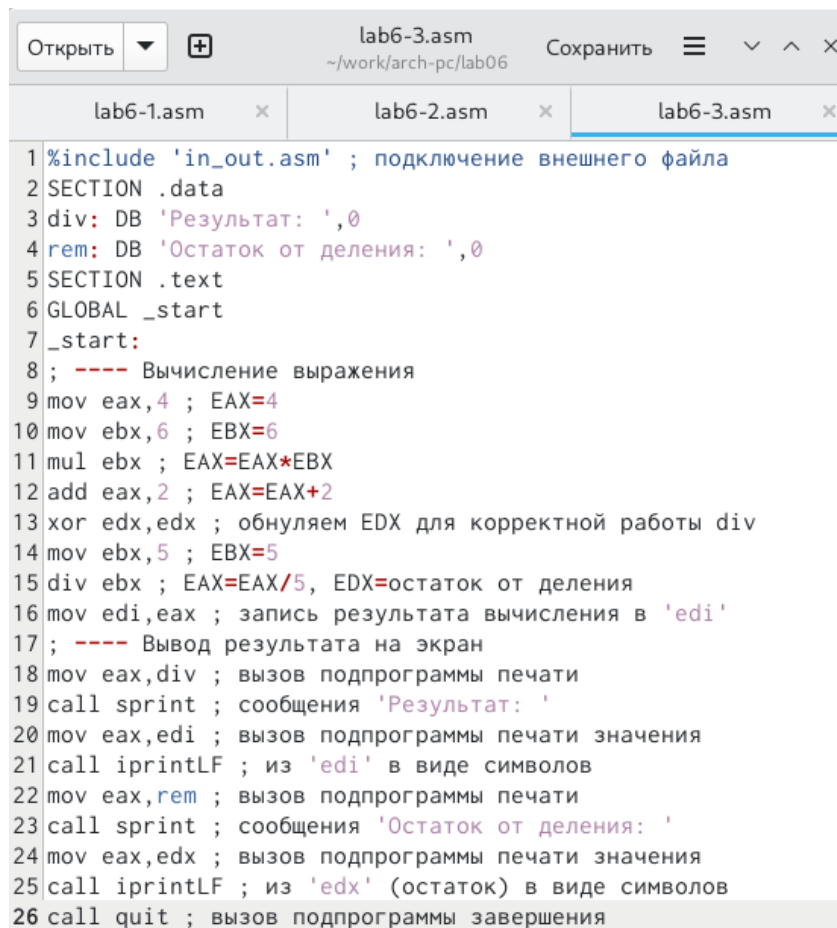
Создаю исполняемый файл и запускаю его (рис. [4.17]).



```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.17: Запуск исполняемого файла

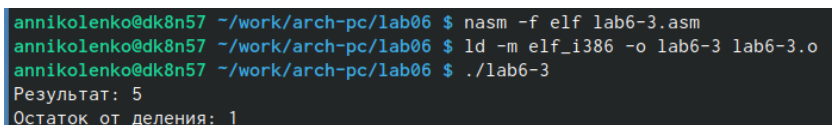
Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2) / 5$ (рис. [4.18]).



```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,4 ; EAX=4
10 mov ebx,6 ; EBX=6
11 mul ebx ; EAX=EAX*EBX
12 add eax,2 ; EAX=EAX+2
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,5 ; EBX=5
15 div ebx ; EAX=EAX/5, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
```

Рис. 4.18: Редактирование файла

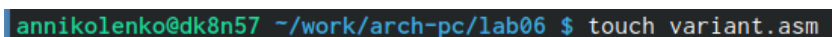
Создаю и запускаю новый исполняемый файл (рис. 19). Проверила его работу (корректность ответа), решив уравнение самостоятельно (рис. [4.19]).



```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 4.19: Запуск исполняемого файла

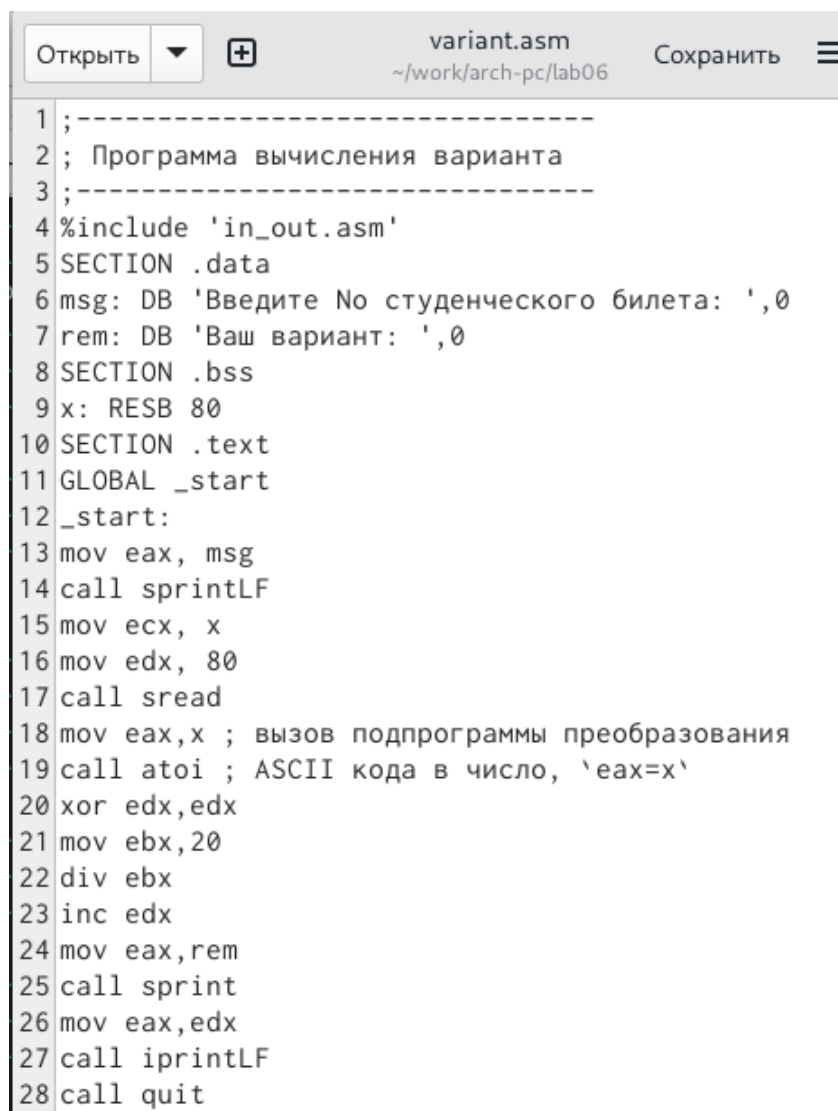
Создаю файл variant.asm с помощью команды touch (рис. [4.20]).



```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ touch variant.asm
```

Рис. 4.20: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. [4.21]).

The image shows a text editor window titled 'variant.asm' with a path '~/.work/arch-pc/lab06'. The editor contains assembly code for a program that calculates a variant number based on a student ticket number. The code includes comments in Russian, data and bss sections, and a main routine that reads input, converts it to a number, and calculates the variant using division and modulo operations. The code is as follows:

```
1 ;-----
2 ; Программа вычисления варианта
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg: DB 'Введите No студенческого билета: ',0
7 rem: DB 'Ваш вариант: ',0
8 SECTION .bss
9 x: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprintLF
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax,x ; вызов подпрограммы преобразования
19 call atoi ; ASCII кода в число, `eax=x`
20 xor edx,edx
21 mov ebx,20
22 div ebx
23 inc edx
24 mov eax,rem
25 call sprint
26 mov eax,edx
27 call iprintLF
28 call quit
```

Рис. 4.21: Редактирование файла

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 15 (рис. [4.22]).

```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132236114
Ваш вариант: 15
```

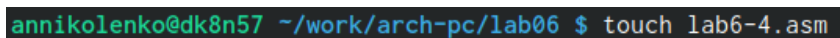
Рис. 4.22: Запуск исполняемого файла

5 Ответы на вопросы

1. За вывод сообщения “Ваш вариант” отвечают строки кода: `mov eax,rem` `call sprint`
2. `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` - вызов 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` прибавляет 1 к значению регистра `edx`
7. За вывод на экран результатов вычислений отвечают строки: `mov eax,edx` `call iprintLF`

6 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch (рис. [6.1]).

A terminal window with a dark background. The prompt is 'annikolenko@dk8n57' in green. The current directory is '~/work/arch-pc/lab06' in blue. The command being executed is '\$ touch lab6-4.asm' in white.

```
annikolenko@dk8n57 ~/work/arch-pc/lab06 $ touch lab6-4.asm
```

Рис. 6.1: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(5 + x)^2 - 3$ (^ - возведение в степень) (Вариант 15) (рис. [6.2]).

```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data ; секция иницированных данных
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5 SECTION .bss ; секция не иницированных данных
6 x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выдел
  80 байт
7 SECTION .text ; Код программы
8 GLOBAL _start ; Начало программы
9 _start: ; Точка входа в программу
10; ---- Вычисление выражения
11 mov eax, msg ; запись адреса выводимого сообщения в eax
12 call sprint ; вызов подпрограммы печати сообщения
13 mov ecx, x ; запись адреса переменной в ecx
14 mov edx, 80 ; запись длины вводимого значения в edx
15 call sread ; вызов подпрограммы ввода сообщения
16 mov eax,x ; вызов подпрограммы преобразования
17 call atoi ; ASCII кода в число, 'eax=x'
18 add eax,5; eax = eax+5 = x + 5
19 mov ebx,eax ; запись значения x в регистр ebx
20 mul ebx; EAX=EAX*EBX = (x+5)*(x+5)
21;call atoi ; ASCII кода в число, 'ebx=x'
22;add ebx,5; ebx = ebx+5 = x + 5
23;mul ebx; EAX=EAX*EBX = (x+5)*(x+5)
24 sub eax,3; eax = eax-3 = (x+5)*(x+5)-3
25 mov edi,eax ; запись результата вычисления в 'edi'
26; ---- Вывод результата на экран
27 mov eax,rem ; вызов подпрограммы печати
28 call sprint ; сообщения 'Результат: '
29 mov eax,edi ; вызов подпрограммы печати значения
30 call iprint ; из 'edi' в виде символов
31 call quit ; вызов подпрограммы завершения

```

Рис. 6.2: Написание программы

Создаю и запускаю исполняемый файл. При вводе значения 5, вывод - 97. (рис. [6.3]).

```

annikolenko@dk8n75 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
annikolenko@dk8n75 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
annikolenko@dk8n75 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 5

```

Рис. 6.3: Запуск исполняемого файла

Повторный запуск исполняемого файла с другим значением $x=1$, вывод - 33 (рис. [6.4]).

```
Результат: 97annikolenko@dk8n75 ~/work/arch-pc/lab06 $ ./lab6-4  
Введите значение переменной x: 1  
Результат: 33annikolenko@dk8n75 ~/work/arch-pc/lab06 $ █
```

Рис. 6.4: Запуск исполняемого файла

7 Листинги программ

1. Программа вывода значения регистра eax

```
%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
```

2. Программа вывода значения регистра eax

```
%include 'in_out.asm' SECTION .text GLOBAL _start
_start: mov eax,'6' mov ebx,'4' add eax,ebx call iprintLF call quit
```

3. Программа вычисления выражения $(5 \times 2 + 3)/3$;-----

; 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 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

4. Программа вычисления вычисления варианта задания по номеру студенческого билета ;----- ; Программа вычисления варианта ;----- %include 'in_out.asm' SECTION .data msg: DB 'Введите No 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

5. Программа для вычисления значения выражения $(5 + x)^2 - 3$. %include 'in_out.asm' ; подключение внешнего файла SECTION .data ; секция иницированных данных msg: DB 'Введите значение переменной x:',0 rem: DB 'Результат:',0 SECTION .bss ; секция не иницированных 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,5; $eax = eax + 5 = x + 5$ mov ebx,eax ; запись значения x в регистр ebx mul ebx; $EAX = EAX * EBX = (x+5)(x+5)$;call atoi ; ASCII кода в число, ebx=x ;add ebx,5; $ebx = ebx + 5 = x + 5$;mul ebx; $EAX = EAX * EBX = (x+5)(x+5)$ sub eax,3; $eax = eax - 3 = (x+5)*(x+5) - 3$ 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

8 Выводы

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

Список литературы