

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Архитектура компьютера

Студент: Краснова К. Г.

Группа: НКАбд-05-24

МОСКВА

2024 г.

Содержание

1	Цель работы.....	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
4.1	Символьные и численные данные в NASM	7
4.2	Выполнение арифметических операций в NASM.....	11
4.2.1	Ответы на вопросы по программе.....	14
4.3	Выполнение заданий для самостоятельной работы.....	14
5	Выводы.....	17
6	Список литературы	17

Список иллюстраций

Рис. 1. Создание директории.....	7
Рис. 2. Редактирование файла	7
Рис. 3. Запуск исполняемого файла.....	8
Рис. 4. Редактирование файла	8
Рис. 5. Запуск исполняемого файла.....	9
Рис. 6. Создание файла.....	9
Рис. 7. Редактирование файла	9
Рис. 8. Запуск исполняемого файла.....	9
Рис. 9. Редактирование файла	10
Рис. 10. Запуск исполняемого файла.....	10
Рис. 11. Редактирование файла	10
Рис. 12. Запуск исполняемого файла.....	11
Рис. 13. Создание файла.....	11
Рис. 14. Редактирование файла	11
Рис. 15. Запуск исполняемого файла.....	12
Рис. 16. Изменение программы	12
Рис. 17. Запуск исполняемого файла.....	13
Рис. 18. Создание файла.....	13
Рис. 19. Редактирование файла	13
Рис. 20. Запуск исполняемого файла.....	14
Рис. 21. Создание файла.....	14
Рис. 22. Написание программы	15
Рис. 23. Запуск исполняемого файла.....	15

1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера N

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 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

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

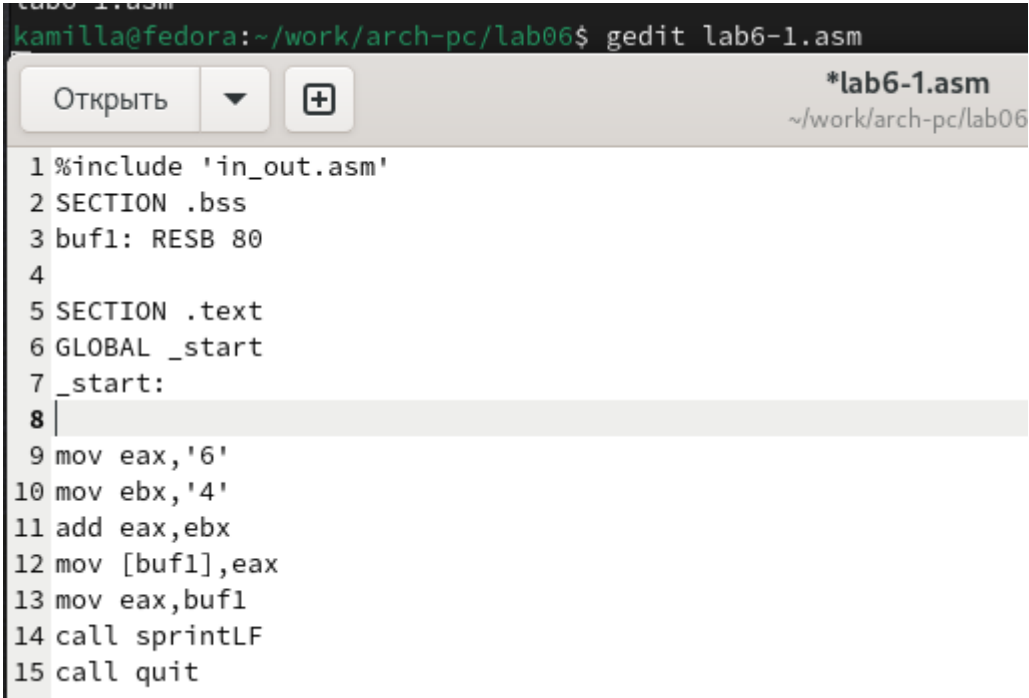
С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6. Перехожу в созданный каталог с помощью утилиты `cd` и с помощью утилиты `touch` создаю файл `lab6-1.asm`. (рис. 1).



```
kamilla@fedora:~/work/arch-pc/la
kamilla@fedora:~$ mkdir ~/work/arch-pc/lab06
kamilla@fedora:~$ cd ~/work/arch-pc/lab06
kamilla@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
kamilla@fedora:~/work/arch-pc/lab06$ ls
lab6-1.asm
kamilla@fedora:~/work/arch-pc/lab06$
```

Рис. 1. Создание директории

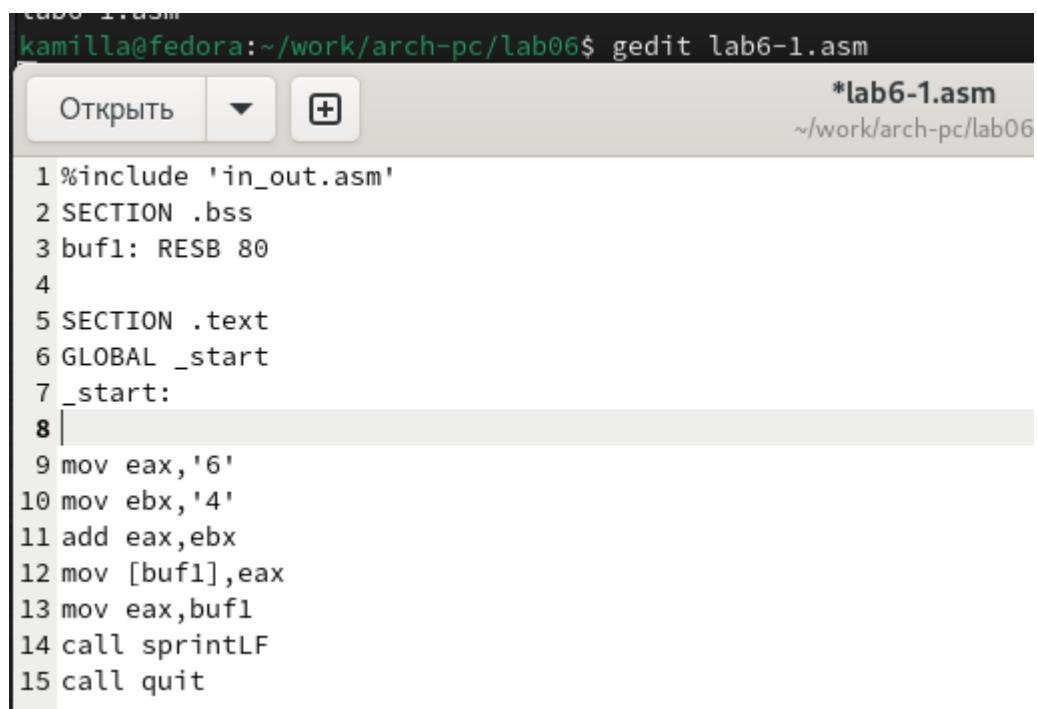
Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 2).



```
kamilla@fedora:~/work/arch-pc/lab06$ gedit lab6-1.asm
Открыть *lab6-1.asm ~/work/arch-pc/lab06
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4
5 SECTION .text
6 GLOBAL _start
7 _start:
8
9 mov eax, '6'
10 mov ebx, '4'
11 add eax, ebx
12 mov [buf1], eax
13 mov eax, buf1
14 call sprintLF
15 call quit
```

Рис. 2. Редактирование файла

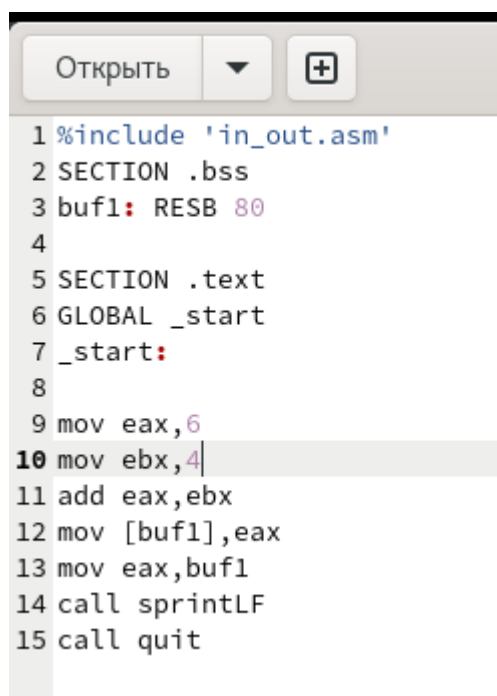
Создаю исполняемый файл программы и запускаю его (рис. 3). Вывод программы: символ `j`, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.



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

Рис. 3. Запуск исполняемого файла

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



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

Рис. 4. Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 5). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.


```

kamilla@fedora:~/work/arch-pc/lab06$ gedit lab6-1.asm
kamilla@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
kamilla@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
kamilla@fedora:~/work/arch-pc/lab06$ ./lab6-1

kamilla@fedora:~/work/arch-pc/lab06$

```

Рис. 5. Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 6).

```

kamilla@fedora:~/work/arch-pc/lab06$ touch lab6-2.asm
kamilla@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
kamilla@fedora:~/work/arch-pc/lab06$

```

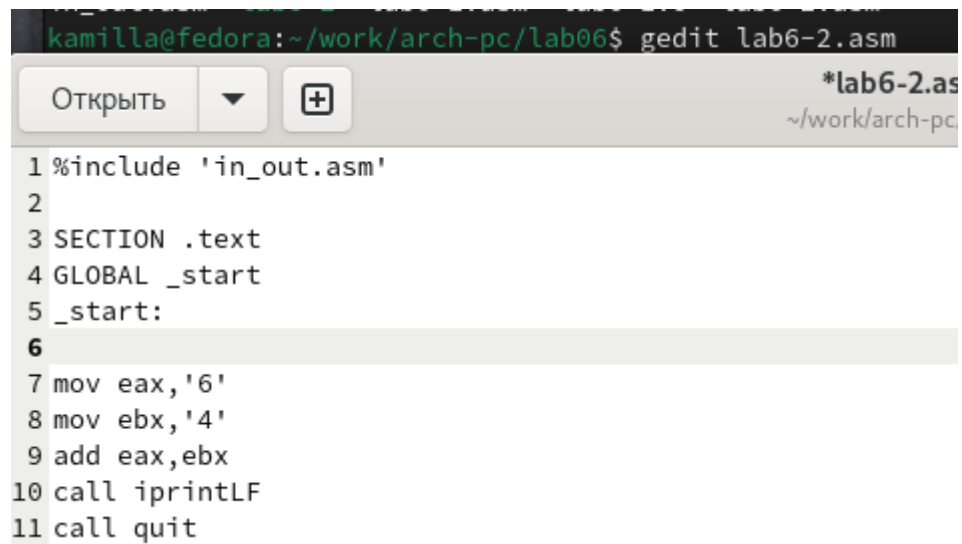
Рис. 6. Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 7).

```

kamilla@fedora:~/work/arch-pc/lab06$ gedit lab6-2.asm

```



```

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

```

Рис. 7. Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 8). 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”.

```

kamilla@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
kamilla@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
kamilla@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
kamilla@fedora:~/work/arch-pc/lab06$

```

Рис. 8. Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 9).

```

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

```

Рис. 9. Редактирование файла

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

```

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

```

Рис. 10. Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 11).

```

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

```

Рис. 11. Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 12). 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.

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

Рис. 12. Запуск исполняемого файла

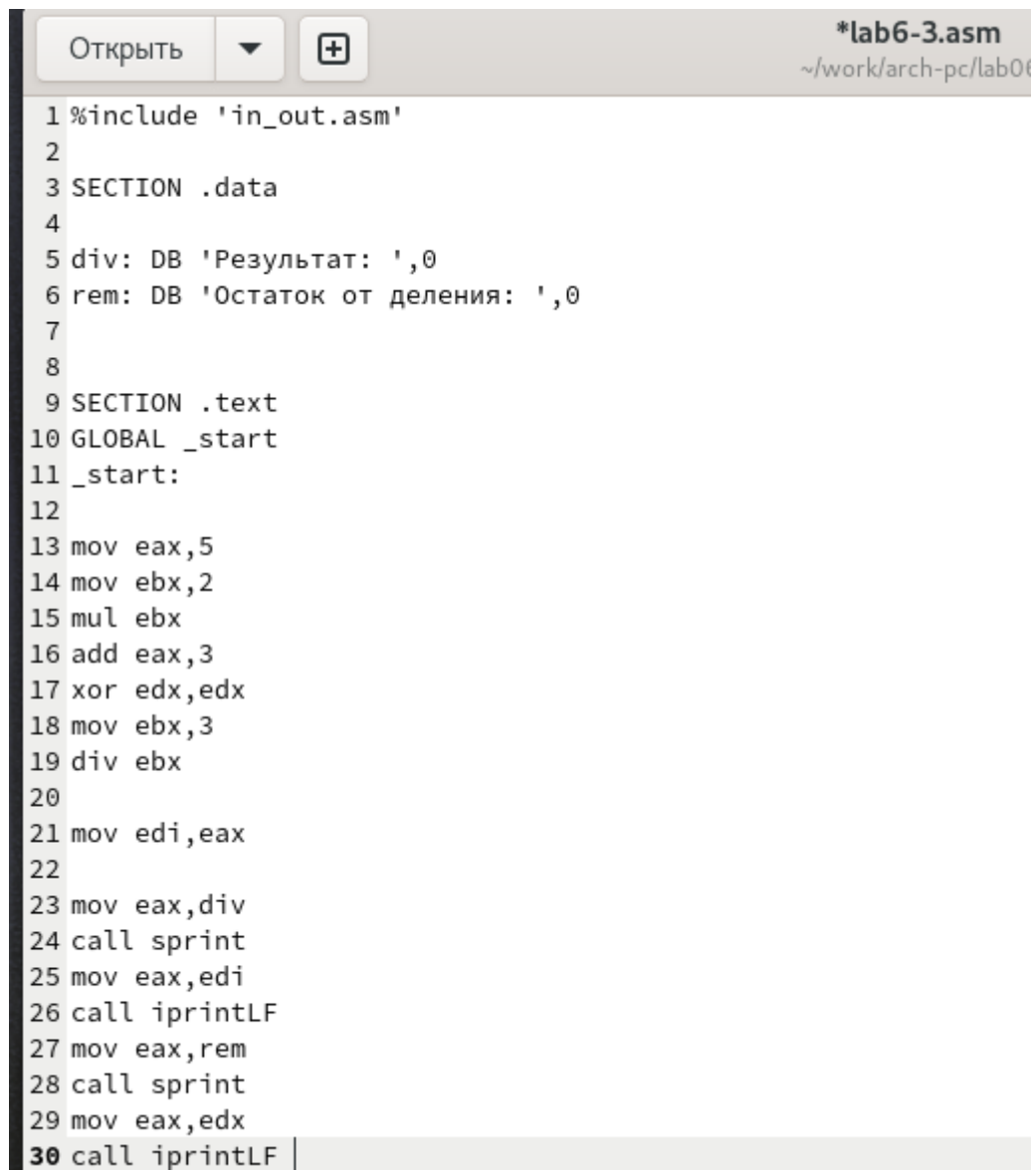
4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. 13).

```
10kamilla@fedora:~/work/arch-pc/lab06$ touch lab6-3.asm
kamilla@fedora:~/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
kamilla@fedora:~/work/arch-pc/lab06$
```

Рис. 13. Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 14).



```
*lab6-3.asm
~/work/arch-pc/lab06

1 %include 'in_out.asm'
2
3 SECTION .data
4
5 div: DB 'Результат: ',0
6 rem: DB 'Остаток от деления: ',0
7
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12
13 mov eax,5
14 mov ebx,2
15 mul ebx
16 add eax,3
17 xor edx,edx
18 mov ebx,3
19 div ebx
20
21 mov edi,eax
22
23 mov eax,div
24 call sprint
25 mov eax,edi
26 call iprintLF
27 mov eax,rem
28 call sprint
29 mov eax,edx
30 call iprintLF
```

Рис. 14. Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 15).

```
kamilla@fedora: ~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
kamilla@fedora: ~/work/arch-pc/lab06$
```

Рис. 15. Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 16).

```
1 %include 'in_out.asm'
2
3 SECTION .data
4
5 div: DB 'Результат: ',0
6 rem: DB 'Остаток от деления: ',0
7
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12
13 mov eax,4
14 mov ebx,6
15 mul ebx
16 add eax,2
17 xor edx,edx
18 mov ebx,5
19 div ebx
20
21 mov edi,eax
22
23 mov eax,div
24 call sprint
25 mov eax,edi
26 call iprintLF
27 mov eax,rem
28 call sprint
29 mov eax,edx
30 call iprintLF
31 call quit
```

Рис. 16. Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 17). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```

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

```

Рис. 17. Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 18).

```

kamilla@fedora:~/work/arch-pc/lab06$ touch variant.asm
kamilla@fedora:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  variant.asm
lab6-1      lab6-1.o   lab6-2.asm  lab6-3    lab6-3.o
kamilla@fedora:~/work/arch-pc/lab06$

```

Рис. 18. Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 19).

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № студенческого билета: ',
4 rem: DB 'Ваш вариант: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x
16 call atoi
17
18 xor edx, edx
19 mov ebx, 20
20 div ebx
21 inc edx
22 mov eax, rem
23 call sprintf
24 mov eax, edx
25 call iprintLF
26 call quit

```

Рис. 19. Редактирование файла

Создаю и запускаю исполняемый файл (рис. 20). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 17.

```

kamilla@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
kamilla@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
kamilla@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246076
Ваш вариант: 17

```

Рис. 20. Запуск исполняемого файла

4.2.1 Ответы на вопросы по программе

4. 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

```

4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch (рис. 21).

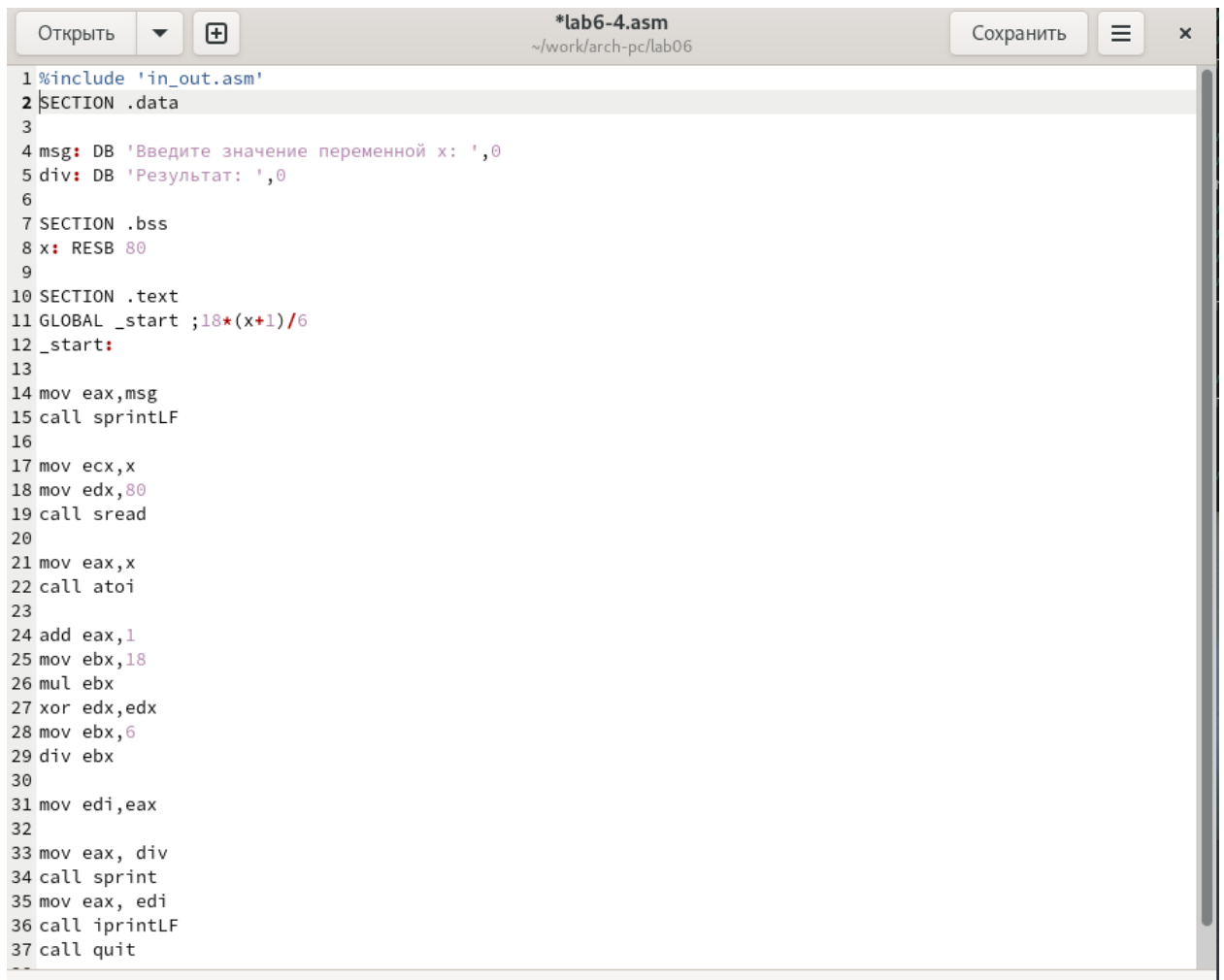
```

kamilla@fedora:~/work/arch-pc/lab06$ touch lab6-4.asm
kamilla@fedora:~/work/arch-pc/lab06$ gedit lab6-4.asm

```

Рис. 21. Создание файла

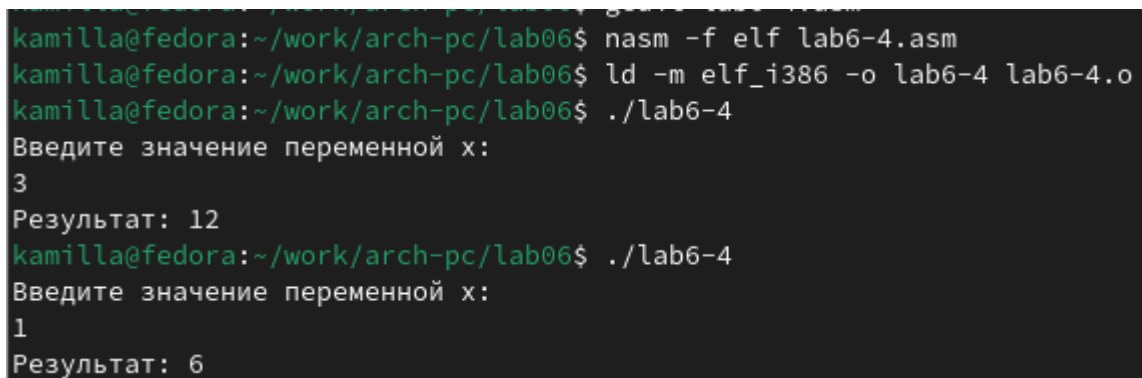
Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(11 + x) * 2 - 6$ (рис. 22). Это выражение было под вариантом 8.



```
1 %include 'in_out.asm'
2 SECTION .data
3
4 msg: DB 'Введите значение переменной x: ',0
5 div: DB 'Результат: ',0
6
7 SECTION .bss
8 x: RESB 80
9
10 SECTION .text
11 GLOBAL _start ;18*(x+1)/6
12 _start:
13
14 mov eax,msg
15 call sprintf
16
17 mov ecx,x
18 mov edx,80
19 call sread
20
21 mov eax,x
22 call atoi
23
24 add eax,1
25 mov ebx,18
26 mul ebx
27 xor edx,edx
28 mov ebx,6
29 div ebx
30
31 mov edi,eax
32
33 mov eax, div
34 call sprintf
35 mov eax, edi
36 call iprintLF
37 call quit
```

Рис. 22. Написание программы

Создаю и запускаю исполняемый файл (рис. 23). При вводе значения 3, вывод — 12. При 1, вывод — 6.



```
kamilla@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
kamilla@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
kamilla@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x:
3
Результат: 12
kamilla@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x:
1
Результат: 6
```

Рис. 23. Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения $18 \cdot (x + 1) / 6$.

```
%include 'in_out.asm'

SECTION .data
```

```
msg: DB 'Введите значение переменной x: ',0
```

```
div: DB 'Результат: ',0
```

```
SECTION .bss
```

```
x: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start ;18*(x+1)/6
```

```
_start:
```

```
mov eax,msg
```

```
call sprintf
```

```
mov ecx,x
```

```
mov edx,80
```

```
call sread
```

```
mov eax,x
```

```
call atoi
```

```
add eax,1
```

```
mov ebx,18
```

```
mul ebx
```

```
xor edx,edx
```

```
mov ebx,6
```

```
div ebx
```

```
mov edi,eax
```

```
mov eax, div
```



```
call sprint  
mov eax, edi  
call iprintLF
```

```
call quit
```

5 Выводы

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

6 Список литературы

1. [Лабораторная работа №7](#)
2. [Таблица ASCII](#)