

# **Отчет по лабораторной работе №6**

**дисциплина: Архитектура компьютера**

Белоусова Елизавета Валентиновна

# Содержание

## **Список иллюстраций**

# 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. - 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 символ- ном виде. Кодирование этой информации производится согласно кодовой табли- це 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, на- пример, 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

Перехожу в каталог, созданный для файлов с программами для лабораторной работы №6 (рис. 1).

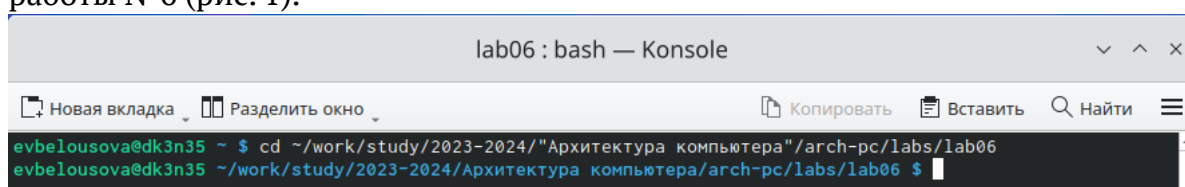


Рис. 1: Перемещение между директориями

С помощью утилиты touch создаю файл lab6-1.asm (рис. 2)

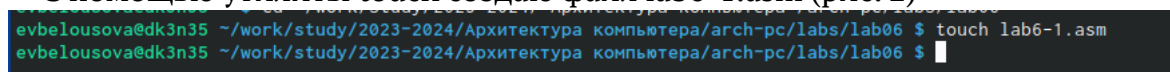


Рис. 2: Создание файла

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, так как он будет использоваться в других программах (рис. 3).

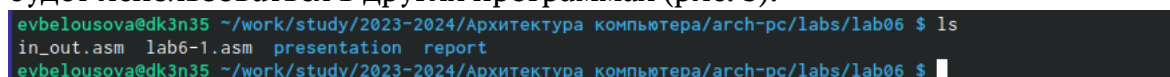


Рис. 3: Создание копии файла

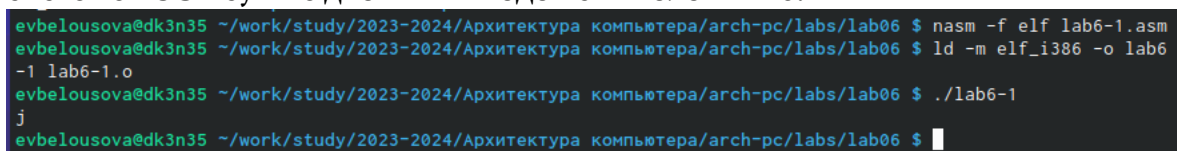
Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. 4).



```
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: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 5). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.



```
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-1.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6
-lab6-1.o
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-1
j
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 5: Запуск исполняемого файла

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



```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, 54
8 mov ebx, 52
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintLF
13 call quit
```

Рис. 6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 7). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

```

evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-1.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6
-lab6-1.o
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-1

evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $

```

Рис. 7: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 8).

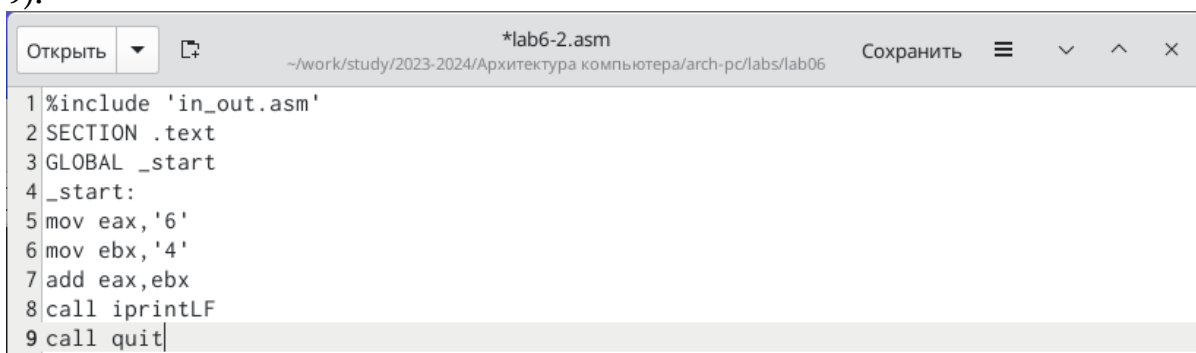
```

evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ touch lab6-2.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $

```

Рис. 8: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 9).



```

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. 9: Редактирование файла

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

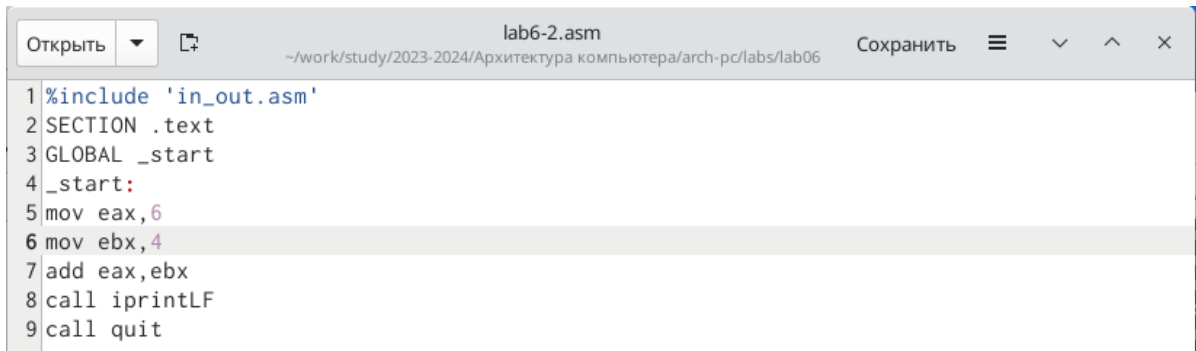
```

evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-2.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6
-lab6-2.o
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-2
106
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $

```

Рис. 10: Запуск исполняемого файла

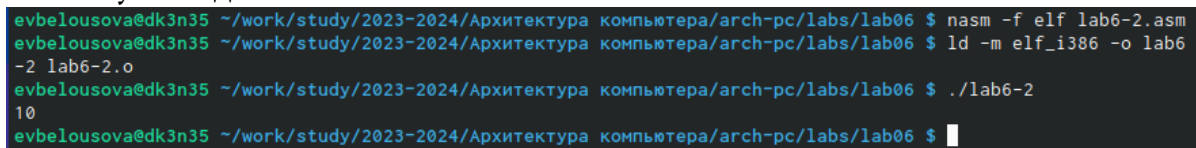
Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 11).



```
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. 11: Редактирование файла

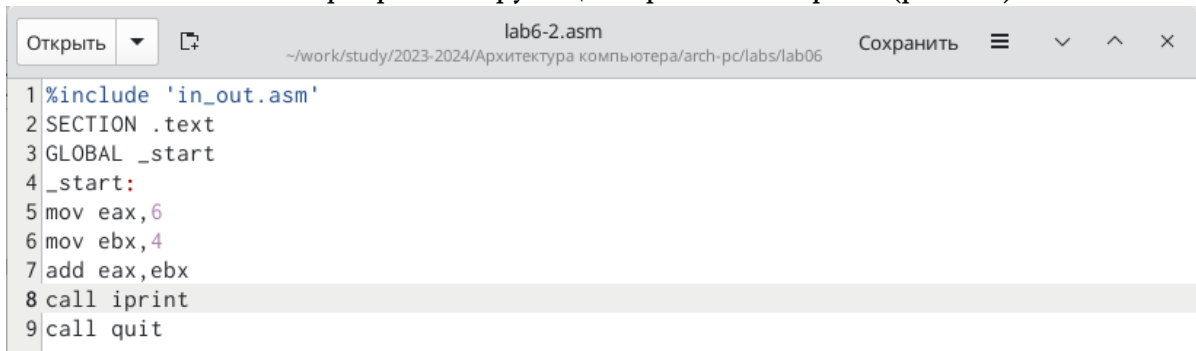
Создаю и запускаю новый исполняемый файл (рис. 12). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.



```
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-2.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-2
10
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 12: Запуск исполняемого файла

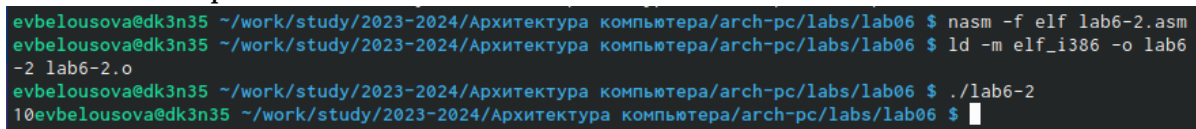
Заменяю в тексте программы функцию iprintLF на iprint (рис. 13).



```
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 iprint
9 call quit
```

Рис. 13: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 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.



```
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-2.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-2
10
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```



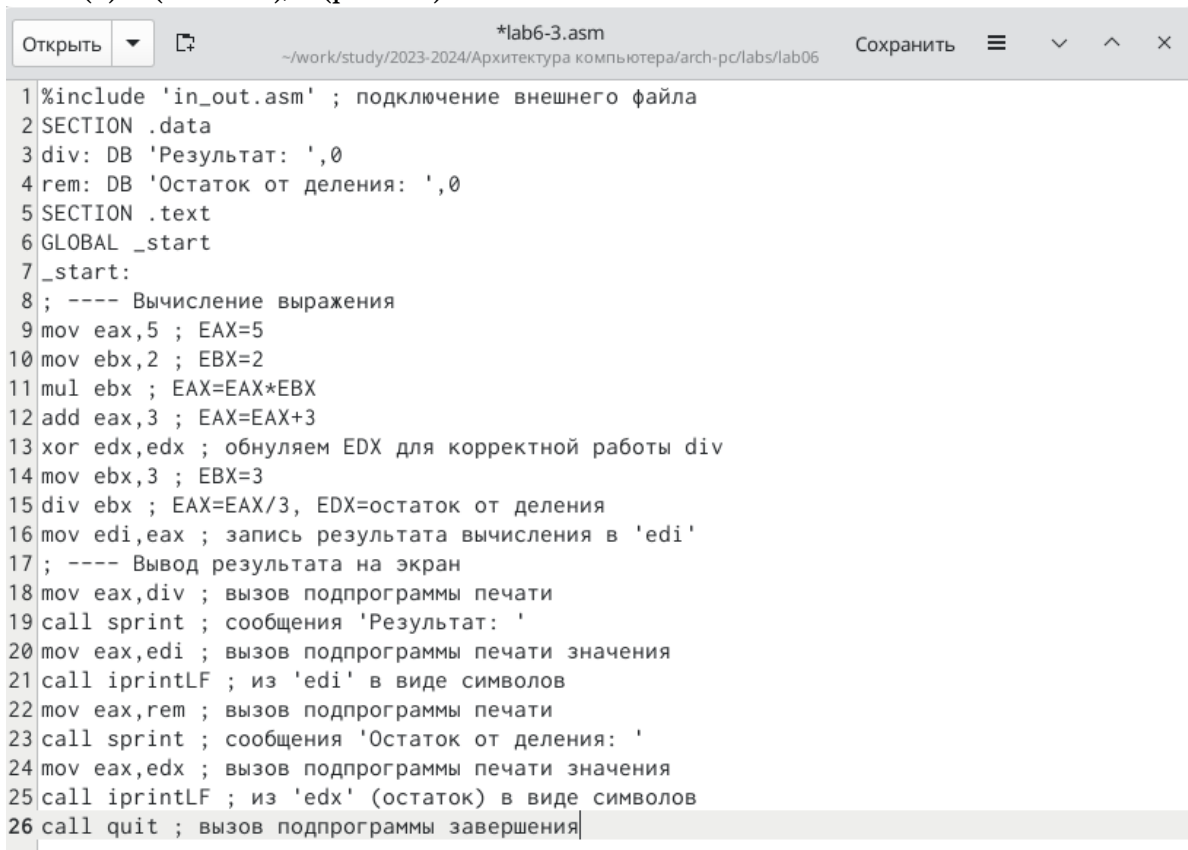
Рис. 14: Запуск исполняемого файла

Создаю файл lab6-3.asm с помощью утилиты touch (рис. 15).

```
10evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ touch lab6-3.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 15: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 * 2 + 3)/3$  (рис. 16).



```
*lab6-3.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06
Сохранить

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
```

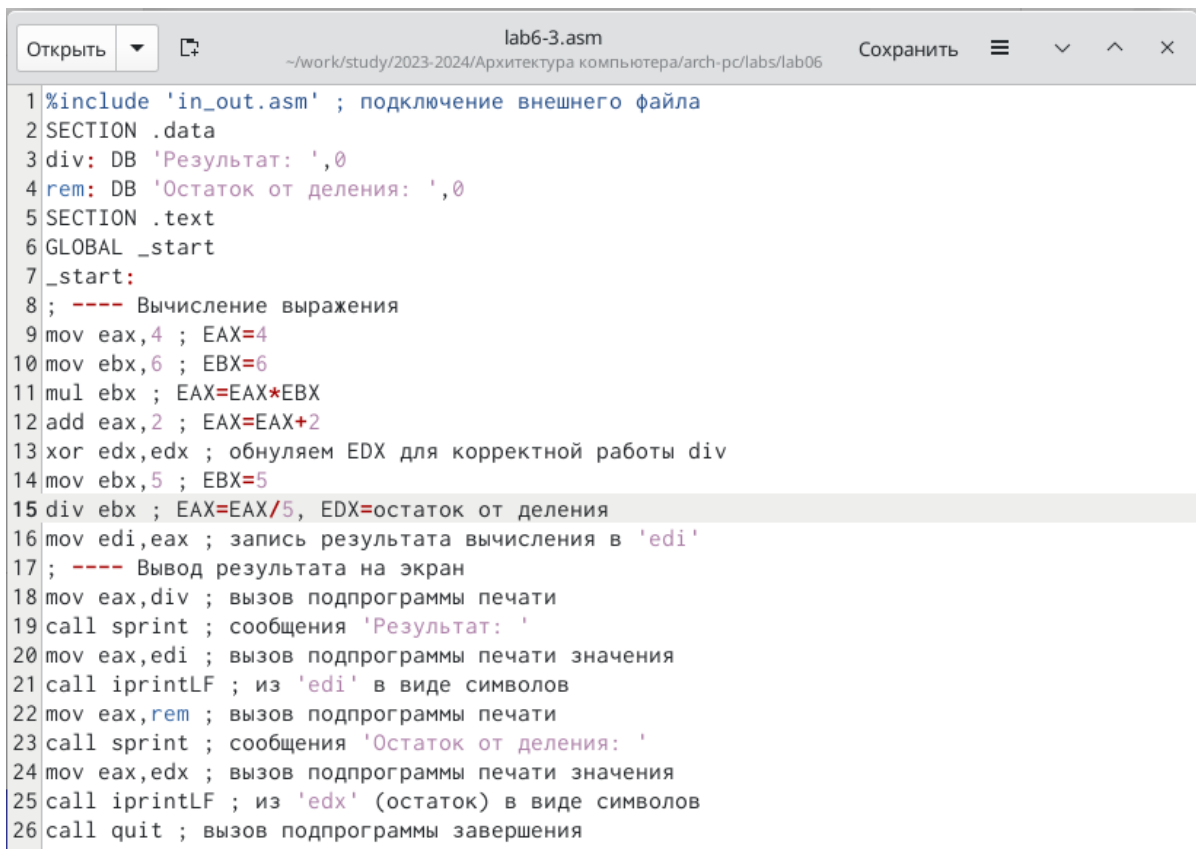
Рис. 16: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 17)

```
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-3.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 17: Запуск исполняемого файла

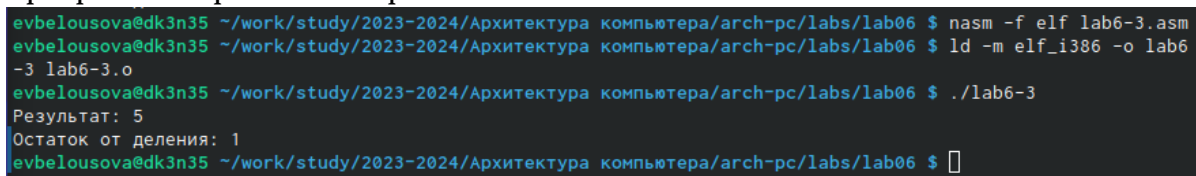
Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4 * 6 + 2)/5$  (рис. 18).



```
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. 18: Изменение текста программы

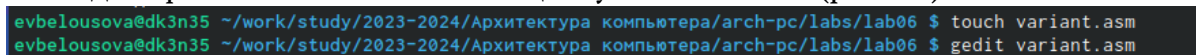
Создаю и запускаю новый исполняемый файл (рис. 19). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.



```
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-3.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 19: Запуск исполняемого файла

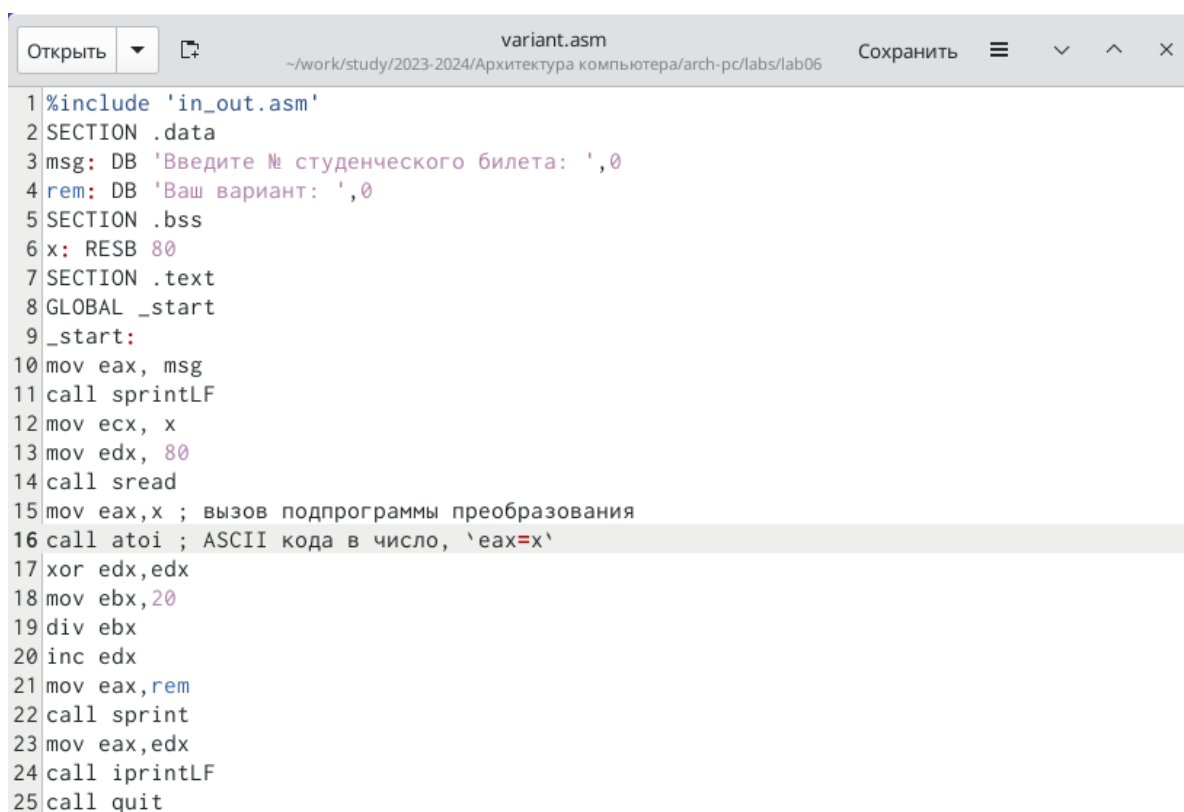
Создаю файл variant.asm с помощью утилиты touch (рис. 20)



```
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ touch variant.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ gedit variant.asm
```

Рис. 20: Создание файла

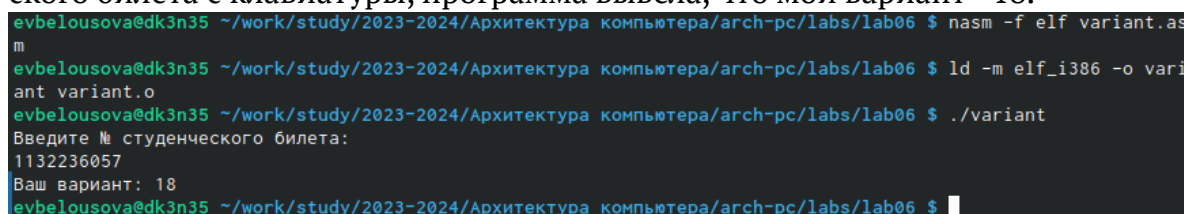
Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 21)



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № студенческого билета: ',0
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 ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, 'eax=x'
17 xor edx,edx
18 mov ebx,20
19 div ebx
20 inc edx
21 mov eax,rem
22 call sprintf
23 mov eax,edx
24 call iprintLF
25 call quit
```

Рис. 21: Редактирование файла

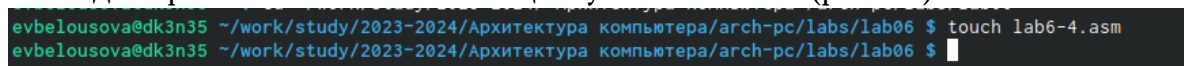
Создаю и запускаю исполняемый файл (рис. 22). Ввожу номер своего студенческого билета с клавиатуры, программа вывела, что мой вариант - 18.



```
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf variant.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o variant variant.o
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./variant
Введите № студенческого билета:
1132236057
Ваш вариант: 18
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 22: Запуск исполняемого файла

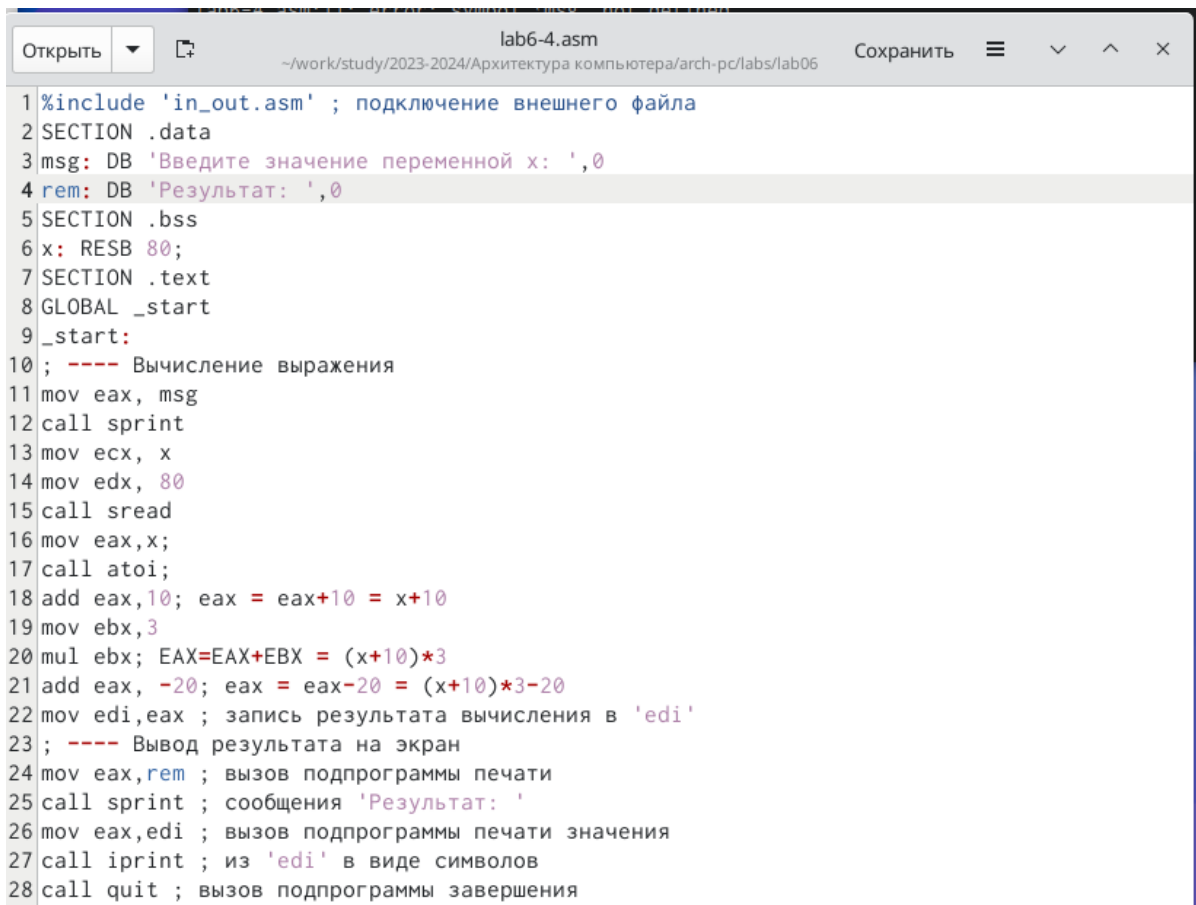
Создаю файл lab6-4.asm с помощью утилиты touch (рис. 23).



```
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ touch lab6-4.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 23: Создание файла

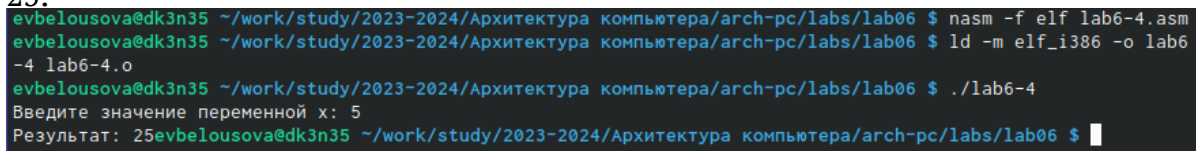
Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения  $3(\text{№} + 10) - 20$  (рис. 24). Это выражение было под вариантом 18.



```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5 SECTION .bss
6 x: RESB 80;
7 SECTION .text
8 GLOBAL _start
9 _start:
10 ; ---- Вычисление выражения
11 mov eax, msg
12 call sprint
13 mov ecx, x
14 mov edx, 80
15 call sread
16 mov eax,x;
17 call atoi;
18 add eax,10; eax = eax+10 = x+10
19 mov ebx,3
20 mul ebx; EAX=EAX*EBX = (x+10)*3
21 add eax, -20; eax = eax-20 = (x+10)*3-20
22 mov edi,eax ; запись результата вычисления в 'edi'
23 ; ---- Вывод результата на экран
24 mov eax,rem ; вызов подпрограммы печати
25 call sprint ; сообщения 'Результат: '
26 mov eax,edi ; вызов подпрограммы печати значения
27 call iprint ; из 'edi' в виде символов
28 call quit ; вызов подпрограммы завершения
```

Рис. 24: Написание программы

Создаю и запускаю исполняемый файл (рис. 25). При вводе значения 5, вывод -



```
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ nasm -f elf lab6-4.asm
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $ ./lab6-4
Введите значение переменной x: 5
Результат: 25evbelousova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06 $
```

Рис. 25: Запуск исполняемого файла

*Ответы на вопросы:* 1. За вывод сообщения “Ваш вариант” отвечают данные 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 еах. 4. За вычисления варианта отвечают данные строки кода: `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 данные строки кода: `mov eax,edx` `call iprintLF`

## **5 Выводы**

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

## **6 Список литературы**

Архитектура ЭВМ