

Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютеров

Симонова Виктория Игоревна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Задание для самостоятельной работы	18
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Создание каталога	8
4.2	Создание файла	8
4.3	Копирую файл in_out	9
4.4	Ввожу текст в файл	9
4.5	Создаю и запускаю исполняемый файл	9
4.6	Изменяю файл	10
4.7	Создаю и запускаю исполняемый файл	10
4.8	Создаю файл	11
4.9	Ввожу код в файл	11
4.10	Создаю и запускаю исполняемый файл	11
4.11	Исправляю	12
4.12	Запуск исполняемого файла с имправленным кодом	12
4.13	Меняю на inprint	13
4.14	Компиляция и запуск файла.	13
4.15	Создаю файл	13
4.16	Ввожу в файл текст программы	14
4.17	Запуск исполняемого файла lab6-3	14
4.18	Изменяю файл lab6-3	15
4.19	Запуск изменённого lab6-3	15
4.20	Ввожу в файл программу для вычисления номера задания	16
4.21	Запуск программы для вычисления варианта	16
4.22	Мой код	18
4.23	Запуск	18

Список таблиц

1 Цель 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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа 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. Схема команды целочисленного сложения `add` (от англ. *addition* - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда `add` работает как с числами со знаком, так и без знака и выглядит следующим образом: `add ,`

4 Выполнение лабораторной работы

##Символьные и численные данные в NASM Создаю каталог для программам лабораторной работы № 6 (рис. [4.1]).

```
[visimonova@fedora ~]$ mkdir ~/work/arch-pc/lab06  
[visimonova@fedora ~]$ cd ~/work/arch-pc/lab06
```

Рис. 4.1: Создание каталога

Перейхожу в него и создаю файл lab6-1.asm (рис. [4.2]).

```
[visimonova@fedora lab06]$ touch lab6-1.asm
```

Рис. 4.2: Создание файла

Копирую в текущий каталог файл in_out.asm, тк он будет использоваться в последующих программах (рис. [4.3]).

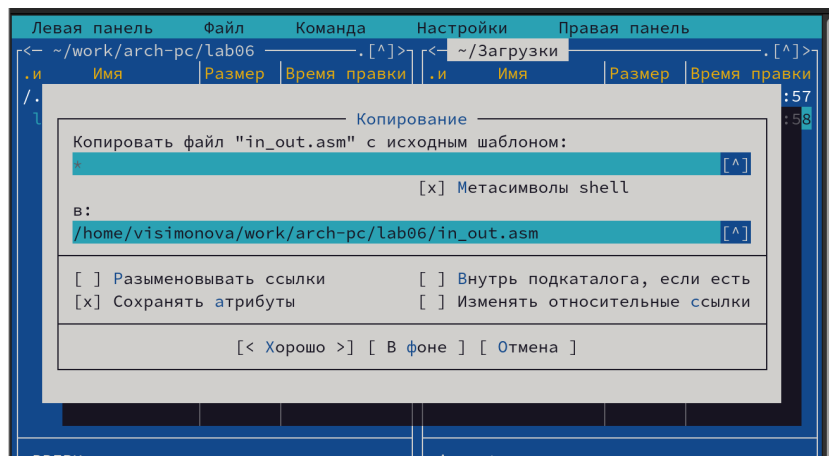


Рис. 4.3: Копирую файл in_out

В файл lab6-1.asm ввожу программу для вывода значений регистра eax (рис. [4.4]).

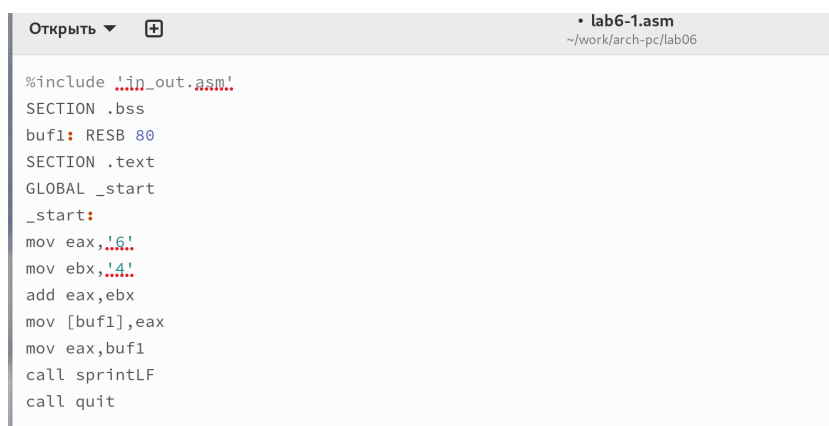


Рис. 4.4: Ввожу текст в файл

Создаю и запускаю исполняемый файл, который выводит символ j, тк сумма двоичных кодов 4 и 6 по системе ASCII соответствует символу j (рис. [4.5]).

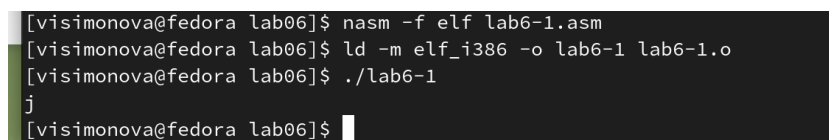
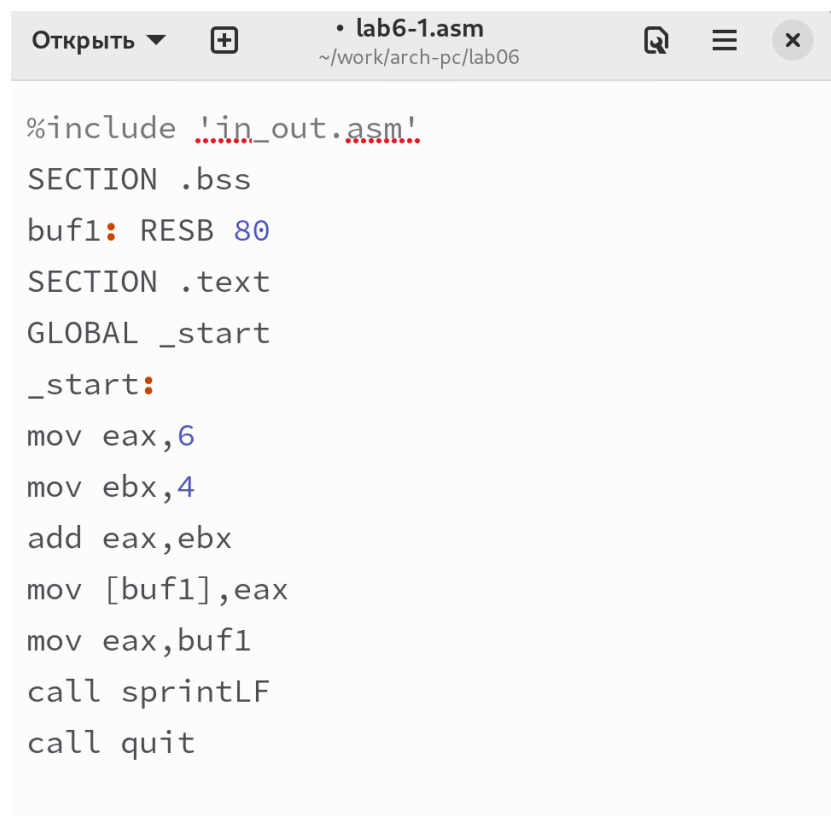


Рис. 4.5: Создаю и запускаю исполняемый файл

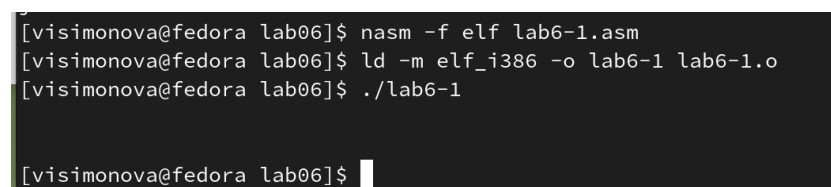
Изменяю '6' и '4' на цифры 6 и 4 (рис. [4.6]).



```
Открыть ▾ + • lab6-1.asm ~/work/arch-pc/lab06
%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. 4.6: Изменяю файл

Создаю и запускаю исполняемый файл. Теперь выводится символ с кодом 10, это символ перевода строки и он не отображается при выводе на экран (рис. [4.7]).



```
[visimonova@fedora lab06]$ nasm -f elf lab6-1.asm
[visimonova@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[visimonova@fedora lab06]$ ./lab6-1

[visimonova@fedora lab06]$
```

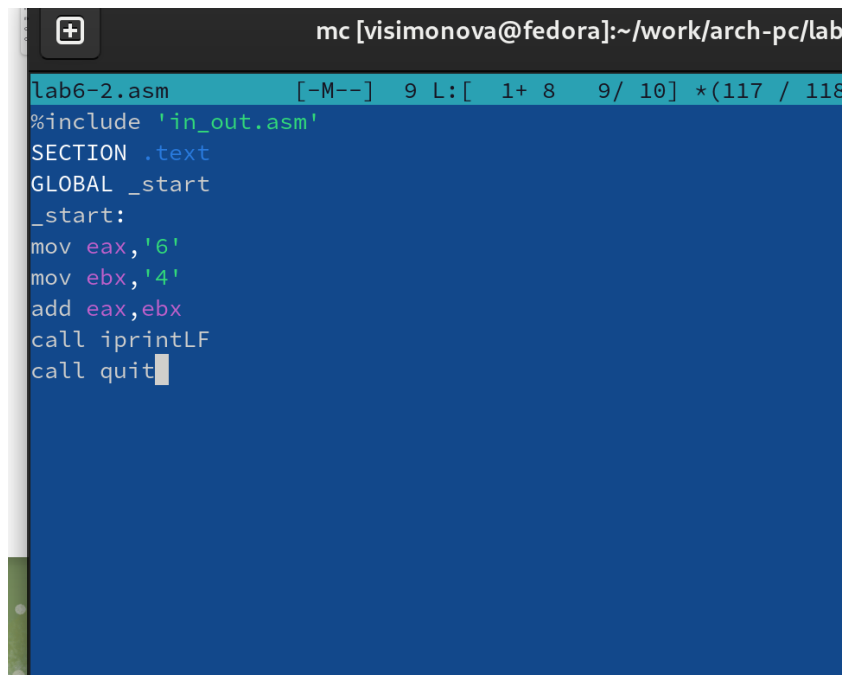
Рис. 4.7: Создаю и запускаю исполняемый файл

Создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 (рис. [4.8]).

```
[visimonova@fedora lab06]$ touch ~/work/arch-pc/lab06/lab6-2.asm
[visimonova@fedora lab06]$
```

Рис. 4.8: Создаю файл

Ввожу в него текст программы для вывода значения регистра ebx (рис. [4.9]).



```
mc [visimonova@fedora]:~/work/arch-pc/lab
lab6-2.asm [-M--] 9 L: [ 1+ 8 9/ 10] *(117 / 118
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 4.9: Ввожу код в файл

Создаю и запускаю исполняемый файл lab6-2, программа вывела число 106 , программа позволяет вывести именно число , а не символ, но мы всё ещё складываем коды (рис. [4.10]).

```
[visimonova@fedora lab06]$ nasm -f elf lab6-2.asm
[visimonova@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[visimonova@fedora lab06]$ ./lab6-2
106
```

Рис. 4.10: Создаю и запускаю исполняемый файл

Изменяю '6' и '4' на цифры 6 и 4 (рис. [4.11]).

```

lab6-2.asm      [-M--]  8 L:[ 1+ 4  5/ 10] *(66 / 114b) 0054 0x036
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
call iprintLF
call quit

```

Рис. 4.11: Исправляю

Создаю и запускаю новый исполняемый файл lab6-2, программа вывела число 10, тк программа складывает сами числа (рис. [4.12]).

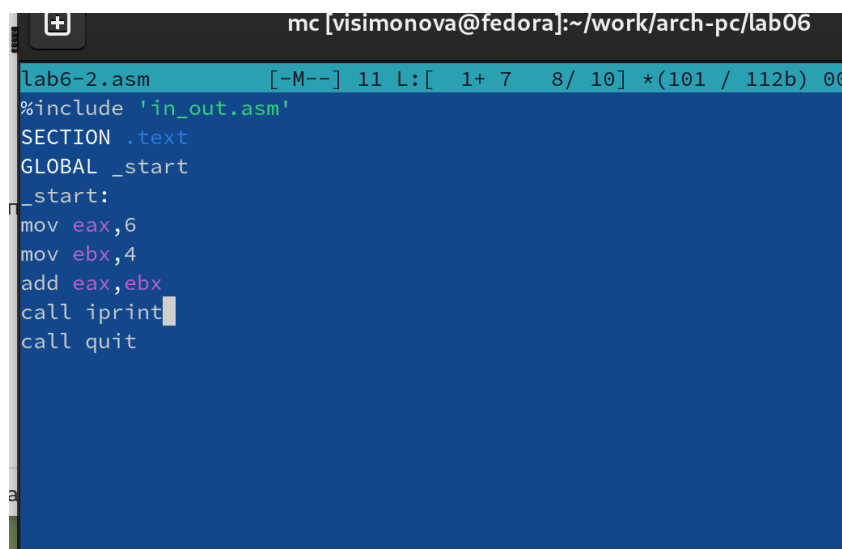
```

[visimonova@fedora lab06]$ nasm -f elf lab6-2.asm
[visimonova@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[visimonova@fedora lab06]$ ./lab6-2
10
[visimonova@fedora lab06]$

```

Рис. 4.12: Запуск исполняемого файла с имправленным кодом

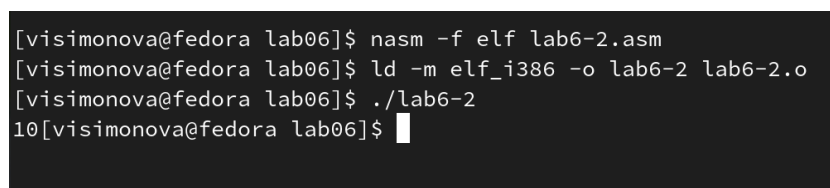
Заменяю функцию iprintLF на inprint (рис. [4.13]).



```
mc [visimonova@fedora]:~/work/arch-pc/lab06
lab6-2.asm      [-M--] 11 L:[ 1+ 7 8/ 10] *(101 / 112b) 00
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 4.13: Меняю на iprint

Создаю и запускаю новый исполняемый файл lab6-2, программа вывела число 10, вывод никак не отличается, тк символ переноса строки не отображался, а сейчас его просто нет (рис. [4.14]).

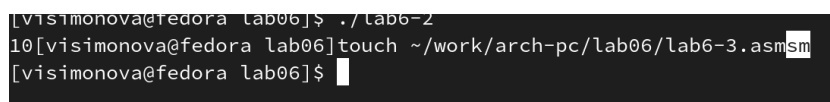


```
[visimonova@fedora lab06]$ nasm -f elf lab6-2.asm
[visimonova@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[visimonova@fedora lab06]$ ./lab6-2
10[visimonova@fedora lab06]$
```

Рис. 4.14: Компиляция и запуск файла.

##Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 (рис. [4.15]).

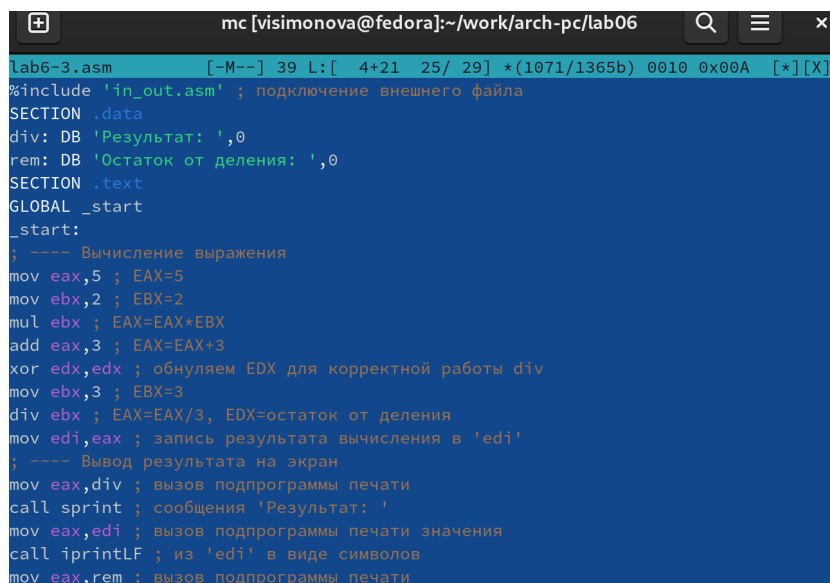


```
[visimonova@fedora lab06]$ ./lab6-2
10[visimonova@fedora lab06]$ touch ~/work/arch-pc/lab06/lab6-3.asm
[visimonova@fedora lab06]$
```

Рис. 4.15: Создаю файл

Ввожу в файл lab6-3.asm программу вычисления выражения $f(x) = (5 * 2 + 3)/3$

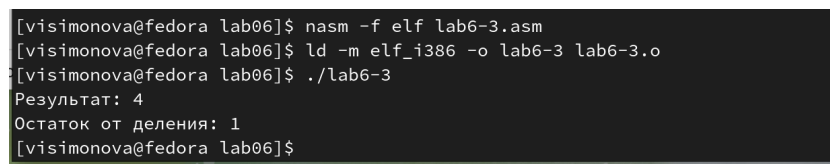
(рис. [4.16]).



```
lab6-3.asm [-M--] 39 L:[ 4+21 25/ 29] *(1071/1365b) 0010 0x00A [*][X]
#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. 4.16: Ввожу в файл текст программы

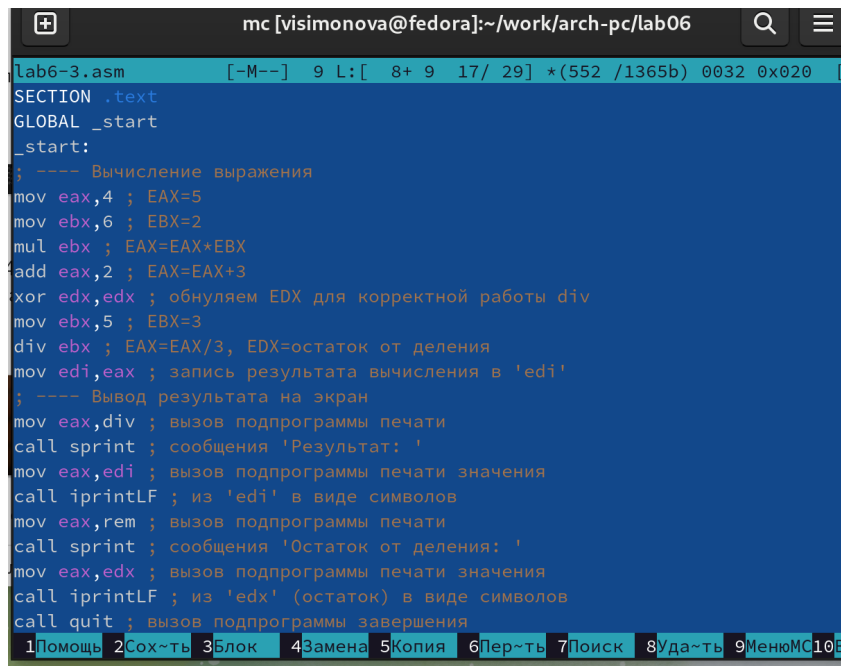
Создаю и запускаю исполняемый файл lab6-3, программа вывела верный результат (рис. [4.17]).



```
[visimonova@fedora lab06]$ nasm -f elf lab6-3.asm
[visimonova@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[visimonova@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1
[visimonova@fedora lab06]$
```

Рис. 4.17: Запуск исполняемого файла lab6-3

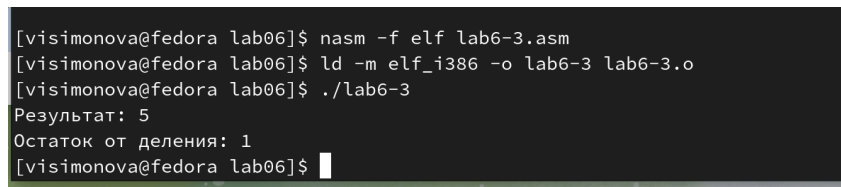
Изменяю программу так, чтобы она высчитывала значение для выражения $f(x) = (4 * 6 + 2)/5$ (рис. [4.18]).



```
lab6-3.asm [-M--] 9 L: [ 8+ 9 17/ 29] *(552 /1365b) 0032 0x020
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ; EAX=5
mov ebx,6 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; 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
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню 10
```

Рис. 4.18: Изменяю файл lab6-3

Создаю и запускаю новый исполняемый файл lab6-3, программа вывела верный результат (рис. [4.19]).



```
[visimonova@fedora lab06]$ nasm -f elf lab6-3.asm
[visimonova@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[visimonova@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
[visimonova@fedora lab06]$
```

Рис. 4.19: Запуск изменённого lab6-3

Создаю файл variant.asm в каталоге ~/work/arch-pc/lab06 и ввожу в него текст для вычисления варианта задания по номеру студенческого билета (рис. [4.20]).

```

variant.asm      [-M--]  9 L:[ 3+21 24/ 28] *(568 /
;-----
%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

```

Рис. 4.20: Ввожу в файл программу для вычисления номера задания

Создаю и запускаю исполняемый файл, ввожу номер своего студенческого билета, результат программы-14 вариант (рис. [4.21]).

```

[visimonova@fedora lab06]$ nasm -f elf variant.asm
[visimonova@fedora lab06]$ ld -m elf_i386 -o variant variant.o
[visimonova@fedora lab06]$ ./variant
Введите № студенческого билета:
1132236012
Ваш вариант: 13
[visimonova@fedora lab06]$

```

Рис. 4.21: Запуск программы для вычисления варианта

Ответы на вопросы

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант:'?

```

mov eax, rem
call sprintf

```


2. Для чего используются следующие инструкции?

```
mov ecx, x ;кладем адрес вводимой строки в регистр ecx  
mov edx, 80; запись длины вводимой строки в регистр edx  
call sread ; ввод программы из внешнего файла , вводим сообщение с клавиатуры
```

3. Для чего используется инструкция “call atoi”?

Вызов программы из внешнего файла , которая отвечает за преобразование ASCII кода.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

```
xor edx,edx; обнуление регистра edx  
mov ebx,20; edx=20  
div ebx; edx=edx/20  
inc edx;edx=edx+1
```

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

В регистр edx

6. Для чего используется инструкция “inc edx”?

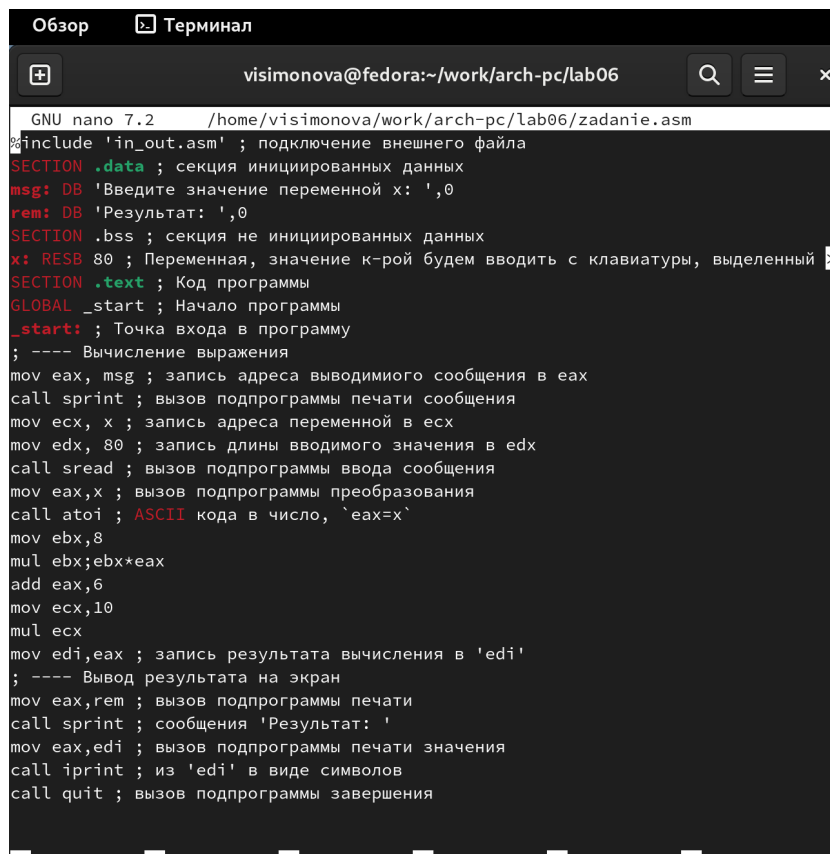
Это инкремент, команда inc ebx увеличивает значение регистра ebx на 1

7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычисления?

```
mov eax,edx  
call iprintLF
```

4.1 Задание для самостоятельной работы

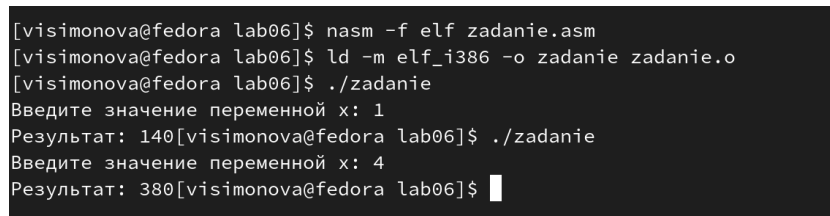
Создаю файл `zadanie.asm` и ввожу в него текст программы для вычисления выражения 13) $f(x) = (8x + 6) \cdot 10$ (рис. [4.22]).



```
GNU nano 7.2 /home/visimonova/work/arch-pc/lab06/zadanie.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 >
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'
mov ebx, 8
mul ebx, eax
add eax, 6
mov ecx, 10
mul ecx
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. 4.22: Мой код

Создаю и запускаю исполняемый файл, ввожу первое и второе значение переменной (рис. [4.23]).



```
[visimonova@fedora lab06]$ nasm -f elf zadanie.asm
[visimonova@fedora lab06]$ ld -m elf_i386 -o zadanie zadanie.o
[visimonova@fedora lab06]$ ./zadanie
Введите значение переменной x: 1
Результат: 140[visimonova@fedora lab06]$ ./zadanie
Введите значение переменной x: 4
Результат: 380[visimonova@fedora lab06]$
```

Рис. 4.23: Запуск

5 Выводы

Я освоила арифметические инструкции языка ассемблера NASM.

Список литературы

1. Лабораторная работа №7
2. Таблица ASCII