

# **Лабораторная работа №6**

**Дисциплина: Архитектура компьютеров**

Воронов Александр Валерьевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Символьные и численные данные в NASM . . . . .	9
4.2	Ответы на вопросы: . . . . .	16
4.3	Задание для самостоятельной работы . . . . .	16
<b>5</b>	<b>Выводы</b>	<b>18</b>
	<b>Список литературы</b>	<b>19</b>

## Список иллюстраций

4.1	Создание каталога и файла . . . . .	9
4.2	Запуск файла . . . . .	10
4.3	Запуск файла . . . . .	10
4.4	Запуск файла . . . . .	11
4.5	Запуск файла . . . . .	12
4.6	Запуск файла . . . . .	12
4.7	Ввод текста из листинга . . . . .	13
4.8	Запуск файла . . . . .	13
4.9	Запуск файла . . . . .	14
4.10	Ввод текста из листинга . . . . .	15
4.11	Запуск файла . . . . .	15
4.12	Ввод текста программы . . . . .	17
4.13	Запуск файла . . . . .	17

## **Список таблиц**

# 1 Цель работы

Целью данной лабораторной работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

1. Символьные и численные данные в NASM.
2. Выполнение арифметических операций в NASM.
3. Задание 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; • непосредственная адресация; • адресация памяти. 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. Допустимые сочетания операндов для команды `add` аналогичны сочетаниям операндов для команды `mov`. Так, например, команда `add eax,ebx` прибавит значение из регистра `eax` к значению из регистра `ebx` и запишет результат в регистр `eax`. Команда целочисленного вычитания `sub` (от англ. *subtraction* – вычитание) работает аналогично команде `add`. Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: `inc` (от англ. *increment*) и `dec` (от англ. *decrement*), которые увеличивают и уменьшают на 1 свой операнд. Операндом может быть регистр или ячейка памяти любого размера. Команды инкремента и декремента выгодны тем, что они занимают меньше места, чем соответствующие команды сложения и вычитания. Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют

различные команды. Для беззнакового умножения используется команда `mul`, для знакового умножения используется команда `imul`. 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. Для выполнения лабораторных работ в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Это:

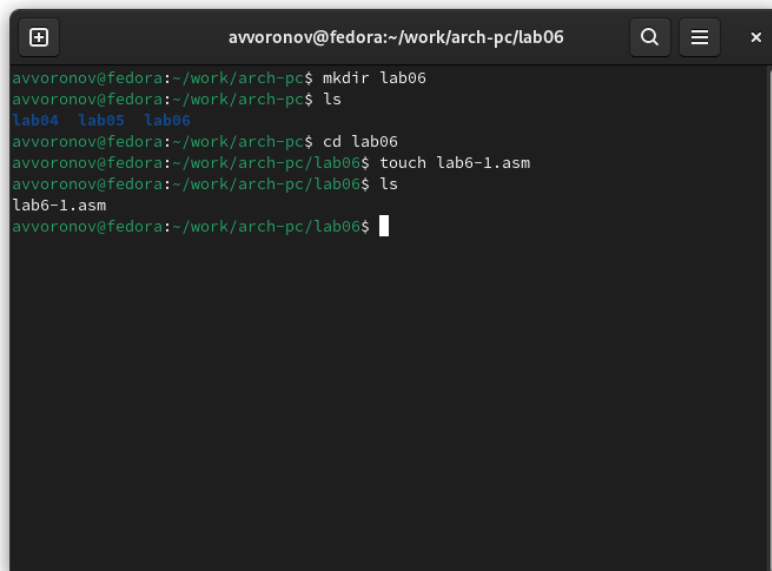
- `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число (`mov eax, ...`).
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки.
- `atoi` – функция преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`mov eax, ...`).



## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

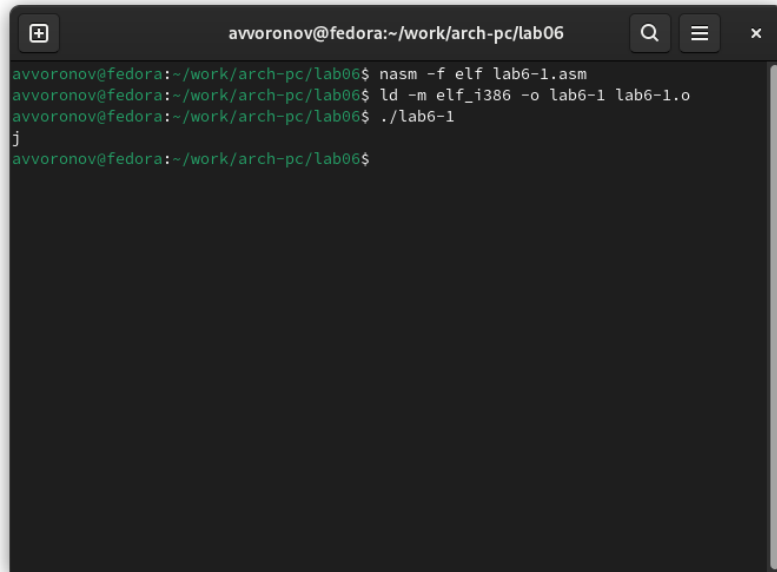
Сначала создаю каталог для программ лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm (рис. 4.1).



```
avvoronov@fedora:~/work/arch-pc/lab06
avvoronov@fedora:~/work/arch-pc$ mkdir lab06
avvoronov@fedora:~/work/arch-pc$ ls
lab04 lab05 lab06
avvoronov@fedora:~/work/arch-pc$ cd lab06
avvoronov@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
avvoronov@fedora:~/work/arch-pc/lab06$ ls
lab6-1.asm
avvoronov@fedora:~/work/arch-pc/lab06$
```

Рис. 4.1: Создание каталога и файла

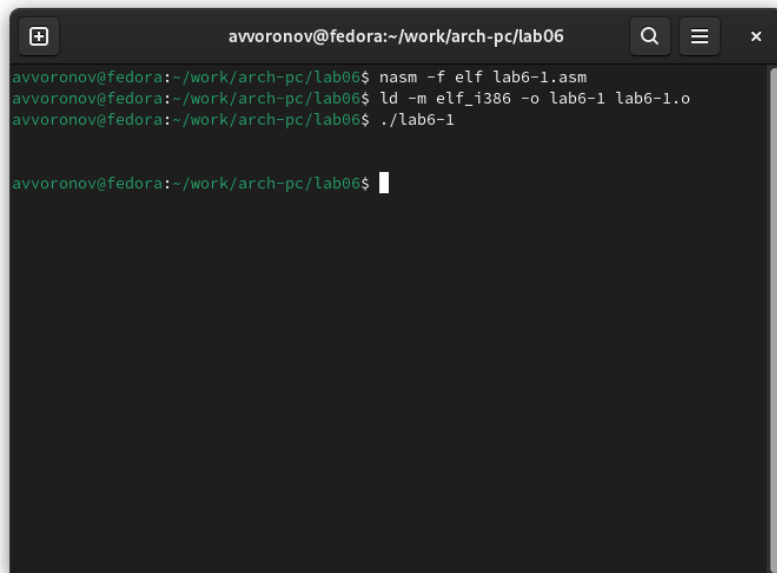
Ввожу в файл lab6-1.asm текст программы из листинга, далее создаю исполняемый файл и запускаю его (рис. 4.2).

A terminal window titled 'avvoronov@fedora:~/work/arch-pc/lab06' with search and menu icons. It shows the following commands and output:

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

Рис. 4.2: Запуск файла

Затем изменяю текст программы и вместо символов, записываю в регистры числа: '6', '4' заменяю на 4, 6. Создаю исполняемый файл и запускаю его (рис. 4.3). Это символ перевода строки, он не отображается при выводе на экран.

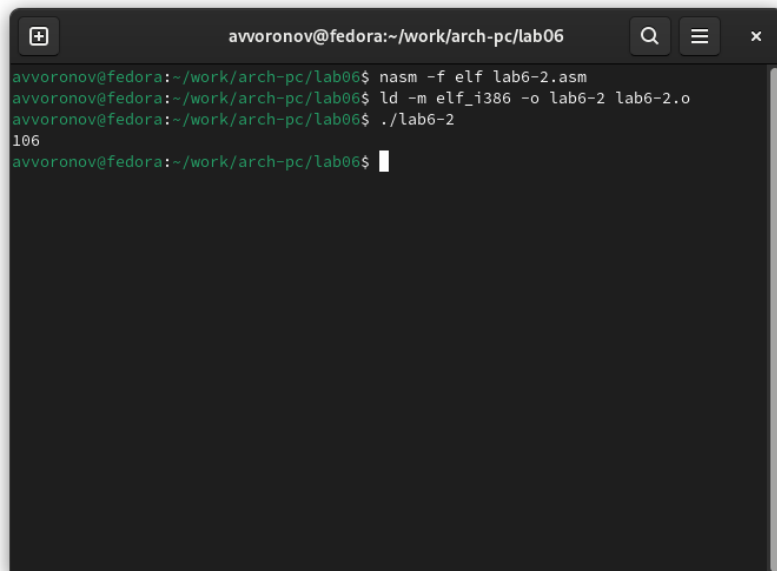
A terminal window titled 'avvoronov@fedora:~/work/arch-pc/lab06' with search and menu icons. It shows the same commands as in Figure 4.2, but the output is empty, indicating a newline character was printed:

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

avvoronov@fedora:~/work/arch-pc/lab06$
```

Рис. 4.3: Запуск файла

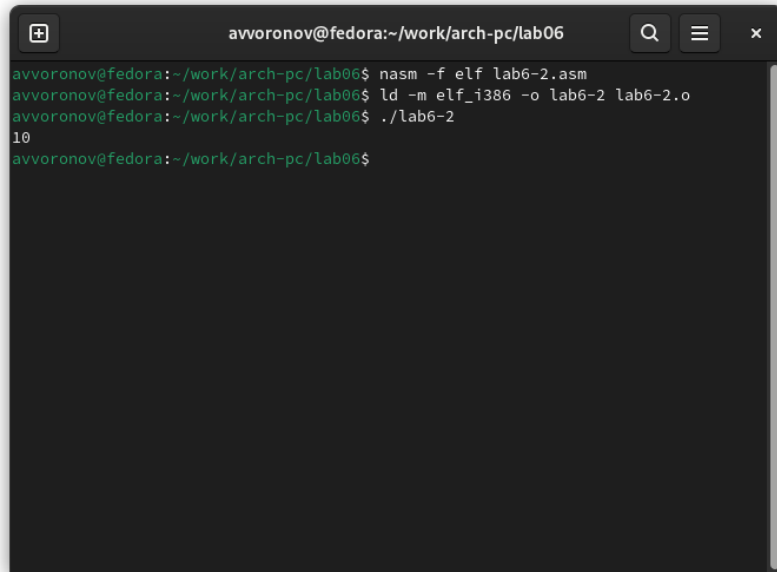
Далее создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввожу в него текст программы из листинга. Создаю исполняемый файл и запускаю его (рис. 4.4).

A terminal window titled 'avvoronov@fedora:~/work/arch-pc/lab06' with search, menu, and close icons. The terminal shows the following commands and output:

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

Рис. 4.4: Запуск файла

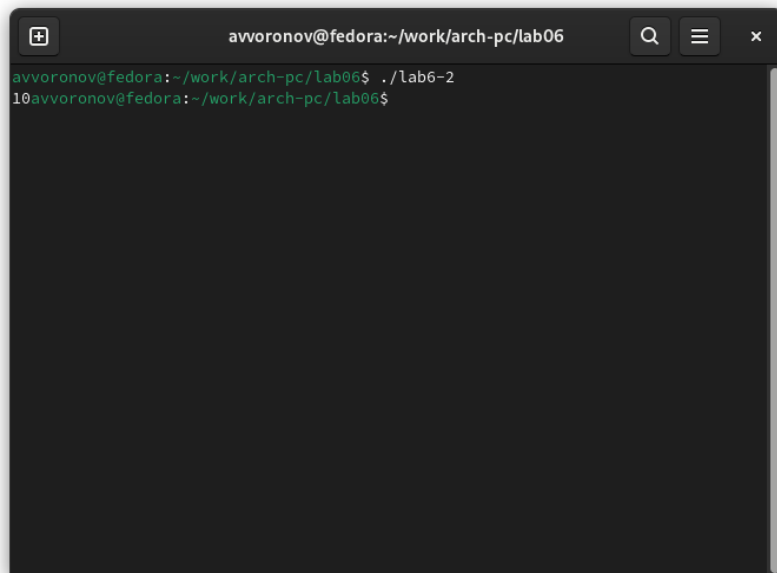
Аналогично предыдущему примеру изменяю символы на числа. Создаю исполняемый файл и запускаю его (рис. 4.5). Программа складывает числа 6 и 4, поэтому вывод 10.



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

Рис. 4.5: Запуск файла

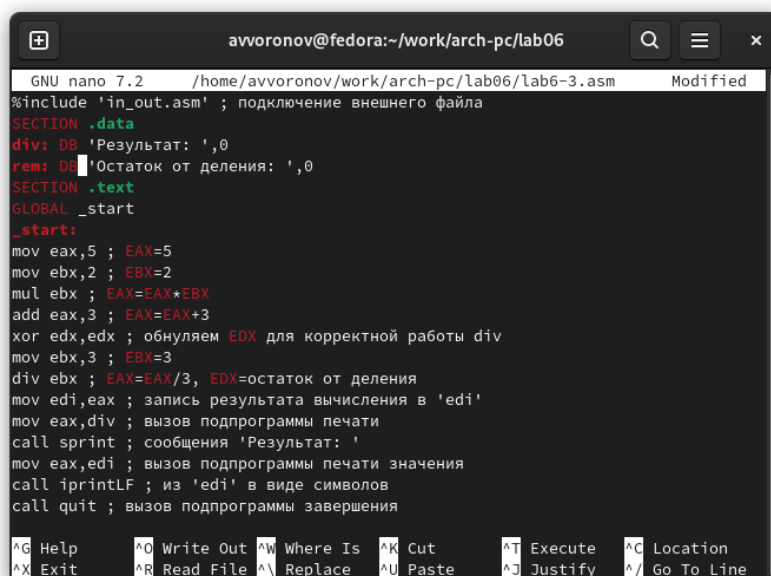
Заменяю функцию `iprintLF` на `iprint`, создаю исполняемый файл и запускаю его (рис. 4.6). Вывод функций отличается тем, что `iprint` не добавляет в выводе символ переноса строки.



```
avvoronov@fedora:~/work/arch-pc/lab06$ ./lab6-2
10avvoronov@fedora:~/work/arch-pc/lab06$
```

Рис. 4.6: Запуск файла

Затем создаю файл lab6-3.asm в каталоге ~/work/arch-pc/lab06. Ввожу текст программы из листинга 6.3 в lab6-3.asm (рис. 4.7).

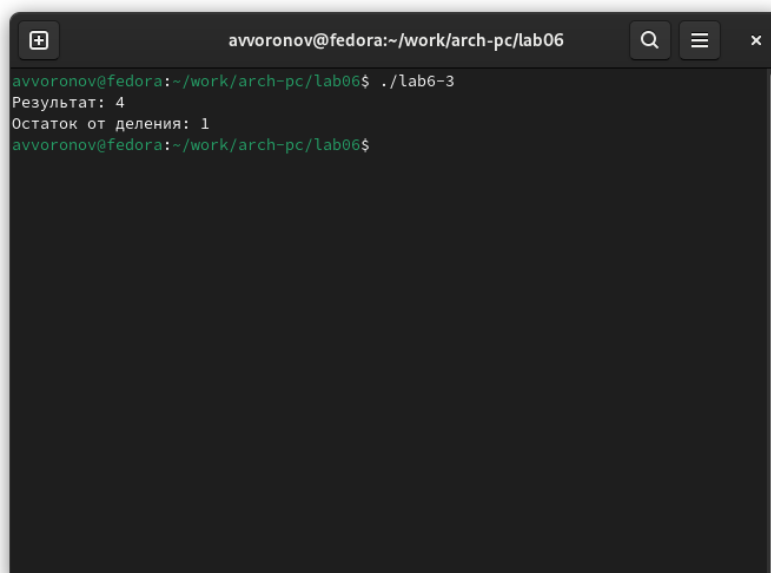


```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab06/lab6-3.asm Modified
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
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'
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 4.7: Ввод текста из листинга

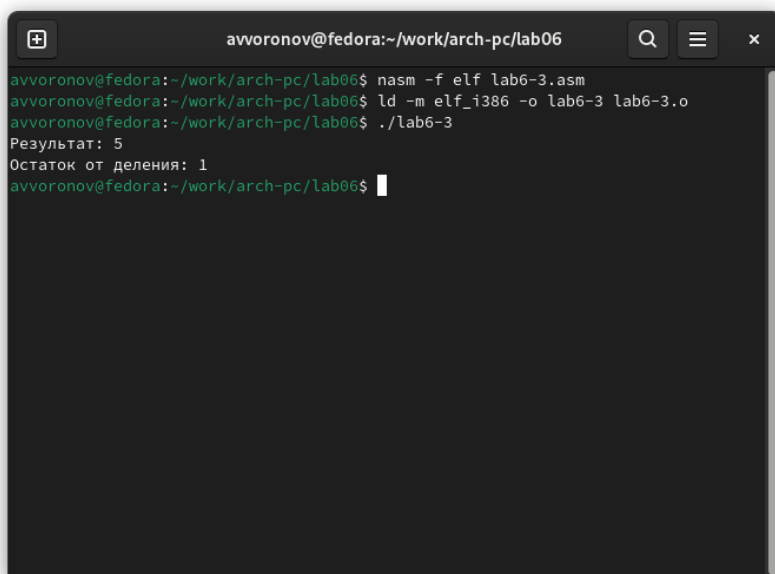
Создаю исполняемый файл и запускаю его (рис. 4.8).



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

Рис. 4.8: Запуск файла

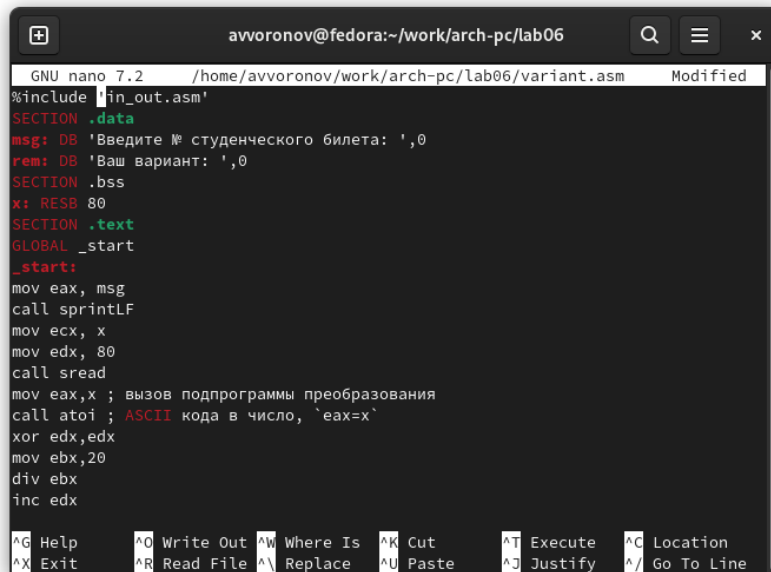
Изменяю текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$ . Создаю исполняемый файл и проверяю его работу (рис. 4.9).



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

Рис. 4.9: Запуск файла

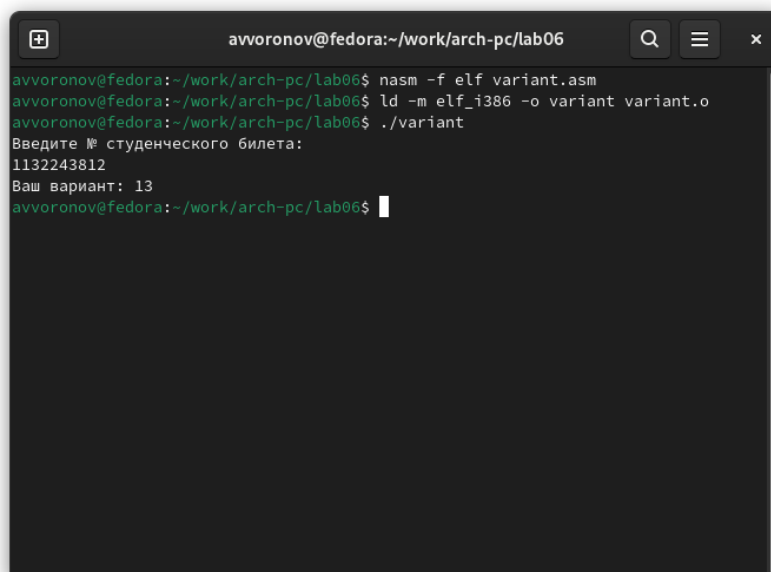
Далее делаю программу вычисления варианта задания для самостоятельной работы по номеру студенческого билета. Создаю файл `variant.asm` в каталоге `~/work/arch-pc/lab06`. Ввожу текст программы из листинга 6.4 в файл `variant.asm` (рис. 4.10).



```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab06/variant.asm Modified
#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 sprintfLF
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
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 4.10: Ввод текста из листинга

Создаю исполняемый файл и запускаю его (рис. 4.11).



```
avvoronov@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
avvoronov@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
avvoronov@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132243812
Ваш вариант: 13
avvoronov@fedora:~/work/arch-pc/lab06$
```

Рис. 4.11: Запуск файла

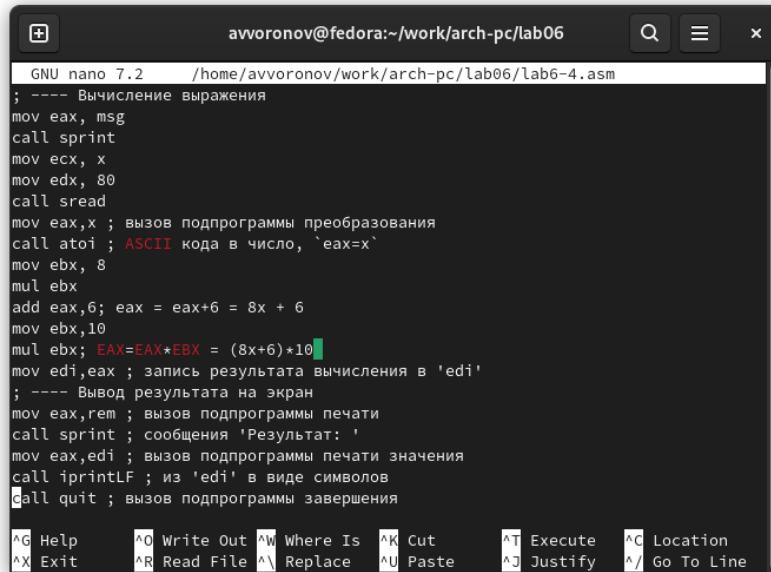
## 4.2 Ответы на вопросы:

1. За вывод на экран сообщения 'Ваш вариант:' отвечают следующие строки:  
`mov eax,rem call sprint`
2. `mov ecx, x` используется, чтобы положить адрес вводимой строки в регистр, `mov edx, 80` используется для записи в регистр длины вводимой строки, `call sread` вызывает подпрограмму из внешнего файла, чтобы вводить сообщения с клавиатуры.
3. Используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр.
4. За вычисление варианта отвечают следующие строки: `xor edx,edx mov ebx,20 div ebx inc edx`
5. В регистр `edx`.
6. `inc edx` используется для увеличения значения регистра `edx` на 1.
7. За вывод на экран результатов вычислений отвечают следующие строки:  
`mov eax,edx call iprintLF`

## 4.3 Задание для самостоятельной работы

Сначала создаю файл `lab6-4.asm` в каталоге `~/work/arch-рс/lab06`. Далее ввожу в файл текст программы для вычисления значения выражения  $(8x + 6) \cdot 10$  (вариант 13) (рис. 4.12).



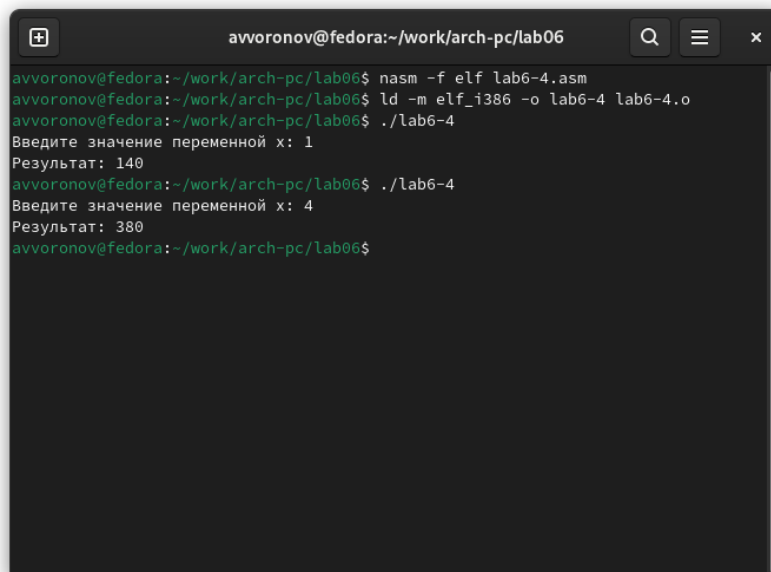


```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab06/lab6-4.asm
; ---- Вычисление выражения
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
mov ebx, 8
mul ebx
add eax, 6; eax = eax+6 = 8x + 6
mov ebx, 10
mul ebx; EAX=EAX*EBX = (8x+6)*10
mov edi, eax ; запись результата вычисления в `edi`
; ---- Вывод результата на экран
mov eax, ret ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprintLF ; из `edi` в виде символов
call quit ; вызов подпрограммы завершения

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^L Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 4.12: Ввод текста программы

Создаю и запускаю исполняемый файл. При вводе  $x1=1$ , вывод - 140. При вводе  $x2=4$ , вывод - 380 (рис. 4.13).



```
avvoronov@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
avvoronov@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
avvoronov@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 140
avvoronov@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 4
Результат: 380
avvoronov@fedora:~/work/arch-pc/lab06$
```

Рис. 4.13: Запуск файла

## **5 Выводы**

В ходе данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

## **Список литературы**