

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: Трофимов Владислав Алексеевич

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

Список иллюстраций	3
1 Цель работы	4
2 Задание.....	5
3 Теоретическое введение	6
4 Выполнение лабораторной работы	8
4.1 Символьные и численные данные в NASM	8
4.2 Выполнение арифметических операций в NASM	12
4.3 Ответы на контрольные вопросы	15
4.4 Задание для самостоятельной работы.....	16
5 Выводы	17
6 Список литературы	18

Список иллюстраций

Рис. 4.1: Создание нового каталога

Рис. 4.2: Сохранение новой программы

Рис. 4.3: Запуск изначальной программы

Рис. 4.4: Измененная программа

Рис. 4.5: Запуск измененной программы

Рис. 4.6: Вторая программа

Рис. 4.7: Вывод второй программы

Рис. 4.8: Вывод измененной второй программы

Рис. 4.9: Замена функции вывода во второй программе

Рис. 4.10: Третья программа

Рис. 4.11: Запуск третьей программы

Рис. 4.12: Изменение третьей программы

Рис. 4.13: Запуск измененной третьей программы

Рис. 4.14: Программа для подсчета варианта

Рис. 4.15: Запуск программы для подсчета варианта

Рис. 4.16: Запуск и проверка программы

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.

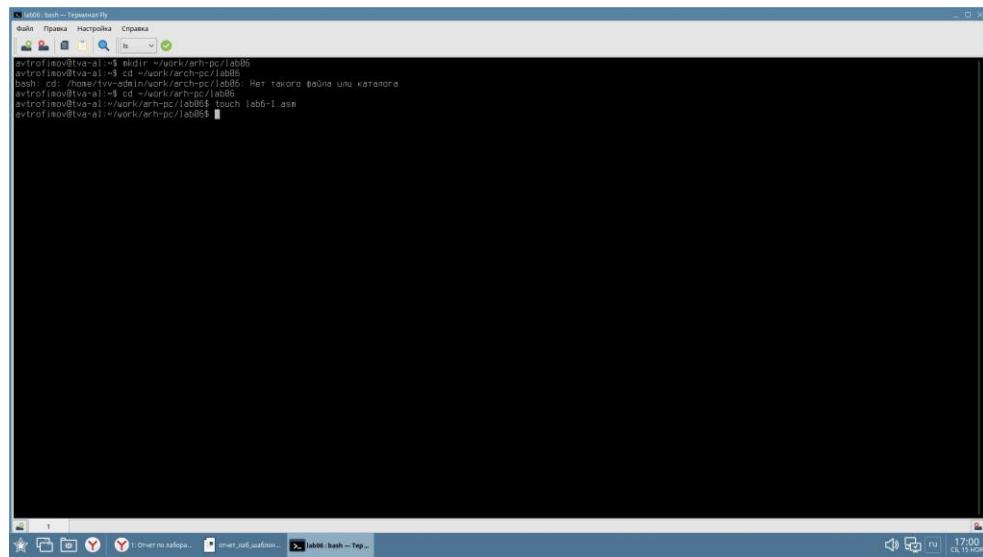
- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`.
- Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`.
- Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию. Ввод информации с клавиатуры и вывод её на экран осуществляется в 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

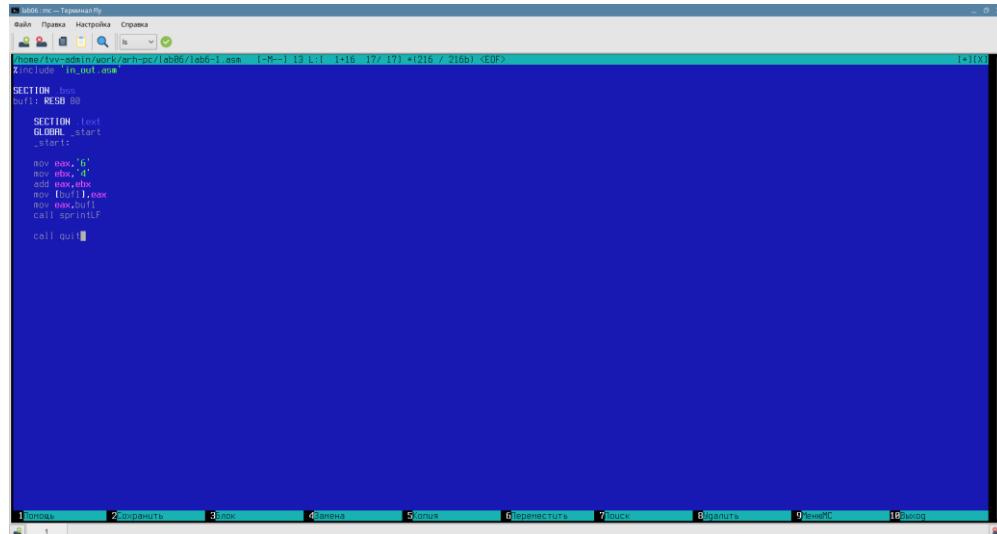
Создаю каталог для программ лабораторной работы №6 и перехожу в него, создаю там файл (рис. 4.1).



```
lab06:bash - Терминал Ry
Файл Правка Настройка Справка
ls
avtrolimov@vda-ai:~$ mkdir ~/work/arth-pc/lab06
avtrolimov@vda-ai:~/work/arth-pc$ cd ~/work/arth-pc/lab06
bash: cd: ~/work/arth-pc/lab06: Нет такого файла или каталога
avtrolimov@vda-ai:~/work/arth-pc$ cd ~/work/arth-pc/lab06
avtrolimov@vda-ai:~/work/arth-pc$ touch lab06-1.asm
avtrolimov@vda-ai:~/work/arth-pc$ lab06-1
```

Рис. 4.1: Создание нового каталога

В созданном файле ввожу программу из листинга (рис. 4.2).



```
lab06:mc -- Терминал Ry
Файл Правка Настройка Справка
/home/tvv-edm@vda-ai:~/work/arth-pc/lab06/lab06-1.asm (-h--) 13 L:( 1+16 )?/ 17) +*(216 / 216b) <EOF>
Z:include in.out asm
SECTION test
GLOBAL start
start:
    mov dx, 'A'
    mov ah, '4'
    add ax, ah
    mov bx, al
    mov bl, ah
    mov bh, al
    call printLF
    call quit
```

Рис. 4.2: Сохранение новой программы

Создаю исполняемый файл и запускаю его, вывод программы отличается от предполагаемого изначально, ибо коды символов в сумме дают символ j по таблице ASCII. (рис. 4.3)

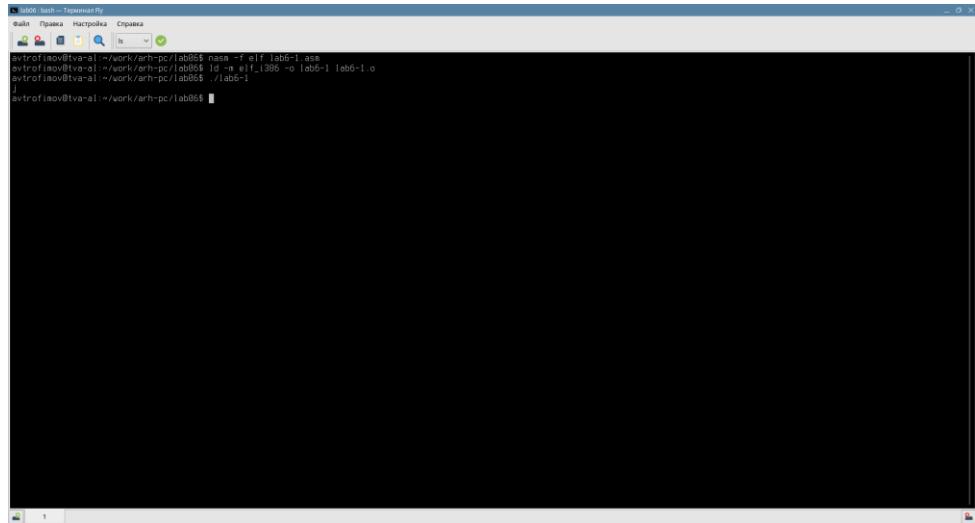


Рис. 4.3: Запуск изначальной программы

Изменяю текст изначальной программы, убрав кавычки (рис. 4.4).

```
SECTION .bss  
        RESB 80  
  
SECTION .text  
GLOBAL _start  
  
.start:  
    mov eax, 2  
    mov ebx, 4  
    add eax,ebx  
    mov [eax], al  
    mov eax,0x4f  
    call sprintf  
  
call quit
```

A screenshot of a terminal window titled "lab06 mc - Терминал Pu". The window has a menu bar with "Файл", "Правка", "Настройка", and "Справка". Below the menu is a toolbar with icons for file operations. The main area of the terminal shows the assembly code for the modified program. The code defines a .bss section with 80 bytes of memory, a .text section starting at _start, and a series of assembly instructions including moves, adds, and prints. The assembly code ends with a quit call.

Рис. 4.4: Измененная программа

На этот раз программа выдала пустую строчку, это связано с тем, что символ 10 означает переход на новую строку (рис. 4.5).

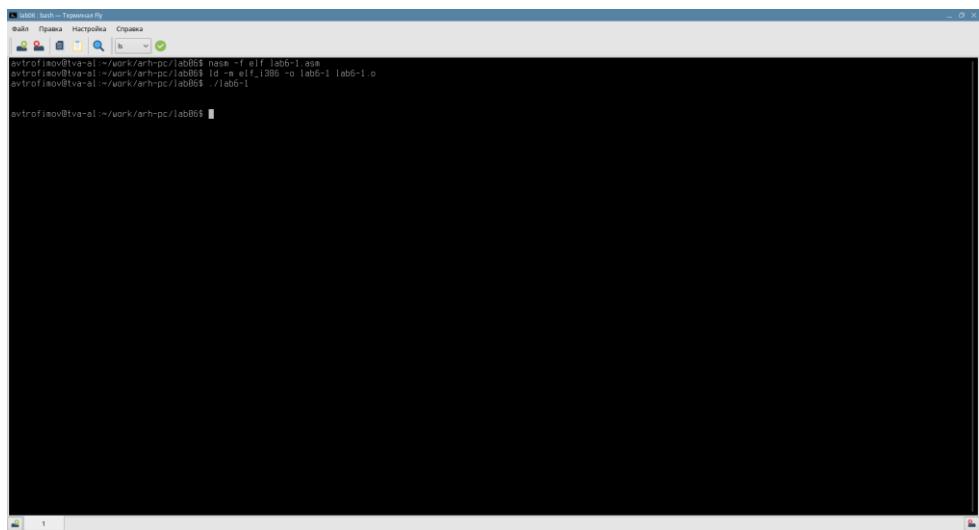


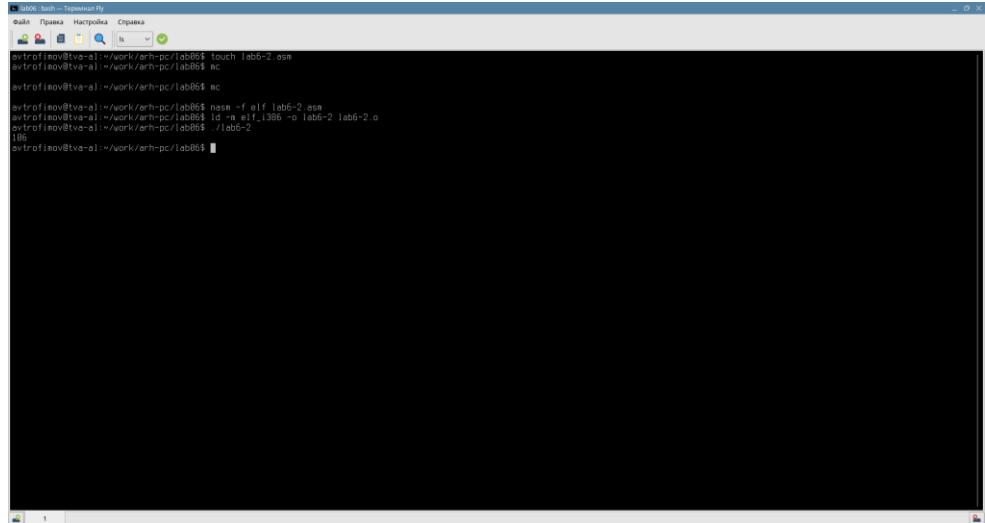
Рис. 4.5: Запуск измененной программы

Создаю новый файл для будущей программы и записываю в нее код из листинга (рис. 4.6).

```
SECTION .text
GLOBAL _start
_start:
    mov eax, '6'
    mov ebx, '4'
    add eax,ebx
    call _printfLF
    call quit
```

Рис. 4.6: Вторая программа

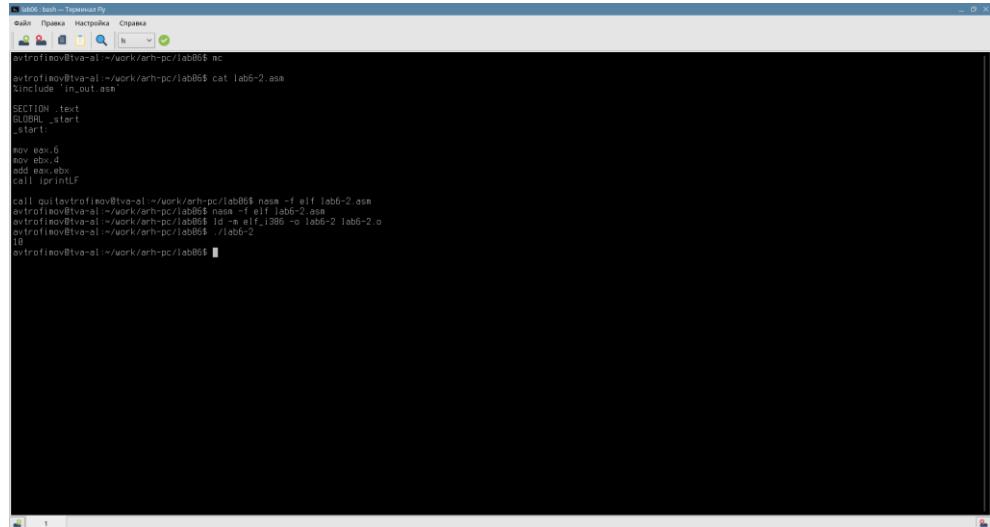
Создаю исполняемый файл и запускаю его, теперь отображается результат 106, программа, как и в первый раз, сложила коды символов, но вывела само число, а не его символ, благодаря замене функции вывода на iprintfLF (рис. 4.7).



```
autrofimov@tva-al:~/work/arm-pc/lab06$ touch lab6-2.asm
autrofimov@tva-al:~/work/arm-pc/lab06$ nc
autrofimov@tva-al:~/work/arm-pc/lab06$ nc
autrofimov@tva-al:~/work/arm-pc/lab06$ nasm -f elf lab6-2.asm
autrofimov@tva-al:~/work/arm-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
autrofimov@tva-al:~/work/arm-pc/lab06$ ./lab6-2
18
```

Рис. 4.7: Вывод второй программы

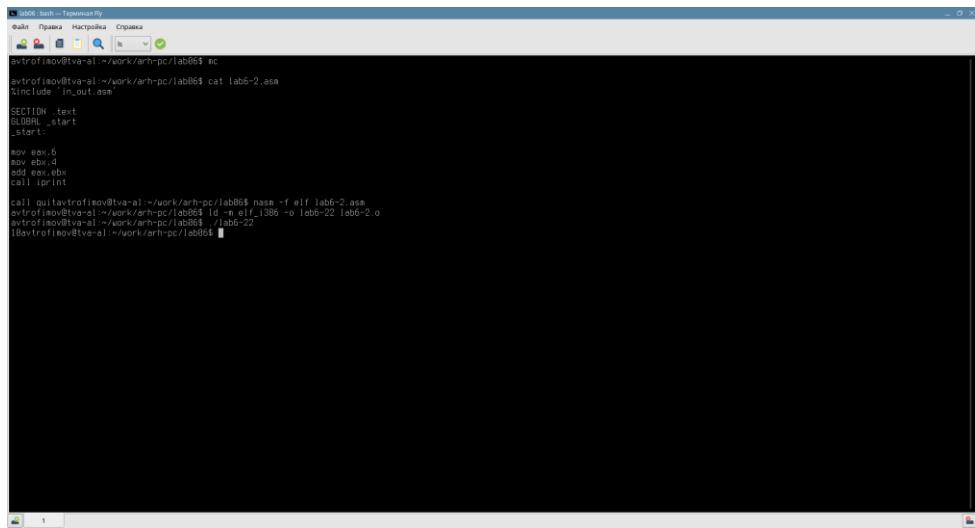
Убрав кавычки в программе, я снова ее запускаю и получаю предполагаемый изначально результат. (рис. 4.8).



```
autrofimov@tva-al:~/work/arm-pc/lab06$ nc
autrofimov@tva-al:~/work/arm-pc/lab06$ cat lab6-2.asm
include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
    mov eax,6
    mov al,4
    add eax,ebx
    call iprintf
call quit
autrofimov@tva-al:~/work/arm-pc/lab06$ nasm -f elf lab6-2.asm
autrofimov@tva-al:~/work/arm-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
autrofimov@tva-al:~/work/arm-pc/lab06$ ./lab6-2
18
```

Рис. 4.8: Вывод измененной второй программы

Заменив функцию вывода на iprintf, я получаю тот же результат, но без переноса строки (рис. 4.9).

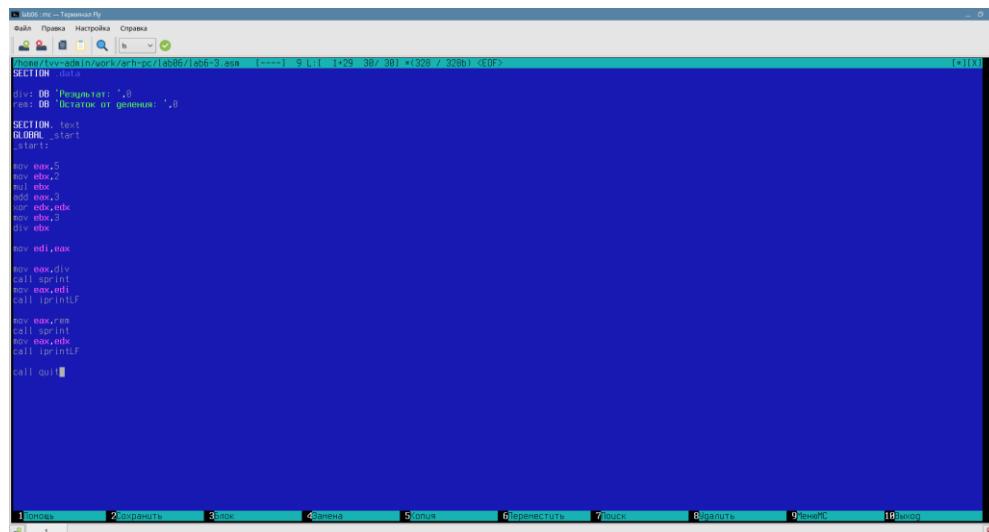


```
lab6: bash -- Терминал Ry
Файл Правка Настройка Справка
avtrofimov@vta-al:~/work/arm-pc/lab6$ nc
avtrofimov@vta-al:~/work/arm-pc/lab6$ cat lab6-2.asm
SECTION .text
GLOBAL _start
_start:
    mov eax,6
    mov ebx,4
    add ebx,ebx
    call print
    call quit
avtrofimov@vta-al:~/work/arm-pc/lab6$ nasm -f elf lab6-2.asm
avtrofimov@vta-al:~/work/arm-pc/lab6$ ld -n elf_i386 -o lab6-22 lab6-2.o
avtrofimov@vta-al:~/work/arm-pc/lab6$ ./lab6-22
16avtrofimov@vta-al:~/work/arm-pc/lab6$
```

Рис. 4.9: Замена функции вывода во второй программе

4.2 Выполнение арифметических операций в NASM

Создаю новый файл и копирую в него содержимое листинга (рис. 4.10).



```
lab6: mc -- Терминал Ry
Файл Правка Настройка Справка
avtrofimov@vta-al:~/work/arm-pc/lab6$ cat lab6-3.asm
div: 0B Текущий: ',0
rest: 0B Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
    mov eax,5
    neg eax,2
    mul ebx
    add eax,3
    xor edx,edx
    mov ebx,3
    div ebx
    mov edi,eax
    mov eax,div
    call sprint
    mov eax,edi
    call sprintLF
    mov eax,rest
    call sprint
    mov eax,edx
    call sprintLF
    call quit
```

Рис. 4.10: Третья программа

Программа выполняет арифметические вычисления, на вывод идет результирующее выражения и его остаток от деления (рис. 4.11).

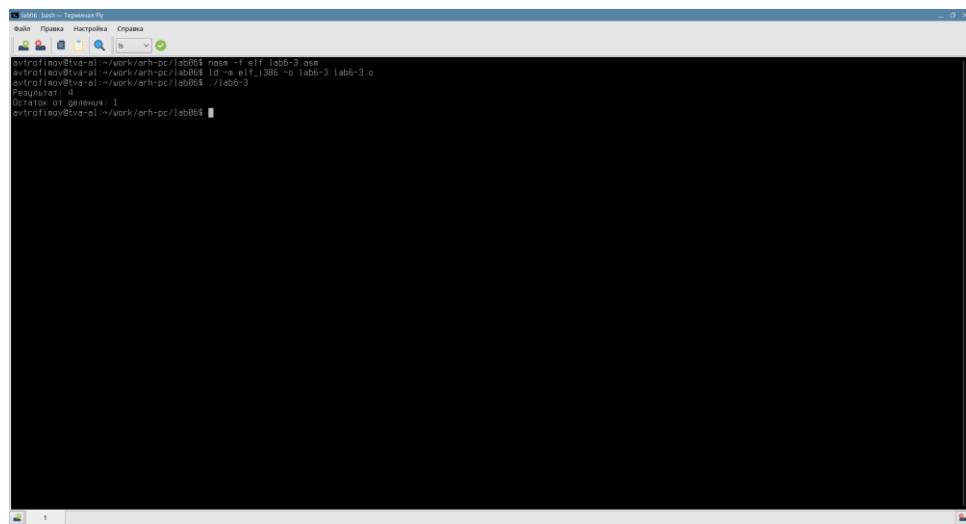


Рис. 4.11: Запуск третьей программы

Заменив переменные в программе для выражения $f(x) = (4*6+2)/5$ (рис. 4.12).

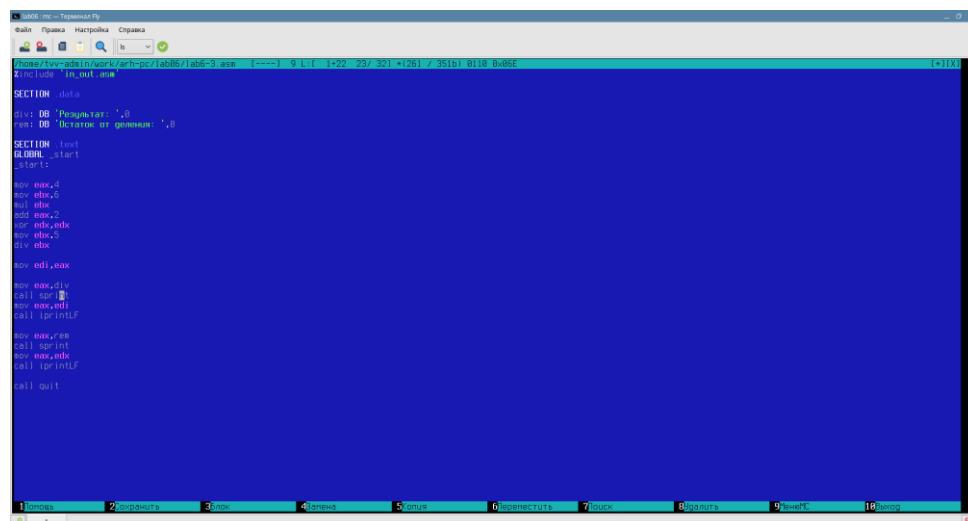


Рис. 4.12: Изменение третьей программы

Запуск программы дает корректный результат (рис. 4.13).

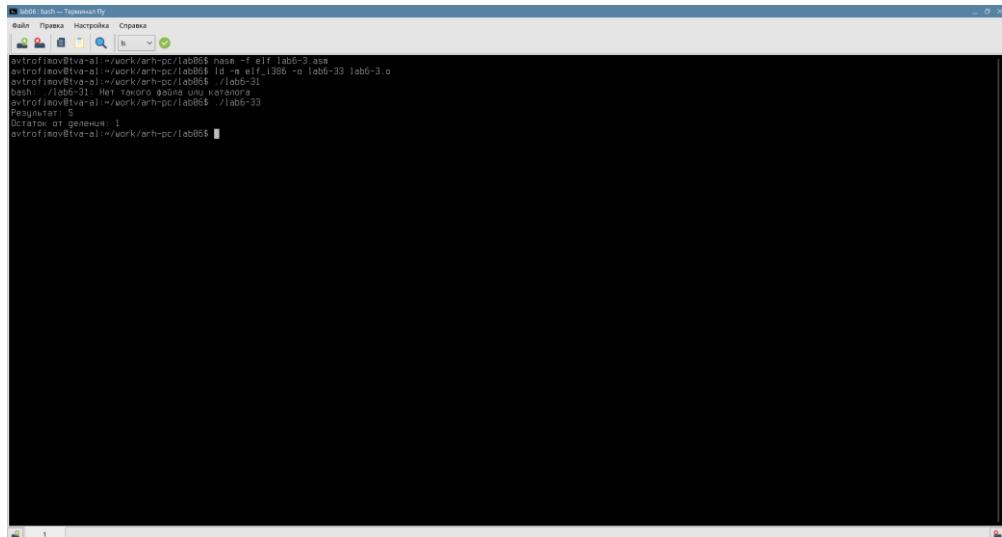


Рис. 4.13: Запуск измененной третьей программы

Создаю новый файл и помещаю текст из листинга (рис. 4.14).

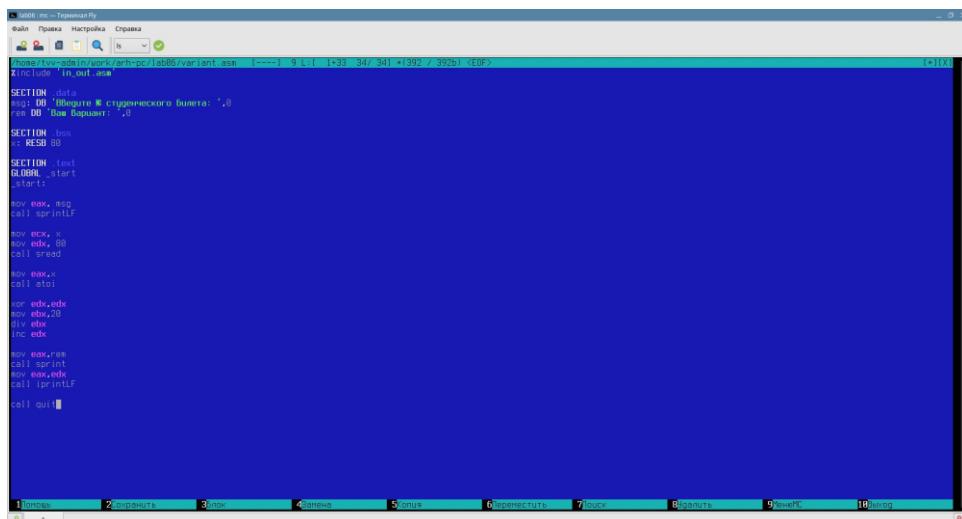


Рис. 4.14: Программа для подсчета варианта

Запустив программу и указав свой номер студенческого билета, я получил свой вариант для дальнейшей работы. (рис. 4.15).

```
lab06 bash - Терминал
Файл Правка Настройка Справка
wvtrofimov@tva-al:~/work/arh-pc/lab06$ touch variant.asm
wvtrofimov@tva-al:~/work/arh-pc/lab06$ nasm -f elf variant.asm
wvtrofimov@tva-al:~/work/arh-pc/lab06$ ld -n elf_i386 -o variant variant.o
wvtrofimov@tva-al:~/work/arh-pc/lab06$ ./variant
163225351
Ваш Вариант: 12
wvtrofimov@tva-al:~/work/arh-pc/lab06$
```

Рис. 4.15: Запуск программы для подсчета варианта

4.3 Ответы на контрольные вопросы

1. 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  
mov ebx,20  
div ebx  
inc edx
```

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

4.4 Задание для самостоятельной работы

В соответствии с выбранным вариантом, я реализую программу для подсчета функции $f(x) = 10 + (31x - 5)$, проверка на нескольких переменных показывает корректное выполнение программы (рис. 4.16).

Ubuntu 16.04 LTS - Terminal

Файл Правка Настройка Справка

```
avtrolimov@vta-al:~/work/arh-pc/lab06$ nasm -f elf lab6-4.asm
avtrolimov@vta-al:~/work/arh-pc/lab06$ ld -n elf_i386 -o lab6-4 lab6-4.o
avtrolimov@vta-al:~/work/arh-pc/lab06$ ./lab6-4
Registers: 59
Регистры: 59
avtrolimov@vta-al:~/work/arh-pc/lab06$ ./lab6-4
Результат: 59
Выведите результат переменной x: 1
Результат: 59
avtrolimov@vta-al:~/work/arh-pc/lab06$ cat lab6-4.asm
#include <in_out.h>
```

SECTION .data

```
msg: DB 'Выведите результат переменной x: ',0
res: DB 'Результат: ',0
```

SECTION .bss

```
x: RESB 80
```

SECTION .text

```
GLOBAL _start
_start:
```

```
    mov eax, msg
    call sprint
    mov ecx, x
    mov edx, 80
    call scread

    mov ebx, x
    call atoi
    mov ebx, 31
    mul ebx
    add ebx, 5
    add ebx, 10
    mov edi, ebx
    mov ebx, res
    call sprint

    mov eax, edi
    call print
    call quite
```

```
avtrolimov@vta-al:~/work/arh-pc/lab06$
```

Рис. 4.16: Запуск и проверка программы

5 Выводы

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

6 Список литературы

Курс ТУИС

Лабораторная работа №6

Программирование на языке ассемблера NASM Столяров А. В.