

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

Лабораторная работа №7. Арифметические операции в NASM.

Кудряшов Артём Николаевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Символьные и численные данные в NASM	7
3.2	Выполнение арифметических операций в NASM	10
3.2.1	Ответы на вопросы	14
3.3	Задание для самостоятельной работы	15
4	Выводы	17

Список иллюстраций

3.1	Новый каталог, файл lab7-1.asm	7
3.2	Текст программы из листинга 7.1	8
3.3	Первая версия программы – вывод ‘j’	8
3.4	Вторая версия программы – вывод перевода строки	9
3.5	Текст программы lab7-2.asm	9
3.6	Третья версия программы – вывод числа 106	9
3.7	Четвёртая версия программы – вывод числа 10	10
3.8	Та же программа, но без перевода строки в конце	10
3.9	Текст программы lab7-3.asm	11
3.10	Результат выполнения программы lab7-3	11
3.11	Текст отредактированной программы для $f(x) = (4*6 + 2)/5$	12
3.12	Запуск отредактированного файла lab7-3	12
3.13	Текст программы variant.asm	13
3.14	Результат работы variant.asm	13
3.15	Текст программы для самостоятельной работы	15
3.16	Запуск файла программы для самостоятельной работы	16

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

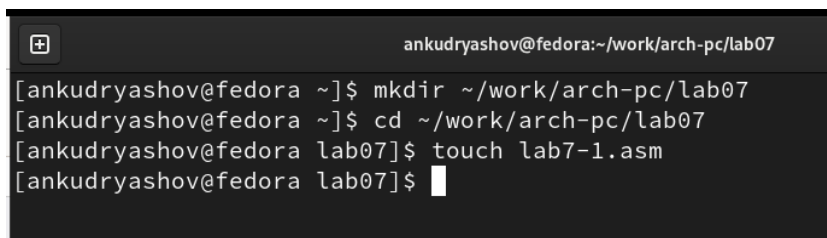
2 Задание

Для выполнения данной лабораторной работы необходимо освоить принцип работы с арифметическими инструкциями языка ассемблера NASM. Для этого следует разобрать различные операторы, используемые для сложения, вычитания, умножения, деления, а также перевода символьного типа в числовой.

3 Выполнение лабораторной работы

3.1 Символьные и численные данные в NASM

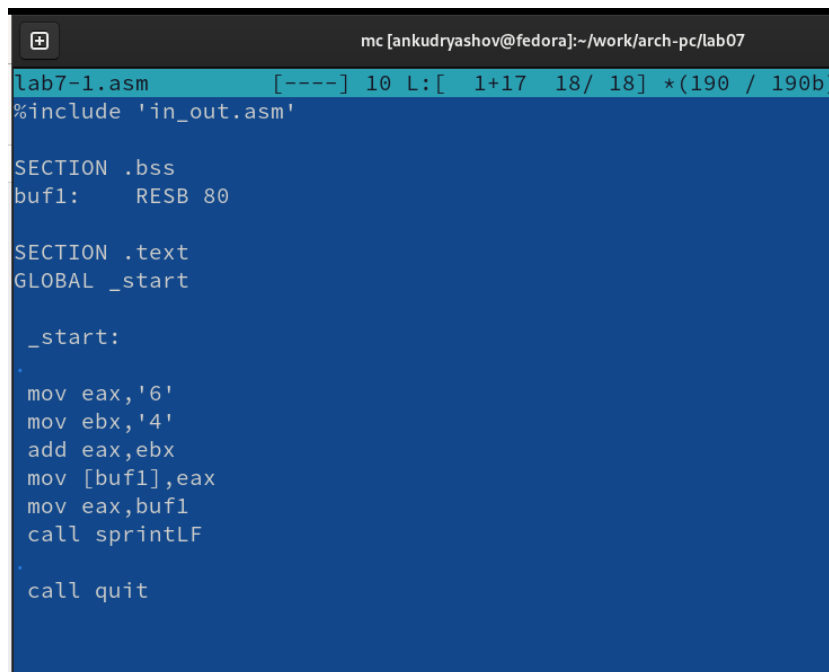
Откроем терминал, создадим каталог для программ в папке arch-pc и с помощью команды touch создадим первый файл lab7-1.asm (рис. 3.1).

A screenshot of a terminal window with a dark background. The title bar at the top shows a window icon and the text 'ankudryashov@fedora:~/work/arch-pc/lab07'. The terminal contains four lines of text: the first line shows the user at the root prompt creating a directory; the second line shows the user changing to the newly created directory; the third line shows the user creating a file named 'lab7-1.asm'; the fourth line shows the prompt after the file has been created.

```
[ankudryashov@fedora ~]$ mkdir ~/work/arch-pc/lab07
[ankudryashov@fedora ~]$ cd ~/work/arch-pc/lab07
[ankudryashov@fedora lab07]$ touch lab7-1.asm
[ankudryashov@fedora lab07]$
```

Рис. 3.1: Новый каталог, файл lab7-1.asm

Введём в файл lab7-1.asm текст программы из листинга 7.1 (рис. 3.2).



```
mc [ankudryashov@fedora]:~/work/arch-pc/lab07
lab7-1.asm [----] 10 L:[ 1+17 18/ 18] *(190 / 190b)
#include 'in_out.asm'

SECTION .bss
buf1:    RESB 80

SECTION .text
GLOBAL _start

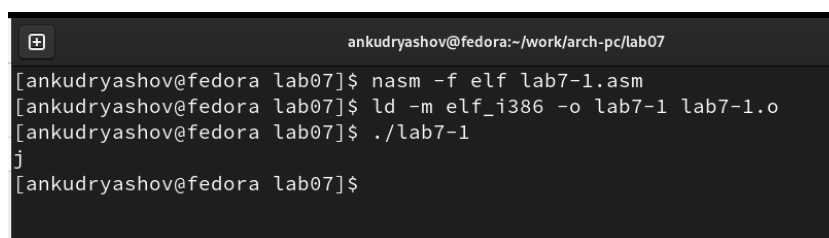
_start:

mov eax,'6'
mov ebx,'4'
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintf

call quit
```

Рис. 3.2: Текст программы из листинга 7.1

Создадим исполняемый файл и запустим его. Как мы видим, программа вывела на экран символ 'j' (рис. 3.3).



```
ankudryashov@fedora:~/work/arch-pc/lab07
[ankudryashov@fedora lab07]$ nasm -f elf lab7-1.asm
[ankudryashov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[ankudryashov@fedora lab07]$ ./lab7-1
j
[ankudryashov@fedora lab07]$
```

Рис. 3.3: Первая версия программы – вывод 'j'

Немного изменим текст программы, записав в регистры числа, а не символы (поменяем `mov eax,'6'` на `mov eax,6` и т.д.). Создадим исполняемый файл и запустим его. В этот раз на экран вывелся символ LF, т.е. перевод строки, потому что его номер в таблице символов ASCII равен 10 (т.к 6+4) (рис. 3.4).


```
ankudryashov@fedora:~/work/arch-pc/lab07
[ankudryashov@fedora lab07]$ nasm -f elf lab7-1.asm
[ankudryashov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[ankudryashov@fedora lab07]$ ./lab7-1

[ankudryashov@fedora lab07]$
```

Рис. 3.4: Вторая версия программы – вывод перевода строки

В нашем каталоге создадим файл lab7-2.asm. Введём в него текст программы из листинга 7.2 (рис. 3.5).

```
mc [ankudryashov@fedora]:~/work/arch-pc/lab07
lab7-2.asm [-M--] 10 L: [ 1+11 12/ 12] *(128 /
%include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

    mov eax,'6'
    mov ebx,'4'
    add eax,ebx
    call iprintLF

    call quit
```

Рис. 3.5: Текст программы lab7-2.asm

Вновь создадим исполняемый файл и запустим его. На этот раз программа выводит на экран число 106 (сумма ASCII кодов 54 + 52) (рис. 3.6).

```
ankudryashov@fedora:~/work/arch-pc/lab07
[ankudryashov@fedora lab07]$ nasm -f elf lab7-2.asm
[ankudryashov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[ankudryashov@fedora lab07]$ ./lab7-2
106
[ankudryashov@fedora lab07]$
```

Рис. 3.6: Третья версия программы – вывод числа 106

Аналогично предыдущему примеру изменим символы на числа. В этот раз программа выведет на экран число 10, т.е. сумму чисел 6 и 4 (рис. 3.7).

```
ankudryashov@fedora:~/work/arch-pc/lab07
[ankudryashov@fedora lab07]$ nasm -f elf lab7-2.asm
[ankudryashov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[ankudryashov@fedora lab07]$ ./lab7-2
10
[ankudryashov@fedora lab07]$
```

Рис. 3.7: Четвёртая версия программы – вывод числа 10

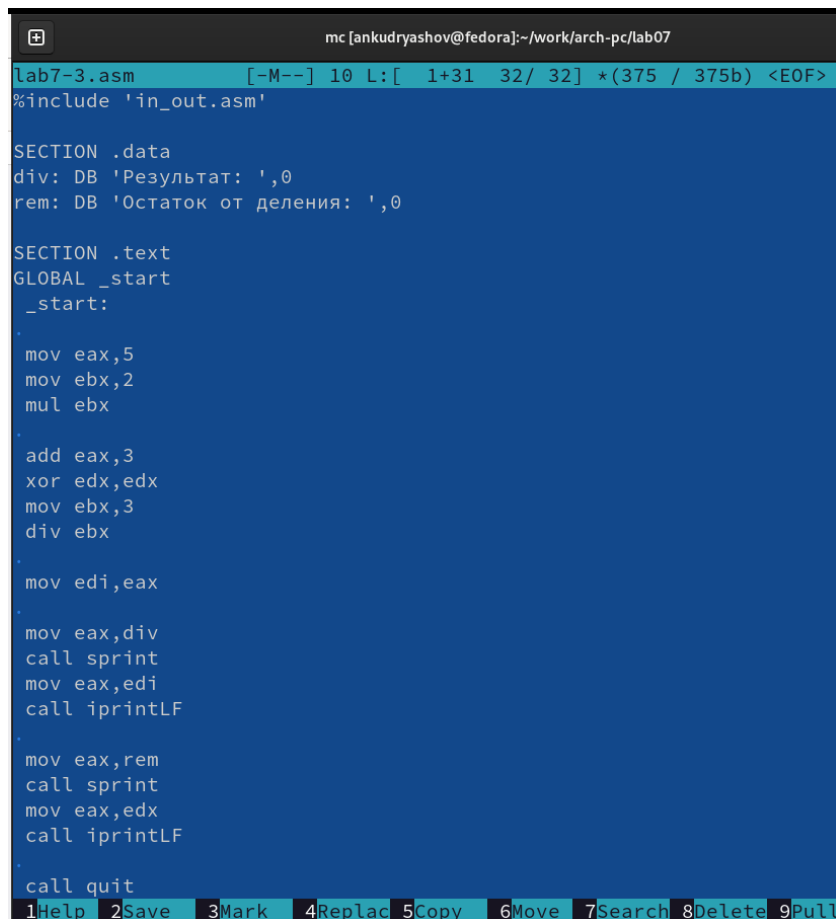
Заменяем функцию `iprintLF` на `iprint`. Создадим и запустим исполняемый файл. Теперь программа выводит число 10, но не ставит после него перевод строки (рис. 3.8).

```
ankudryashov@fedora:~/work/arch-pc/lab07
[ankudryashov@fedora lab07]$ nasm -f elf lab7-2.asm
[ankudryashov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[ankudryashov@fedora lab07]$ ./lab7-2
10[ankudryashov@fedora lab07]$
```

Рис. 3.8: Та же программа, но без перевода строки в конце

3.2 Выполнение арифметических операций в NASM

Вычислим значение функции $f(x) = (5 * 2 + 3) / 3$. Создадим файл `lab7-3.asm` и скопируем в него текст программы из листинга 7.3 (рис. 3.9).



```
lab7-3.asm [-M--] 10 L: [ 1+31 32/ 32] *(375 / 375b) <EOF>
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

SECTION .text
GLOBAL _start
_start:

    mov eax,5
    mov ebx,2
    mul ebx

    add eax,3
    xor edx,edx
    mov ebx,3
    div ebx

    mov edi,eax

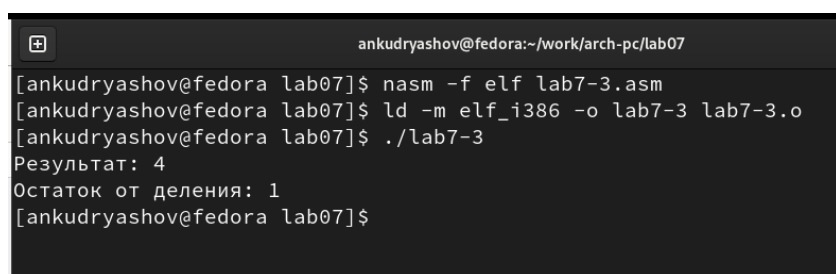
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF

    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF

    call quit
```

Рис. 3.9: Текст программы lab7-3.asm

Создадим исполняемый файл и запустим его. Увидим в выводе целую часть ответа 4 и остаток от деления 1 (рис. 3.10).



```
ankudryashov@fedora:~/work/arch-pc/lab07
[ankudryashov@fedora lab07]$ nasm -f elf lab7-3.asm
[ankudryashov@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[ankudryashov@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[ankudryashov@fedora lab07]$
```

Рис. 3.10: Результат выполнения программы lab7-3

Изменим текст программы так, чтобы она вычисляла $f(x) = (4*6 + 2)/5$ (рис. 3.11).

```
mc [ankudryashov@fedora]:~/work/arch-pc/lab07
lab7-3.asm [-M--] 10 L: [ 1+31 32/ 32] *(375 / 375b) <EOF>
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

SECTION .text
GLOBAL _start
_start:

mov eax,4
mov ebx,6
mul ebx

add eax,2
xor edx,edx
mov ebx,5
div ebx

mov edi,eax

mov eax,div
call sprint
mov eax,edi
call iprintLF

mov eax,rem
call sprint
mov eax,edx
call iprintLF

call quit
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9Pull
```

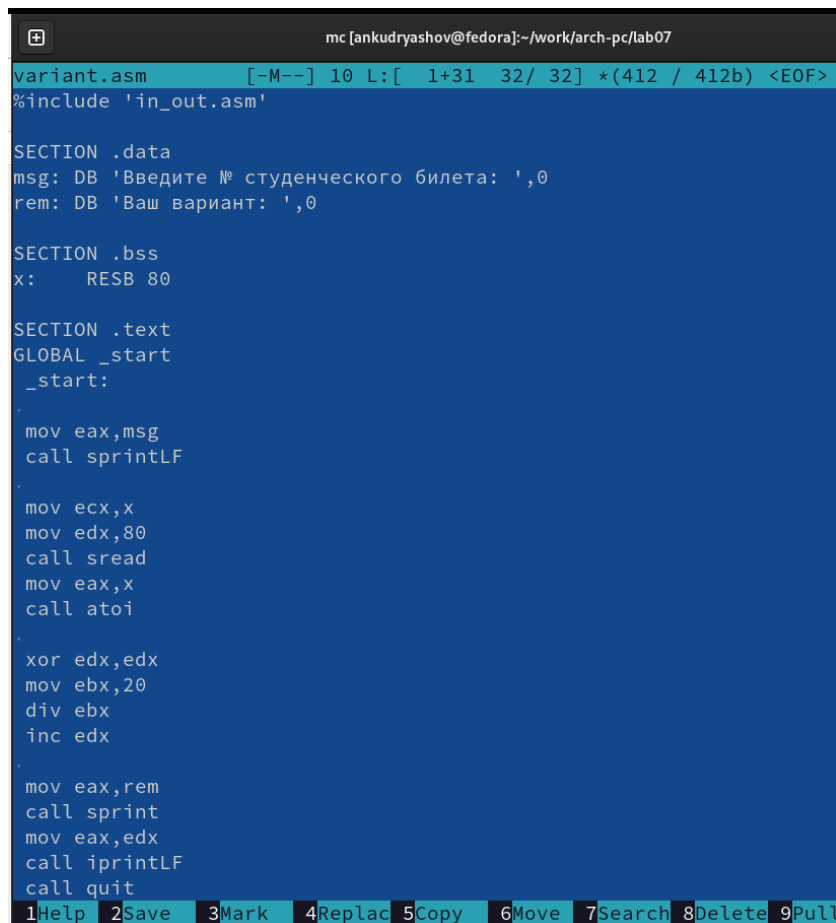
Рис. 3.11: Текст отредактированной программы для $f(x) = (4*6 + 2)/5$

Создадим и запустим исполняемый файл (рис. 3.12).

```
ankudryashov@fedora:~/work/arch-pc/lab07
[ankudryashov@fedora lab07]$ nasm -f elf lab7-3.asm
[ankudryashov@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[ankudryashov@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[ankudryashov@fedora lab07]$
```

Рис. 3.12: Запуск отредактированного файла lab7-3

В качестве еще одного примера создадим программу вычисления варианта задания по номеру студенческого билета. Скопируем в файл variant.asm текст программы из листинга 7.4 (рис. 3.13).



```
variant.asm [-M--] 10 L:[ 1+31 32/ 32] *(412 / 412b) <EOF>
#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 sprintLF

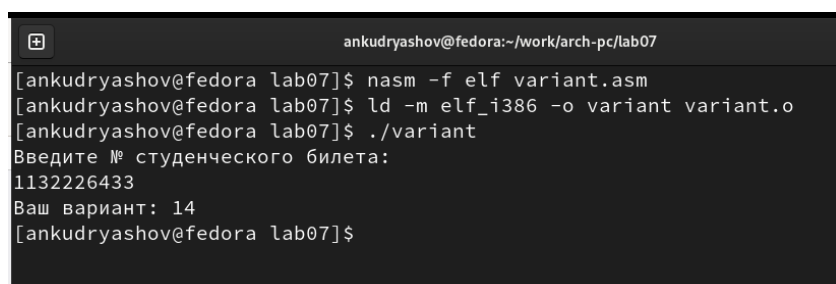
mov ecx,x
mov edx,80
call sread
mov eax,x
call atoi

xor edx,edx
mov ebx,20
div ebx
inc edx

mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Рис. 3.13: Текст программы variant.asm

В результате выполнения получим вариант №14. Проверим: $1132226433 \% 20 = 13$, $13 + 1 = 14$, верно (рис. 3.14).



```
[ankudryashov@fedora lab07]$ nasm -f elf variant.asm
[ankudryashov@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[ankudryashov@fedora lab07]$ ./variant
Введите № студенческого билета:
1132226433
Ваш вариант: 14
[ankudryashov@fedora lab07]$
```

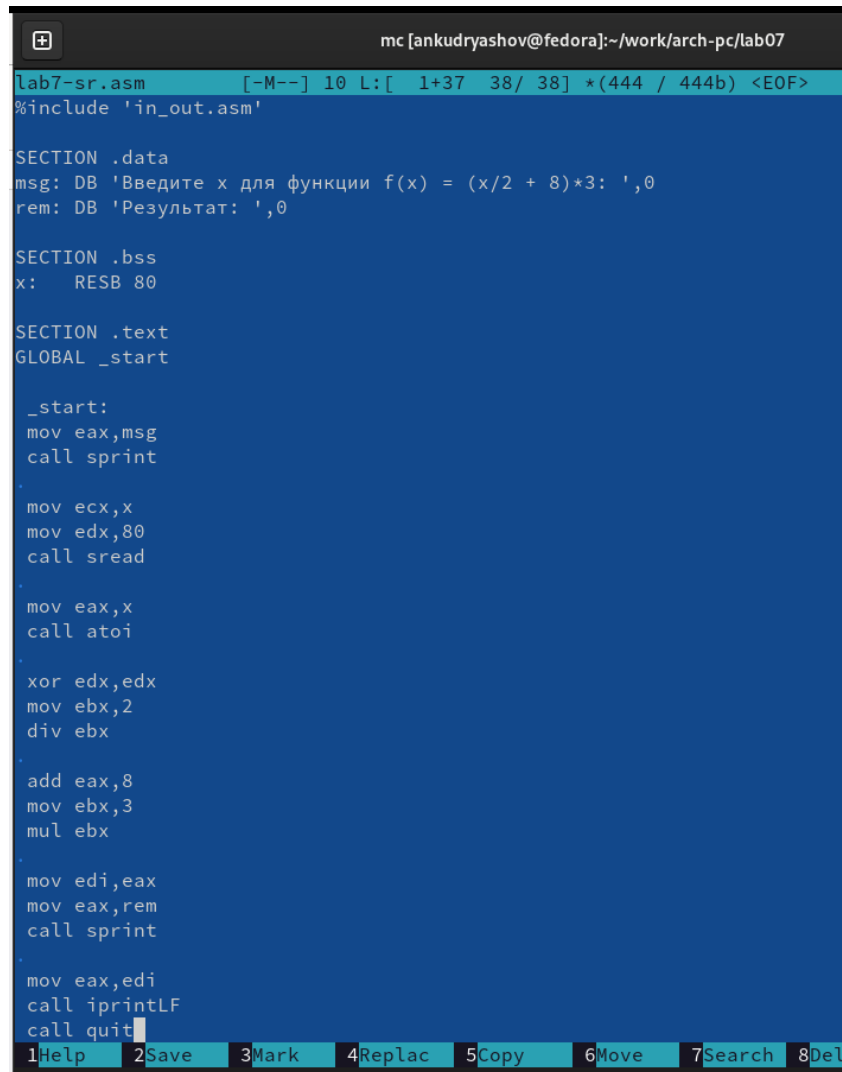
Рис. 3.14: Результат работы variant.asm

3.2.1 Ответы на вопросы

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? Переменная `rem` хранит текст ‘Ваш вариант:’, в строке `mov eax,rem` адрес `rem` помещается в `eax`, затем в строке `call sprint` значение из `eax` выводится на экран.
2. Для чего используются следующие инструкции? `push ecx, x` `mov edx, 80` `call sread` Для считывания значения переменной `x` от пользователя. Первая инструкция указывает, что записать результат считывания нужно в `x`, вторая инструкция указывает максимальную длину строки, последняя инструкция вызывает функцию, считывающую значение, введённое пользователем.
3. Для чего используется инструкция “`call atoi`”? Для преобразования ASCII кода в число
4. Какие строки листинга 7.4 отвечают за вычисления варианта? `mov ebx,20` `div ebx` `inc edx`
5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? В регистр `edx`.
6. Для чего используется инструкция “`inc edx`”? Для увеличения значения, хранящегося в `edx` на 1.
7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений? `mov eax,rem` `call sprint` `mov eax,edx` `call iprintLF` Здесь первые две строки выводят на экран сообщение ‘Ваш вариант:’, а вторые две строки выводят сам номер варианта, посчитанный в ходе работы программы.

3.3 Задание для самостоятельной работы

Получили вариант №14. Функция имеет вид $f(x) = (x/2 + 8) \cdot 3$, а x_1 и x_2 равны 1 и 4 соответственно. Создадим файл lab7-sr.asm и запишем туда текст программы, выполняющей необходимые вычисления (рис. 3.15).



```
mc [ankudryashov@fedora]:~/work/arch-pc/lab07
lab7-sr.asm [-M--] 10 L: [ 1+37 38/ 38] *(444 / 444b) <EOF>
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите x для функции f(x) = (x/2 + 8)*3: ',0
rem: 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 sread

mov eax,x
call atoi

xor edx,edx
mov ebx,2
div ebx

add eax,8
mov ebx,3
mul ebx

mov edi,eax
mov eax,rem
call sprint

mov eax,edi
call iprintLF
call quit
```

Рис. 3.15: Текст программы для самостоятельной работы

Создадим и запустим исполняемый файл lab7-sr. Введём значение 1. Получим $(1/2 + 8) \cdot 3 = 8 \cdot 3 = 24$. Теперь введём значение 4. Получим $(4/2 + 8) \cdot 3 = 10 \cdot 3 = 30$. Всё верно (рис. 3.16).

```
ankudryashov@fedora:~/work/arch-pc/lab07
[ankudryashov@fedora lab07]$ nasm -f elf lab7-sr.asm
[ankudryashov@fedora lab07]$ ld -m elf_i386 -o lab7-sr lab7-sr.o
[ankudryashov@fedora lab07]$ ./lab7-sr
Введите x для функции f(x) = (x/2 + 8)*3: 1
Результат: 24
[ankudryashov@fedora lab07]$ ./lab7-sr
Введите x для функции f(x) = (x/2 + 8)*3: 4
Результат: 30
[ankudryashov@fedora lab07]$
```

Рис. 3.16: Запуск файла программы для самостоятельной работы

4 Выводы

В результате выполнения лабораторной работы удалось в полной мере освоить работу с арифметическими операциями и переводом из символьного в числовой тип на языке ассемблера NASM.