

Отчёт по лабораторной работе 6

Архитектура компьютеров

Агджабекова Эся Рустамовна НПИбд-01-24

Содержание

| | | |
|----------|--|-----------|
| 1 | Цель работы | 5 |
| 2 | Выполнение лабораторной работы | 6 |
| 2.1 | Ответы на вопросы по программе variant.asm | 18 |
| 2.2 | Самостоятельное задание | 19 |
| 3 | Выводы | 22 |

Список иллюстраций

| | | |
|------|---|----|
| 2.1 | Подготовила каталог | 6 |
| 2.2 | Программа в файле lab6-1.asm | 7 |
| 2.3 | Запуск программы lab6-1.asm | 8 |
| 2.4 | Программа в файле lab6-1.asm | 9 |
| 2.5 | Запуск программы lab6-1.asm | 9 |
| 2.6 | Программа в файле lab6-2.asm | 10 |
| 2.7 | Запуск программы lab6-2.asm | 11 |
| 2.8 | Программа в файле lab6-2.asm | 12 |
| 2.9 | Запуск программы lab6-2.asm | 13 |
| 2.10 | Запуск программы lab6-2.asm | 13 |
| 2.11 | Программа в файле lab6-3.asm | 14 |
| 2.12 | Запуск программы lab6-3.asm | 14 |
| 2.13 | Программа в файле lab6-3.asm | 15 |
| 2.14 | Запуск программы lab6-3.asm | 16 |
| 2.15 | Программа в файле variant.asm | 17 |
| 2.16 | Запуск программы variant.asm | 17 |
| 2.17 | Программа в файле task.asm | 20 |
| 2.18 | Запуск программы task.asm | 21 |

Список таблиц

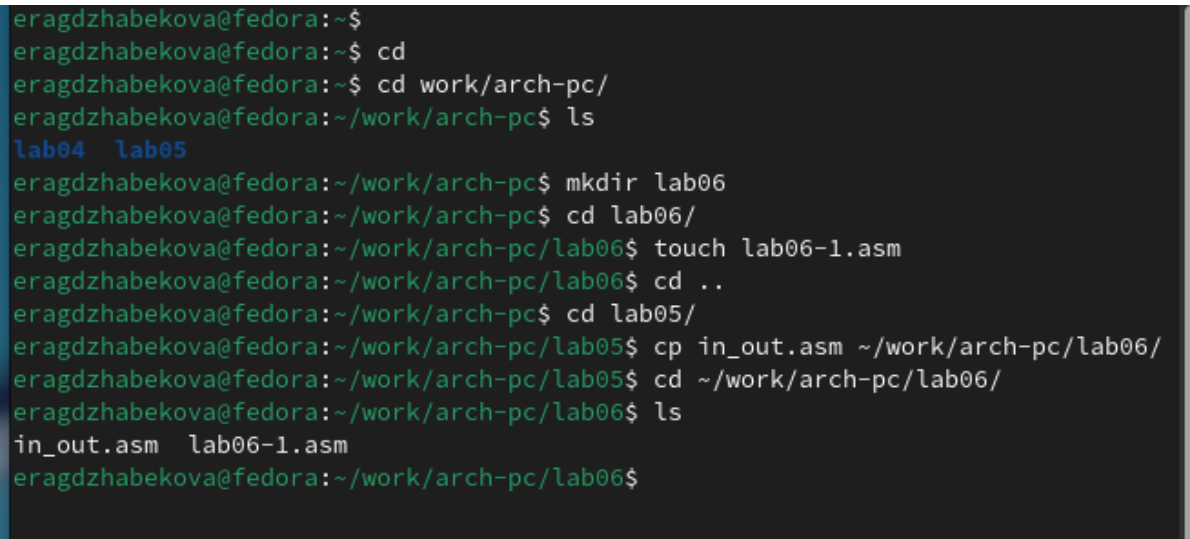
1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

Я создала каталог для программ лабораторной работы № 6, перешла в него и создала файл lab6-1.asm.

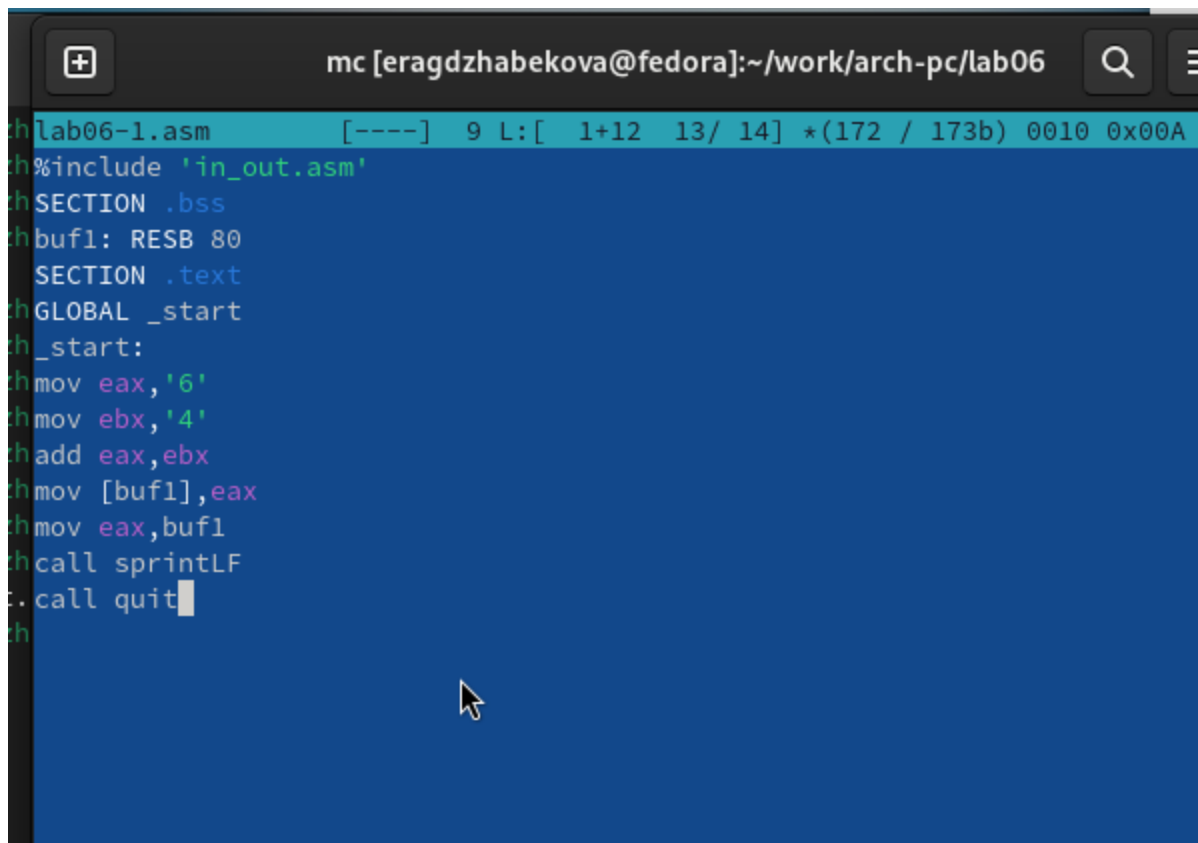
(рис. 2.1)



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

Рис. 2.1: Подготовила каталог

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр еах.



```
mc [eragdzhabekova@fedora]:~/work/arch-pc/lab06
lab06-1.asm [----] 9 L: [ 1+12 13/ 14] *(172 / 173b) 0010 0x00A
#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. 2.2: Программа в файле lab6-1.asm

В данной программе (рис. 2.2) я записываю символ '6' в регистр `eax` (`mov eax, '6'`), а символ '4' в регистр `ebx` (`mov ebx, '4'`).

Затем я добавляю значение регистра `ebx` к значению в регистре `eax` (`add eax, ebx`, результат сложения записывается в регистр `eax`).

После этого я вывожу результат.

Однако, для использования функции `sprintf`, необходимо, чтобы в регистре `eax` был записан адрес, поэтому я использую дополнительную переменную.

Я записываю значение регистра `eax` в переменную `buf1` (`mov [buf1], eax`), а затем записываю адрес переменной `buf1` в регистр `eax` (`mov eax, buf1`) и вызываю функцию `sprintf`.

```
eragdzhbekova@fedora:~/work/arch-pc/lab06$  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f lef lab06-1.asm  
nasm: fatal: unrecognised output format `lef' - use -hf for a list  
Type nasm -h for help.  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./lab06-1  
j  
eragdzhbekova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.3: Запуск программы lab6-1.asm

В данном случае, когда мы ожидаем увидеть число 10 при выводе значения регистра `eax`, фактическим результатом будет символ 'j'.

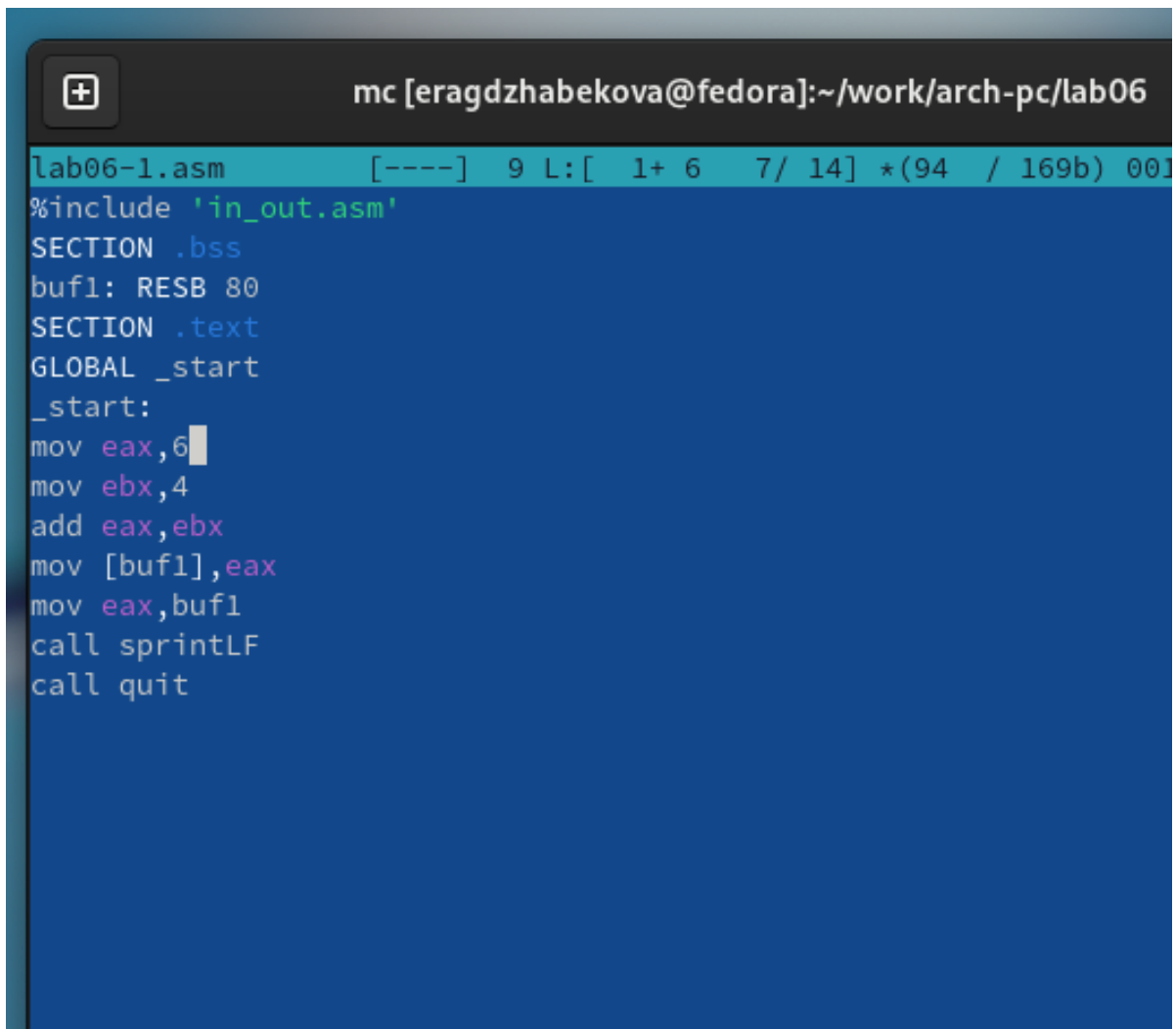
Это происходит из-за того, что код символа 'б' равен 00110110 в двоичном представлении (или 54 в десятичном представлении),

а код символа '4' равен 00110100 (или 52 в десятичном представлении).

Когда я выполняю команду `add eax, ebx`, результатом будет сумма кодов - 01101010 (или 106 в десятичном представлении),

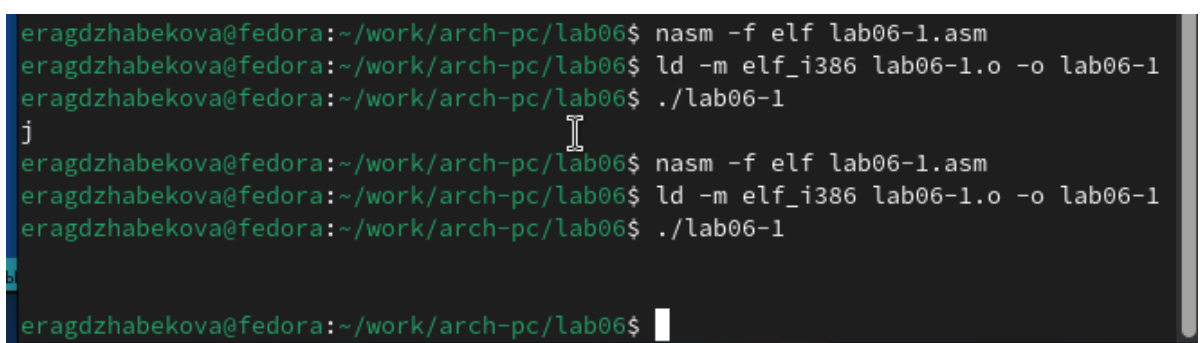
который соответствует символу 'j'. (рис. 2.3)

Далее я изменила текст программы и вместо символов записала в регистры числа. (рис. 2.4)



```
lab06-1.asm [----] 9 L: [ 1+ 6 7/ 14] *(94 / 169b) 001
#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 printf
call _exit
```

Рис. 2.4: Программа в файле lab6-1.asm



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

Рис. 2.5: Запуск программы lab6-1.asm

Как и в предыдущем случае, при выполнении программы мы не получим число

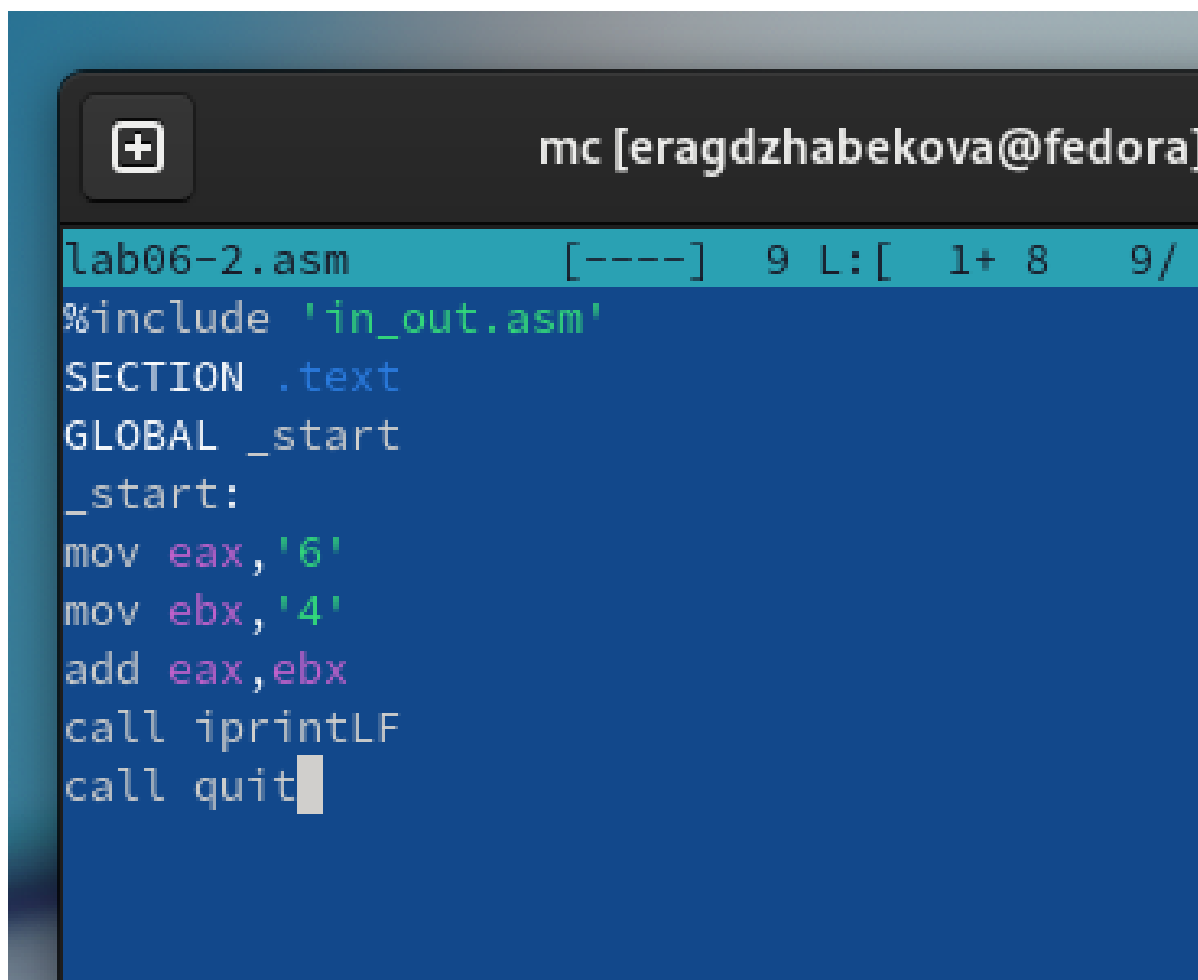
10.

Вместо этого выводится символ с кодом 10, который представляет собой символ конца строки (возврат каретки).

Этот символ не отображается в консоли, но он добавляет пустую строку.

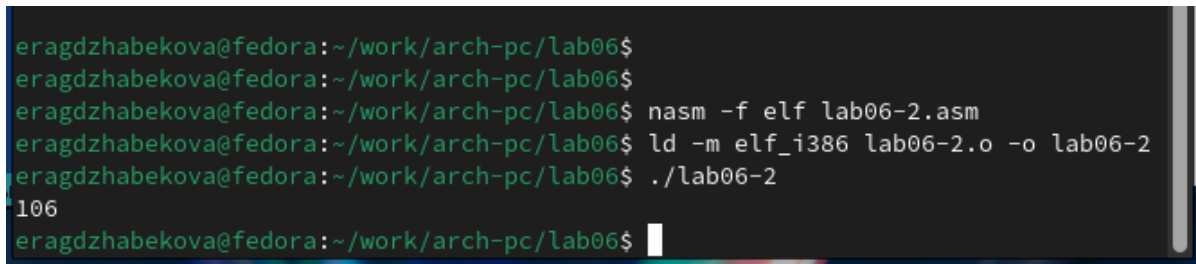
Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно.

Я преобразовала текст программы с использованием этих функций. (рис. 2.6)



```
mc [eragdzhbekova@fedora]
lab06-2.asm [----] 9 L: [ 1+ 8 9/
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

Рис. 2.6: Программа в файле `lab6-2.asm`



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

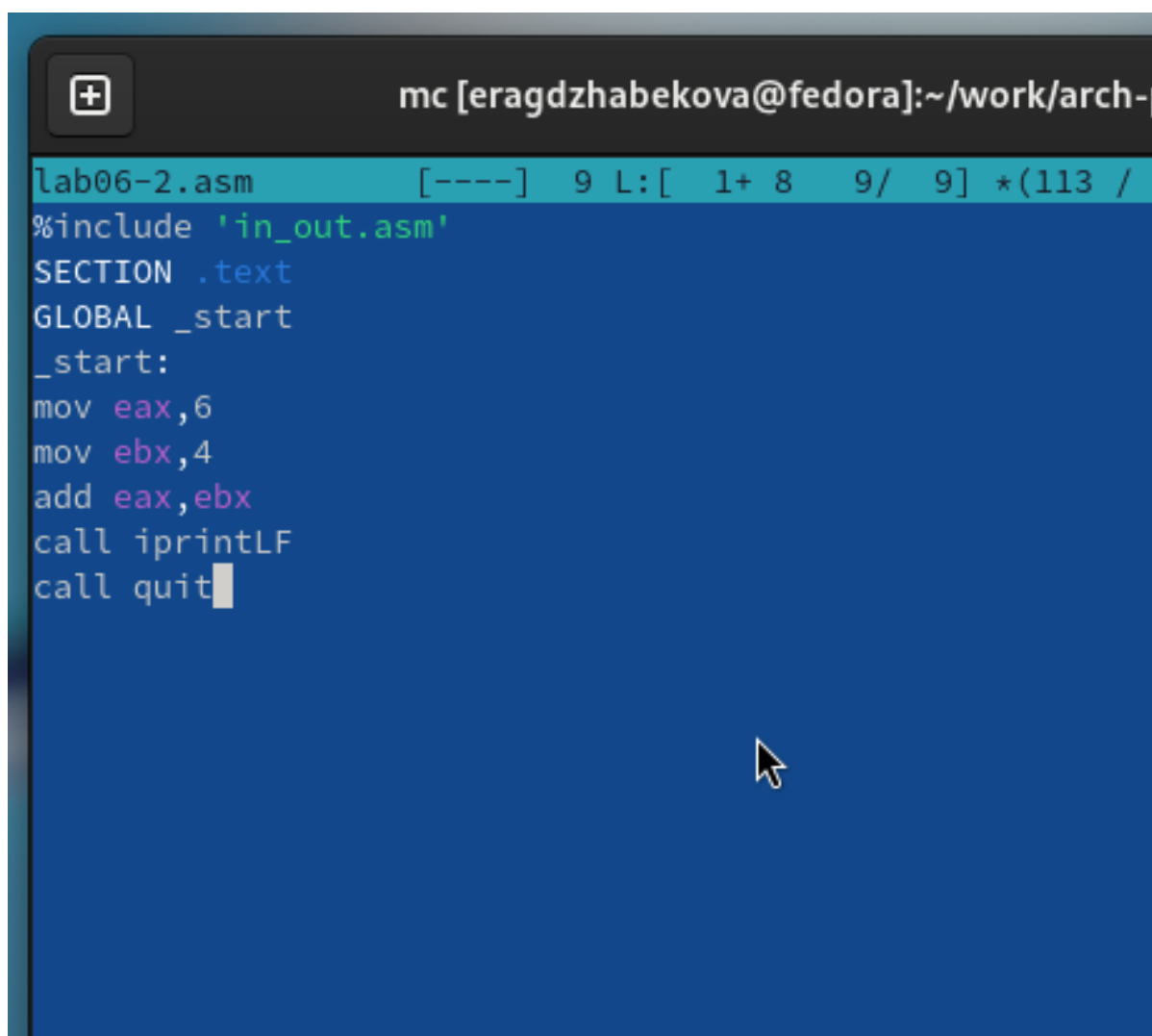
Рис. 2.7: Запуск программы lab6-2.asm

В результате выполнения программы я получаю число 106. (рис. 2.7)

В данном случае, как и в первом примере, команда `add` складывает коды символов '6' и '4' ($54+52=106$).

Однако, в отличие от предыдущей программы, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру, я изменила символы на числа. (рис. 2.8)



```
mc [eragdzhbekova@fedora]:~/work/arch-  
lab06-2.asm [----] 9 L:[ 1+ 8 9/ 9] *(113 /  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
call quit
```

Рис. 2.8: Программа в файле lab6-2.asm

Функция iprintLF позволяет вывести число, и операндами были числа (а не коды символов).

Поэтому получаем число 10. (рис. 2.9)

```
eragdzhbekova@fedora:~/work/arch-pc/lab06$  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./lab06-2  
106  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./lab06-2  
10  
eragdzhbekova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.9: Запуск программы lab6-2.asm

Я заменила функцию `iprintLF` на `iprint`. Создала исполняемый файл и запустила его.

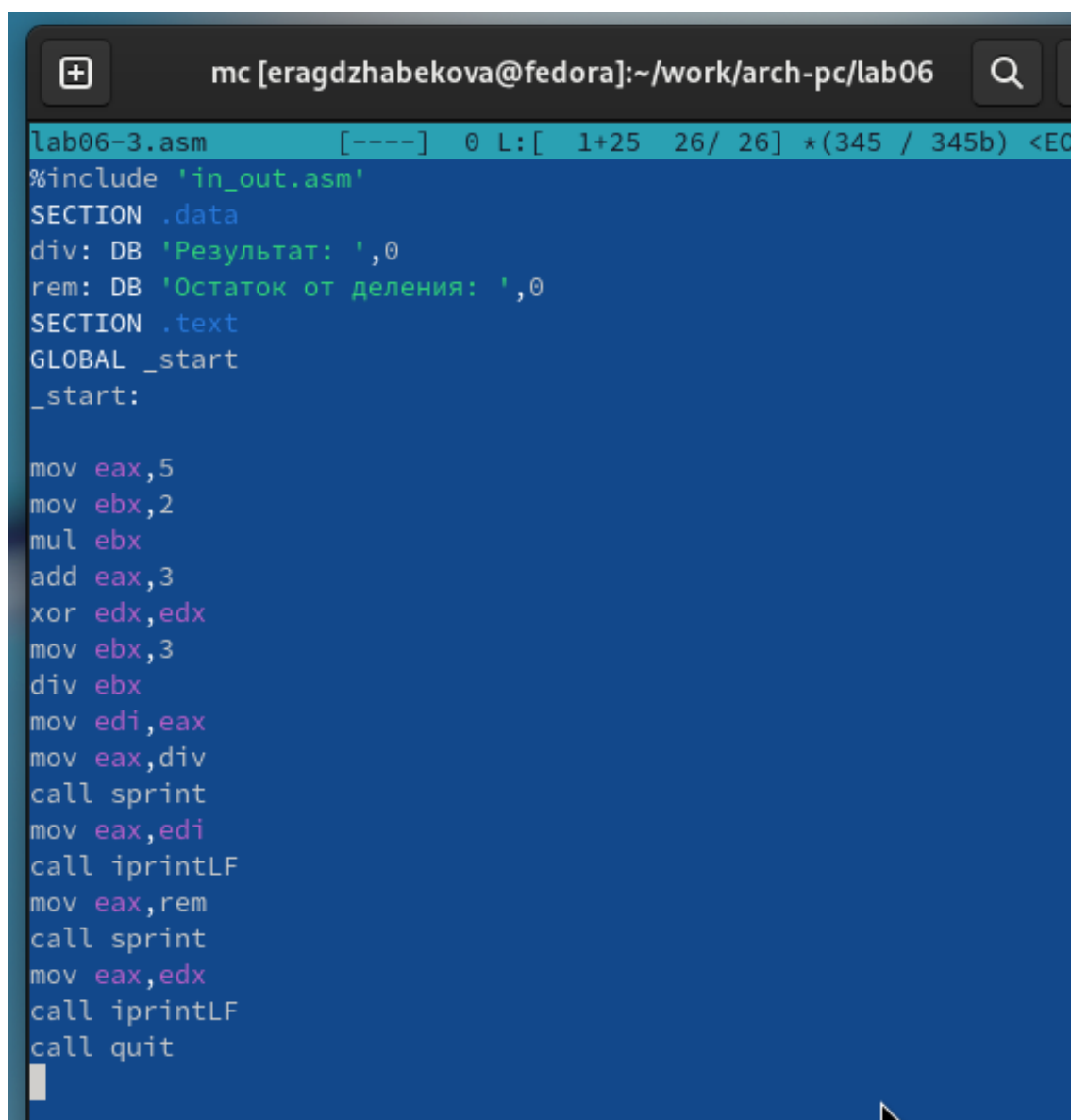
Вывод отличается тем, что нет переноса строки. (рис. 2.10)

```
eragdzhbekova@fedora:~/work/arch-pc/lab06$  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./lab06-2  
106  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./lab06-2  
10  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./lab06-2  
10eragdzhbekova@fedora:~/work/arch-pc/lab06$  
eragdzhbekova@fedora:~/work/arch-pc/lab06$  
eragdzhbekova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.10: Запуск программы lab6-2.asm

В качестве примера выполнения арифметических операций в NASM привожу программу вычисления арифметического выражения (рис. 2.11) (рис. 2.12)

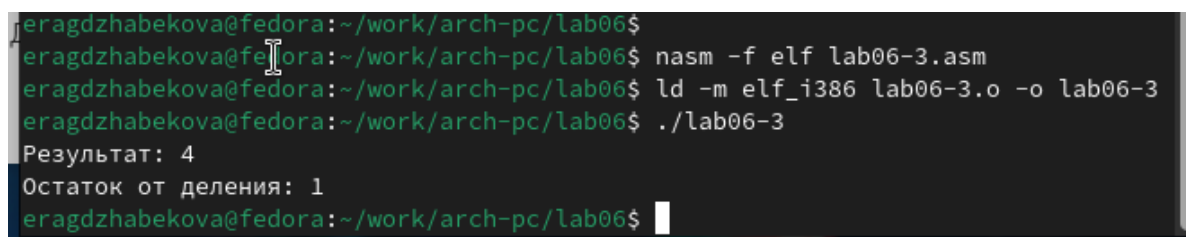
$$f(x) = (5 * 2 + 3) / 3$$



```
mc [eragdzhbekova@fedora]:~/work/arch-pc/lab06
lab06-3.asm [----] 0 L: [ 1+25 26/ 26] *(345 / 345b) <EO
#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. 2.11: Программа в файле lab6-3.asm



```
eragdzhbekova@fedora:~/work/arch-pc/lab06$
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./lab06-3
Результат: 4
Остаток от деления: 1
eragdzhbekova@fedora:~/work/arch-pc/lab06$
```

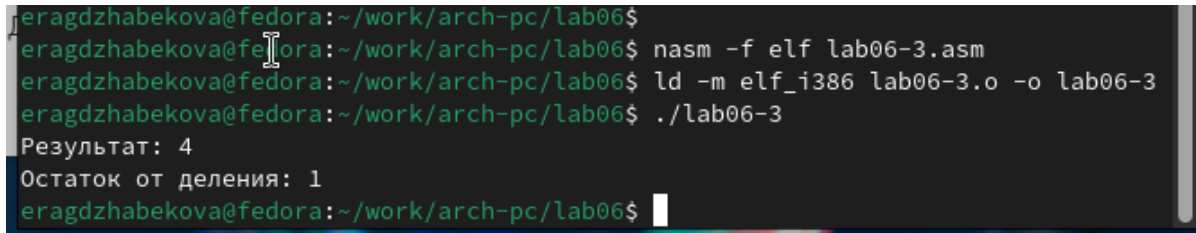
Рис. 2.12: Запуск программы lab6-3.asm

Я изменила текст программы для вычисления выражения

$$f(x) = (4 * 6 + 2) / 5$$

.

Создала исполняемый файл и проверила его работу. (рис. 2.13) (рис. 2.14)



```
eragdzhbekova@fedora:~/work/arch-pc/lab06$  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./lab06-3  
Результат: 4  
Остаток от деления: 1  
eragdzhbekova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.13: Программа в файле lab6-3.asm

```
lab06-3.asm [----] 7 L:[ 1+14 15/ 26] *(222 / 345b) 001
#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
```

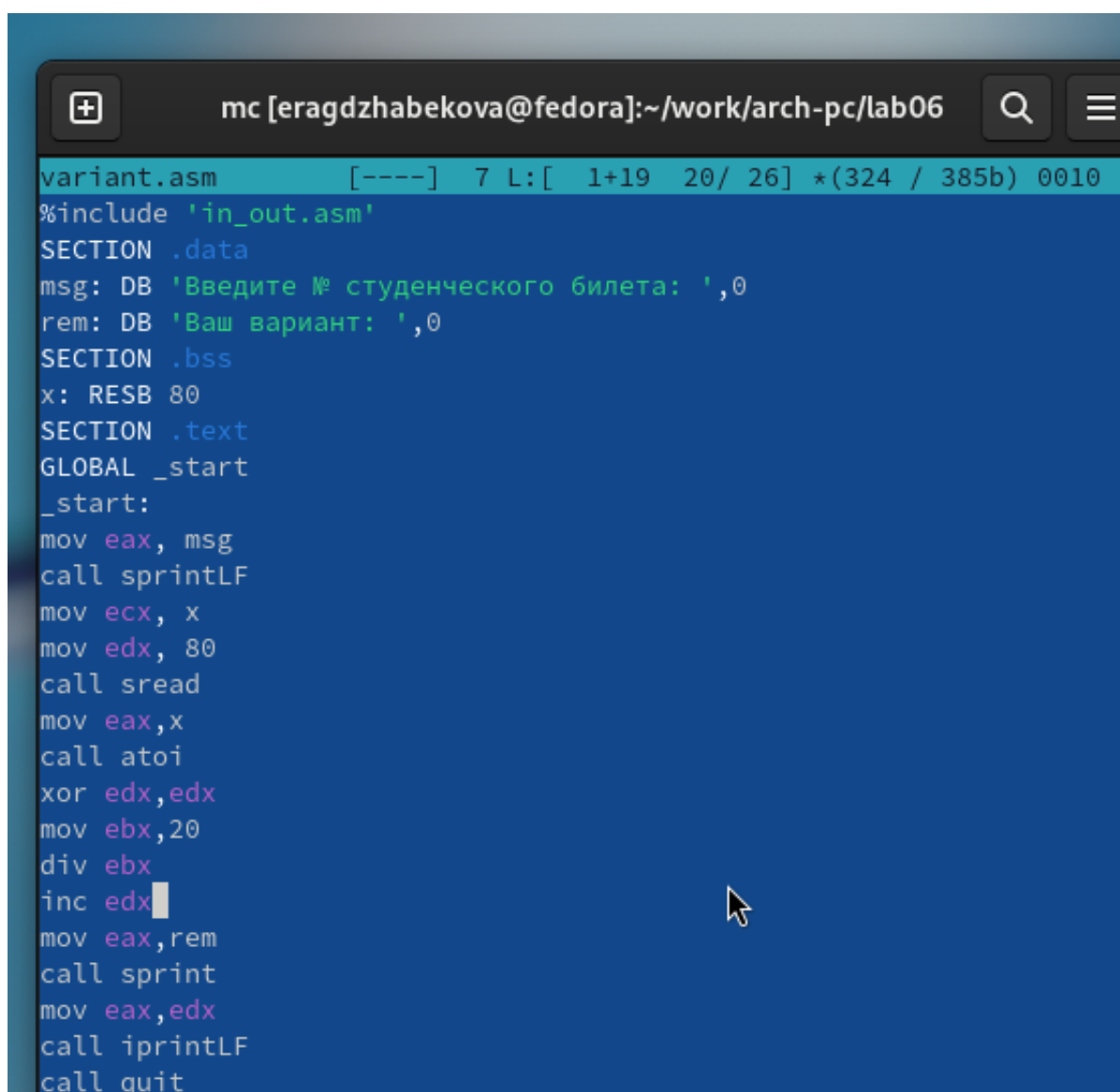
Рис. 2.14: Запуск программы lab6-3.asm

В качестве другого примера рассматриваю программу вычисления варианта задания по номеру студенческого билета. (рис. 2.15) (рис. 2.16)

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры.

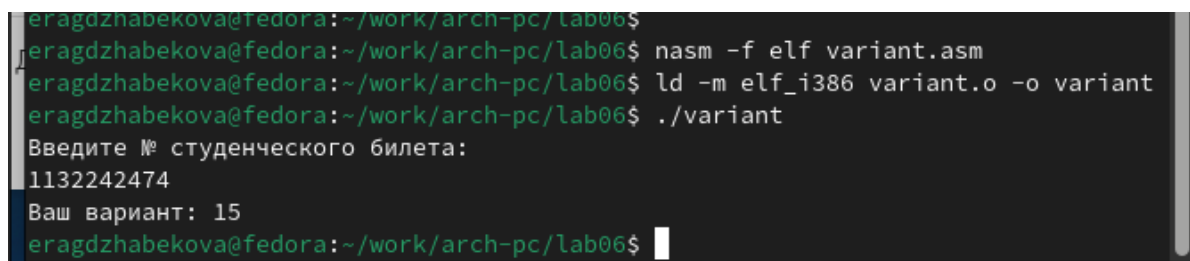
Как отмечалось выше, ввод с клавиатуры осуществляется в символьном виде, и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа.

Для этого может быть использована функция `atoi` из файла `in_out.asm`.



```
variant.asm [----] 7 L:[ 1+19 20/ 26] *(324 / 385b) 0010 [
#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. 2.15: Программа в файле `variant.asm`



```
eragdzhbekova@fedora:~/work/arch-pc/lab06$
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132242474
Ваш вариант: 15
eragdzhbekova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.16: Запуск программы `variant.asm`

2.1 Ответы на вопросы по программе variant.asm

1. **Какие строки листинга отвечают за вывод на экран сообщения ‘Ваш вариант:’?**

Строка “mov eax, 0” перекладывает в регистр значение переменной с фразой “Ваш вариант:”.

Строка “call sprint” вызывает подпрограмму вывода строки.

2. **Для чего используются следующие инструкции?**

Инструкция “nasm” используется для компиляции кода на языке ассемблера NASM.

Инструкция “mov ecx, x” используется для перемещения значения переменной x в регистр ecx.

Инструкция “mov edx, 80” используется для перемещения значения 80 в регистр edx.

Инструкция “call read” вызывает подпрограмму для считывания значения студенческого билета из консоли.

3. **Для чего используется инструкция “call atoi”?**

Инструкция “call atoi” используется для преобразования введенных символов в числовой формат.

4. **Какие строки листинга отвечают за вычисления варианта?**

Строка “xor edx, edx” обнуляет регистр edx.

Строка “mov ebx, 20” записывает значение 20 в регистр ebx.

Строка “div ebx” выполняет деление номера студенческого билета на 20.

Строка “inc edx” увеличивает значение регистра edx на 1.

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

Остаток от деления записывается в регистр edx.

6. Для чего используется инструкция “inc edx”?

Инструкция “inc edx” используется для увеличения значения в регистре edx на 1, в соответствии с формулой вычисления варианта.

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

Строка “mov eax, edx” перекладывает результат вычислений в регистр eax. Строка “call iprintLF” вызывает подпрограмму для вывода значения на экран.

2.2 Самостоятельное задание

Я написала программу вычисления выражения $y = f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений.

Вид функции $f(x)$ выбрала из таблицы 6.3 вариантов заданий в соответствии с номером, полученным при выполнении лабораторной работы.

Создала исполняемый файл и проверила его работу для значений x_1 и x_2 из 6.3.

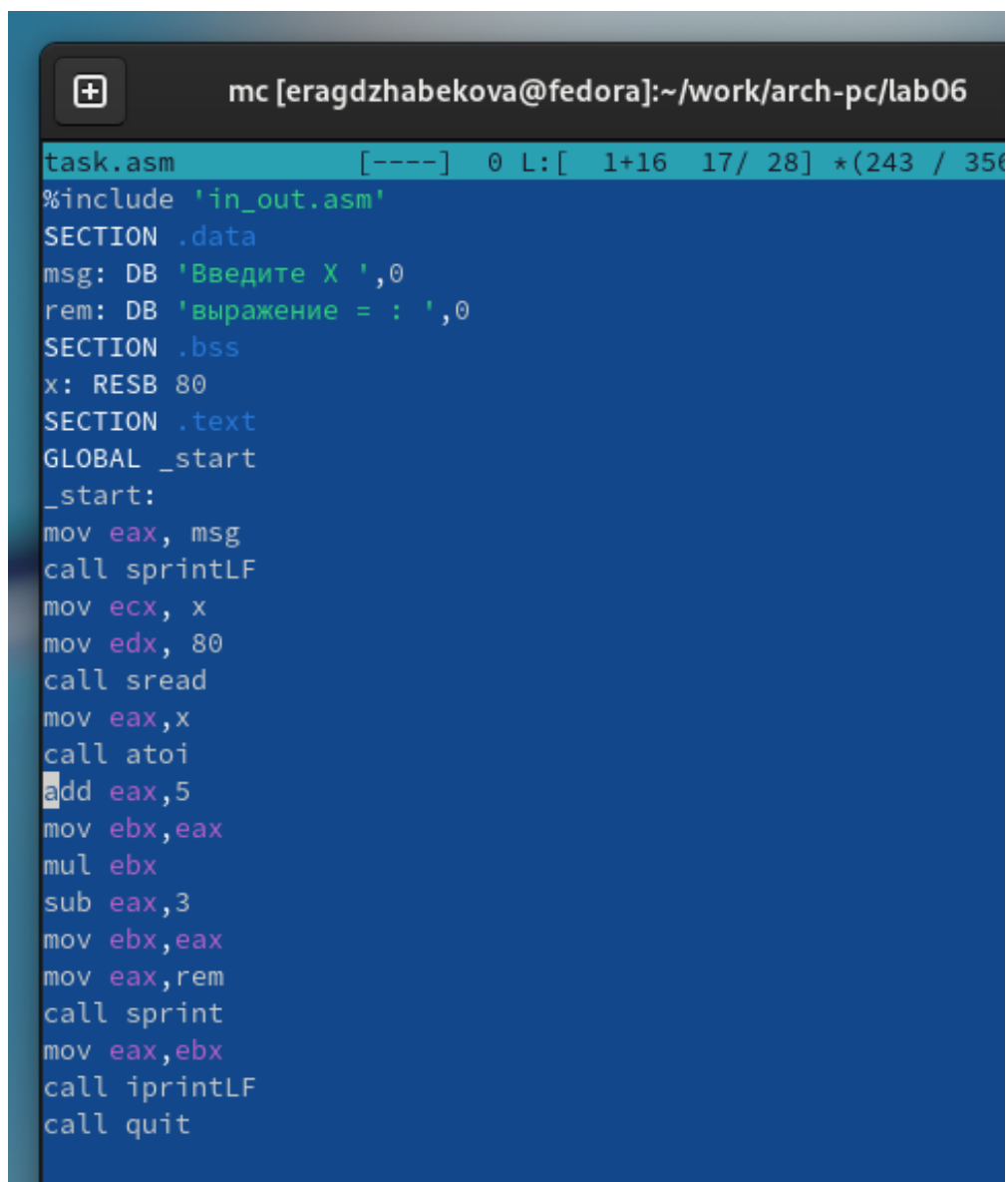
Получили вариант 15 -

$$(5 + x)^2 - 3$$

для

$$x_1 = 5, x_2 = 1$$

(рис. 2.17) (рис. 2.18)



```
task.asm [----] 0 L: [ 1+16 17/ 28] *(243 / 356
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите X ',0
rem: DB 'выражение = : ',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
add eax, 5
mov ebx, eax
mul ebx
sub eax, 3
mov ebx, eax
mov eax, rem
call sprint
mov eax, ebx
call iprintLF
call quit
```

Рис. 2.17: Программа в файле task.asm

```
eragdzhbekova@fedora:~/work/arch-pc/lab06$  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ nasm -f elf task.asm  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 task.o -o task  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./task  
Введите X  
5  
выражение = : 97  
eragdzhbekova@fedora:~/work/arch-pc/lab06$ ./task  
Введите X  
1  
выражение = : 33  
eragdzhbekova@fedora:~/work/arch-pc/lab06$
```

Рис. 2.18: Запуск программы task.asm

3 Выводы

Изучили работу с арифметическими операциями.