

# **Отчёта по лабораторной работе 7**

**Освоение арифметических инструкций языка ассемблера NASM.**

Сиссе Мохамед Ламин

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	28
	Список литературы	29

## Список иллюстраций

4.1	Пример программы . . . . .	10
4.2	Работа программы . . . . .	11
4.3	Пример программы . . . . .	12
4.4	Работа программы . . . . .	13
4.5	Пример программы . . . . .	14
4.6	Работа программы . . . . .	15
4.7	Пример программы . . . . .	16
4.8	Работа программы . . . . .	17
4.9	Работа программы . . . . .	18
4.10	Пример программы . . . . .	19
4.11	Работа программы . . . . .	20
4.12	Пример программы . . . . .	21
4.13	Работа программы . . . . .	22
4.14	Пример программы . . . . .	23
4.15	Работа программы . . . . .	24
4.16	Пример программы . . . . .	26
4.17	Работа программы . . . . .	27

## Список таблиц

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

1. Изучите примеры программ.
2. Напишите программу вычисления выражения в соответствии с вариантом.
3. Загрузите файлы на GitHub.

### 3 Теоретическое введение

В основном наборе инструкций входят разные вариации четырех арифметических действий: сложение, вычитание, умножение, деление. Важно помнить, что в результате арифметических действий меняются некоторые биты регистра флагов, что позволяет выполнять команду условного перехода, т.е. разветвлять программу на основе результат операции. Замечу, что для команд сложения и вычитания справедливыми являются отмеченное выше для операндов команды `mov`. К командам сложения можно отнести: `add` – обычное сложение, `adc` – сложение с добавлением результату флага переноса в качестве единицы (если флаг равен нулю, то команда эквивалентна команде `add`), `xadd` – сложение, с предварительным обменом данных между операндами, `inc` – прибавление единицы к содержимому операнда. Несколько примеров: `add %rbx, dt` (или `addq, dt`, где четко указано, что складываются 64-битовые величины) – к содержимому области памяти `dt` добавляется содержимое регистра `rbx` и результат помещается в `dt`; `adc %rdx, %rdx` – удвоение содержимого регистра `rdx` плюс добавление значения флага переноса; `incl ll` – увеличение на единицу содержимого памяти по адресу `ll`. При этом явно указывается, что операнд имеет размер 32 бита (`dword`).

К командам вычитания можно отнести следующие инструкции процессора x86-64: `sub` – обычное вычитание, `sbb` – вычитание из результата флага переноса в качестве единицы (если флаг равен нулю, то команда эквивалентна `sub`), `dec` – вычитание единицы из результата, `neg` – вычитание значения операнда из 0. Несколько примеров: `sub %rax, ll` – из содержимого `ll` вычитается содержимое

регистра `rax` (или явно `subq %rax, ll`, где указывается, что операнды имеют 64-размер), и результат помещается в `ll`; `subw go, %ax` – вычитание из содержимого `ax` числа по адресу `go`, результат помещается в `ax`; `sbb %rdx, %rax` – вычитание с дополнительным вычитанием флага переноса (из числа в `rax` вычитается число в `rdx` и результат в `rax`); `decbl` – вычитание единицы из байта, расположенного по адресу `l`. Следует отметить еще специальную команду `cmpr`, которая во всем похожа на команду `sub`, кроме одного – результат вычитания никуда не помещается. Инструкция используется специально, для сравнения операндов.

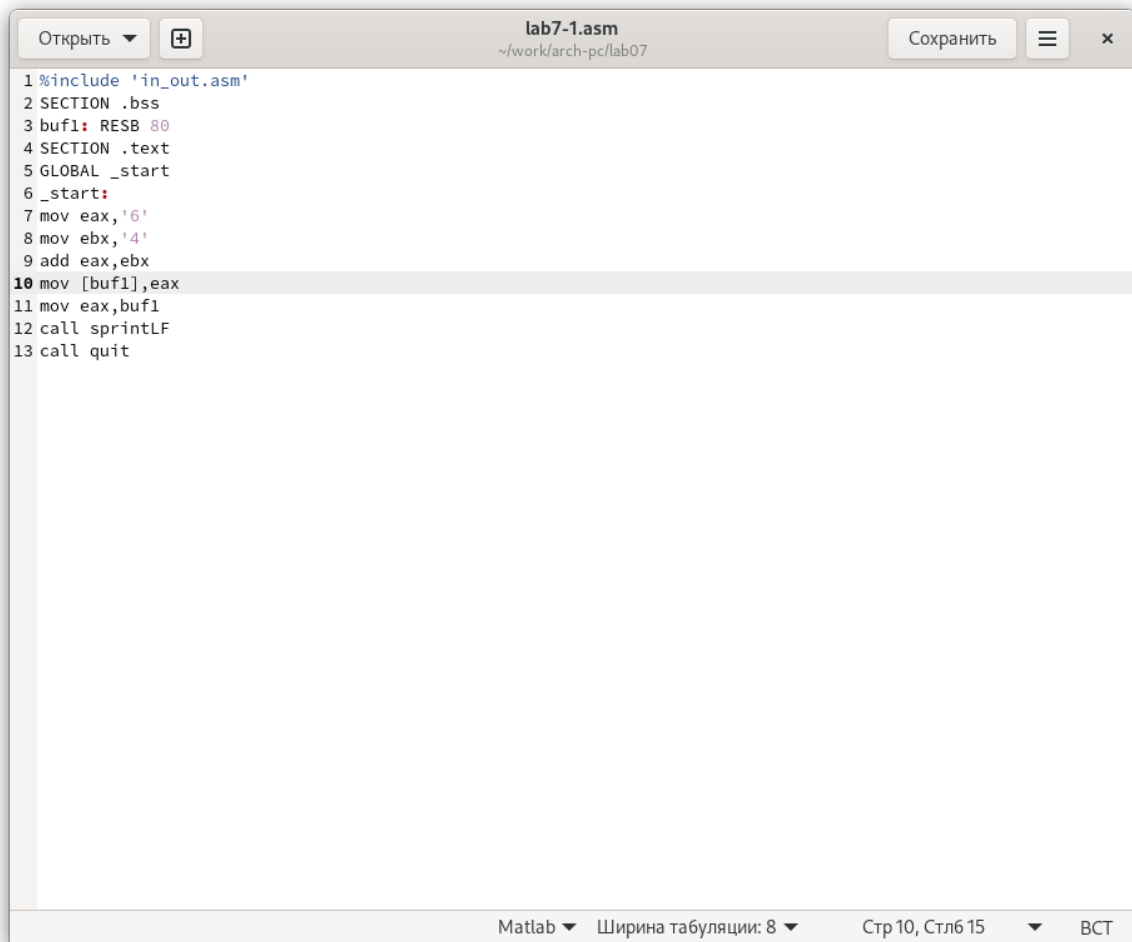
Две основные команды умножения: `mul` – умножение беззнаковых чисел, `imul` – умножение знаковых чисел. Команда содержит один операнд – регистр или адрес памяти. В зависимости от размера операнда данные помещаются: в `ax`, `dx : ax`, `edx : eax`, `rdx : rax`. Например: `mull ll` – содержимое памяти с адресом `ll` будет умножено на содержимое `eax` (не забываем о суффиксе `l`), а результат отправлен в пару регистров `edx : eax`; `mul %dl` – умножить содержимое регистра `dl` на содержимое регистра `al`, а результат положить в `ax`; `mul %r8` – умножить содержимое регистра `r8` на содержимое регистра `rax`, а результат положить в пару регистров `rdx : rax`.

Для деления (целого) также предусмотрены две команды: `div` – беззнаковое деление, `idiv` – знаковое деление. Инструкция также имеет один операнд – делитель. В зависимости от его размера результат помещается: `al` – результат деления, `ah` – остаток от деления; `ax` – результат деления, `dx` – остаток от деления; `eax` – результат деления, `edx` – остаток от деления; `rax` – результат деления, `rdx` – остаток от деления. Приведем примеры: `divl dv` – содержимое `edx : eax` делится на делитель, находящийся в памяти по адресу `dv` и результат деления помещается в `eax`, остаток в `edx`; `div %rsi` – содержимое `rdx : rax` делится на содержимое `rsi`, результат помещается в `rax`, остаток в `rdx`.



## 4 Выполнение лабораторной работы

1. Создайте каталог для программ лабораторной работы № 6, перейдите в него и создайте файл lab7-1.asm:
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр eax. (рис. 4.1, 4.2)

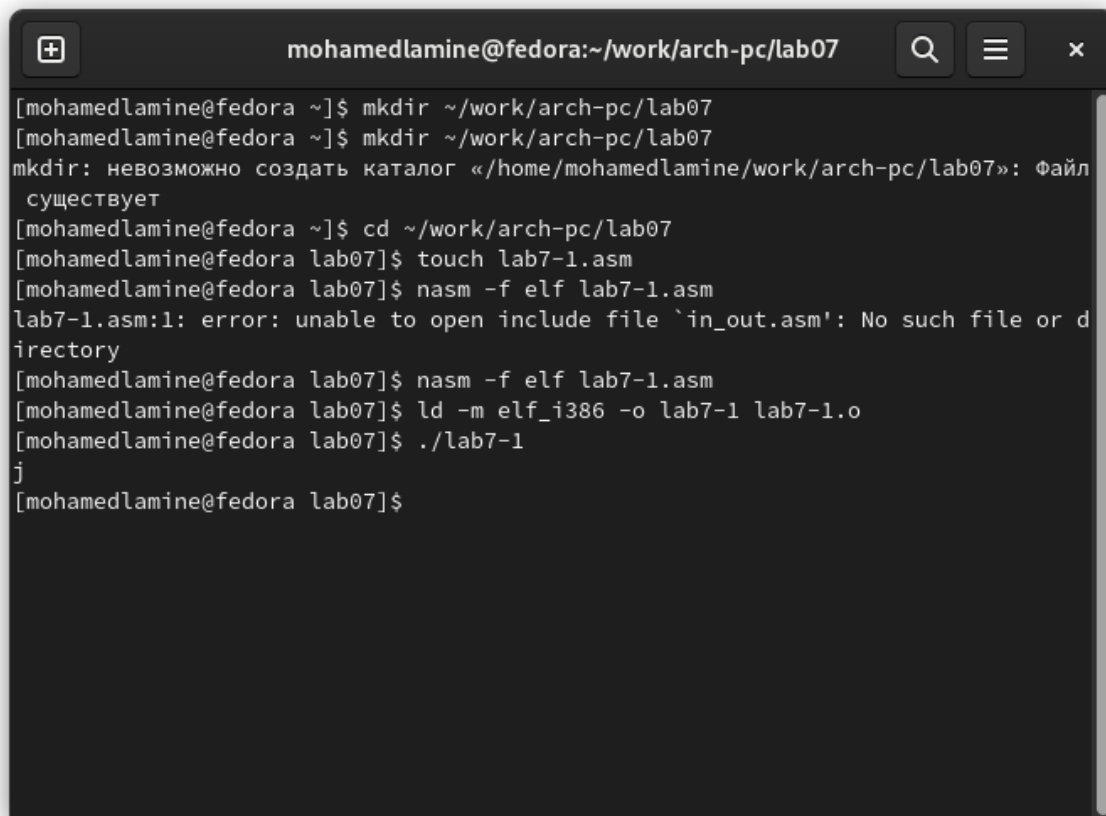


The image shows a MATLAB editor window titled 'lab7-1.asm' with a file path of '~\work\arch-pc\lab07'. The window contains assembly code for an x86\_64 architecture. The code includes a header file 'in\_out.asm', defines a buffer 'buf1' of size 80, and sets up a global section starting at '\_start'. The main logic involves moving the value '6' into 'eax', adding '4' to it, and storing the result at the memory address pointed to by 'buf1'. Finally, it calls 'sprintf' to format the data and 'quit' to end the program.

```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintf
13 call quit
```

The status bar at the bottom indicates the editor is in 'Matlab' mode, with a tab width of 8, and the cursor is at line 10, column 15.

Рис. 4.1: Пример программы

A terminal window titled 'mohamedlamine@fedora:~/work/arch-pc/lab07' with search, menu, and close icons. It shows a series of shell commands and their outputs. The commands include creating a directory, changing to it, creating a file, and using nasm and ld to compile and link an assembly program. An error message is shown for a missing include file.

```
[mohamedlamine@fedora ~]$ mkdir ~/work/arch-pc/lab07
[mohamedlamine@fedora ~]$ mkdir ~/work/arch-pc/lab07
mkdir: невозможно создать каталог «/home/mohamedlamine/work/arch-pc/lab07»: Файл
существует
[mohamedlamine@fedora ~]$ cd ~/work/arch-pc/lab07
[mohamedlamine@fedora lab07]$ touch lab7-1.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
lab7-1.asm:1: error: unable to open include file `in_out.asm': No such file or d
irectory
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mohamedlamine@fedora lab07]$ ./lab7-1
j
[mohamedlamine@fedora lab07]$
```

Рис. 4.2: Работа программы

3. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы (Листинг 1) следующим образом: (рис. 4.3, 4.4)

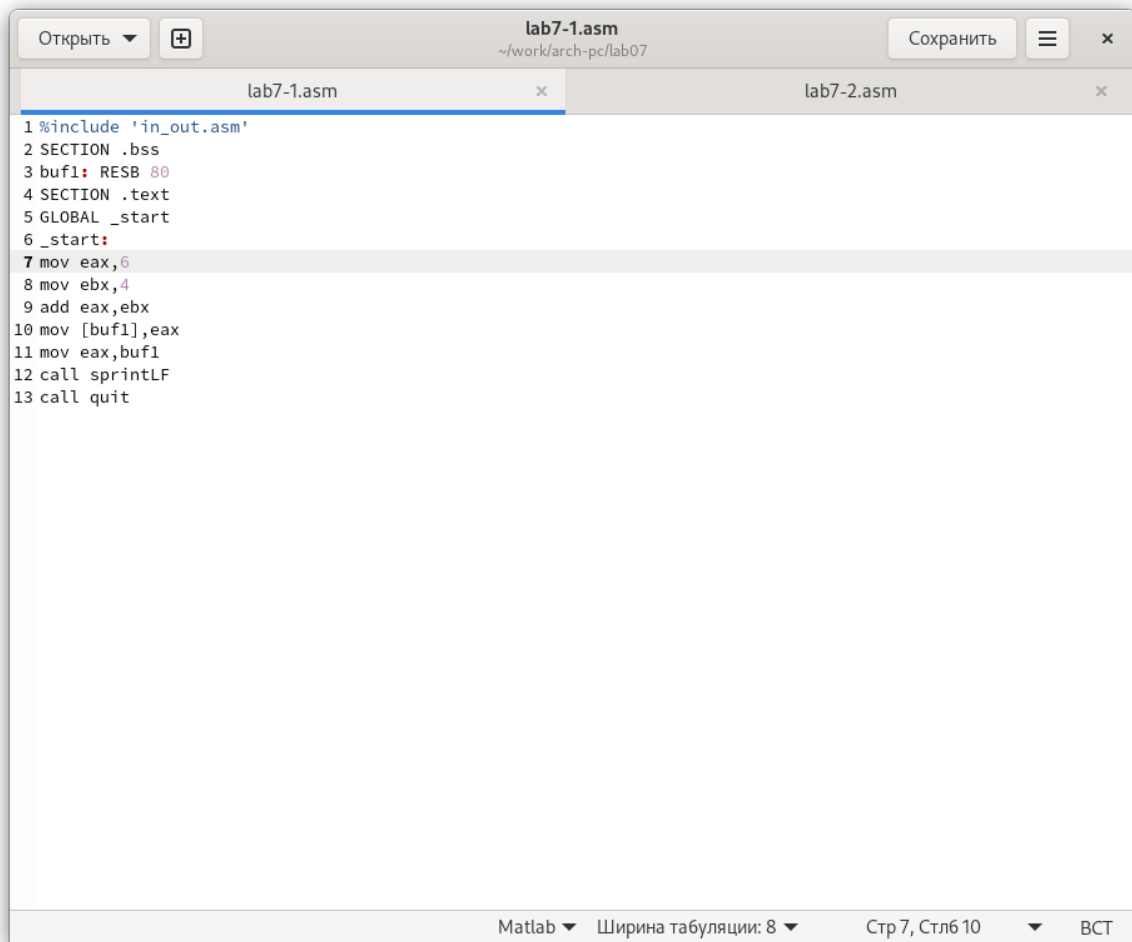
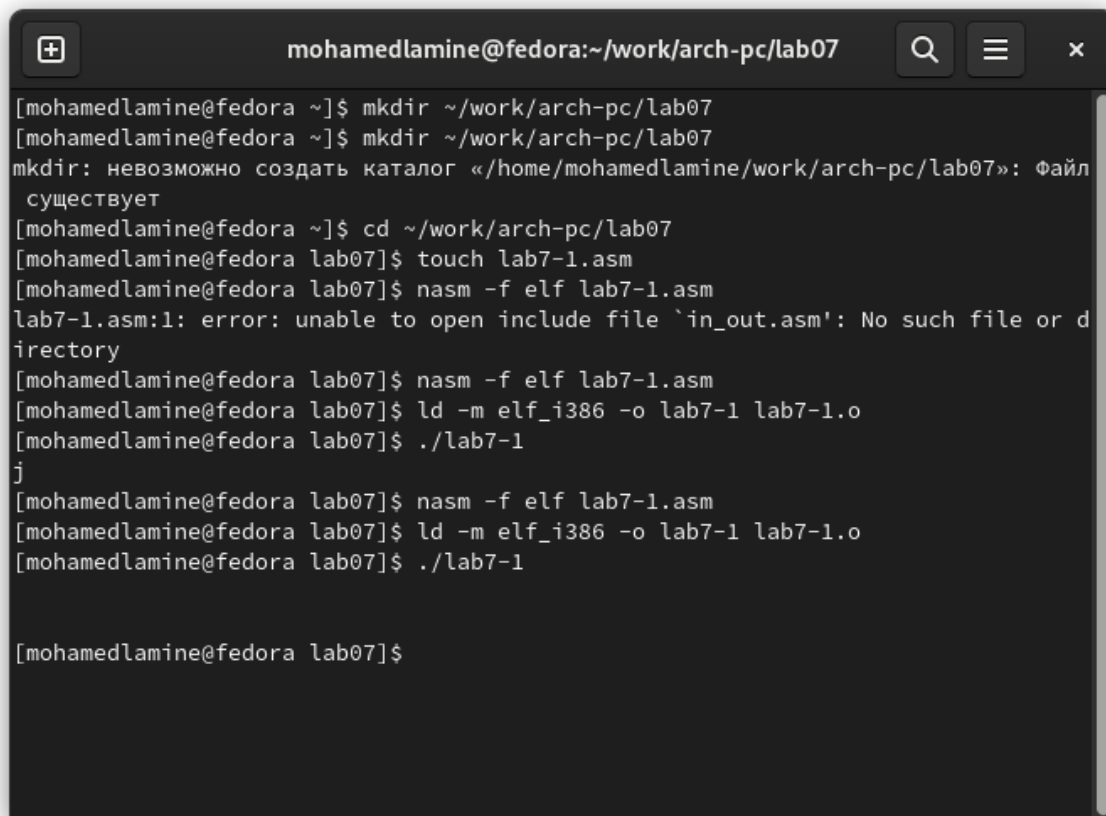


Рис. 4.3: Пример программы

A terminal window titled 'mohamedlamine@fedora:~/work/arch-pc/lab07' with search, menu, and close icons. The terminal shows the following commands and output:

```
[mohamedlamine@fedora ~]$ mkdir ~/work/arch-pc/lab07
[mohamedlamine@fedora ~]$ mkdir ~/work/arch-pc/lab07
mkdir: невозможно создать каталог «/home/mohamedlamine/work/arch-pc/lab07»: Файл
существует
[mohamedlamine@fedora ~]$ cd ~/work/arch-pc/lab07
[mohamedlamine@fedora lab07]$ touch lab7-1.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
lab7-1.asm:1: error: unable to open include file `in_out.asm': No such file or d
irectory
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mohamedlamine@fedora lab07]$ ./lab7-1
j
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mohamedlamine@fedora lab07]$ ./lab7-1

[mohamedlamine@fedora lab07]$
```

Рис. 4.4: Работа программы

Никакой символ не виден, но он есть. Это возврат каретки LF.

4. Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 7.1 с использованием этих функций. (рис. 4.5, 4.6)

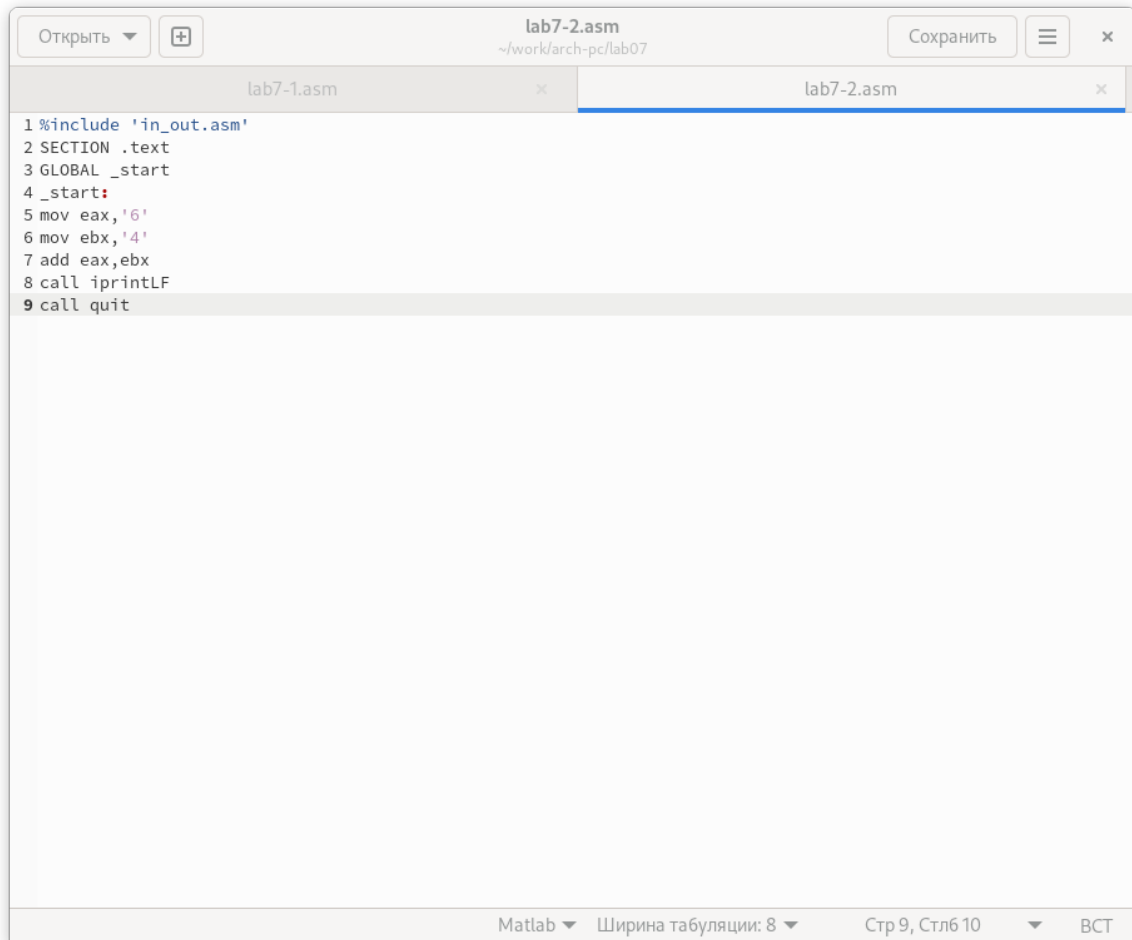
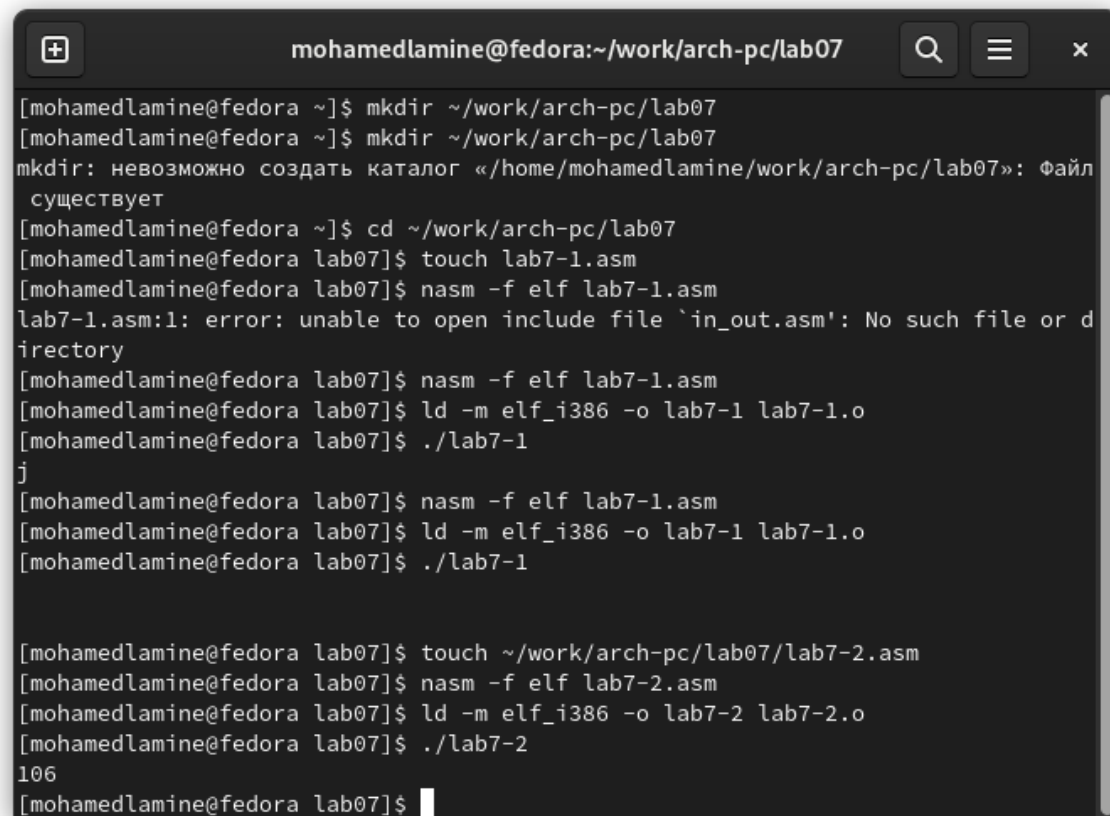


Рис. 4.5: Пример программы

A screenshot of a terminal window with a dark background. The title bar shows the user 'mohamedlamine' on a 'fedora' machine, in the directory '~/work/arch-pc/lab07'. The terminal contains the following commands and output:

```
[mohamedlamine@fedora ~]$ mkdir ~/work/arch-pc/lab07
[mohamedlamine@fedora ~]$ mkdir ~/work/arch-pc/lab07
mkdir: невозможно создать каталог «/home/mohamedlamine/work/arch-pc/lab07»: Файл
существует
[mohamedlamine@fedora ~]$ cd ~/work/arch-pc/lab07
[mohamedlamine@fedora lab07]$ touch lab7-1.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
lab7-1.asm:1: error: unable to open include file `in_out.asm': No such file or d
irectory
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mohamedlamine@fedora lab07]$ ./lab7-1
j
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mohamedlamine@fedora lab07]$ ./lab7-1

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

Рис. 4.6: Работа программы

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда `add` складывает коды символов '6' и '4' ( $54+52=106$ ). Однако, в отличие от программы из листинга 7.1, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

5. Аналогично предыдущему примеру изменим символы на числа. (рис. 4.7, 4.8)

Создайте исполняемый файл и запустите его. Какой результат будет получен при исполнении программы? – получили число 10

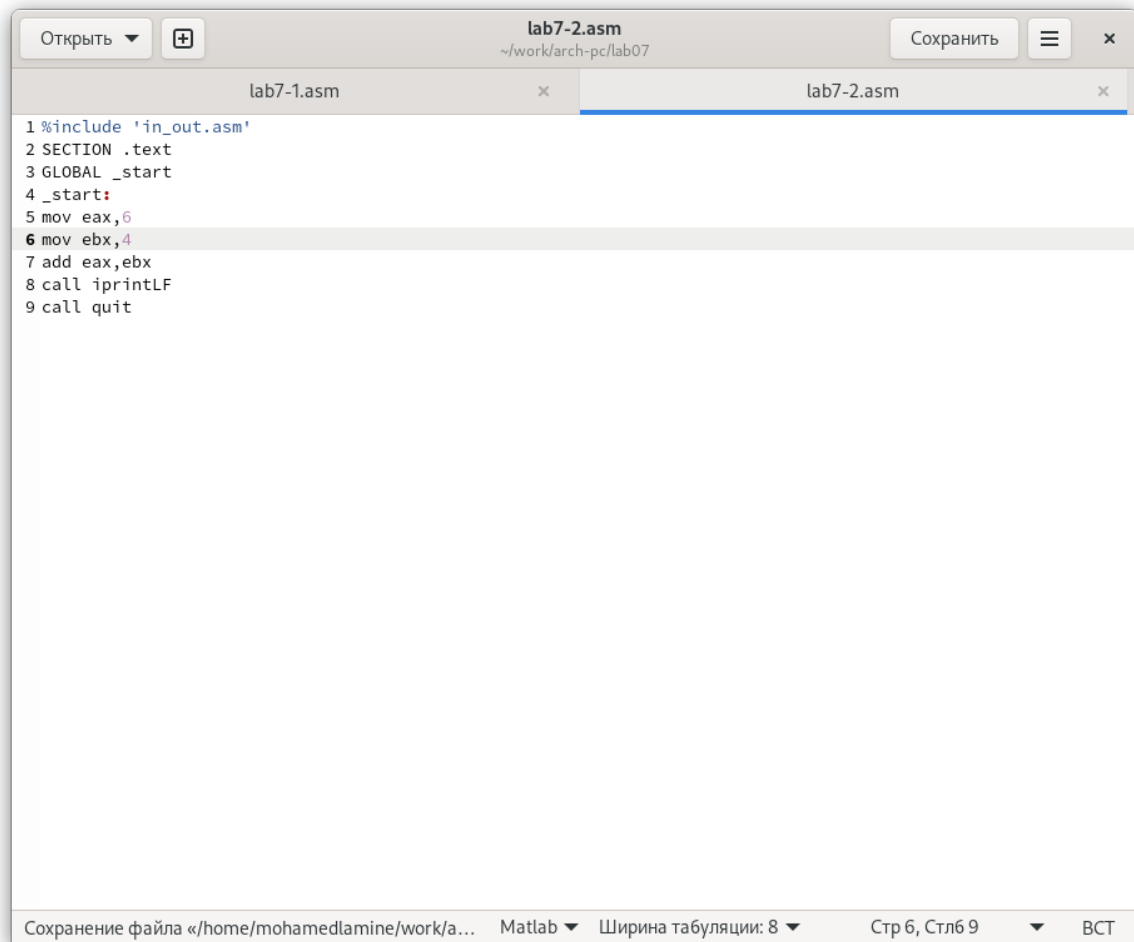
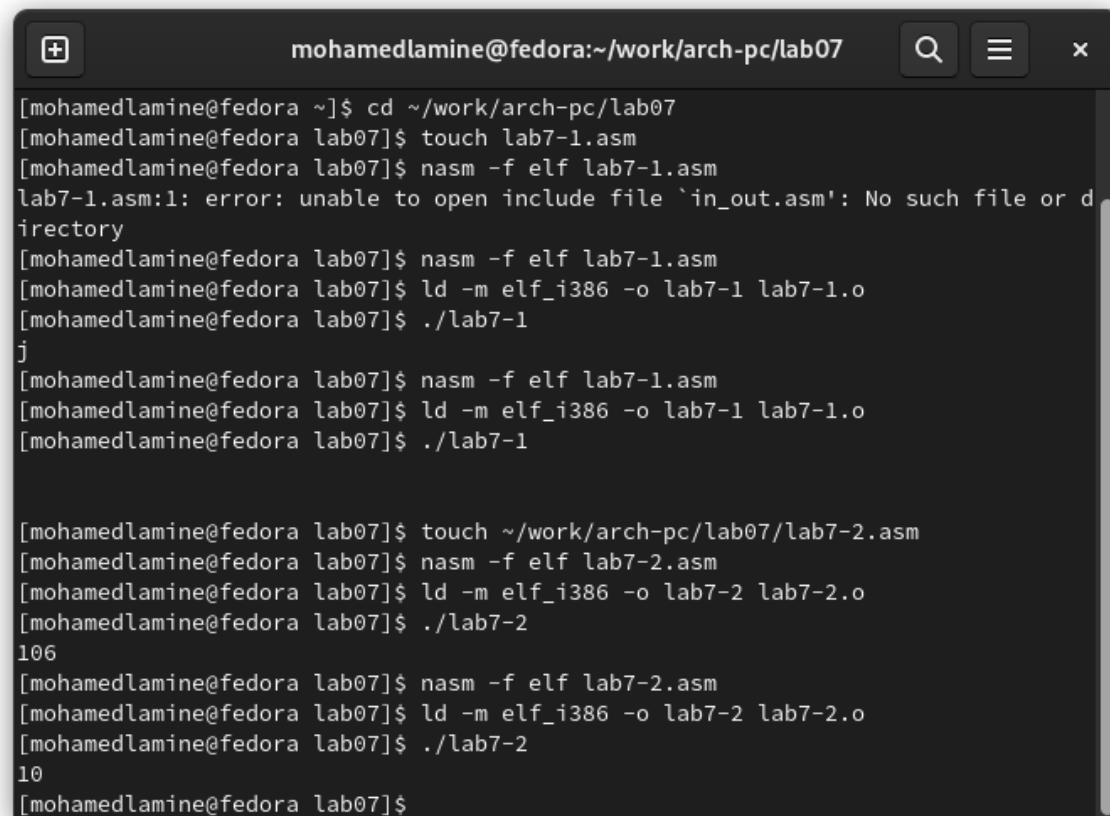


Рис. 4.7: Пример программы



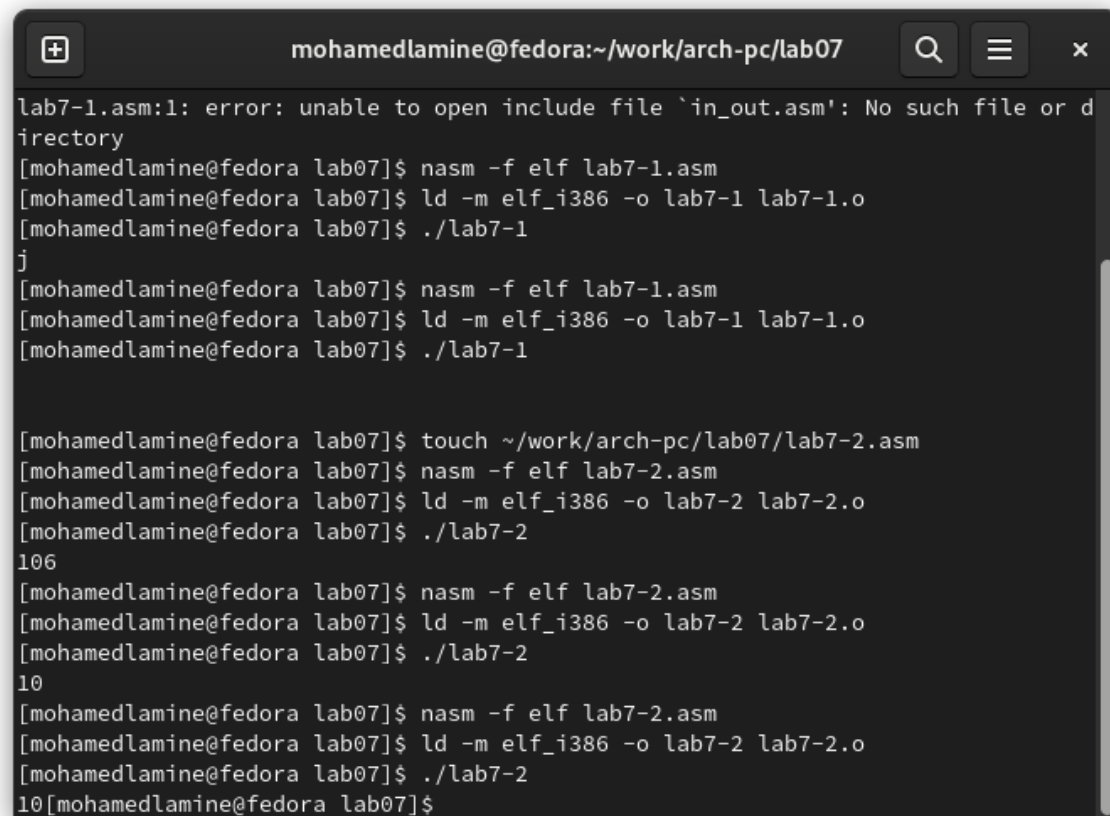


```
mohamedlamine@fedora:~/work/arch-pc/lab07
[mohamedlamine@fedora ~]$ cd ~/work/arch-pc/lab07
[mohamedlamine@fedora lab07]$ touch lab7-1.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
lab7-1.asm:1: error: unable to open include file `in_out.asm': No such file or directory
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mohamedlamine@fedora lab07]$ ./lab7-1
j
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mohamedlamine@fedora lab07]$ ./lab7-1

[mohamedlamine@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
106
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
10
[mohamedlamine@fedora lab07]$
```

Рис. 4.8: Работа программы

Замените функцию `iprintLF` на `iprint`. Создайте исполняемый файл и запустите его. Чем отличается вывод функций `iprintLF` и `iprint`? - Вывод отличается что нет переноса строки. (рис. 4.9)



```
mohamedlamine@fedora:~/work/arch-pc/lab07
lab7-1.asm:1: error: unable to open include file 'in_out.asm': No such file or directory
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mohamedlamine@fedora lab07]$ ./lab7-1
j
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-1.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[mohamedlamine@fedora lab07]$ ./lab7-1

[mohamedlamine@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
106
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
10
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
10[mohamedlamine@fedora lab07]$
```

Рис. 4.9: Работа программы

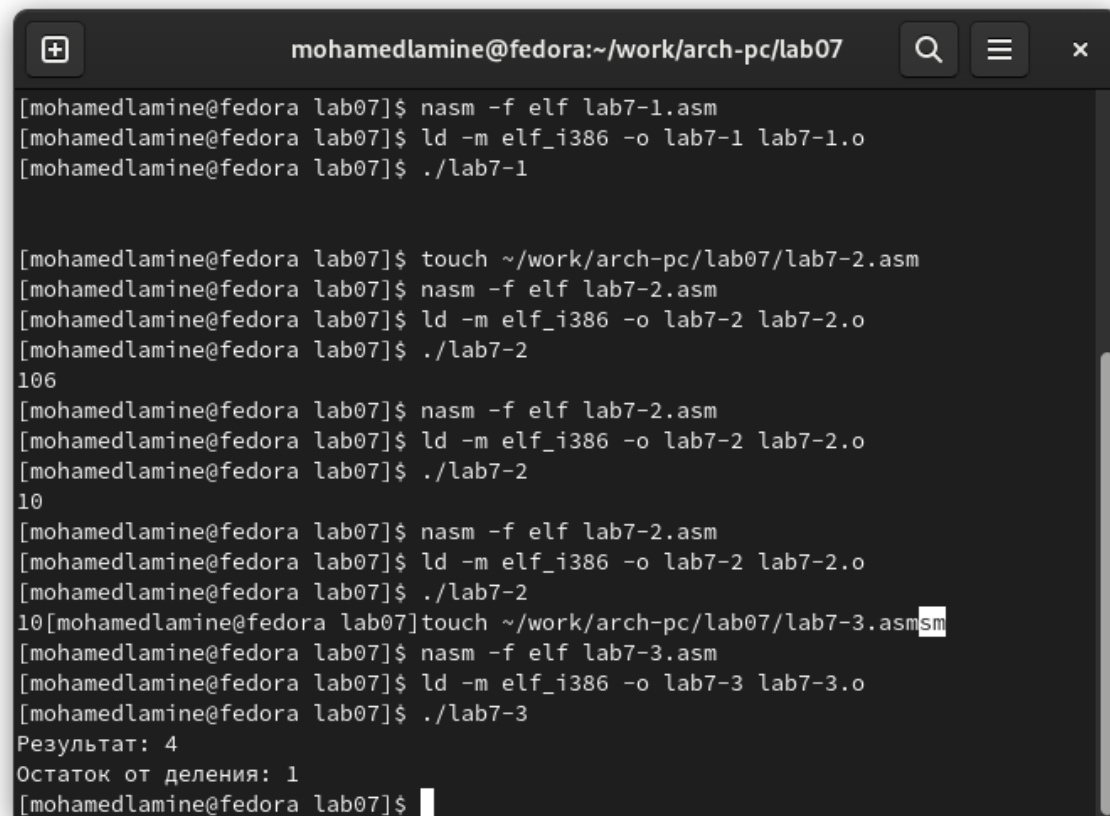
6. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения

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

. (рис. 4.10, рис. 4.11)

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,5 ; EAX=5
10 mov ebx,2 ; EBX=2
11 mul ebx ; EAX=EAX*EBX
12 add eax,3 ; EAX=EAX+3
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,3 ; EBX=3
15 div ebx ; EAX=EAX/3, EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
```

Рис. 4.10: Пример программы



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

[mohamedlamine@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
106
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
10
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
10[mohamedlamine@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-3.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-3.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[mohamedlamine@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[mohamedlamine@fedora lab07]$
```

Рис. 4.11: Работа программы

Измените текст программы для вычисления выражения

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

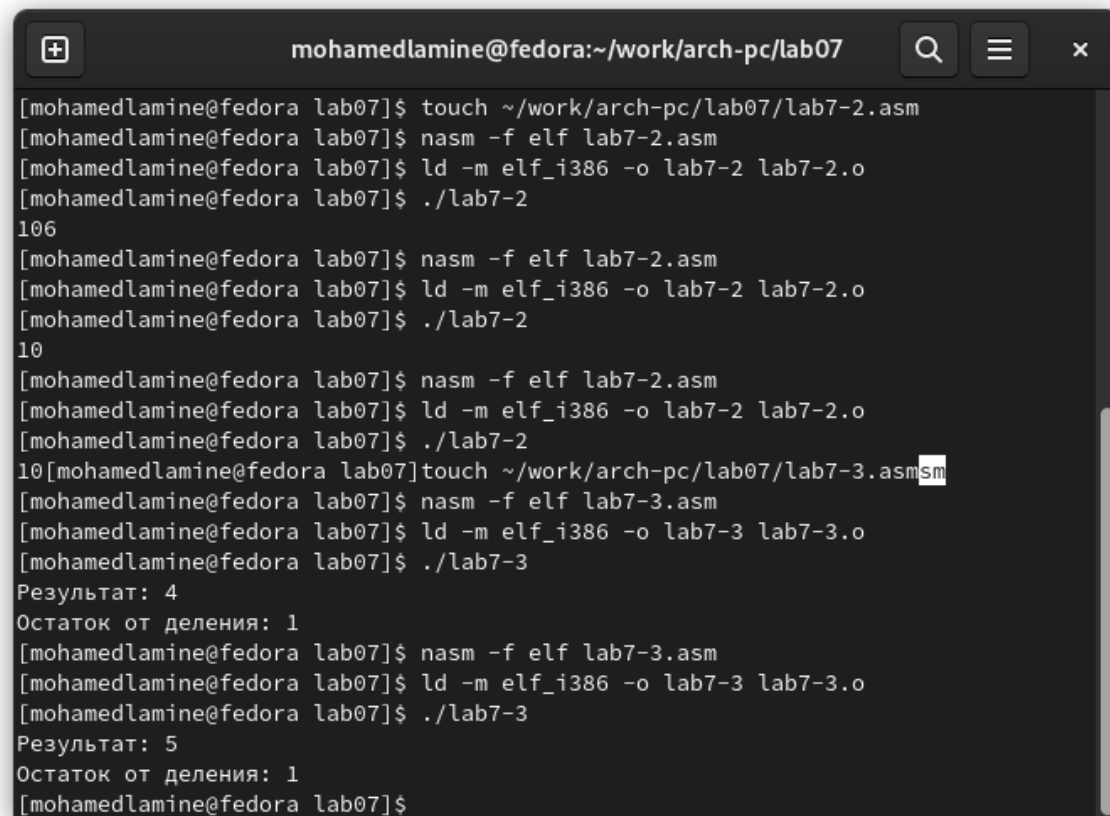
. Создайте исполняемый файл и проверьте его работу. (рис. 4.12, рис. 4.13)

The screenshot shows a MATLAB editor window with the title bar 'lab7-3.asm' and the path '~\work\arch-pc\lab07'. The window contains three tabs: 'lab7-1.asm', 'lab7-2.asm', and 'lab7-3.asm'. The 'lab7-3.asm' tab is active and displays the following assembly code:

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 ; ---- Вычисление выражения
9 mov eax,4 ;
10 mov ebx,6 ;
11 mul ebx ;
12 add eax,2 ;
13 xor edx,edx ; обнуляем EDX для корректной работы div
14 mov ebx,5 ;
15 div ebx ; EDX=остаток от деления
16 mov edi,eax ; запись результата вычисления в 'edi'
17 ; ---- Вывод результата на экран
18 mov eax,div ; вызов подпрограммы печати
19 call sprint ; сообщения 'Результат: '
20 mov eax,edi ; вызов подпрограммы печати значения
21 call iprintLF ; из 'edi' в виде символов
22 mov eax,rem ; вызов подпрограммы печати
23 call sprint ; сообщения 'Остаток от деления: '
24 mov eax,edx ; вызов подпрограммы печати значения
25 call iprintLF ; из 'edx' (остаток) в виде символов
26 call quit ; вызов подпрограммы завершения
```

The status bar at the bottom indicates 'Matlab', 'Ширина табуляции: 8', 'Стр 26, Стлб 42', and 'ВСТ'.

Рис. 4.12: Пример программы



```
mohamedlamine@fedora:~/work/arch-pc/lab07
[mohamedlamine@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
106
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
10
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
10[mohamedlamine@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-3.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-3.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[mohamedlamine@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-3.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[mohamedlamine@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[mohamedlamine@fedora lab07]$
```

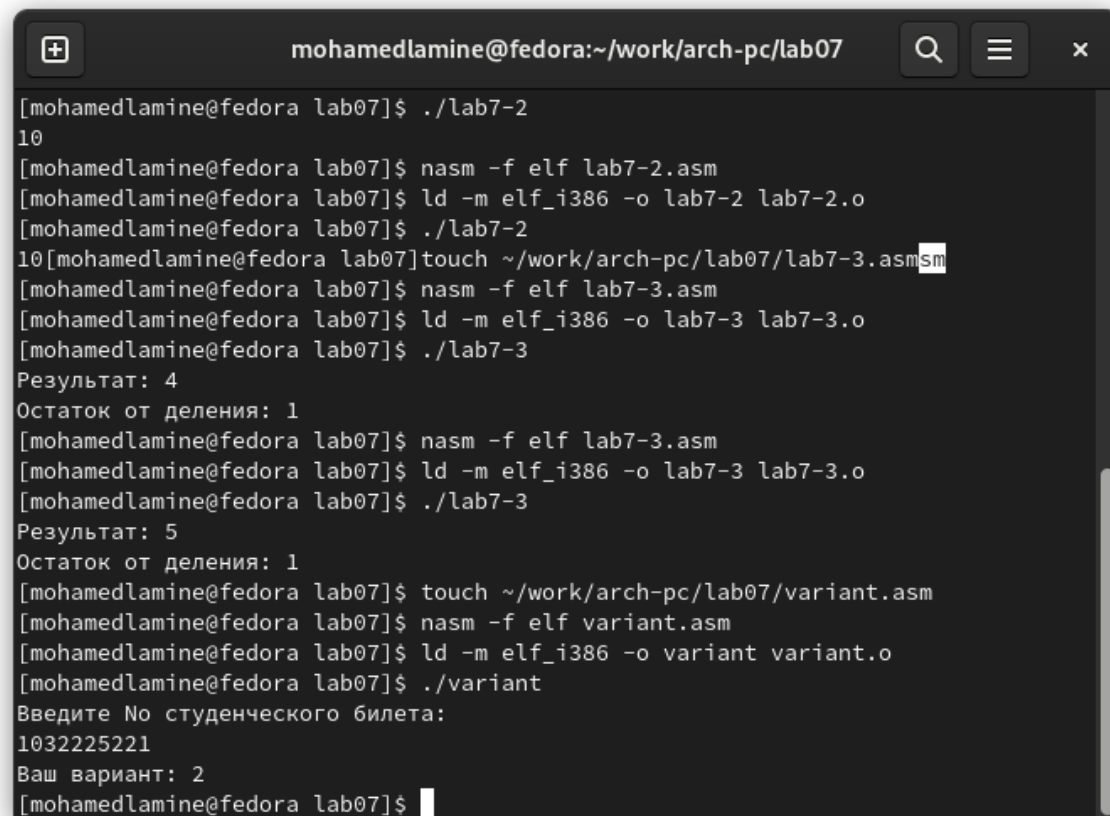
Рис. 4.13: Работа программы

7. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму: (рис. 4.14, рис. 4.15)

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите No студенческого билета: ',0
4 rem: DB 'Ваш вариант: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintf
12 mov ecx, x
13 mov edx, 80
14 call read
15 mov eax,x ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, `eax=x`
17 xor edx,edx
18 mov ebx,20
19 div ebx
20 inc edx
21 mov eax,rem
22 call sprintf
23 mov eax,edx
24 call iprintLF
25 call quit
```

Сохранение файла «/home/mohamedlamine/work/a... Matlab Ширина табуляции: 8 Стр 9, Стлб 8 ВСТ

Рис. 4.14: Пример программы



```
mohamedlamine@fedora:~/work/arch-pc/lab07
[mohamedlamine@fedora lab07]$ ./lab7-2
10
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-2.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[mohamedlamine@fedora lab07]$ ./lab7-2
10[mohamedlamine@fedora lab07]touch ~/work/arch-pc/lab07/lab7-3.asm
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-3.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[mohamedlamine@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[mohamedlamine@fedora lab07]$ nasm -f elf lab7-3.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[mohamedlamine@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[mohamedlamine@fedora lab07]$ touch ~/work/arch-pc/lab07/variant.asm
[mohamedlamine@fedora lab07]$ nasm -f elf variant.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[mohamedlamine@fedora lab07]$ ./variant
Введите No студенческого билета:
1032225221
Ваш вариант: 2
[mohamedlamine@fedora lab07]$
```

Рис. 4.15: Работа программы

- Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? – `mov eax,rem` – перекладывает в регистр значение переменной с фразой ‘Ваш вариант:’ `call sprint` – вызов подпрограммы вывода строки
- Для чего используются следующие инструкции? `nasm mov ecx, x mov edx, 80 call sread`

Считывает значение студбилета в переменную X из консоли

- Для чего используется инструкция “`call atoi`”? - эта подпрограмма переводит введенные символы в числовой формат
- Какие строки листинга 7.4 отвечают за вычисления варианта? `xor edx,edx mov ebx,20 div ebx`



- В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”? 1 байт AH 2 байта DX 4 байта EDX – наш случай
- Для чего используется инструкция “inc edx”? по формуле вычисления варианта нужно прибавить единицу
- Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений mov eax,edx – результат перекладывается в регистр eax call iprintLF – вызов подпрограммы вывода

8. Написать 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  $f(x)$  выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений  $x_1$  и  $x_2$  из 6.3. (рис. 4.16, рис. 4.17)

Получили вариант 2 -

$$(12x + 3)^5$$

для  $x=1$  и  $x=3$

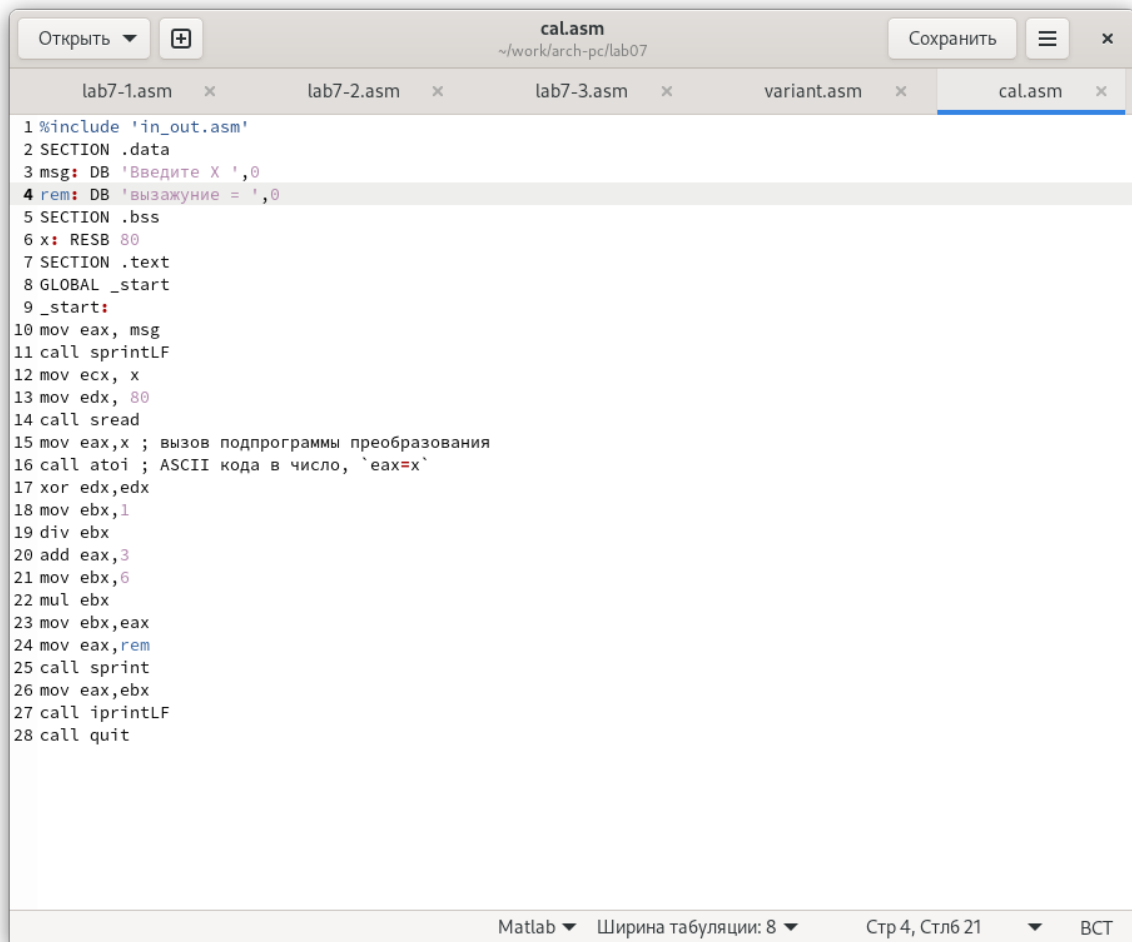
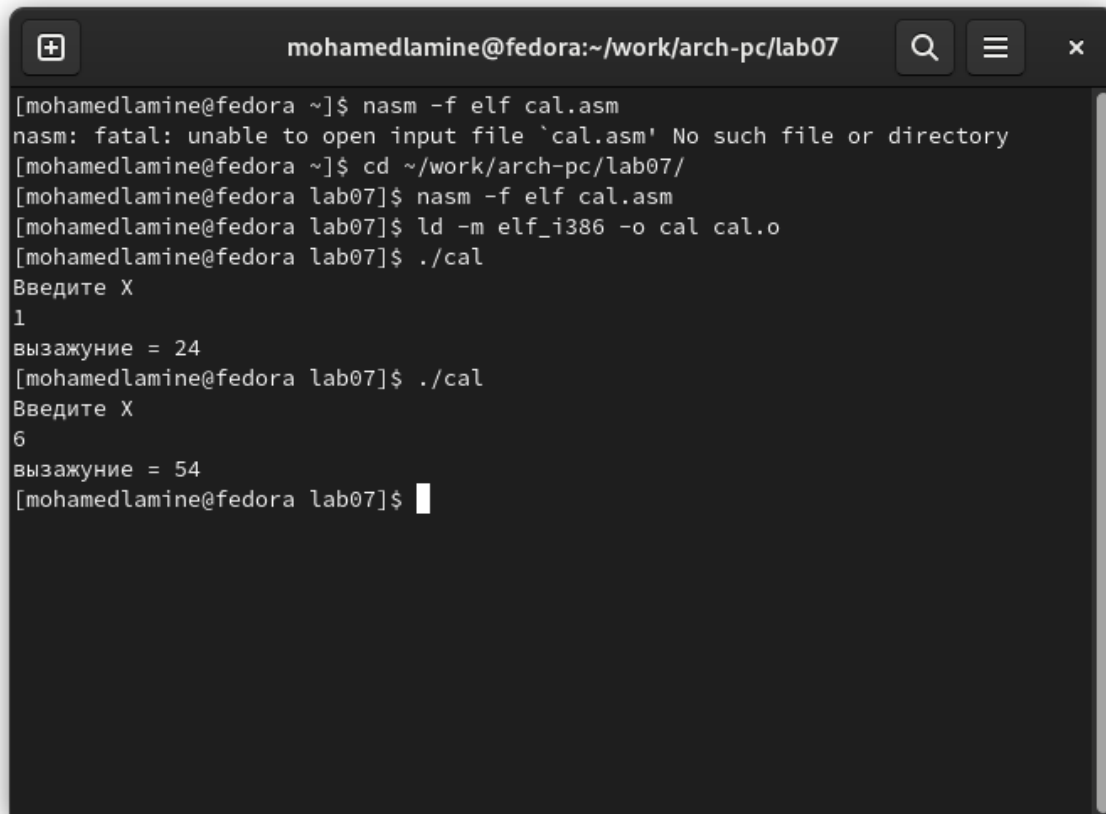


Рис. 4.16: Пример программы



```
mohamedlamine@fedora:~/work/arch-pc/lab07
[mohamedlamine@fedora ~]$ nasm -f elf cal.asm
nasm: fatal: unable to open input file `cal.asm' No such file or directory
[mohamedlamine@fedora ~]$ cd ~/work/arch-pc/lab07/
[mohamedlamine@fedora lab07]$ nasm -f elf cal.asm
[mohamedlamine@fedora lab07]$ ld -m elf_i386 -o cal cal.o
[mohamedlamine@fedora lab07]$ ./cal
Введите X
1
вызажуние = 24
[mohamedlamine@fedora lab07]$ ./cal
Введите X
6
вызажуние = 54
[mohamedlamine@fedora lab07]$
```

Рис. 4.17: Работа программы

## **5 Выводы**

Изучили работу с арифметическими операциями

# Список литературы

1. Расширенный ассемблер: NASM
2. MASM, TASM, FASM, NASM под Windows и Linux