

# **Отчёта по лабораторной работе 6**

**Освоение арифметических инструкций языка ассемблера NASM.**

Мельников Максим

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	25
	Список литературы	26

## Список иллюстраций

4.1	Пример программы . . . . .	9
4.2	Работа программы . . . . .	10
4.3	Пример программы . . . . .	11
4.4	Работа программы . . . . .	12
4.5	Пример программы . . . . .	13
4.6	Работа программы . . . . .	13
4.7	Пример программы . . . . .	14
4.8	Работа программы . . . . .	15
4.9	Работа программы . . . . .	15
4.10	Пример программы . . . . .	16
4.11	Работа программы . . . . .	17
4.12	Пример программы . . . . .	18
4.13	Работа программы . . . . .	19
4.14	Пример программы . . . . .	20
4.15	Работа программы . . . . .	21
4.16	Пример программы . . . . .	23
4.17	Работа программы . . . . .	24

## 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.

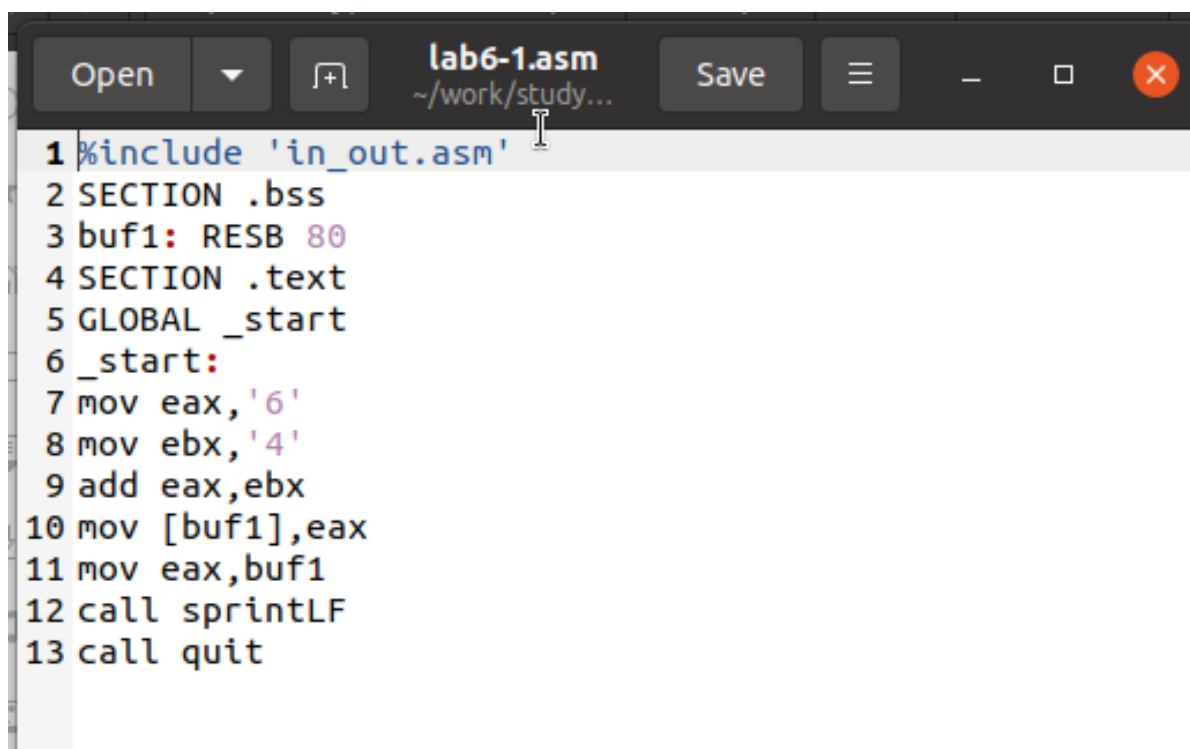
Две основные команды умножения: `mul` – умножение беззнаковых чисел, `imul` – умножение знаковых чисел. Команда содержит один операнд – регистр или адрес памяти. В зависимости от размера операнда данные помещаются: в `ax`, `dx : ax`, `edx : eax`, `rdx : rax`. Например: `mull ll` – содержимое памяти с 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`.



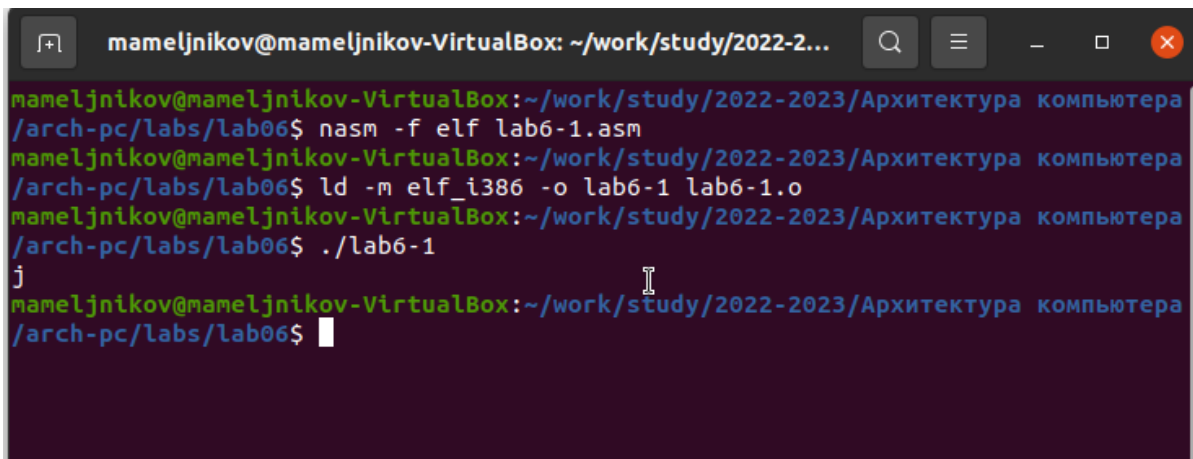
## 4 Выполнение лабораторной работы

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)



```
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 sprintLF
13 call quit
```

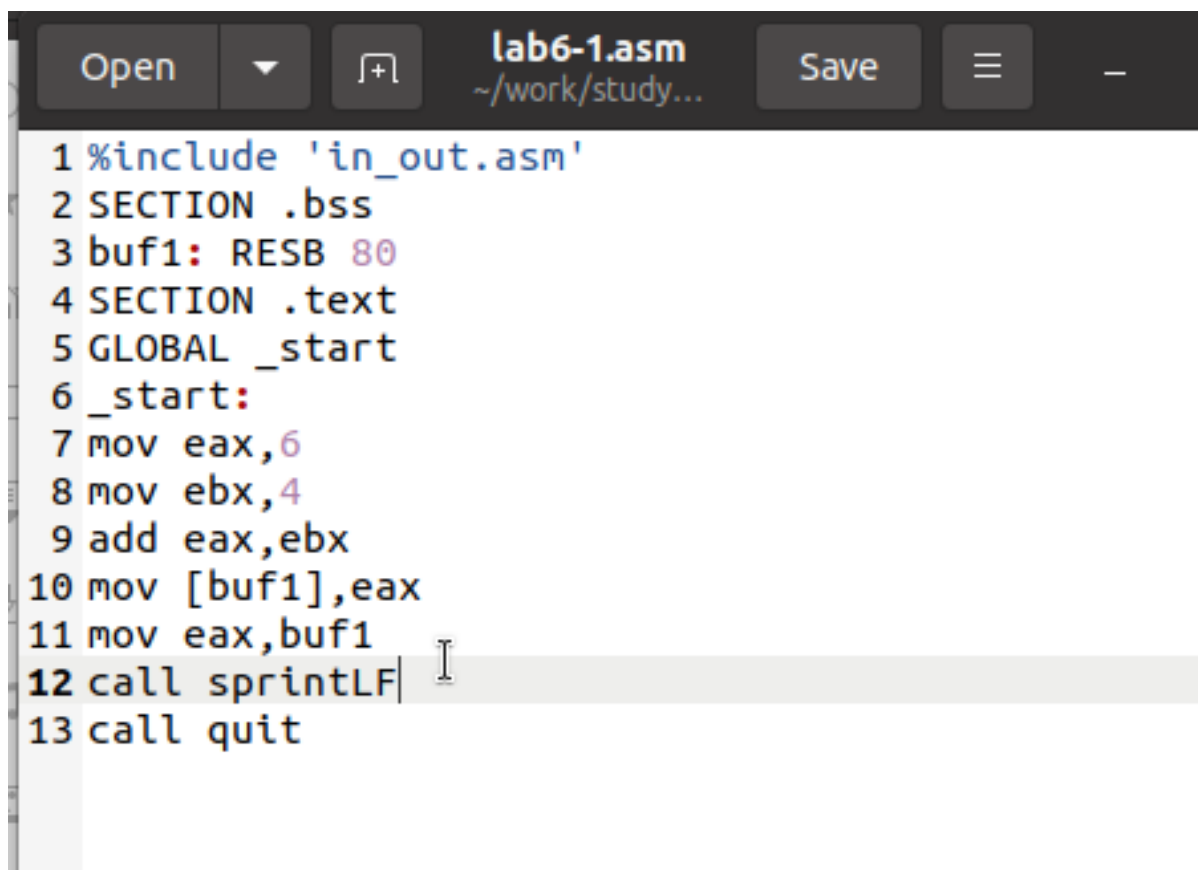
Рис. 4.1: Пример программы

A screenshot of a terminal window with a dark background. The window title is "mameljnikov@mameljnikov-VirtualBox: ~/work/study/2022-2...". The terminal shows the following commands and output:

```
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab06$ nasm -f elf lab6-1.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab06$ ./lab6-1
j
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab06$
```

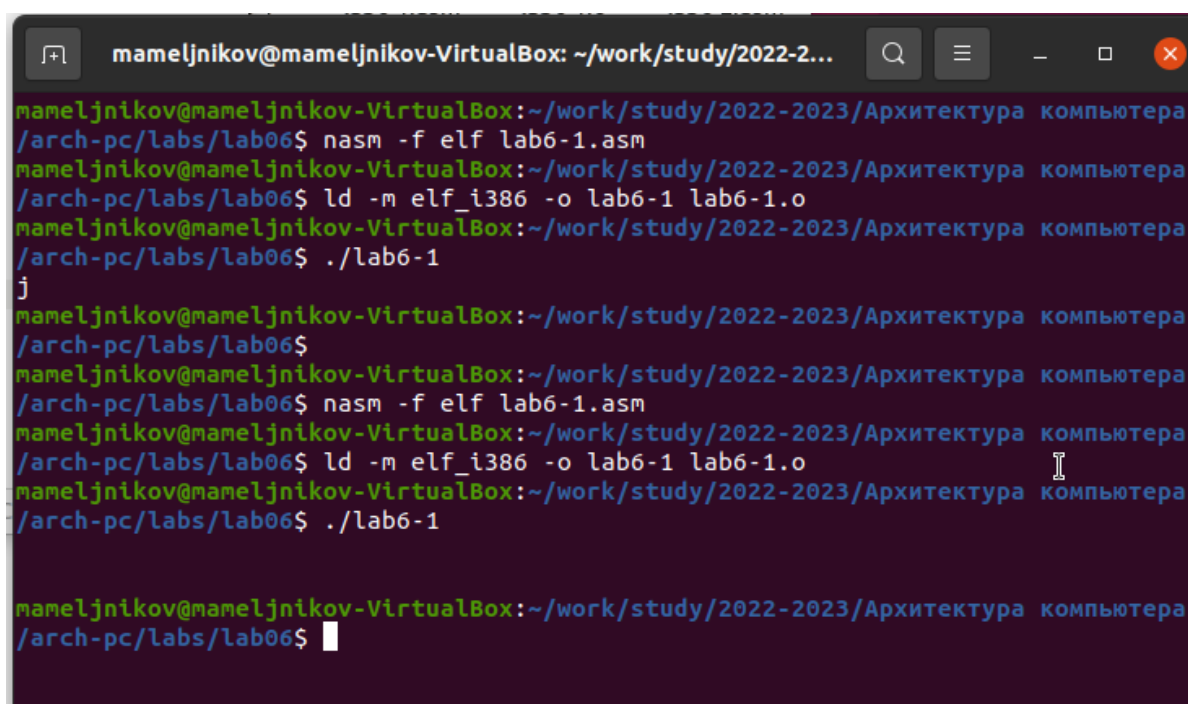
Рис. 4.2: Работа программы

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



```
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 sprintLF
13 call quit
```

Рис. 4.3: Пример программы

A screenshot of a terminal window titled 'mameljnikov@mameljnikov-VirtualBox: ~/work/study/2022-2...'. The terminal shows a series of commands and their outputs. The commands are: 'nasm -f elf lab6-1.asm', 'ld -m elf\_i386 -o lab6-1 lab6-1.o', and './lab6-1'. The outputs are: 'j', 'mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06\$', 'mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06\$', 'mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06\$', and 'mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06\$'. The terminal has a dark background with green and yellow text. The window has standard Linux window controls at the top right.

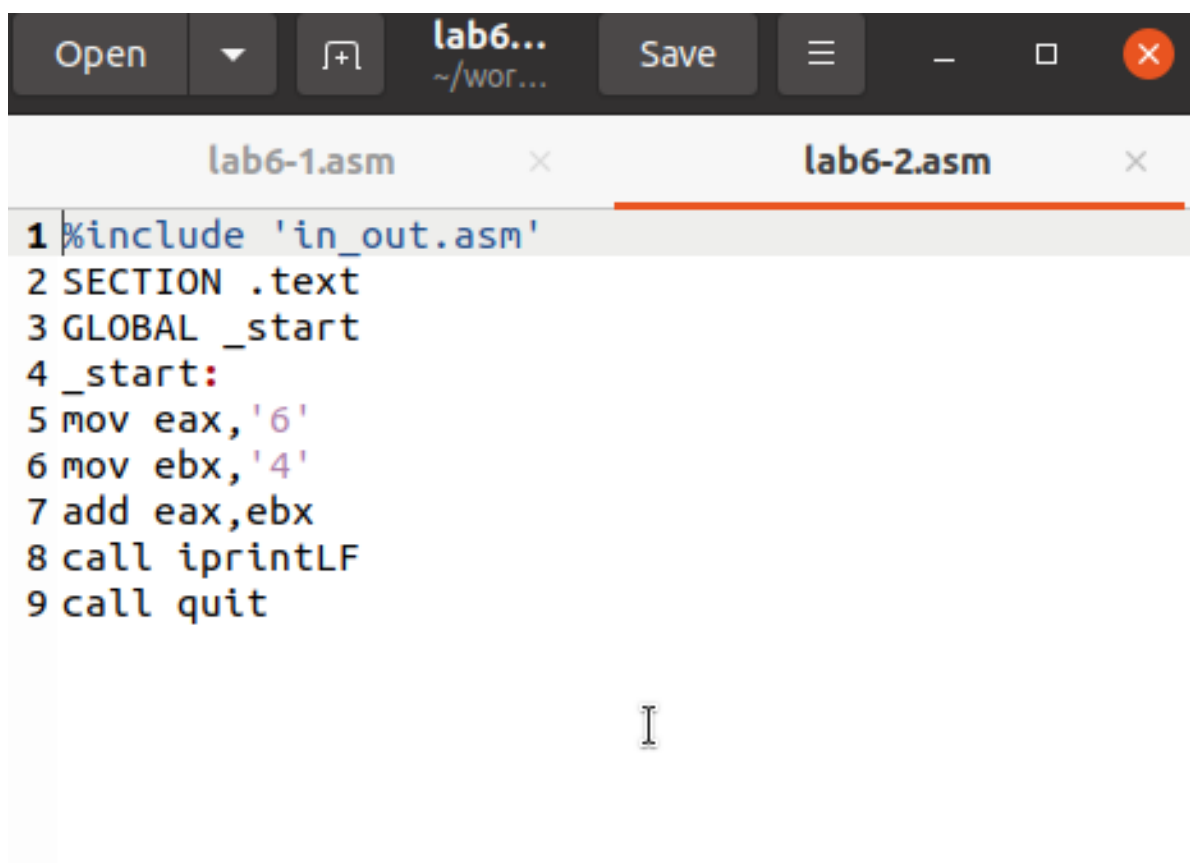
```
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06$ nasm -f elf lab6-1.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06$ ./lab6-1
j
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06$ nasm -f elf lab6-1.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06$ ./lab6-1

mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера /arch-pc/labs/lab06$
```

Рис. 4.4: Работа программы

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

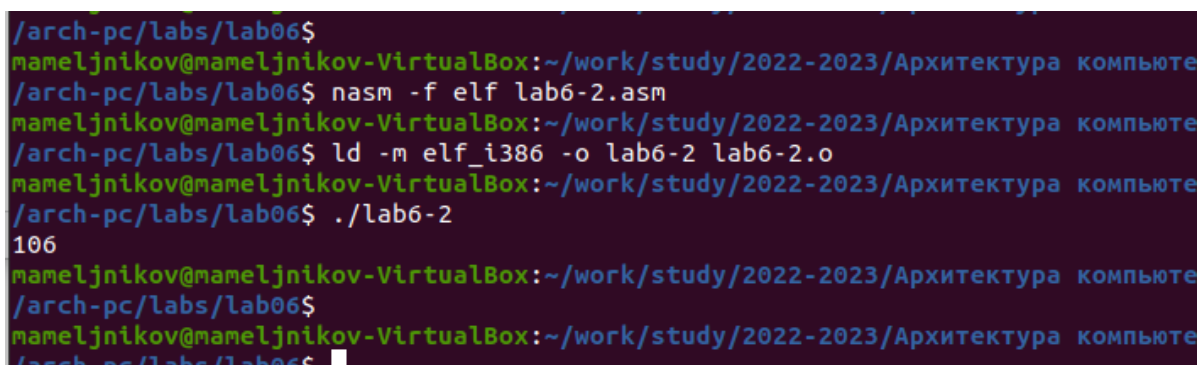
4. Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 7.1 с использованием этих функций. (рис. 4.5, 4.6)



The screenshot shows a code editor window with two tabs: 'lab6-1.asm' and 'lab6-2.asm'. The 'lab6-2.asm' tab is active and contains the following assembly code:

```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, '6'
6 mov ebx, '4'
7 add eax, ebx
8 call iprintLF
9 call quit
```

Рис. 4.5: Пример программы



The screenshot shows a terminal window with the following commands and output:

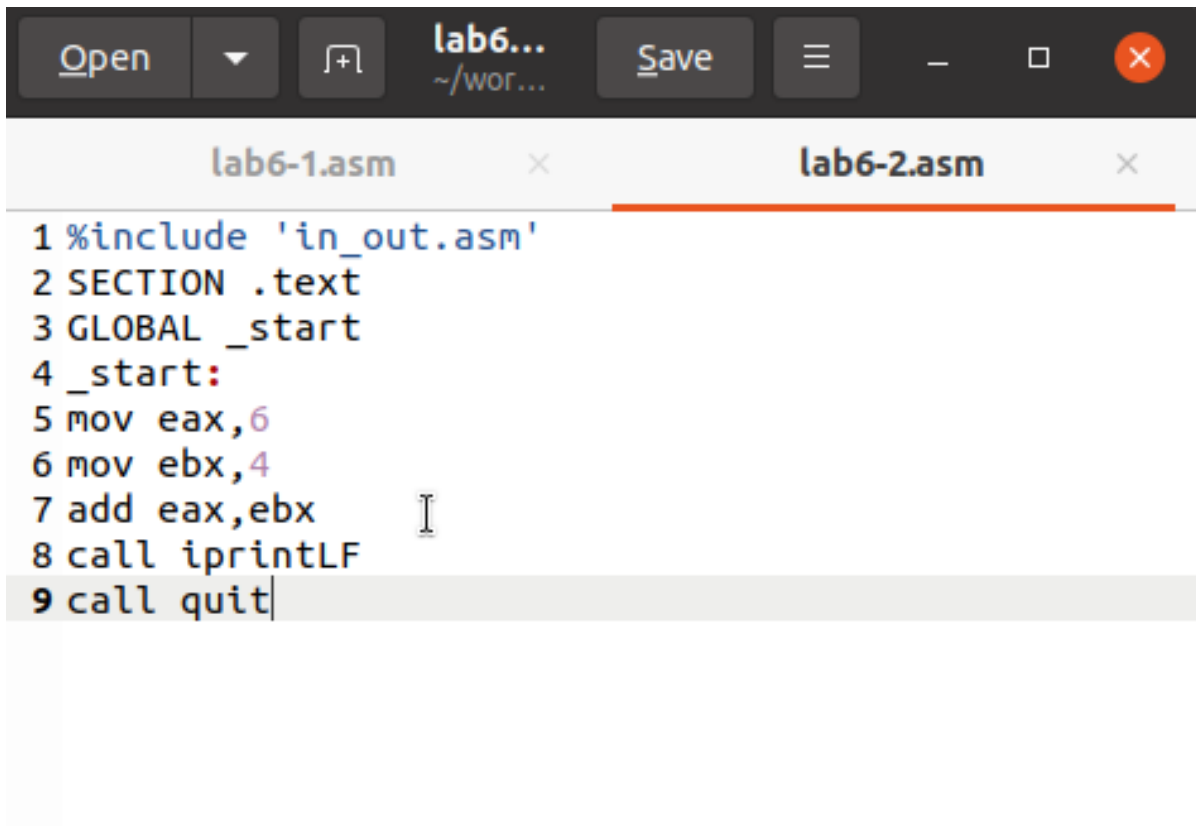
```
/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьюте
/arch-pc/labs/lab06$ nasm -f elf lab6-2.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьюте
/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьюте
/arch-pc/labs/lab06$ ./lab6-2
106
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьюте
/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьюте
/arch-pc/labs/lab06$
```

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



```
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintLF
9 call quit
```

Рис. 4.7: Пример программы

```

/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ nasm -f elf lab6-2.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ./lab6-2
106
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ nasm -f elf lab6-2.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ./lab6-2
10
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$

```

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)

```

mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ nasm -f elf lab6-2.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ./lab6-2
10
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ nasm -f elf lab6-2.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ./lab6-2
10mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьюте
ра
/arch-pc/labs/lab06$

```

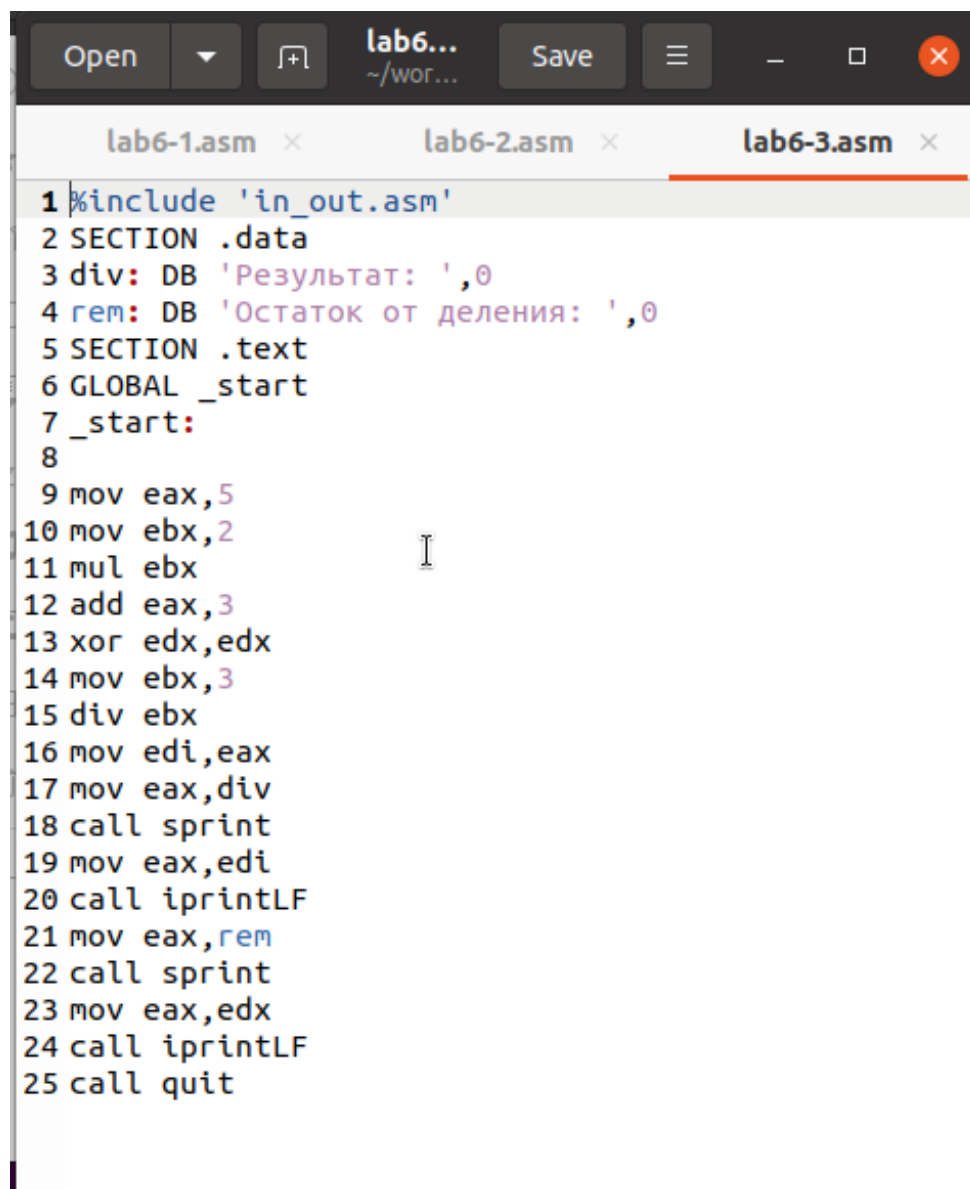
Рис. 4.9: Работа программы

6. В качестве примера выполнения арифметических операций в NASM приве-

дем программу вычисления арифметического выражения

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

. (рис. 4.10, рис. 4.11)



```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8
9 mov eax,5
10 mov ebx,2
11 mul ebx
12 add eax,3
13 xor edx,edx
14 mov ebx,3
15 div ebx
16 mov edi,eax
17 mov eax,div
18 call sprint
19 mov eax,edi
20 call iprintLF
21 mov eax,rem
22 call sprint
23 mov eax,edx
24 call iprintLF
25 call quit
```

Рис. 4.10: Пример программы



```

/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компь
/arch-pc/labs/lab06$ nasm -f elf lab6-3.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компь
/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компь
/arch-pc/labs/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компь
/arch-pc/labs/lab06$

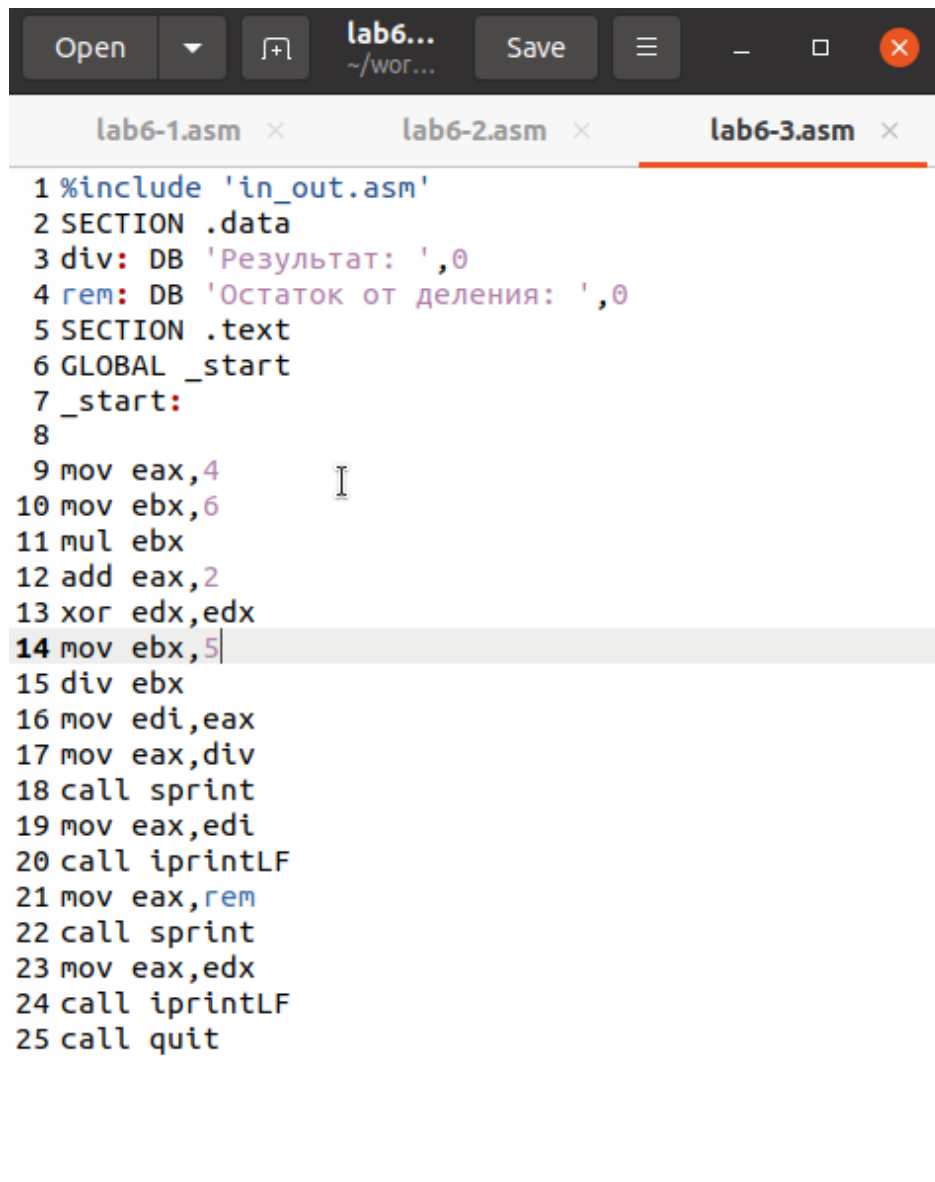
```

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)



```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8
9 mov eax,4
10 mov ebx,6
11 mul ebx
12 add eax,2
13 xor edx,edx
14 mov ebx,5
15 div ebx
16 mov edi,eax
17 mov eax,div
18 call sprint
19 mov eax,edi
20 call iprintLF
21 mov eax,rem
22 call sprint
23 mov eax,edx
24 call iprintLF
25 call quit
```

Рис. 4.12: Пример программы

```

mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ nasm -f elf lab6-3.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ nasm -f elf lab6-3.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$

```

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)

The image shows a screenshot of an assembly code editor window. The window has a dark theme and a menu bar with 'Open', 'Save', and other standard icons. Below the menu bar, there are three tabs: 'lab6-2.asm', 'lab6-3.asm', and 'variant.asm'. The 'variant.asm' tab is active and highlighted with a red underline. The code is written in assembly language and is as follows:

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № 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 sread
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
26 |
```

The code is a program that prompts the user to enter a student ticket number. It reads the input, converts it from ASCII to a decimal number using the `atoi` function, and then calculates the variant number by dividing the ticket number by 20 and taking the remainder. The result is then printed using the `iprintLF` function. The program ends with a `quit` call.

Рис. 4.14: Пример программы

```
mameljnikov@mameljnikov-VirtualBox: ~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ nasm -f elf variant.asm
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ld -m elf_i386 -o variant variant.o
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
/arch-pc/labs/lab06$ ./variant
Введите № студенческого билета:
1132222837
Ваш вариант: 18
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера
```

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

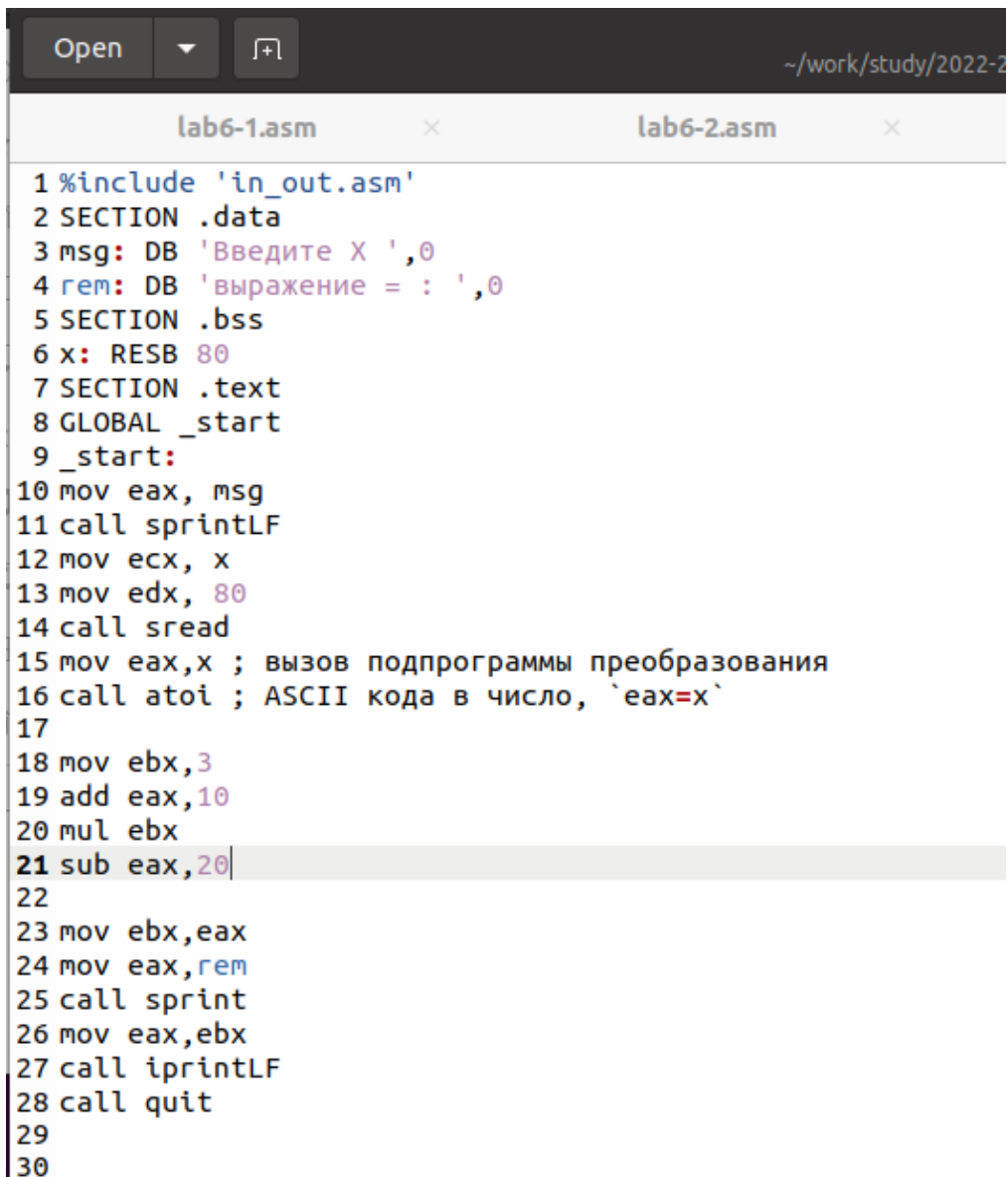
- Для чего используется инструкция “`call atoi`”? - эта подпрограмма переводит введенные символы в числовой формат
- Какие строки листинга 7.4 отвечают за вычисления варианта? `xor edx,edx mov ebx,20 div ebx`
- В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? 1 байт 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 вычисления `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 18 -

$$3(x + 10) - 20$$

для  $x=1$  и  $5$



```
Open  ▾  [icon]  ~/work/study/2022-2
lab6-1.asm  ×  lab6-2.asm  ×

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите X ',0
4 rem: DB 'выражение = : ',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 sread
15 mov eax, x ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, `eax=x`
17
18 mov ebx, 3
19 add eax, 10
20 mul ebx
21 sub eax, 20
22
23 mov ebx, eax
24 mov eax, rem
25 call sprintf
26 mov eax, ebx
27 call iprintLF
28 call quit
29
30
```

Рис. 4.16: Пример программы

```
/arch-pc/labs/lab06$  
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера  
/arch-pc/labs/lab06$ nasm -f elf calc.asm  
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера  
/arch-pc/labs/lab06$ ld -m elf_i386 -o calc calc.o  
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера  
/arch-pc/labs/lab06$ ./calc  
Введите X  
1  
выражение = : 13  
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера  
/arch-pc/labs/lab06$ ./calc  
Введите X  
5  
выражение = : 25  
mameljnikov@mameljnikov-VirtualBox:~/work/study/2022-2023/Архитектура компьютера  
/arch-pc/labs/lab06$
```

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