

Отчёт по лабораторной работе 6

Архитектура компьютера

Артем Олейников

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Задание для самостоятельной работы	21
3	Выводы	24

Список иллюстраций

2.1	Код программы lab6-1.asm	7
2.2	Компиляция и запуск программы lab6-1.asm	8
2.3	Код программы lab6-1.asm	9
2.4	Компиляция и запуск программы lab6-1.asm	9
2.5	Код программы lab6-2.asm	10
2.6	Компиляция и запуск программы lab6-2.asm	11
2.7	Код программы lab6-2.asm	12
2.8	Компиляция и запуск программы lab6-2.asm	13
2.9	Код программы lab6-2.asm	14
2.10	Компиляция и запуск программы lab6-2.asm	14
2.11	Код программы lab6-3.asm	15
2.12	Компиляция и запуск программы lab6-3.asm	16
2.13	Код программы lab6-3.asm	17
2.14	Компиляция и запуск программы lab6-3.asm	17
2.15	Код программы variant.asm	19
2.16	Компиляция и запуск программы variant.asm	19
2.17	Код программы program.asm	22
2.18	Компиляция и запуск программы program.asm	23

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

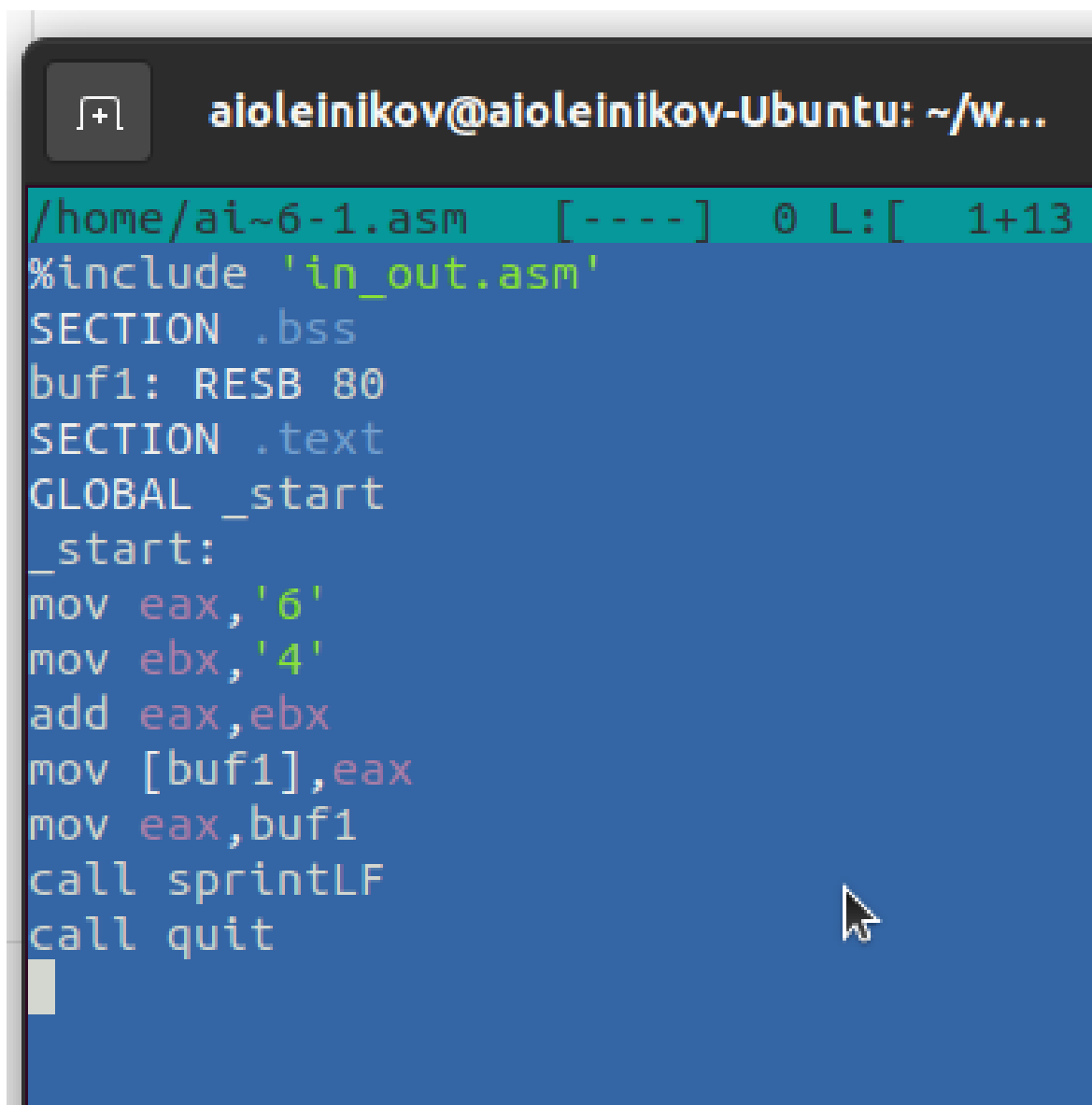
2 Выполнение лабораторной работы

Я создал папку для хранения файлов шестой лабораторной работы, перешел в нее и сформировал файл с исходным кодом lab6-1.asm.

В ходе этой лабораторной мы изучим примеры программ, демонстрирующих вывод символов и цифровых данных на экран. Эти программы будут оперировать данными, помещенными в регистр `eax`.

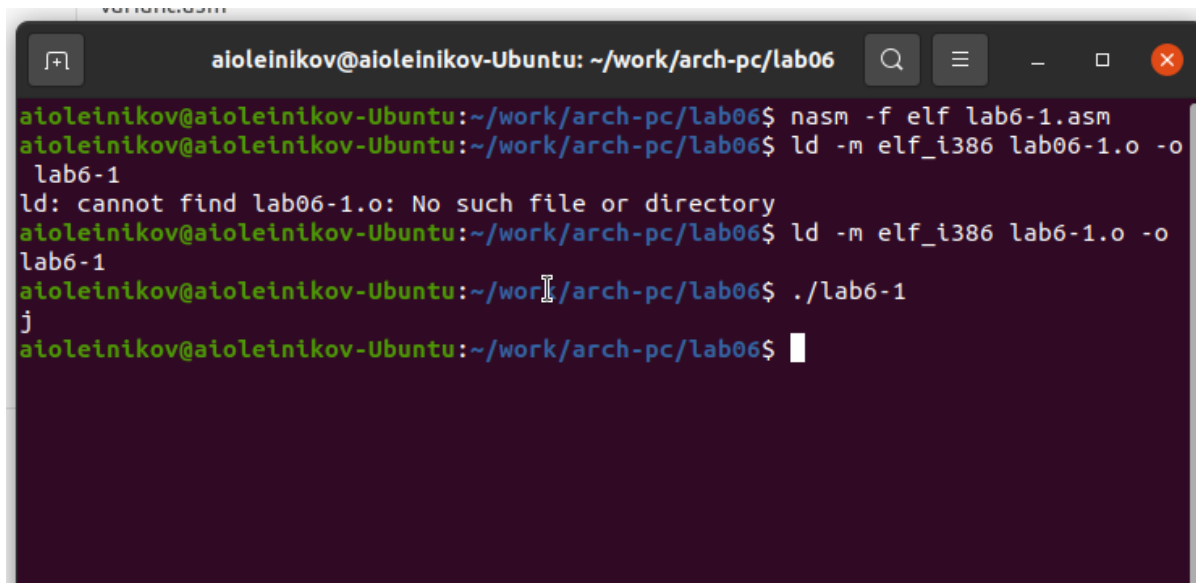
В одной из программ мы помещаем символ '6' в регистр `eax` с помощью инструкции (`mov eax, '6'`) и символ '4' в регистр `ebx` (`mov ebx, '4'`). После этого выполняем сложение значений, находящихся в регистрах `eax` и `ebx` (`add eax, ebx`), и отображаем полученный итог.

Чтобы воспользоваться функцией `sprintf`, которая ожидает адрес в регистре `eax`, мы применяем вспомогательную переменную. Сначала мы копируем содержимое регистра `eax` в переменную `buf1` (`mov [buf1], eax`), затем загружаем адрес `buf1` обратно в регистр `eax` (`mov eax, buf1`) и выполняем вызов функции `sprintf`.

A terminal window with a dark background. The title bar shows a window icon, a plus sign, and the text 'aioleinikov@aioleinikov-Ubuntu: ~/w...'. The terminal content shows assembly code for a file named 'lab6-1.asm'. The code includes a header line with file name, line numbers, and a line count. It then includes 'in_out.asm', defines a .bss section with a buffer 'buf1' of 80 bytes, and a .text section. The main code starts at '_start:', moves '6' to 'eax', '4' to 'ebx', adds them, stores the result in 'buf1', calls 'sprintf' (labeled as 'sprintfLF' in the image), and finally calls 'quit'.

```
/home/ai~6-1.asm  [----]  0  L:[  1+13
%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 sprintfLF
call quit
```

Рис. 2.1: Код программы lab6-1.asm

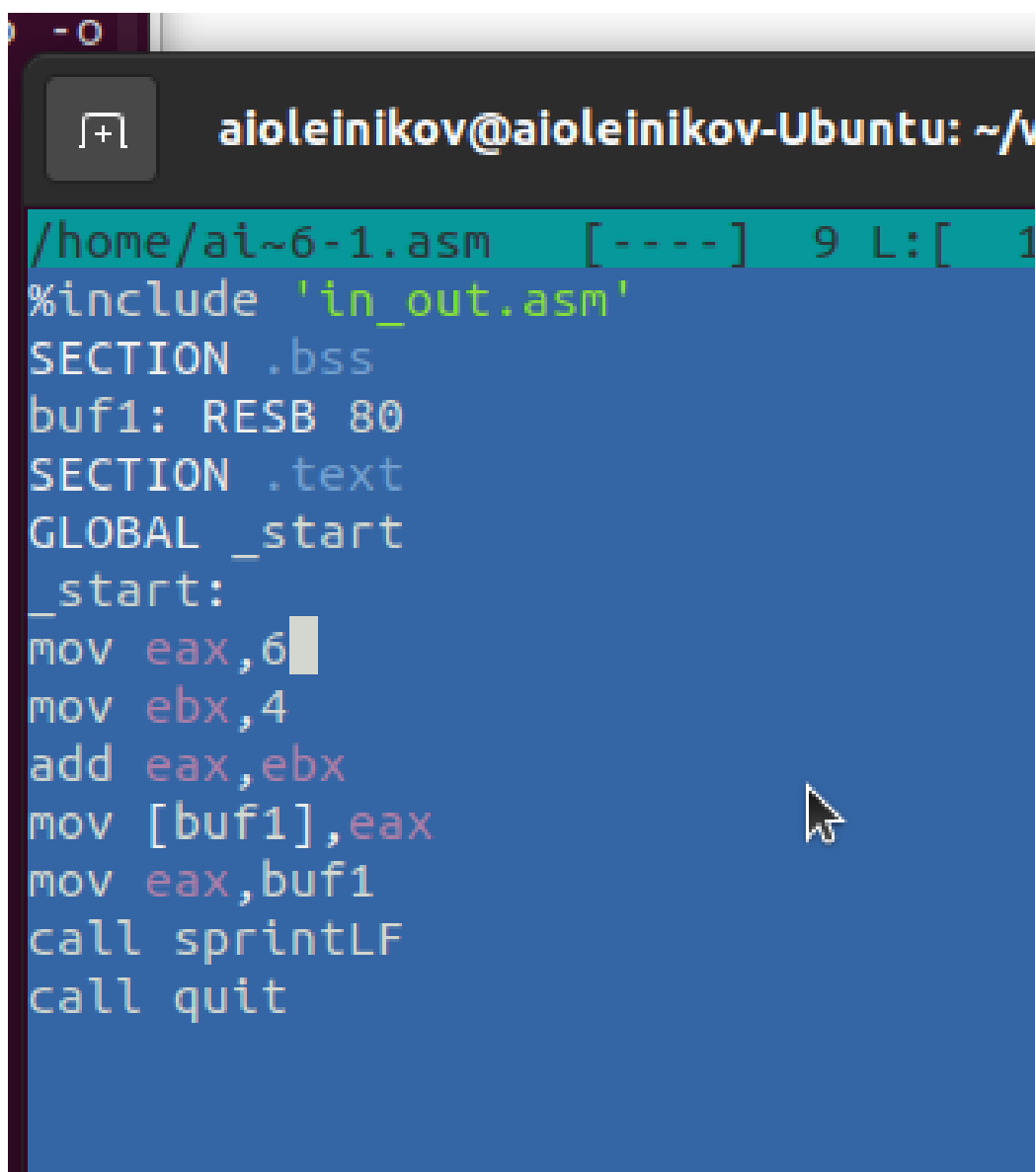
A terminal window titled 'aioleinikov@aioleinikov-Ubuntu: ~/work/arch-pc/lab06'. The terminal shows the following commands and output:

```
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab6-1
ld: cannot find lab06-1.o: No such file or directory
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ./lab6-1
j
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.2: Компиляция и запуск программы lab6-1.asm

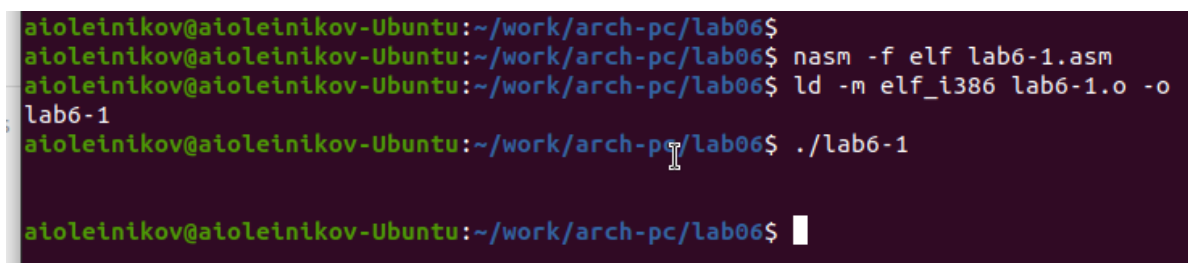
В этом примере мы ожидаем на выходе число 10 после сложения значений регистров `eax` и `ebx`. Тем не менее, на экране появится символ 'j', так как двоичный код символа '6' равен 00110110 (что соответствует 54 в десятичной системе), а символа '4' – 00110100 (или 52 в десятичной системе). Инструкция `add eax, ebx` приводит к записи в регистр `eax` суммы этих кодов – 01101010 (или 106 в десятичной системе), что соответствует коду символа 'j'.

После этого я внес изменения в код программы, заменив символы на числовые значения в регистрах.



```
aioleinikov@aioleinikov-Ubuntu: ~/work/arch-pc/lab06
/home/ai~6-1.asm [----] 9 L:[ 1
#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. 2.3: Код программы lab6-1.asm



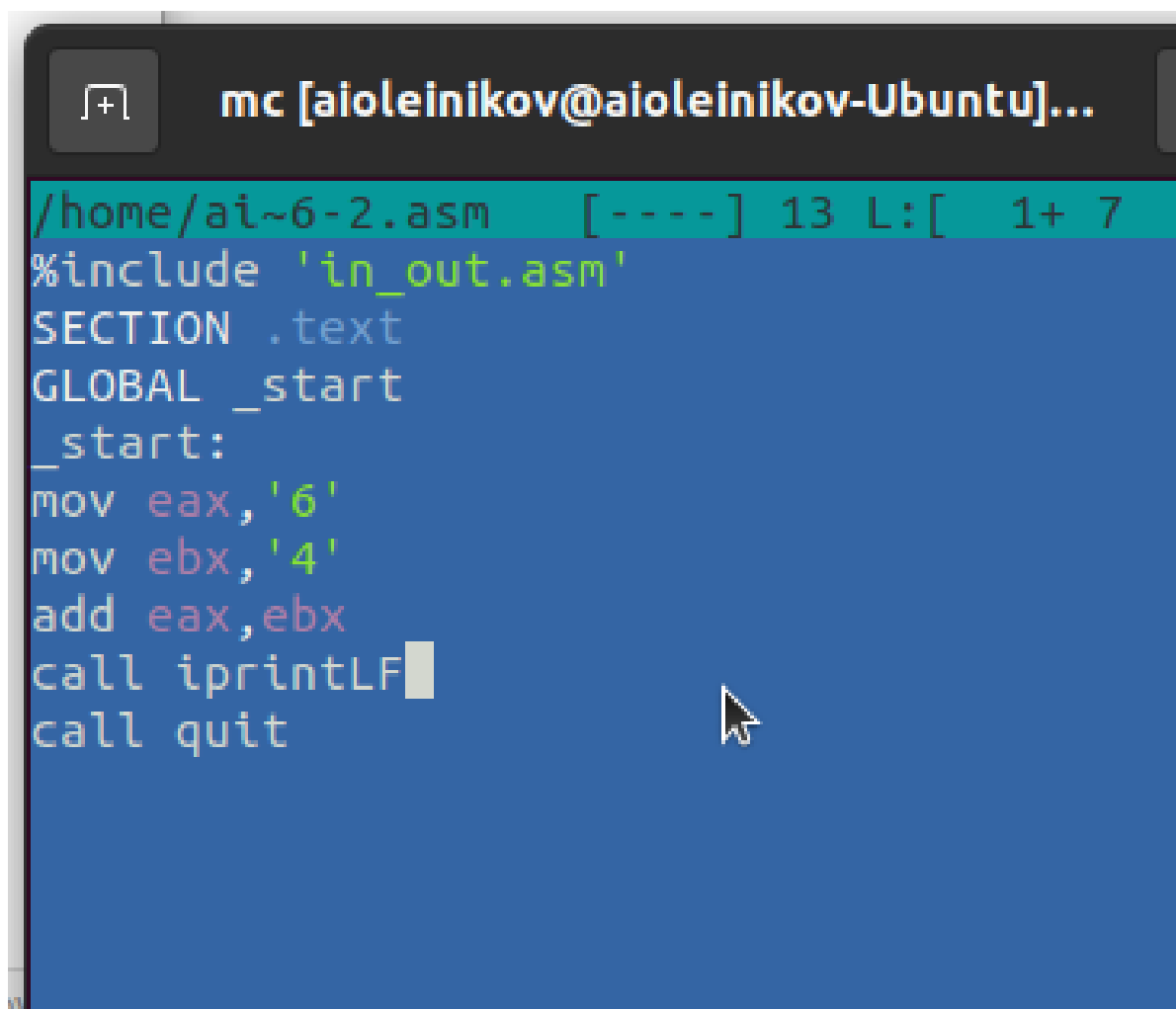
```
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o
lab6-1
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ./lab6-1

aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.4: Компиляция и запуск программы lab6-1.asm

Так же, как и в прошлый раз, при запуске программы мы не увидим число 10. Вместо этого пройдет вывод символа, который соответствует коду 10. Этот символ представляет собой перевод строки (или возврат каретки). На экране консоли он невидим, но создает пустую строку.

В файле `in_out.asm` внедрены вспомогательные процедуры для преобразования ASCII-символов в числовые значения и наоборот. Я внес изменения в код программы, применив эти процедуры.



```
mc [aioleinikov@aioleinikov-Ubuntu]...  
/home/ai~6-2.asm  [ - - - - ] 13 L: [ 1+ 7  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,'6'  
mov ebx,'4'  
add eax,ebx  
call iprintLF  
call quit
```

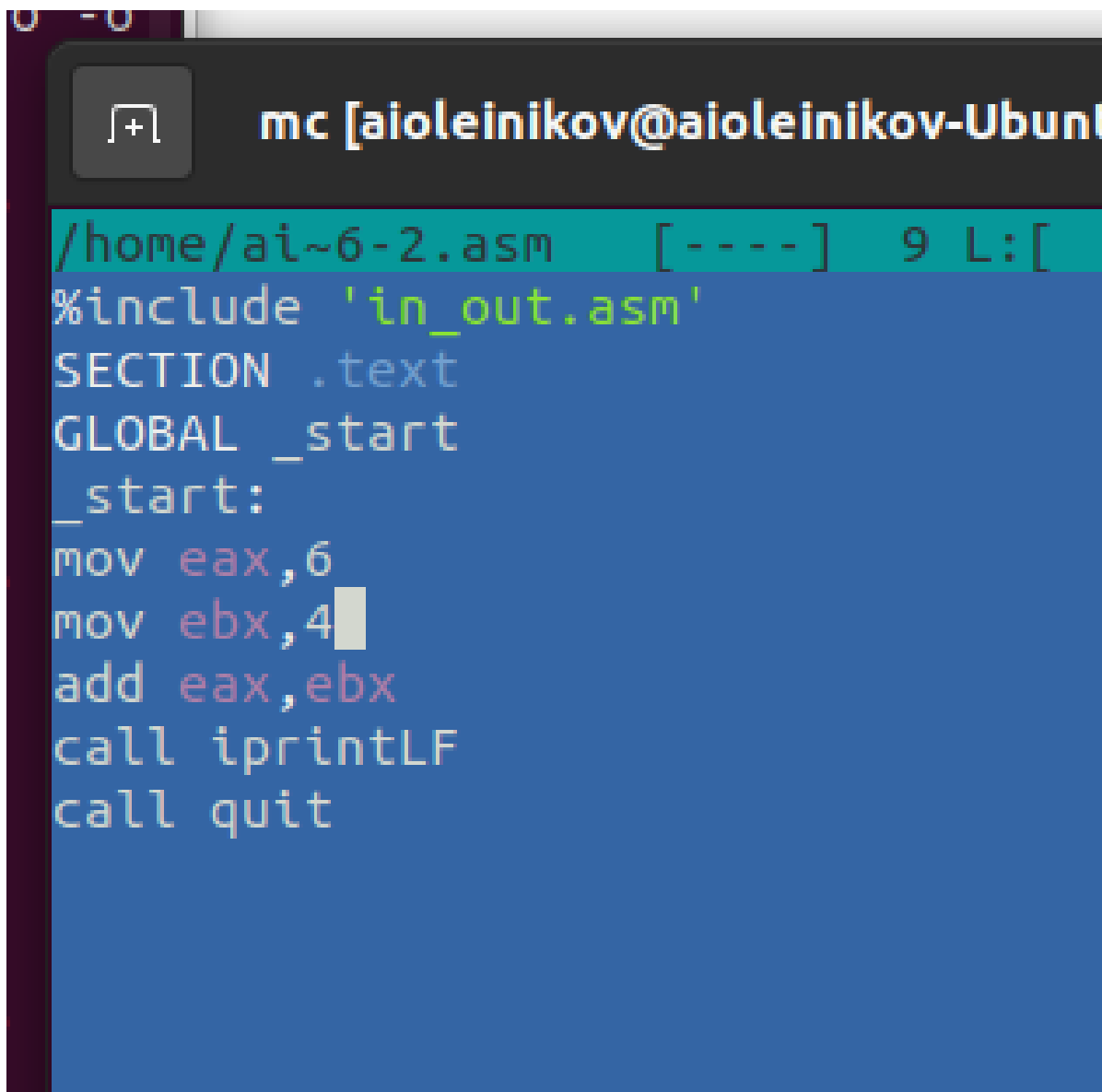
Рис. 2.5: Код программы `lab6-2.asm`

```
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$  
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm  
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o  
lab6-2  
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ./lab6-2  
106  
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.6: Компиляция и запуск программы lab6-2.asm

После запуска программы на экран будет выведено число 106. В этом случае, подобно первому примеру, команда `add` суммирует численные значения символов '6' и '4' ($54+52=106$). Но в отличие от прошлой версии программы, функция `iprintLF` позволяет отобразить именно число, а не символ с соответствующим числовым кодом.

Таким же образом, как и в предыдущем случае, мы провели замену символов на их числовые эквиваленты.



```
mc [aioleinikov@aioleinikov-Ubun...  
/home/ai~6-2.asm [----] 9 L:[  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
call quit
```

Рис. 2.7: Код программы lab6-2.asm

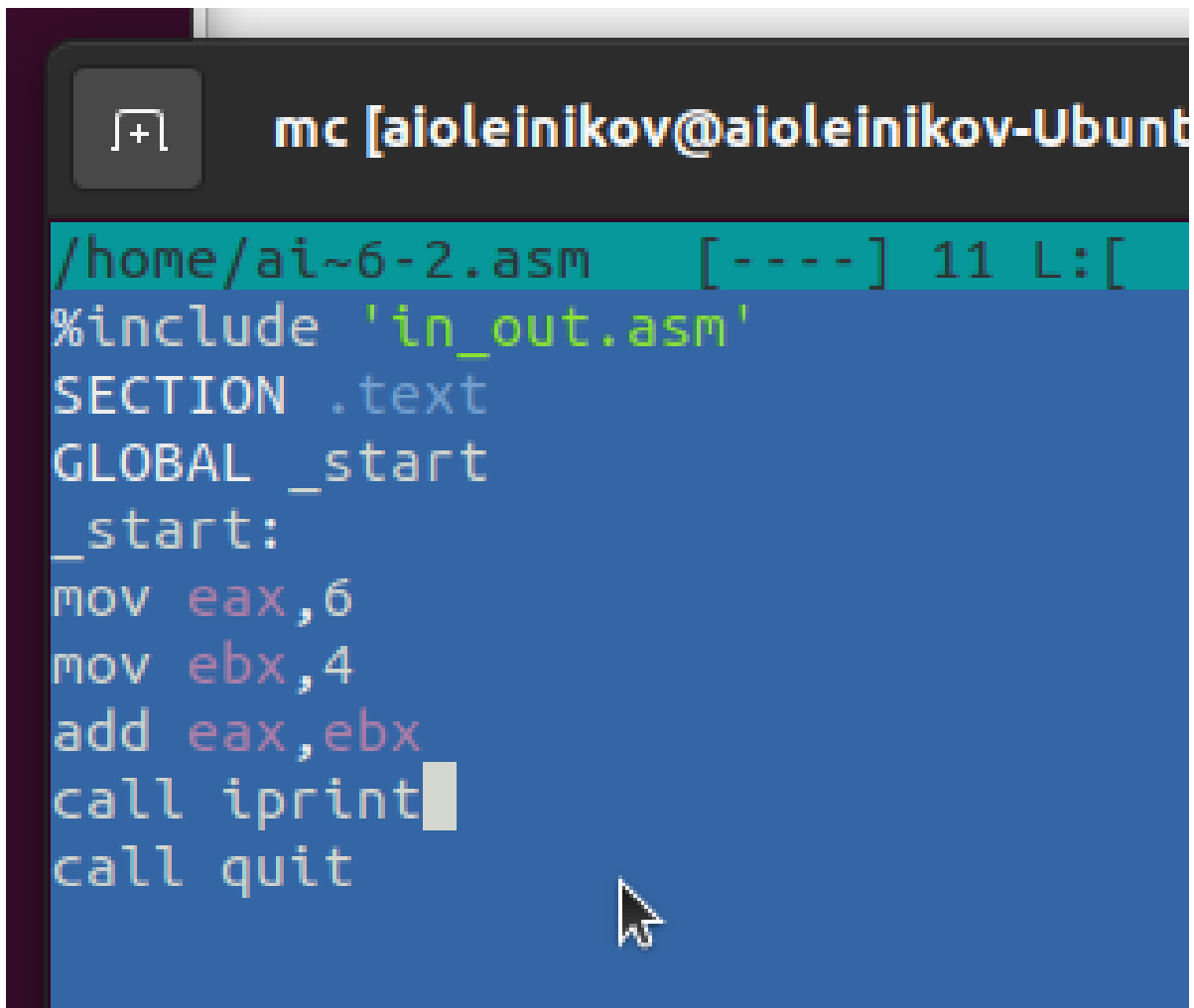
Функция `iprintLF` предназначена для вывода числовых значений, и в данном контексте она работает с числами, а не с их символьными кодами. В результате мы видим на экране число 10.

```
aioleinikov@aioleinikov-Ubuntu: ~/work/arch-pc/lab06$  
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$  
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm  
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o  
lab6-2  
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ./lab6-2  
10  
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.8: Компиляция и запуск программы lab6-2.asm

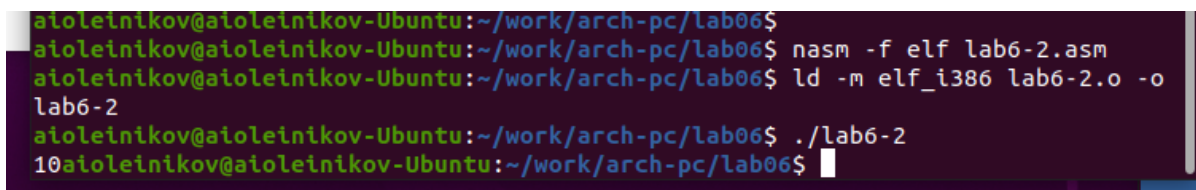
Функция `iprintLF` используется для отображения чисел, и в этой ситуации в качестве операндов выступают числа (не их символьные коды), что приводит к выводу числа 10.

Мы произвели замену функции `iprintLF` на `iprint`. Скомпилировали программу, создали исполняемый файл и запустили его. Разница в выводе заключается в отсутствии перевода строки.



```
mc [aioleinikov@aioleinikov-Ubuntu]
/home/ai~6-2.asm [----] 11 L:[
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 2.9: Код программы lab6-2.asm



```
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o
lab6-2
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ./lab6-2
10aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.10: Компиляция и запуск программы lab6-2.asm

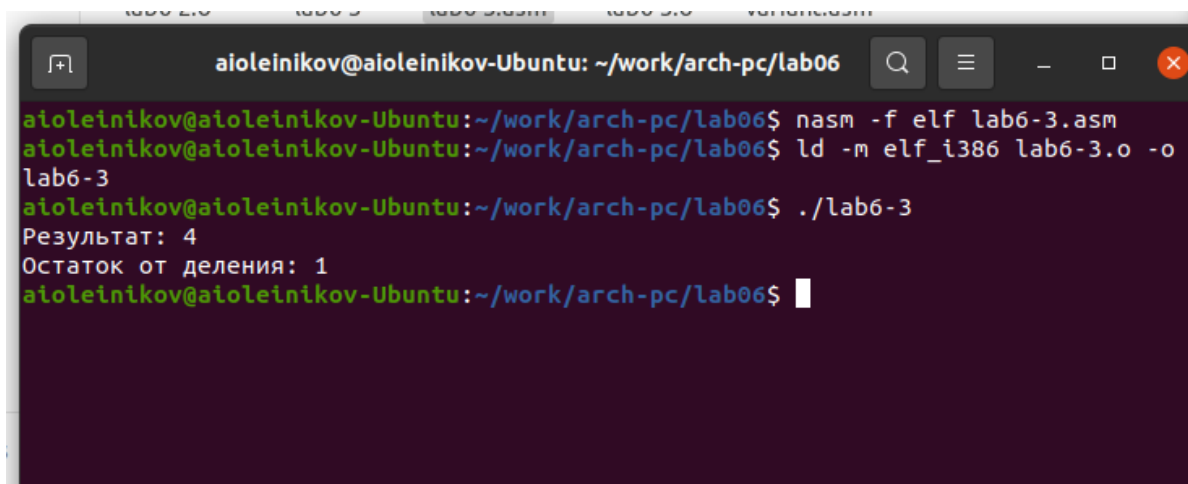
В рамках изучения выполнения арифметических действий в NASM была рассмотрена программа для расчета арифметической функции

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



```
mc [aioleinikov@aioleinikov-Ubuntu]...  
/home/ai~6-3.asm [----] 0 L:[ 1+25 26/ 2  
%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: Код программы lab6-3.asm

A terminal window titled 'aioleinikov@aioleinikov-Ubuntu: ~/work/arch-pc/lab06'. The terminal shows the following commands and output:

```
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.12: Компиляция и запуск программы lab6-3.asm

Затем модифицировал код программы для того, чтобы она могла вычислять функцию. После создания исполняемого файла он провел его тестирование.

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

.


```
mc [aioleinikov@aioleinikov-Ubuntu]
/home/ai~6-3.asm [----] 9 L:[ 1
#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
```

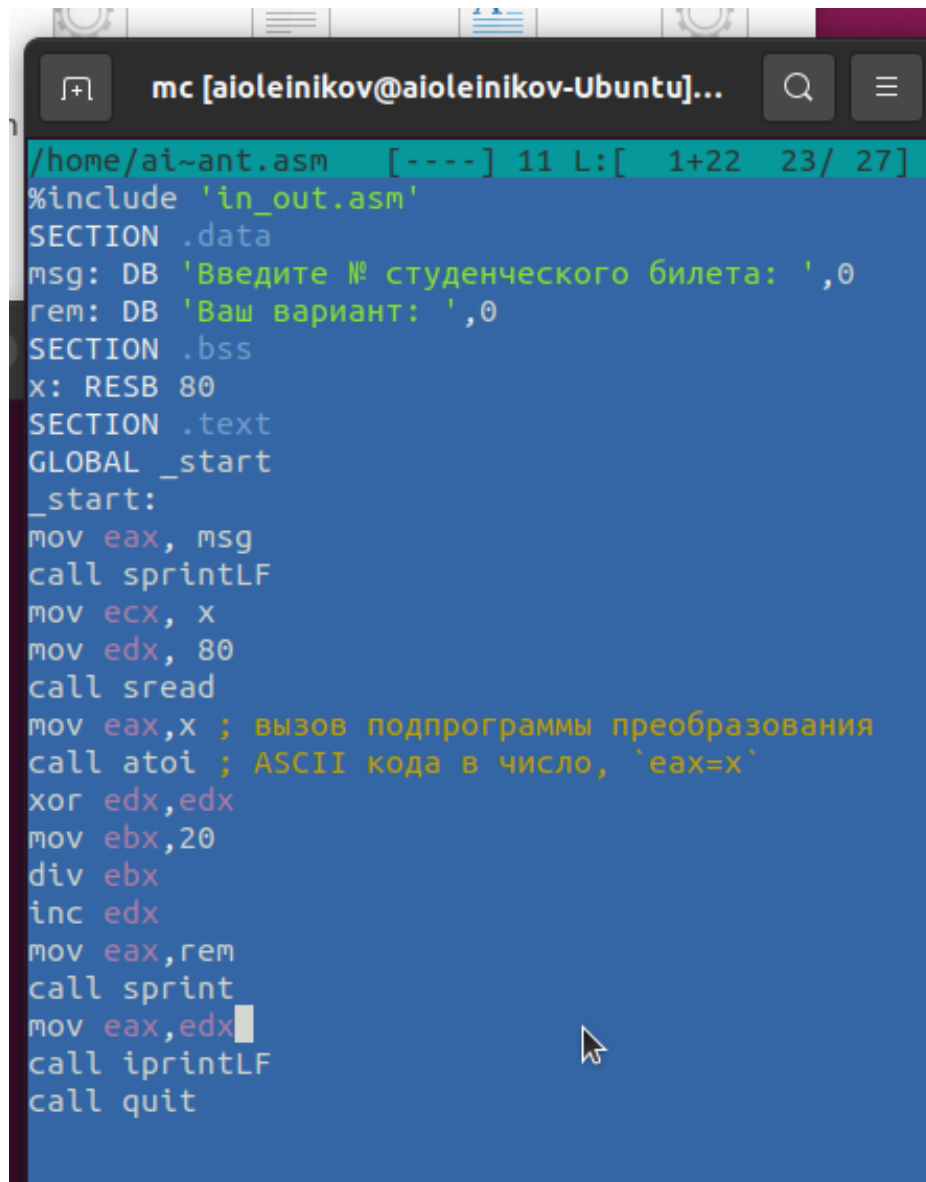
Рис. 2.13: Код программы lab6-3.asm

```
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o
lab6-3
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.14: Компиляция и запуск программы lab6-3.asm

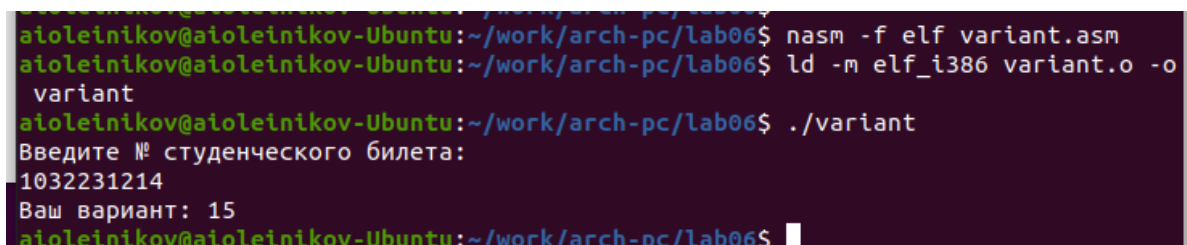
В качестве другого примера мы рассмотрели программу вычисления варианта задания по номеру студенческого билета.

В этом примере число для выполнения арифметических действий пользователь вводит с клавиатуры. Как было отмечено ранее, ввод осуществляется в виде символов, и для того чтобы арифметические операции в NASM были выполнены правильно, эти символы необходимо преобразовать в числовой формат. Для конвертации можно применить функцию `atoi`, которая содержится в файле `in_out.asm`.



```
mc [aioleinikov@aioleinikov-Ubuntu]...
/home/ai~ant.asm [----] 11 L:[ 1+22 23/ 27]
%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 ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
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: Код программы variant.asm



```
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf variant.asm
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o
variant
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032231214
Ваш вариант: 15
aioleinikov@aioleinikov-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.16: Компиляция и запуск программы variant.asm

ответы на вопросы

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

Команда `mov eax, ptr` загружает в регистр `eax` строку с текстом “Ваш вариант:”. После этого команда `call sprint` инициирует процедуру, которая выводит эту строку на экран.

2. Для чего используются следующие инструкции?

Команда `mov ecx, x` копирует значение из регистра `ecx` в переменную `x`. Команда `mov edx, 80` присваивает регистру `edx` число 80. Команда `call sread` запускает процедуру чтения данных из стандартного ввода.

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

Команда `call atoi` конвертирует строку символов в целое число.

4. Какие строки листинга отвечают за вычисления варианта?

Команды, выполняющие вычисление варианта, включают: `xor edx, edx` для обнуления регистра `edx`, `mov ebx, 20` для присвоения регистру `ebx` числа 20, `div ebx` для деления значения в аккумуляторе на значение в `ebx`, и `inc edx` для увеличения результата в регистре `edx` на единицу.

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

Остаток от деления после выполнения команды `div ebx` записывается в регистр `edx`.

6. Для чего используется инструкция “`inc edx`”?

Команда `inc edx` служит для инкремента, то есть увеличения значения в регистре `edx` на один. Это необходимо для корректного расчета варианта по заданной формуле, где к остатку от деления добавляется единица.

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

Команда `mov eax, edx` помещает результат вычислений в регистр `eax`. Затем команда `call iprintLF` активирует процедуру, которая выводит результат на экран с переводом строки.

2.1 Задание для самостоятельной работы

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.

Вариант 15 - $(5 + x)^2 - 3$ для $x = 5, x = 1$

```
mc [aioleinikov@aioleinikov-Ubuntu]:~/work/arch-pc/lab06
/home/ai~ram.asm [B- -] 9 L: [ 1+20 21/ 31] *(390
#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 ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`

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: Код программы program.asm

при $x = 5$ $f(x) = 97$

при $x = 1$ $f(x) = 33$

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

Рис. 2.18: Компиляция и запуск программы program.asm

Программа считает верно.

3 Выводы

Изучили работу с арифметическими операциями.