

Отчёт по лабораторной работе 6

Архитектура компьютера

Ходжамедов Давуд НБИбд-02-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	28

Список иллюстраций

2.1	Программа в файле lab6-1.asm	7
2.2	Запуск программы lab6-1.asm	8
2.3	Программа в файле lab6-1.asm	9
2.4	Запуск программы lab6-1.asm	10
2.5	Программа в файле lab6-2.asm	11
2.6	Запуск программы lab6-2.asm	12
2.7	Программа в файле lab6-2.asm	13
2.8	Запуск программы lab6-2.asm	14
2.9	Запуск программы lab6-2.asm	15
2.10	Программа в файле lab6-3.asm	16
2.11	Запуск программы lab6-3.asm	17
2.12	Программа в файле lab6-3.asm	18
2.13	Запуск программы lab6-3.asm	20
2.14	Программа в файле variant.asm	21
2.15	Запуск программы variant.asm	22
2.16	Программа в файле task.asm	25
2.17	Запуск программы task.asm	27

Список таблиц

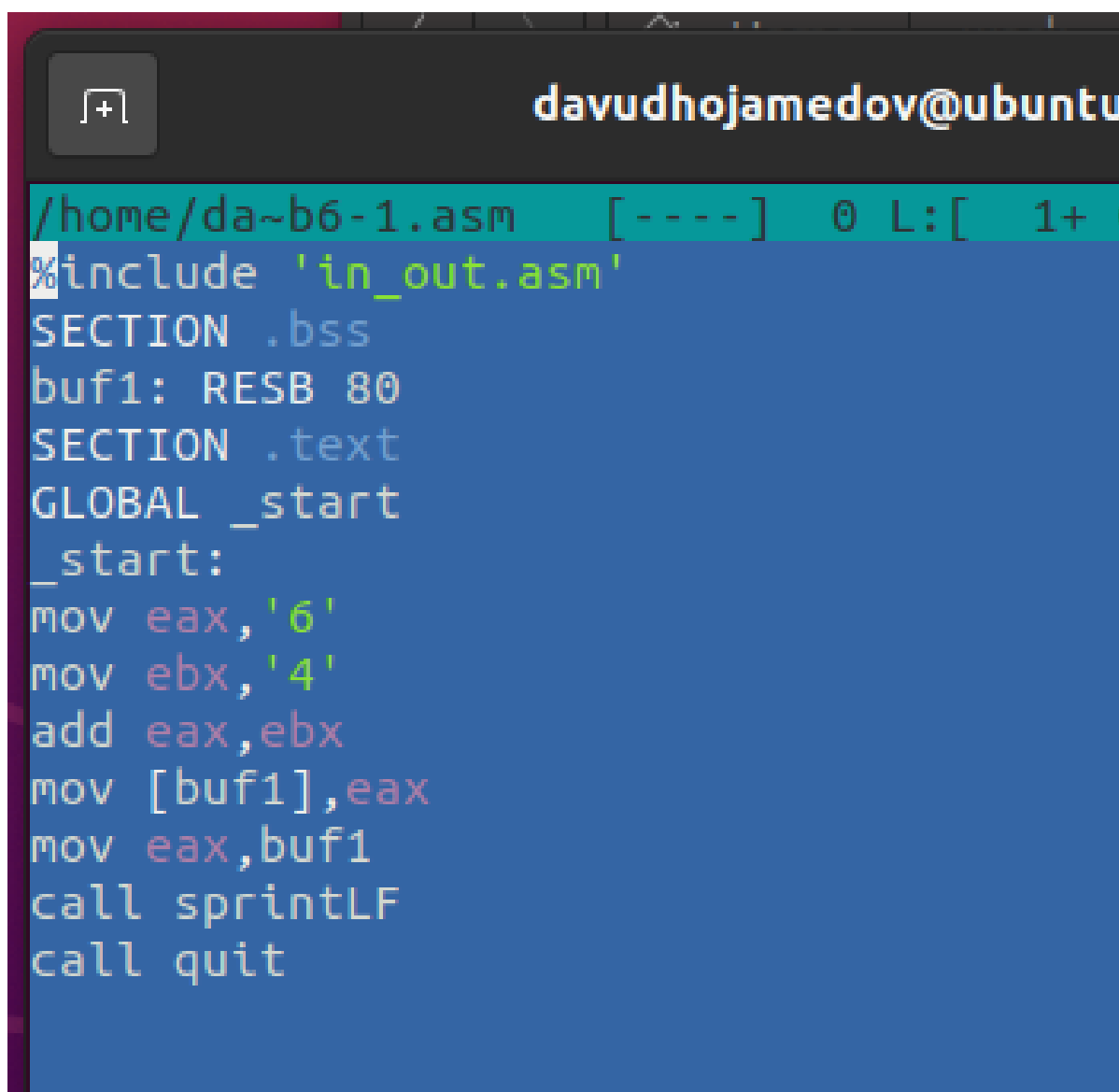
1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

1. Создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab6-1.asm.
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр еах.

В данной программе в регистр еах записывается символ 6 (`mov еах, '6'`), в регистр ебх символ 4 (`mov ебх, '4'`). Далее к значению в регистре еах прибавляем значение регистра ебх (`add еах, ебх`, результат сложения запишется в регистр еах). Далее выводим результат. Так как для работы функции `sprintLF` в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра еах в переменную `buf1` (`mov [buf1], еах`), а затем запишем адрес переменной `buf1` в регистр еах (`mov еах, buf1`) и вызовем функцию `sprintLF`.



```

/home/da~b6-1.asm  [ - - - - ]  0  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 sprintfLF
call quit

```

Рис. 2.1: Программа в файле lab6-1.asm

Привожу код программы в отчете

```

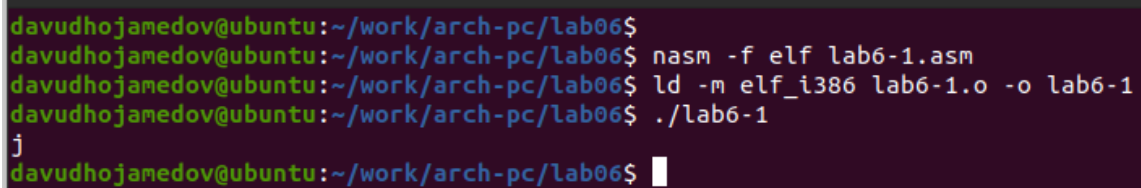
#include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start

```

```

_start:
mov  eax,'6'
mov  ebx,'4'
add  eax,ebx
mov  [buf1],eax
mov  eax,buf1
call sprintfLF
call quit

```



```

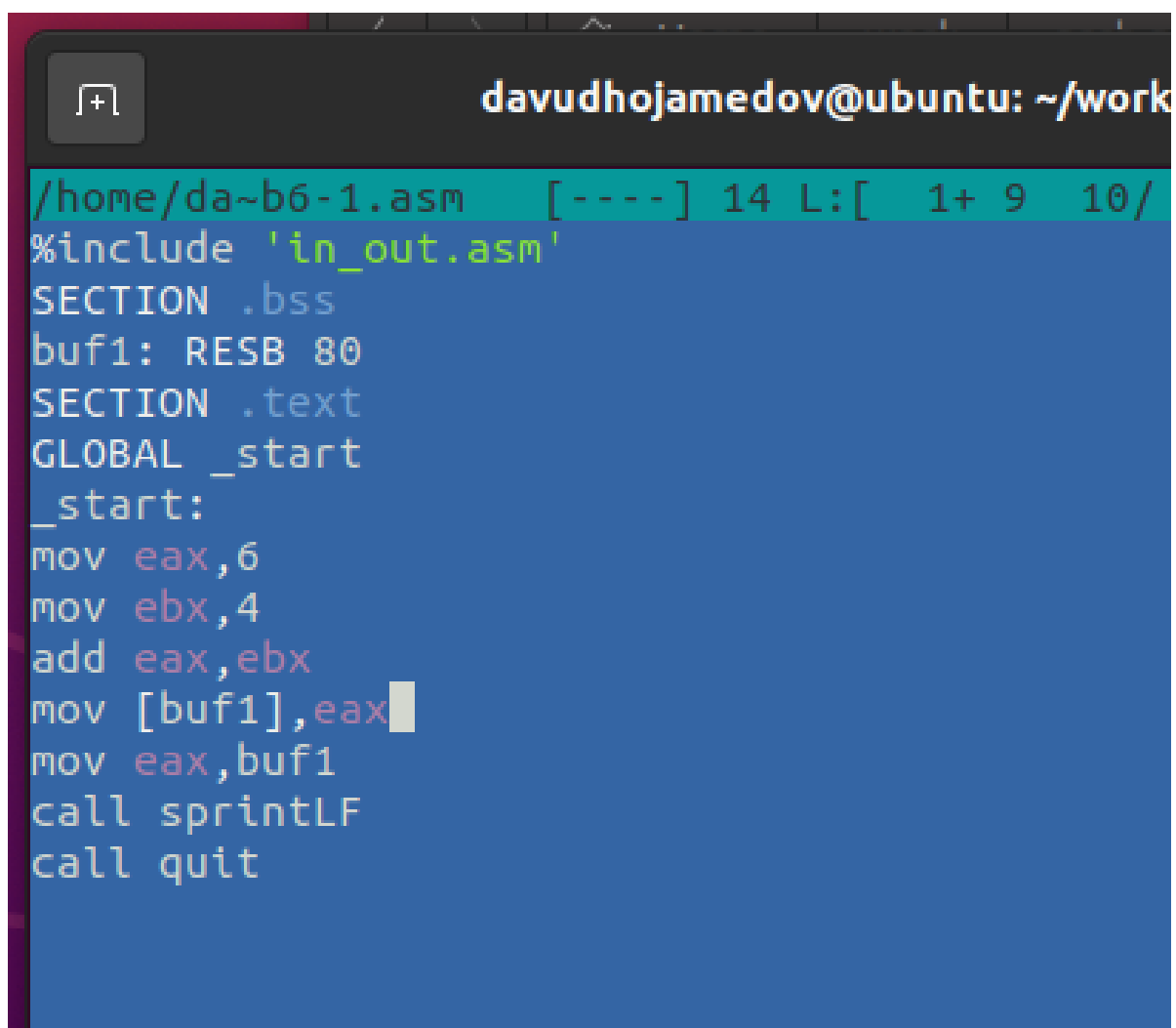
davudhojamedov@ubuntu:~/work/arch-pc/lab06$
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./lab6-1
j
davudhojamedov@ubuntu:~/work/arch-pc/lab06$

```

Рис. 2.2: Запуск программы lab6-1.asm

В данном случае при выводе значения регистра `eax` мы ожидаем увидеть число 10. Однако результатом будет символ `j`. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда `add eax,ebx` запишет в регистр `eax` сумму кодов – 01101010 (106), что в свою очередь является кодом символа `j`.

3. Далее изменяю текст программы и вместо символов, запишем в регистры числа.

A screenshot of a terminal window with a dark background. The title bar shows the user 'davudhojamedov@ubuntu' and the directory '~/work'. The terminal displays assembly code for a file named 'lab6-1.asm'. The code includes a header file 'in_out.asm', defines a buffer 'buf1' of size 80 in the .bss section, and defines the .text section. The main routine '_start' initializes 'eax' to 6, 'ebx' to 4, adds them to get 10, stores the result in 'buf1', prints it with 'sprintf', and then calls 'quit'.

```

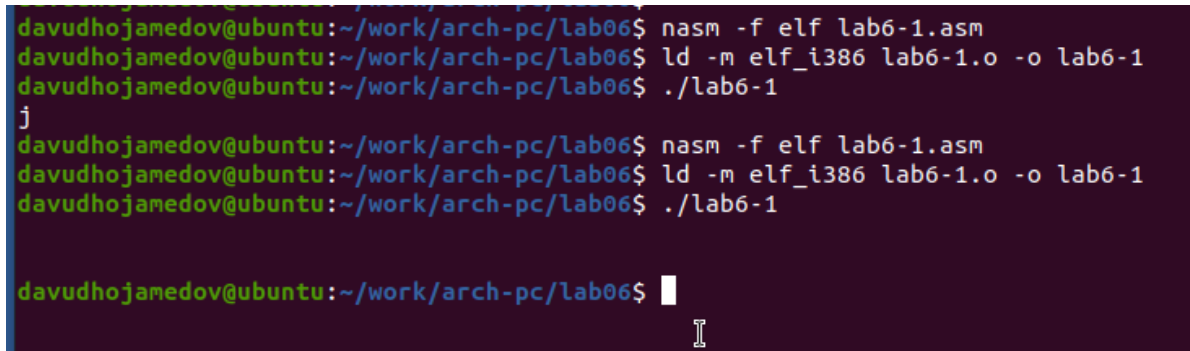
/home/da~b6-1.asm  [ - - - - ] 14 L:[ 1+ 9 10/
#include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
mov  [buf1],eax
mov  eax,buf1
call sprintf
call quit
```

Рис. 2.3: Программа в файле lab6-1.asm

Привожу код программы в отчете

```
%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 sprintLF
call quit
```

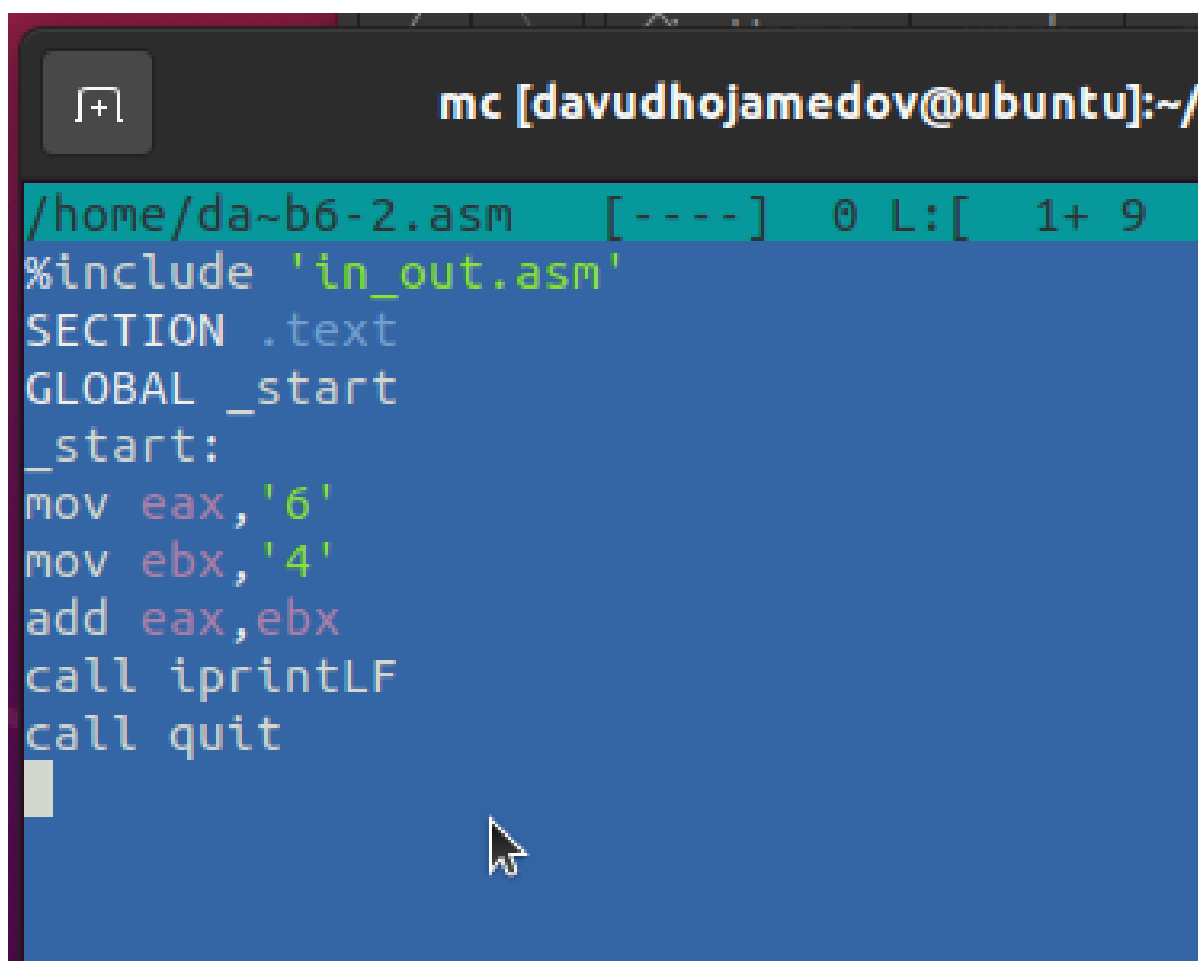
A terminal window with a dark purple background and green text. The prompt is 'davudhojamedov@ubuntu:~/work/arch-pc/lab06\$'. The user enters 'nasm -f elf lab6-1.asm', followed by 'ld -m elf_i386 lab6-1.o -o lab6-1', and then './lab6-1'. The output is a single character 'j'. The user then repeats the same three commands. The output is again 'j'. The prompt is shown again at the bottom with a cursor and a small icon below it.

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

Рис. 2.4: Запуск программы lab6-1.asm

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Это символ конца строки (возврат каретки). В консоле он не отображается, но добавляет пустую строку.

4. Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовал текст программы с использованием этих функций.



```
mc [davudhojamedov@ubuntu]:~/
/home/da~b6-2.asm [----] 0 L:[ 1+ 9
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 2.5: Программа в файле lab6-2.asm

Привожу код программы в отчете

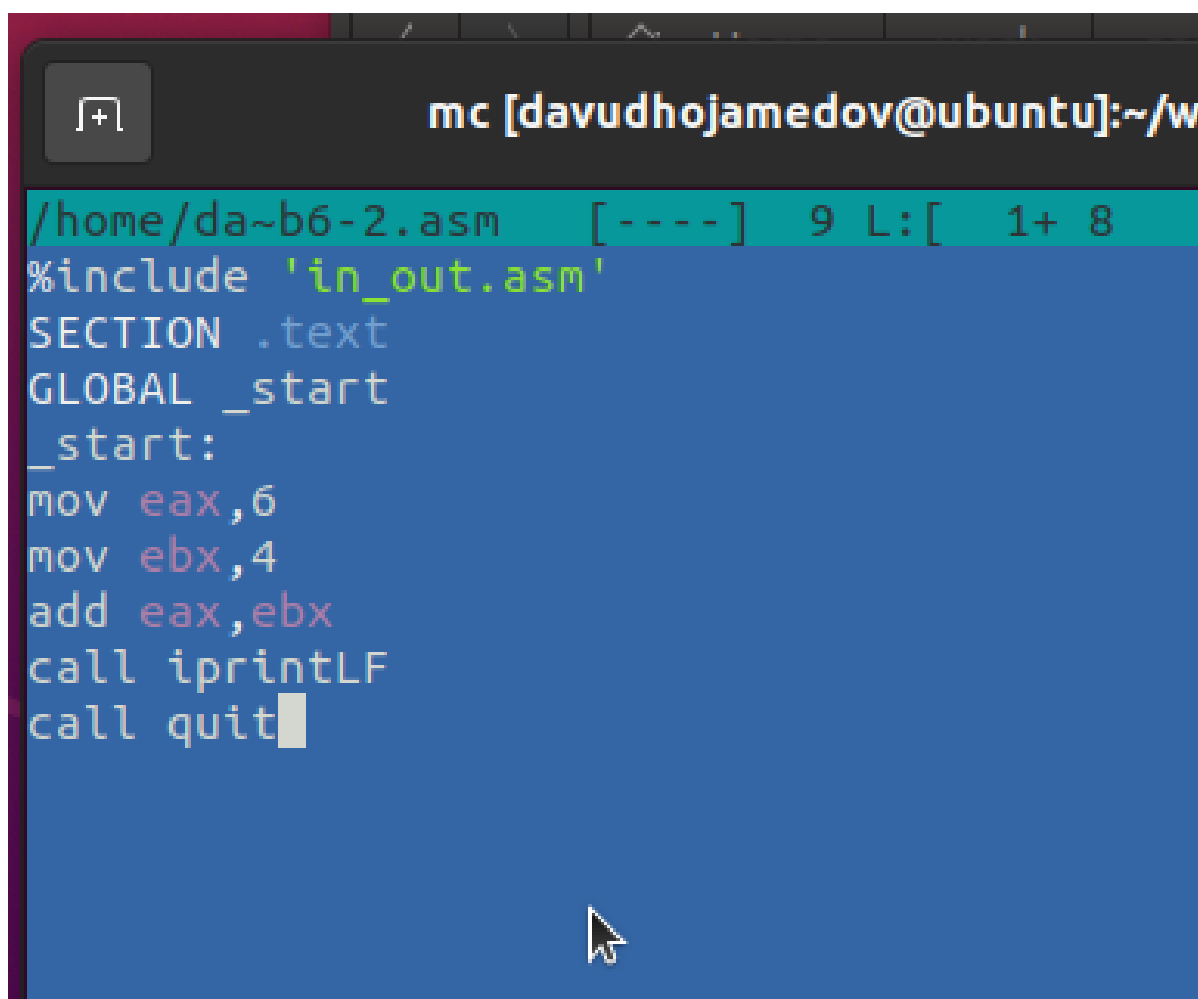
```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

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

Рис. 2.6: Запуск программы lab6-2.asm

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 прошлой программы, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

5. Аналогично предыдущему примеру изменим символы на числа.

A screenshot of a terminal window with a dark background. The title bar at the top shows a window icon and the text 'mc [davudhojamedov@ubuntu]:~/w'. The terminal content shows the path '/home/da~b6-2.asm' followed by a status bar '[- - - -] 9 L: [1+ 8]'. The code being displayed is assembly language: '%include \'in_out.asm\'', 'SECTION .text', 'GLOBAL _start', '_start:', 'mov eax,6', 'mov ebx,4', 'add eax,ebx', 'call iprintLF', and 'call quit' with a cursor at the end of the last line.

```
mc [davudhojamedov@ubuntu]:~/w
/home/da~b6-2.asm [ - - - - ] 9 L: [ 1+ 8 ]
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

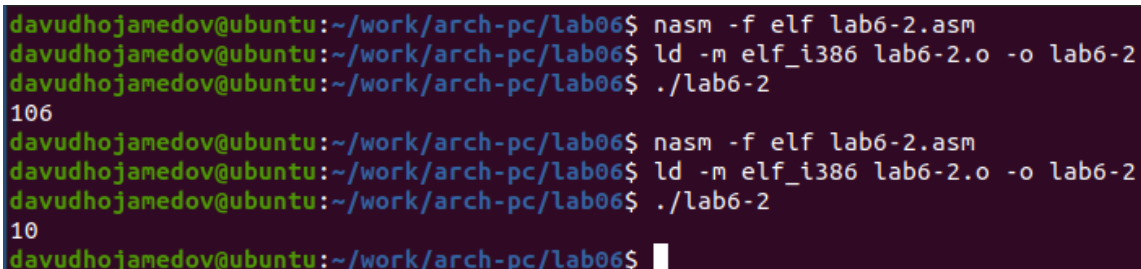
Рис. 2.7: Программа в файле lab6-2.asm

Привожу код программы в отчете

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
```

```
call quit
```

Функция `iprintLF` позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10.



```
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./lab6-2
106
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./lab6-2
10
davudhojamedov@ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.8: Запуск программы `lab6-2.asm`

Заменяю функцию `iprintLF` на `iprint`. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки.

Привожу код программы в отчете

```
%include 'in_out.asm'

SECTION .text

GLOBAL _start

_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

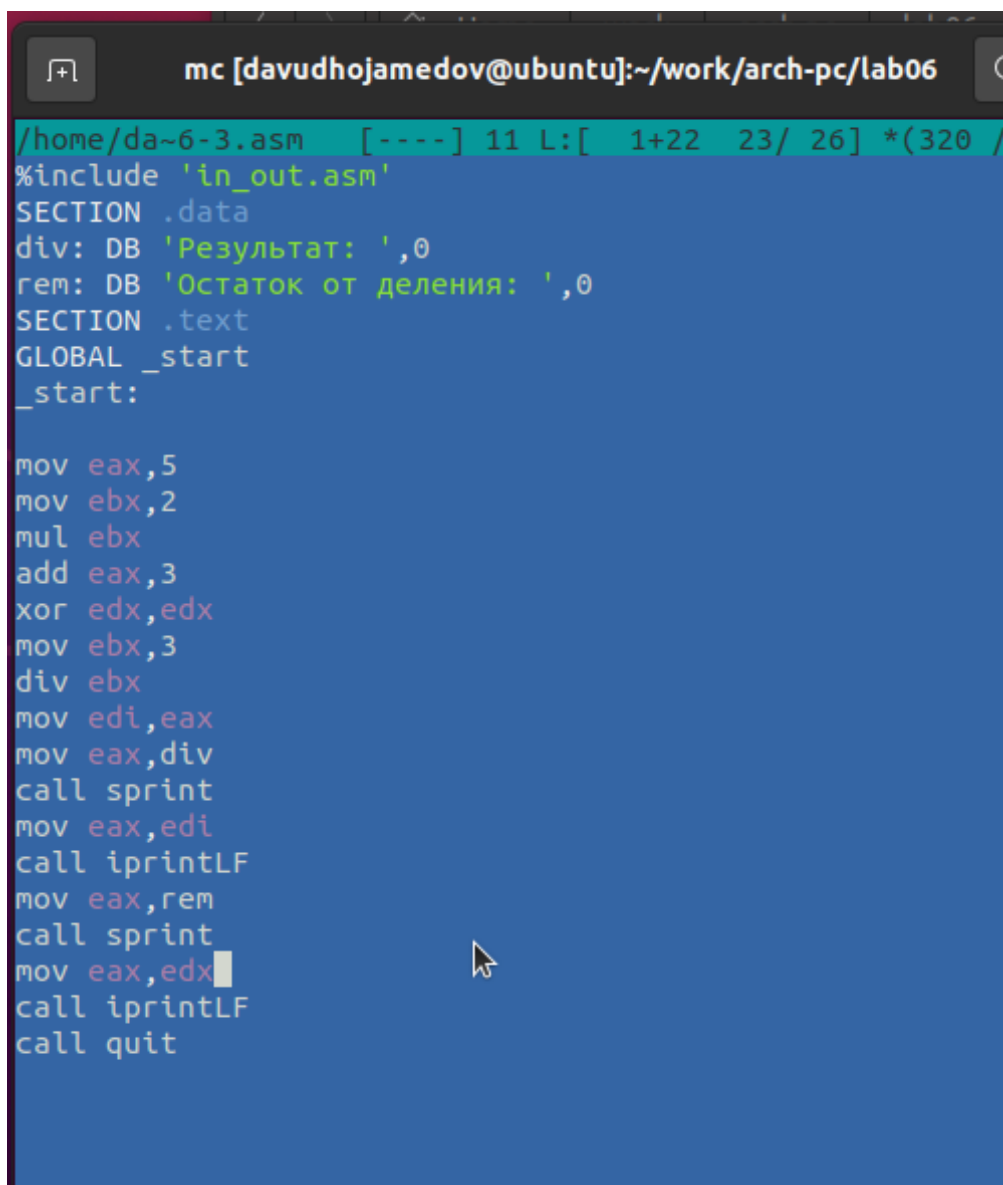
```
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./lab6-2
106
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./lab6-2
10
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./lab6-2
10davudhojamedov@ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.9: Запуск программы lab6-2.asm

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

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

.



```
mc [davudhojamedov@ubuntu]:~/work/arch-pc/lab06
/home/da~6-3.asm [----] 11 L: [ 1+22 23/ 26] *(320 /
#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. 2.10: Программа в файле lab6-3.asm

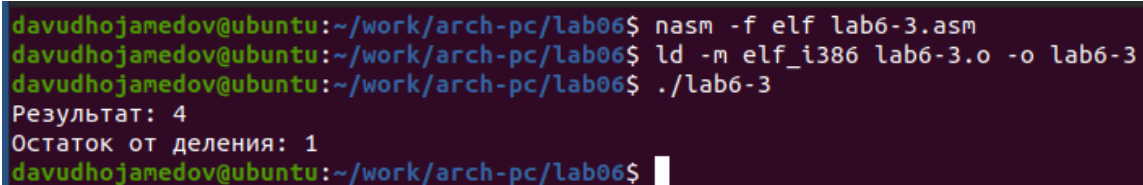
Также размещаю код программы в отчете.

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
```



```
GLOBAL _start
_start:
```

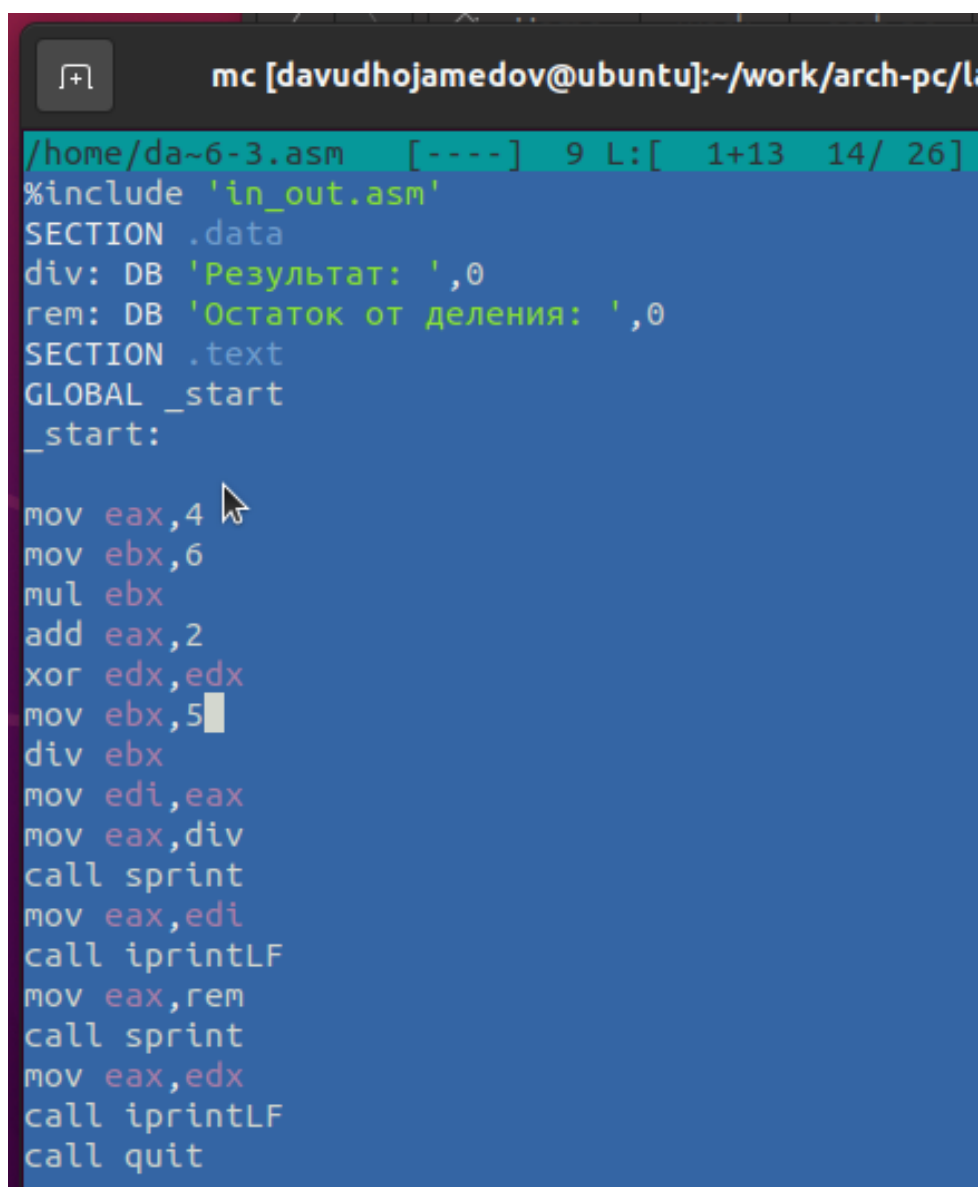
```
mov eax,5
mov ebx,2
mul ebx
add eax,3
xor edx,edx
mov ebx,3
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```



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

Рис. 2.11: Запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$. Создал исполняемый файл и проверил его работу.



```
mc [davudhojamedov@ubuntu]:~/work/arch-pc/l
/home/da~6-3.asm [----] 9 L:[ 1+13 14/ 26]
#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. 2.12: Программа в файле lab6-3.asm

Также размещаю код программы в отчете.

```
%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
```

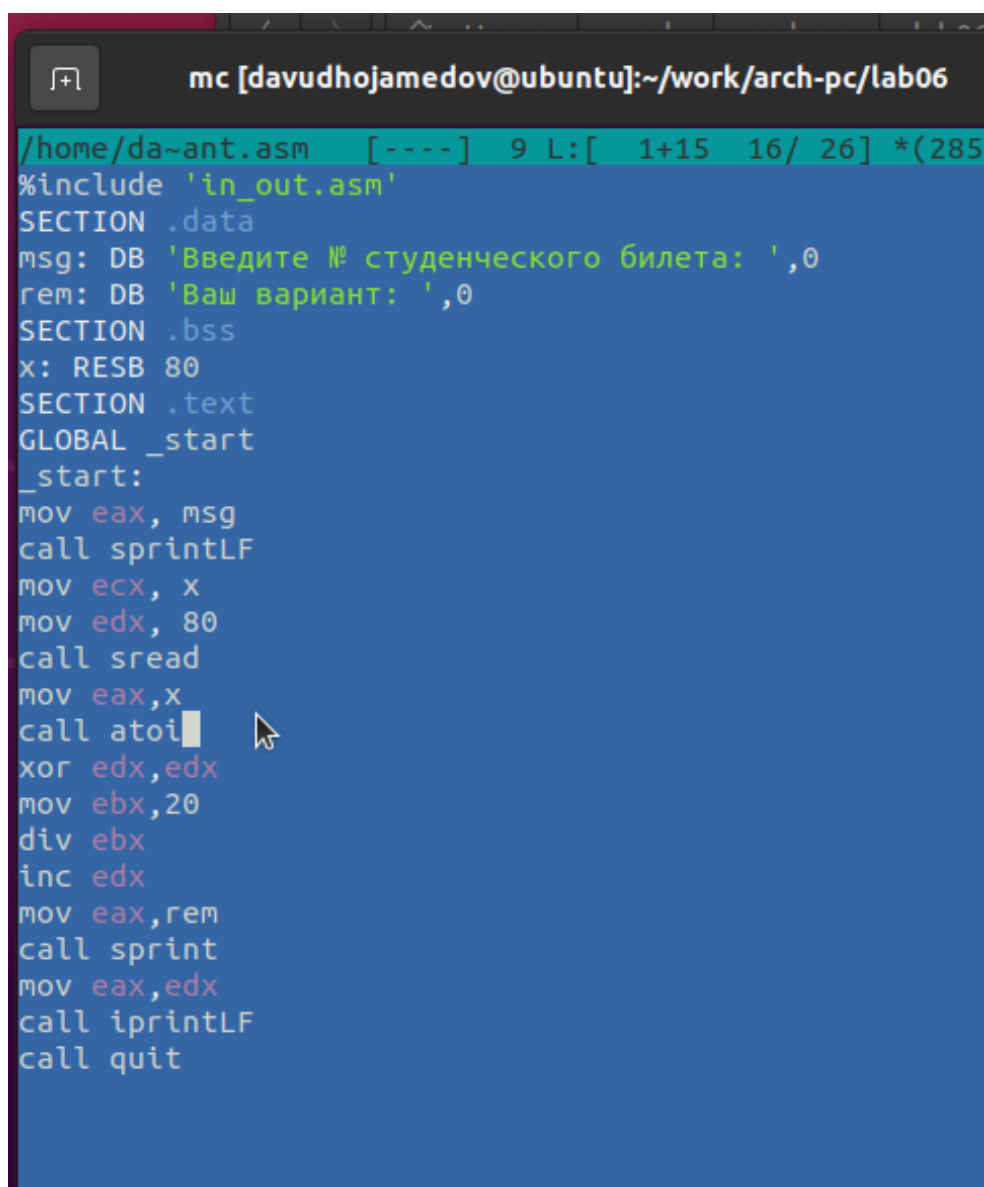
```
call quit
```

```
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
davudhojamedov@ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.13: Запуск программы lab6-3.asm

7. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета.

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`.



```
mc [davudhojamedov@ubuntu]:~/work/arch-pc/lab06
/home/da~ant.asm [---] 9 L:[ 1+15 16/ 26] *(285
#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. 2.14: Программа в файле variant.asm

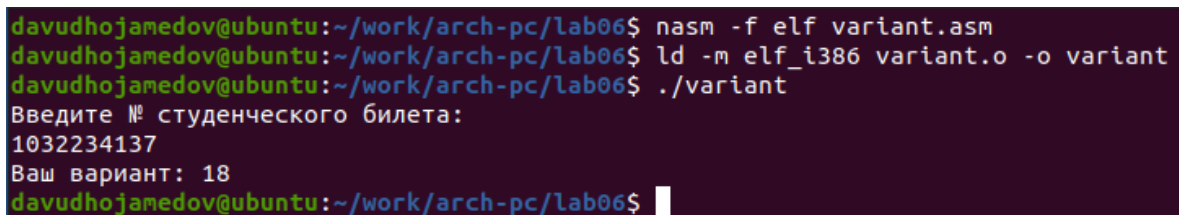
Также размещаю код программы в отчете.

```
%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

```



```

davudhojamedov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf variant.asm
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032234137
Ваш вариант: 18
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ █

```

Рис. 2.15: Запуск программы variant.asm

ответы на вопросы

1. Какие строки листинга 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

2. Для чего используются следующие инструкции?

```
nasm
mov ecx, x
mov edx, 80
call sread
```

Считывает значение студбилета в переменную X из консоли

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

Эта подпрограмма переводит введенные символы в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?

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

Здесь происходит деление номера студ билета на 20. В регистре `edx` хранится остаток, к нему прибавляется 1.

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

регистр `edx`

6. Для чего используется инструкция “inc edx”?

по формуле вычисления варианта нужно прибавить единицу

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

mov eax,edx – результат перекладывается в регистр eax

call iprintLF – вызов подпрограммы вывода

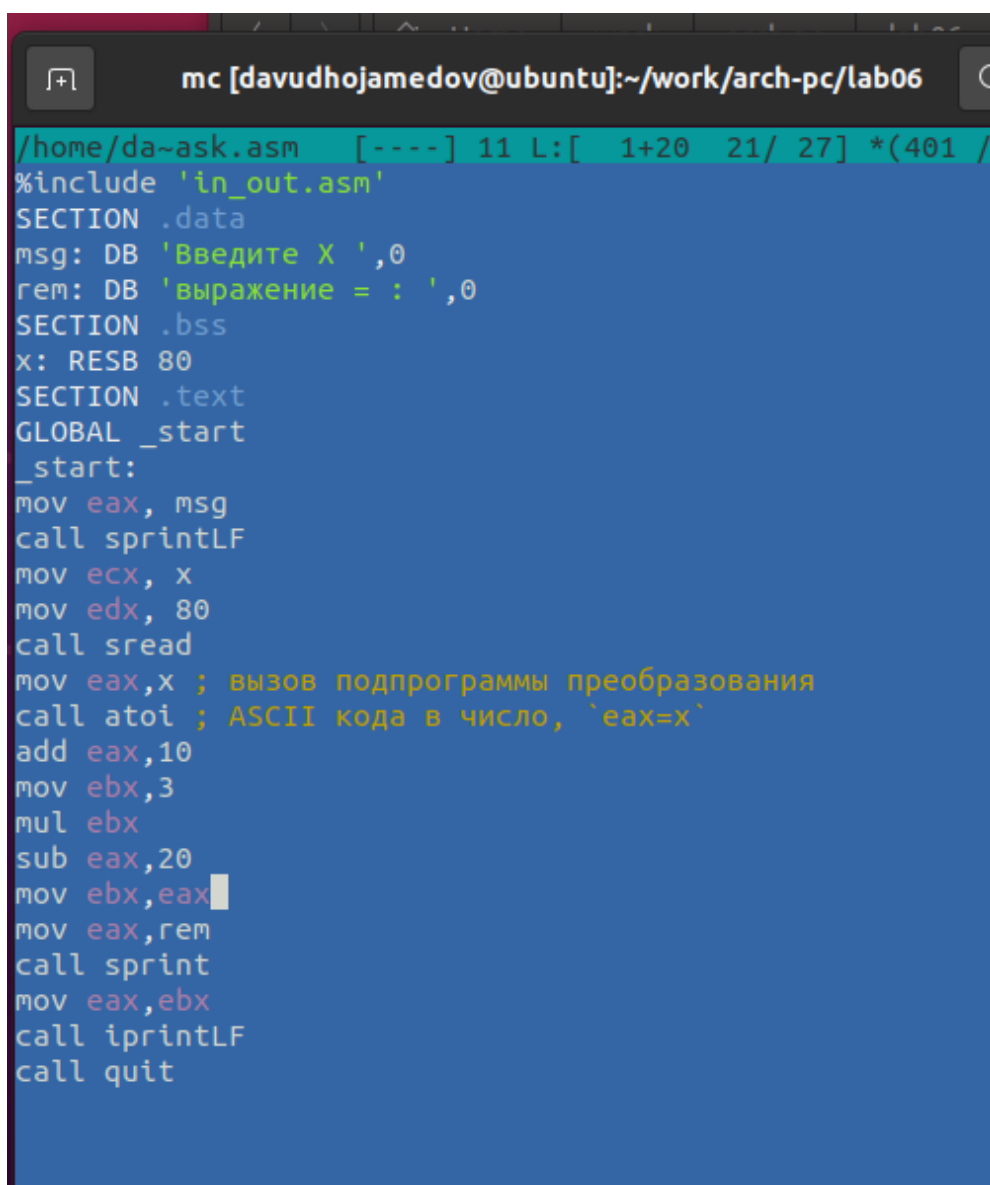
8. Написать программу вычисления выражения $y = f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений x_1 и x_2 из 6.3.

Получили вариант 18 -

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

для

$$x_1 = 1, x_2 = 5$$



```
mc [davudhojamedov@ubuntu]:~/work/arch-pc/lab06
/home/da~ask.asm [----] 11 L:[ 1+20 21/ 27] *(401 /
#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, 10
mov ebx, 3
mul ebx
sub eax, 20
mov ebx, eax
mov eax, rem
call sprint
mov eax, ebx
call iprintLF
call quit
```

Рис. 2.16: Программа в файле task.asm

Также размещаю код программы в отчете.

```
%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 sprintf
mov ecx, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
add eax,10
mov ebx,3
mul ebx
sub eax,20
mov eax,rem
call sprintf
mov eax,ebx
call iprintLF
call quit

```

```
davudhojamedov@ubuntu: ~/work/arch-pc/lab06$  
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf task.asm  
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 task.o -o task  
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./task  
Введите X  
1  
выражение = : 13  
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./task  
Введите X  
20  
выражение = : 70  
davudhojamedov@ubuntu:~/work/arch-pc/lab06$ ./task  
Введите X  
5  
выражение = : 25  
davudhojamedov@ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.17: Запуск программы task.asm

3 Выводы

Изучили работу с арифметическими операциями.