

Отчёт по лабораторной работе 6

Архитектура компьютеров и операционные системы

Тарасова Алина НКАбд 05-23

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Символьные и численные данные в NASM	7
3.2	Выполнение арифметических операций в NASM	14
3.3	Ответы на вопросы	19
3.4	Задание для самостоятельной работы	21
4	Выводы	24

Список иллюстраций

3.1	Программа lab6-1.asm	8
3.2	Запуск программы lab6-1.asm	9
3.3	Программа lab6-1.asm	10
3.4	Запуск программы lab6-1.asm	11
3.5	Программа lab6-2.asm	12
3.6	Запуск программы lab6-2.asm	12
3.7	Программа lab6-2.asm	13
3.8	Запуск программы lab6-2.asm	14
3.9	Запуск программы lab6-2.asm	14
3.10	Программа lab6-3.asm	15
3.11	Запуск программы lab6-3.asm	15
3.12	Программа lab6-3.asm	16
3.13	Запуск программы lab6-3.asm	17
3.14	Программа variant.asm	18
3.15	Запуск программы variant.asm	19
3.16	Программа calc.asm	22
3.17	Запуск программы calc.asm	23

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Изучить синтаксис арифметических операций в ассемблере
2. Разобрать примеры программ
3. Выполнить самостоятельное задание # Теоретическое введение

В ассемблере можно выделить следующие базовые операции:

- Схема команды целочисленного сложения `add` (от англ. `addition` - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда.
- Команда целочисленного вычитания `sub` (от англ. `subtraction` – вычитание) работает аналогично команде `add`.
- Существуют специальные команды: `inc` (от англ. `increment`) и `dec` (от англ. `decrement`), которые увеличивают и уменьшают на 1 свой операнд.
- Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды. Для беззнакового умножения используется команда `mul` (от англ. `multiply` – умножение), для знакового умножения используется команда `imul`.
- Для деления, как и для умножения, существует 2 команды `div` (от англ. `divide` - деление) и `idiv`

3 Выполнение лабораторной работы

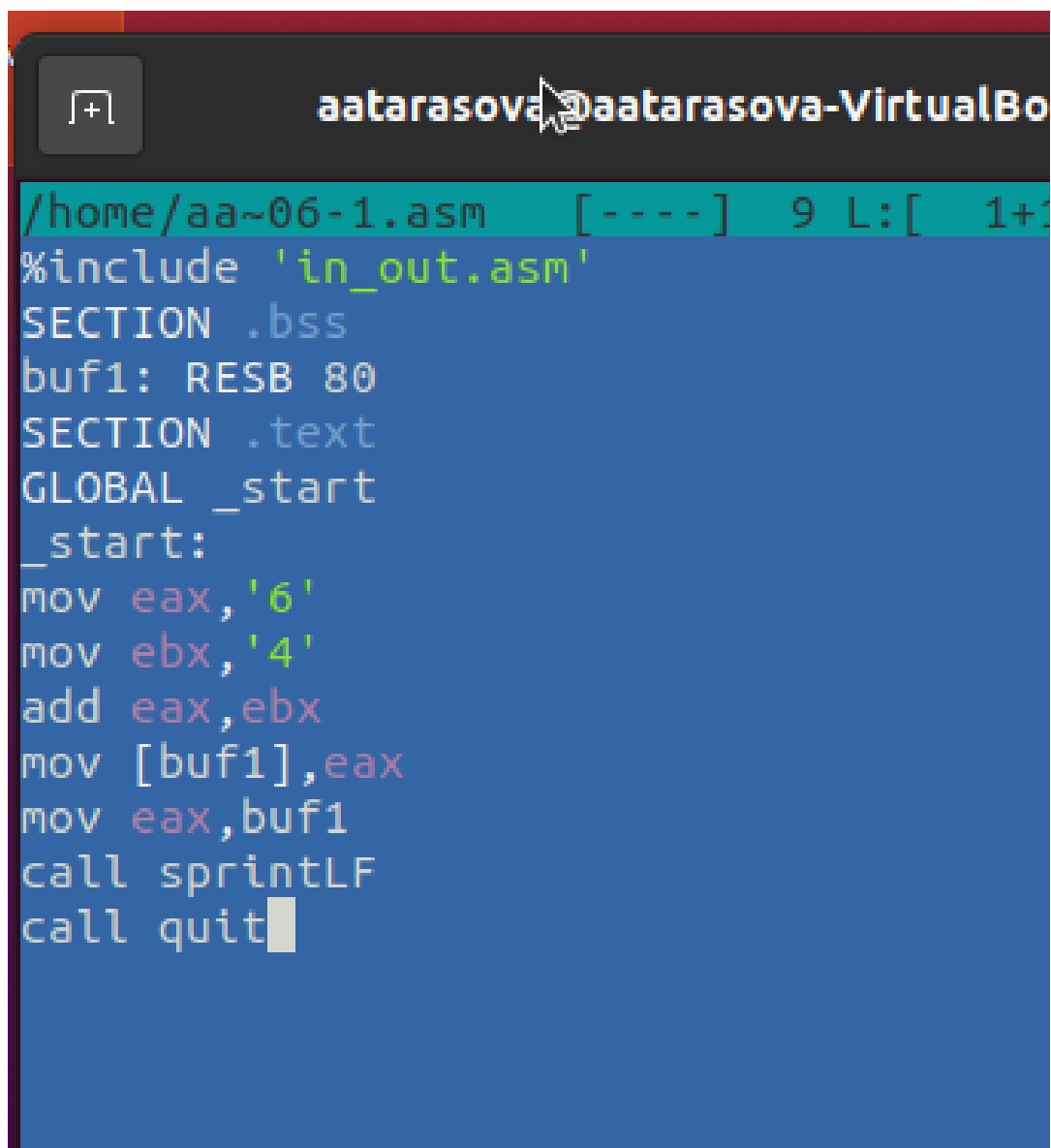
3.1 Символьные и численные данные в NASM

Создаю каталог для программам лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm.

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. (рис. [3.1]) (рис. [3.2])

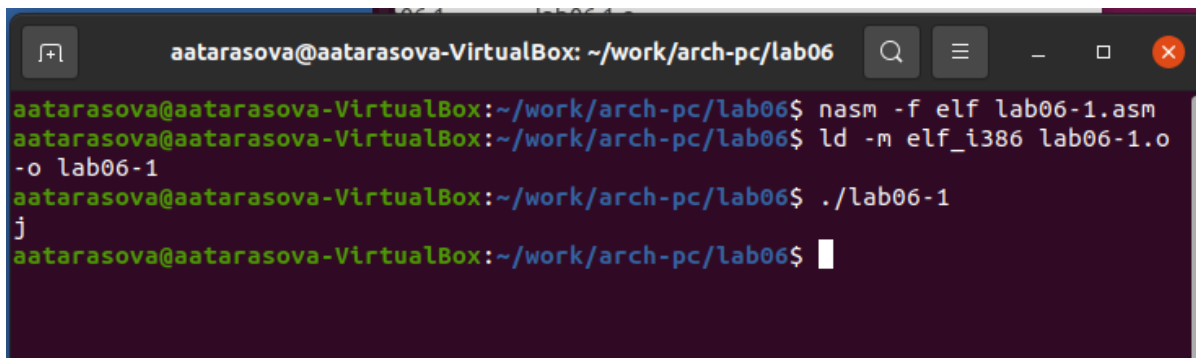
Так как для работы функции `sprintLF` в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра еах в переменную `buf1` (`mov [buf1],еах`), а затем запишем адрес переменной `buf1` в регистр еах (`mov еах,buf1`) и вызовем функцию `sprintLF`.



The image shows a terminal window with a dark background. The title bar at the top reads "aatarasova@aatarasova-VirtualBo". The terminal content displays assembly code for a file named "/home/aa~06-1.asm". The code includes a header file, defines a buffer in the .bss section, and contains logic in the .text section to calculate the sum of 6 and 4, store it in the buffer, and print it with a newline before calling a quit function.

```
/home/aa~06-1.asm  [ - - - - ]  9  L: [ 1+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. 3.1: Программа lab6-1.asm

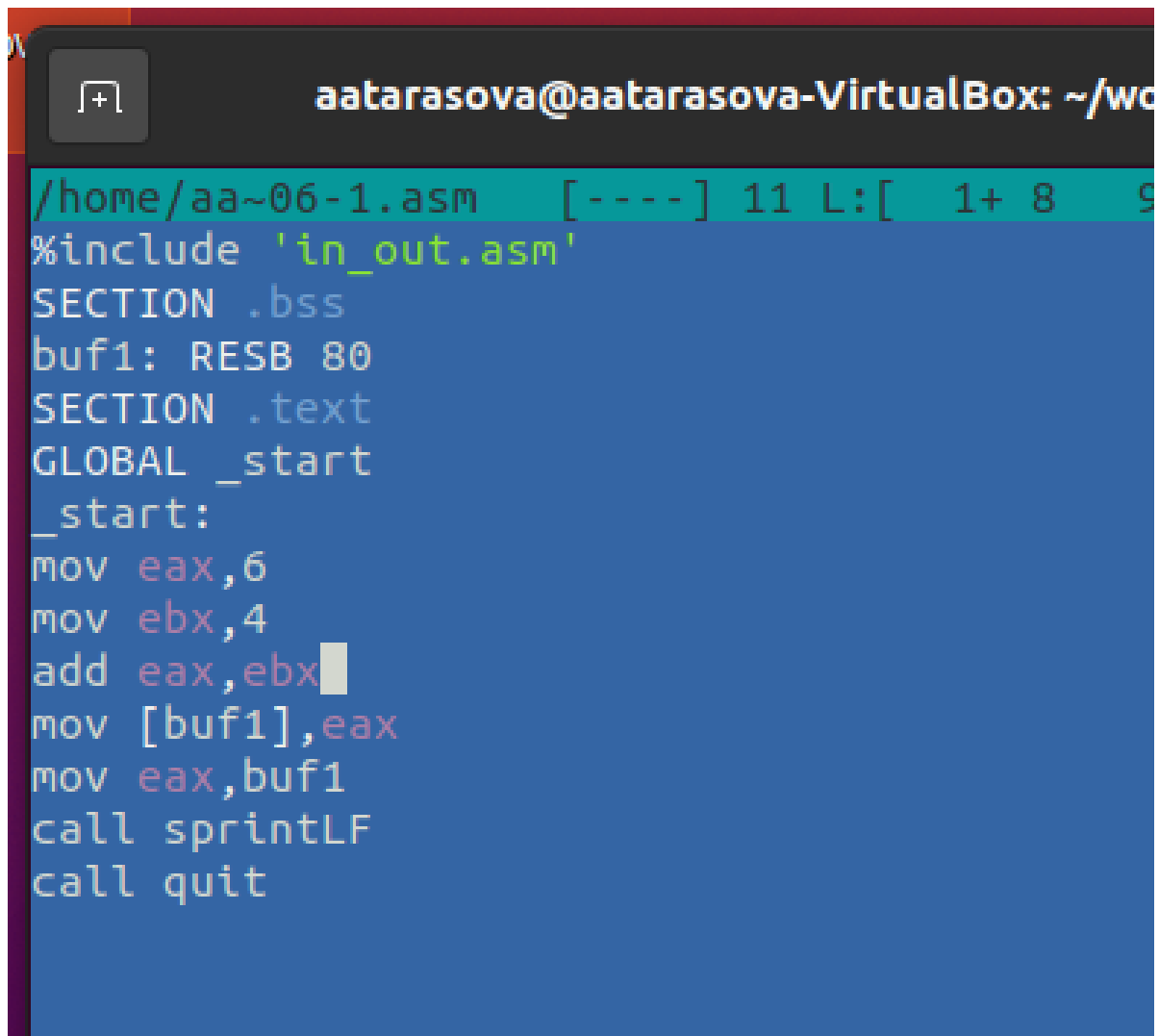


```
aatarasova@aatarasova-VirtualBox: ~/work/arch-pc/lab06
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o
-o lab06-1
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ./lab06-1
j
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$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

Рис. 3.2: Запуск программы lab6-1.asm

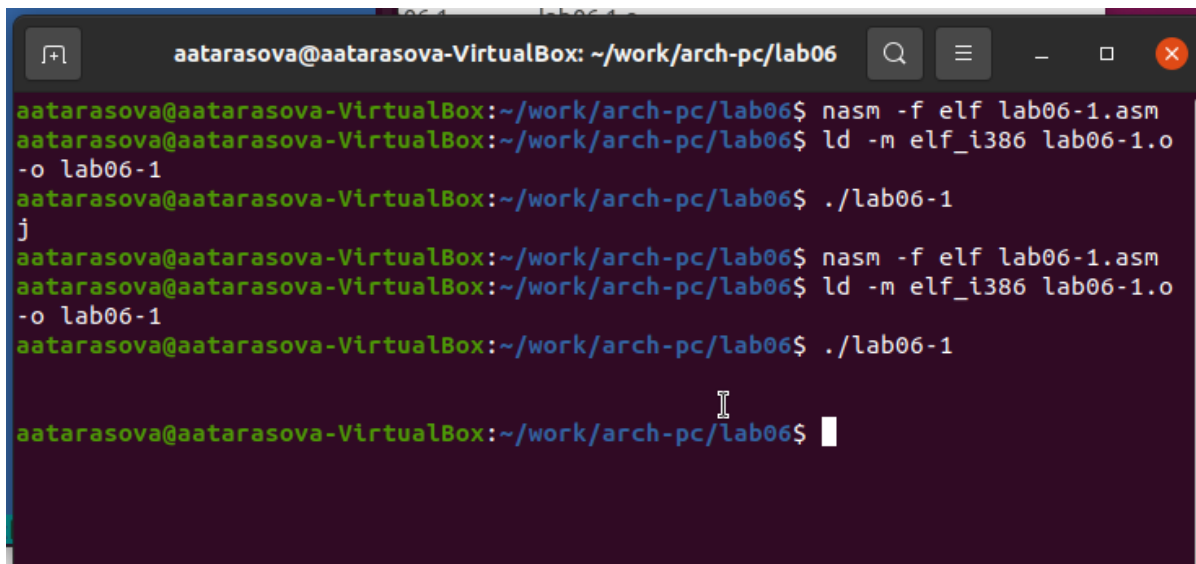
В данном случае при выводе значения регистра `eax` мы ожидаем увидеть число 10. Однако результатом будет символ `j`. Это происходит потому, что код символа `б` равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа `4` – 00110100 (52). Команда `add eax,ebx` запишет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

Далее изменяю текст программы и вместо символов, запишем в регистры числа. (рис. [3.3]) (рис. [3.4])



```
aatarasova@aatarasova-VirtualBox: ~/wo
/home/aa~06-1.asm [ - - - - ] 11 L: [ 1+ 8 9
#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. 3.3: Программа lab6-1.asm



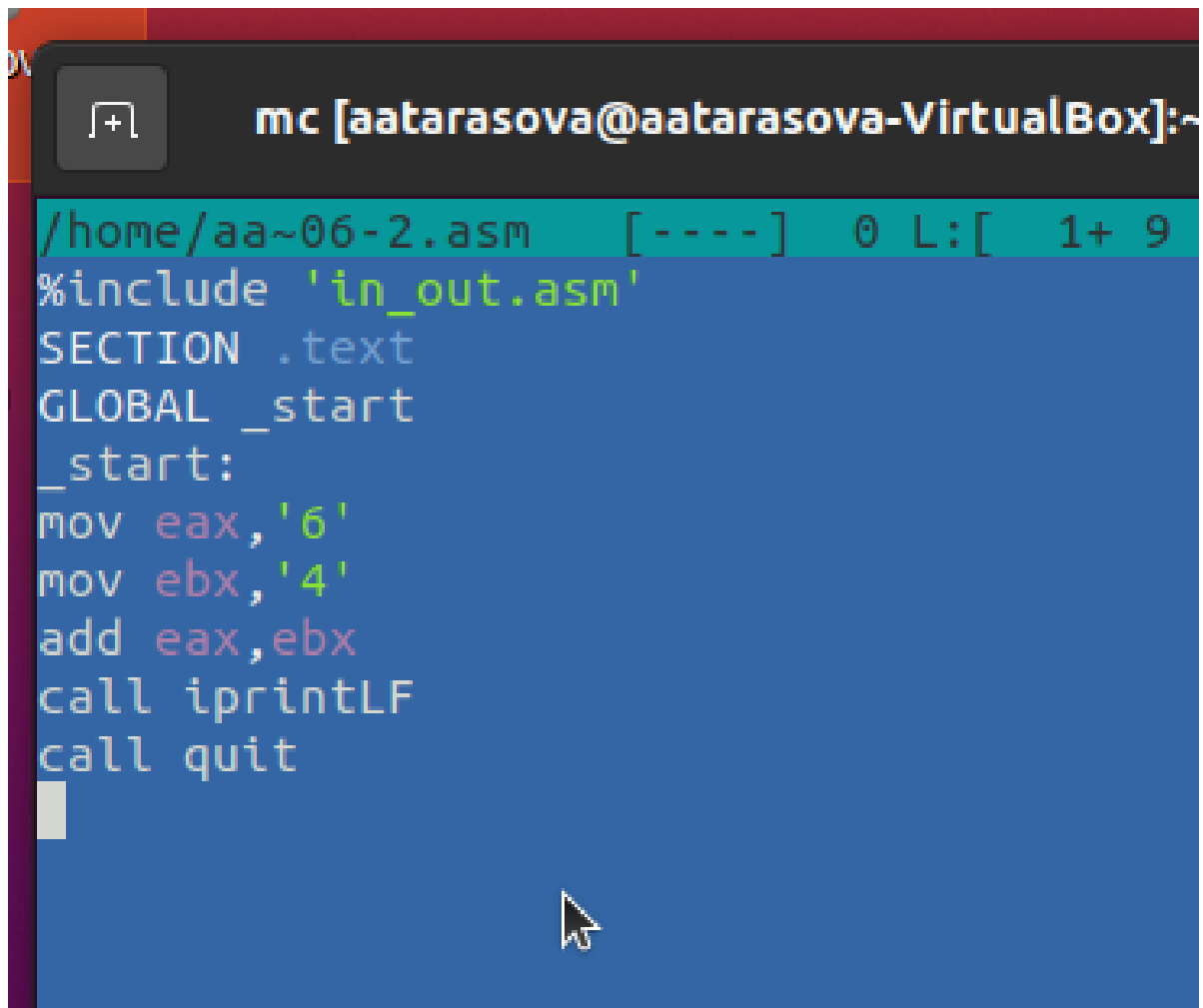
```
aatarasova@aatarasova-VirtualBox: ~/work/arch-pc/lab06
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o
-o lab06-1
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ./lab06-1
j
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$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

aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$
```

Рис. 3.4: Запуск программы lab6-1.asm

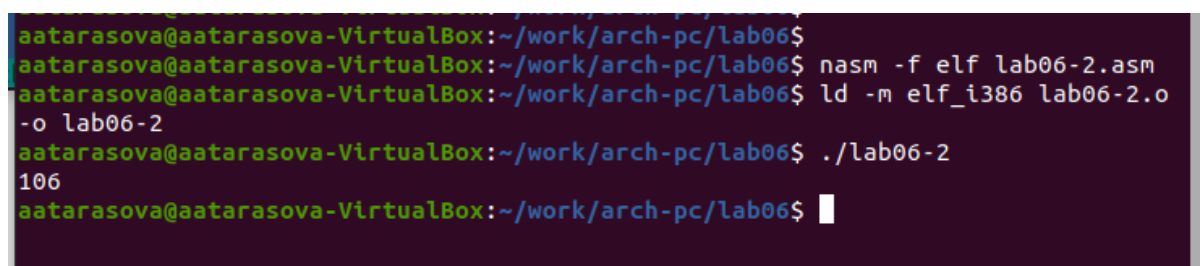
Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Это символ конца строки (возврат каретки). В консоле он не отображается, но добавляет пустую строку.

Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразую текст программы с использованием этих функций. (рис. [3.5]) (рис. [3.6])



```
mc [aatarasova@aatarasova-VirtualBox]:~  
/home/aa~06-2.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. 3.5: Программа lab6-2.asm



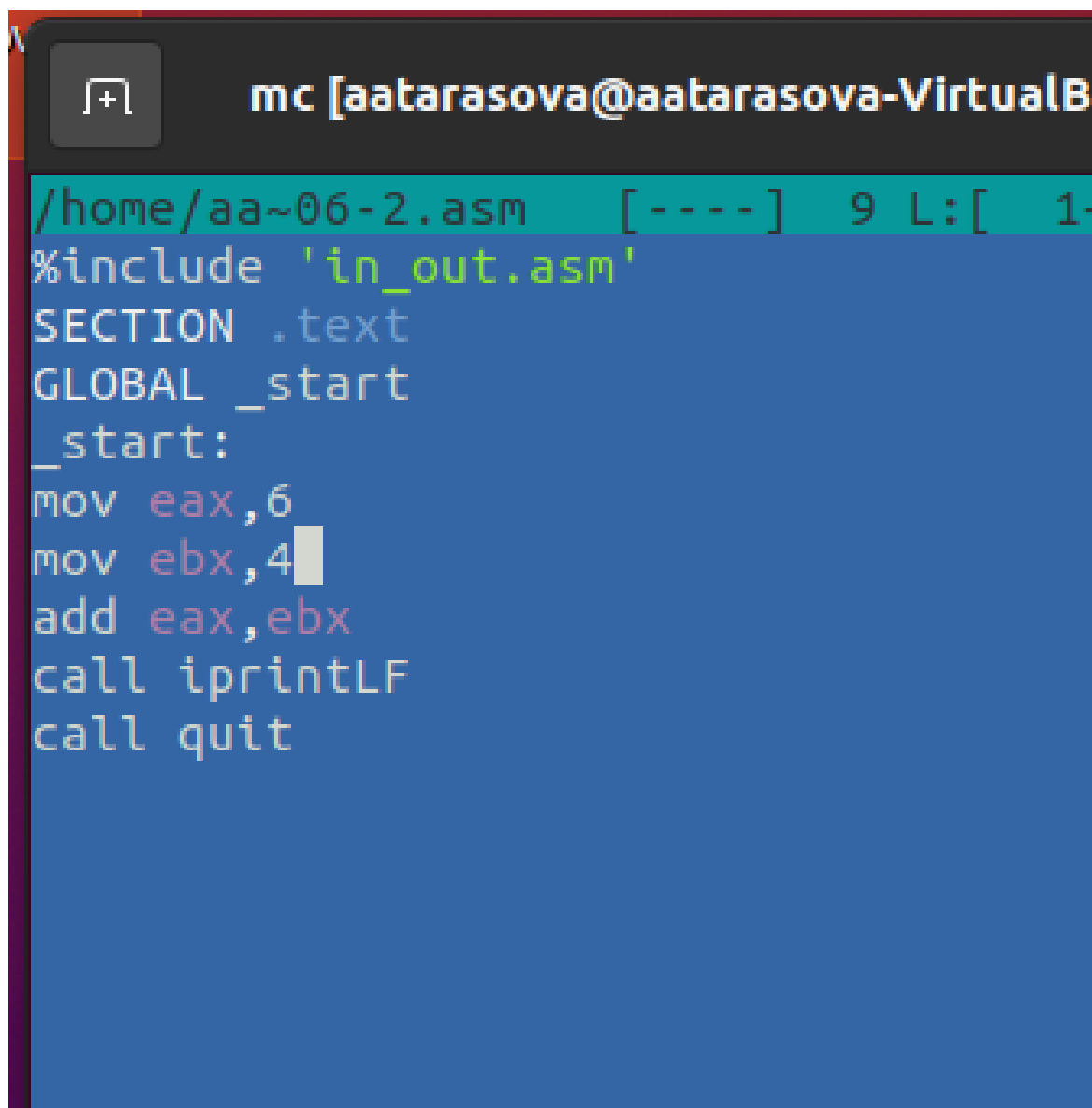
```
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$  
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm  
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o  
-o lab06-2  
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ./lab06-2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

Рис. 3.6: 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.

Аналогично предыдущему примеру изменим символы на числа. (рис. [3.7]) (рис. [3.8])



```
mc [aatarasova@aatarasova-VirtualBo  
/home/aa~06-2.asm [ - - - - ] 9 L:[ 1+  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
call quit
```

Рис. 3.7: Программа lab6-2.asm

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

```

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

```

Рис. 3.8: Запуск программы lab6-2.asm

Заменяла функцию `iprintLF` на `iprint`. Создала исполняемый файл и запустила его. Вывод отличается тем, что нет переноса строки. (рис. [3.9])

```

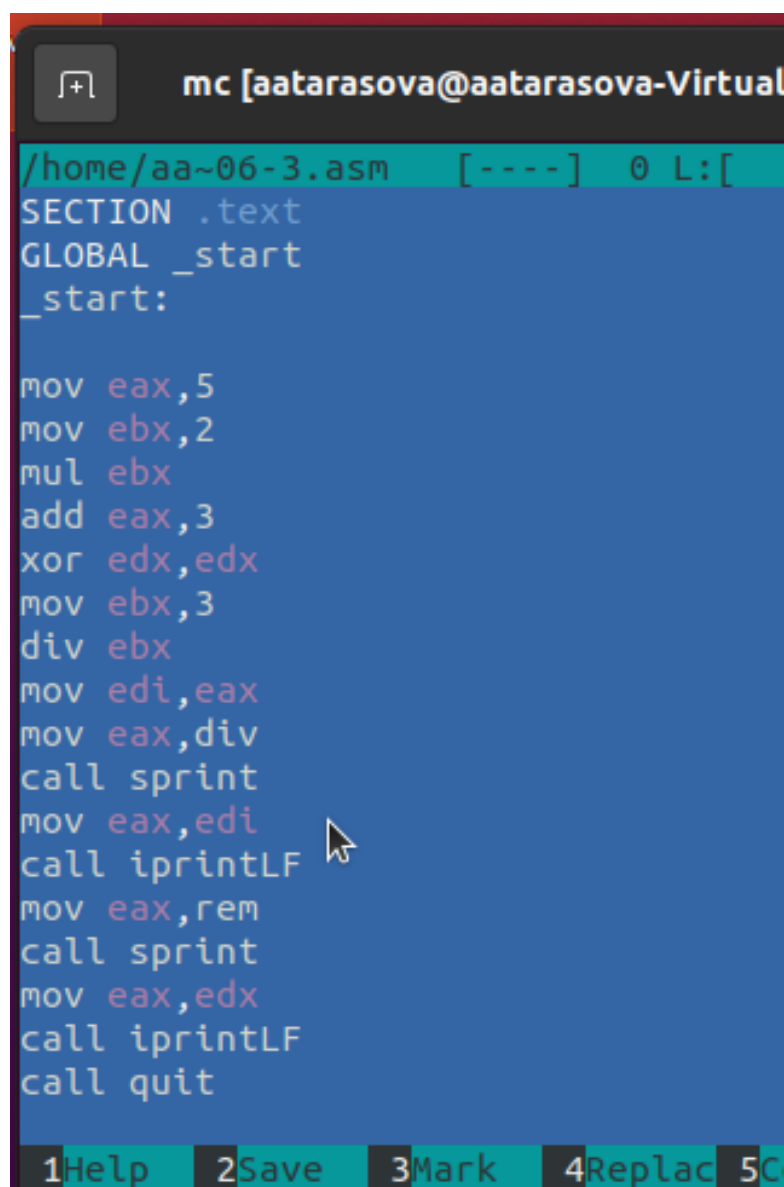
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o
-o lab06-2
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ./lab06-2
106
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$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

```

Рис. 3.9: Запуск программы lab6-2.asm

3.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3) / 3$. (рис. [3.10]) (рис. [3.11])



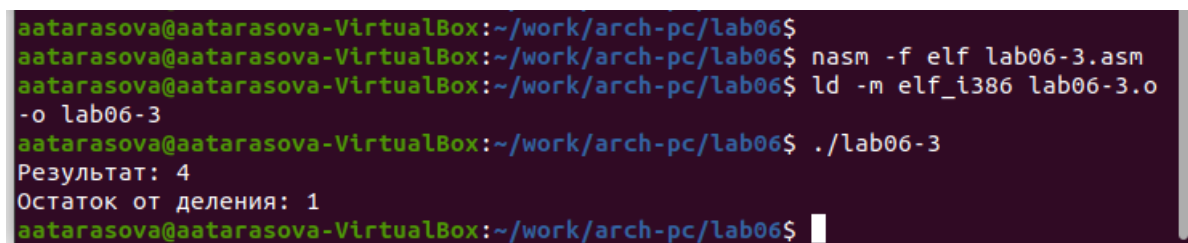
The screenshot shows a text editor window titled "mc [aatarasova@aatarasova-Virtual...". The editor displays the assembly code for a file named "/home/aa~06-3.asm". The code is as follows:

```
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
```

At the bottom of the editor, there is a menu bar with the following items: 1Help, 2Save, 3Mark, 4Replac, 5C.

Рис. 3.10: Программа lab6-3.asm

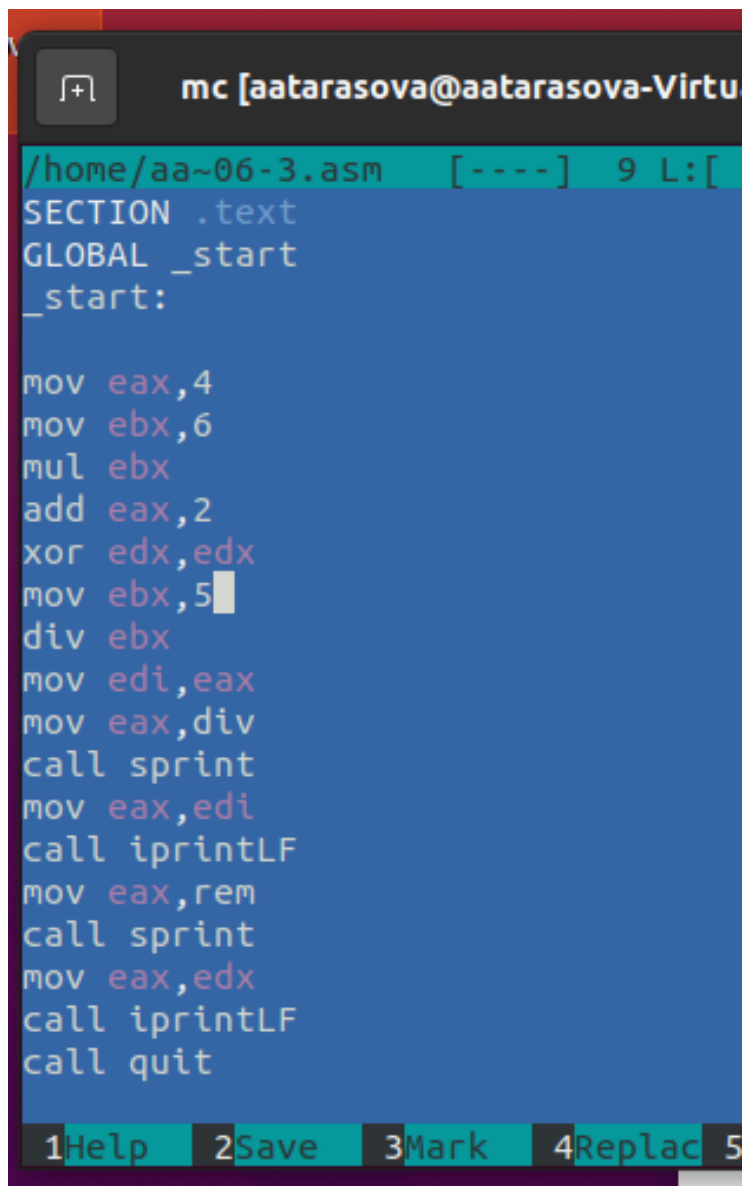


The screenshot shows a terminal window with the following commands and output:

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

Рис. 3.11: Запуск программы lab6-3.asm

Изменила текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$.
Создала исполняемый файл и проверила его работу. (рис. [3.12]) (рис. [3.13])



```
mc [aatarasova@aatarasova-Virtu
/home/aa~06-3.asm [----] 9 L:[
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

1Help 2Save 3Mark 4Replac 5
```

Рис. 3.12: Программа lab6-3.asm


```

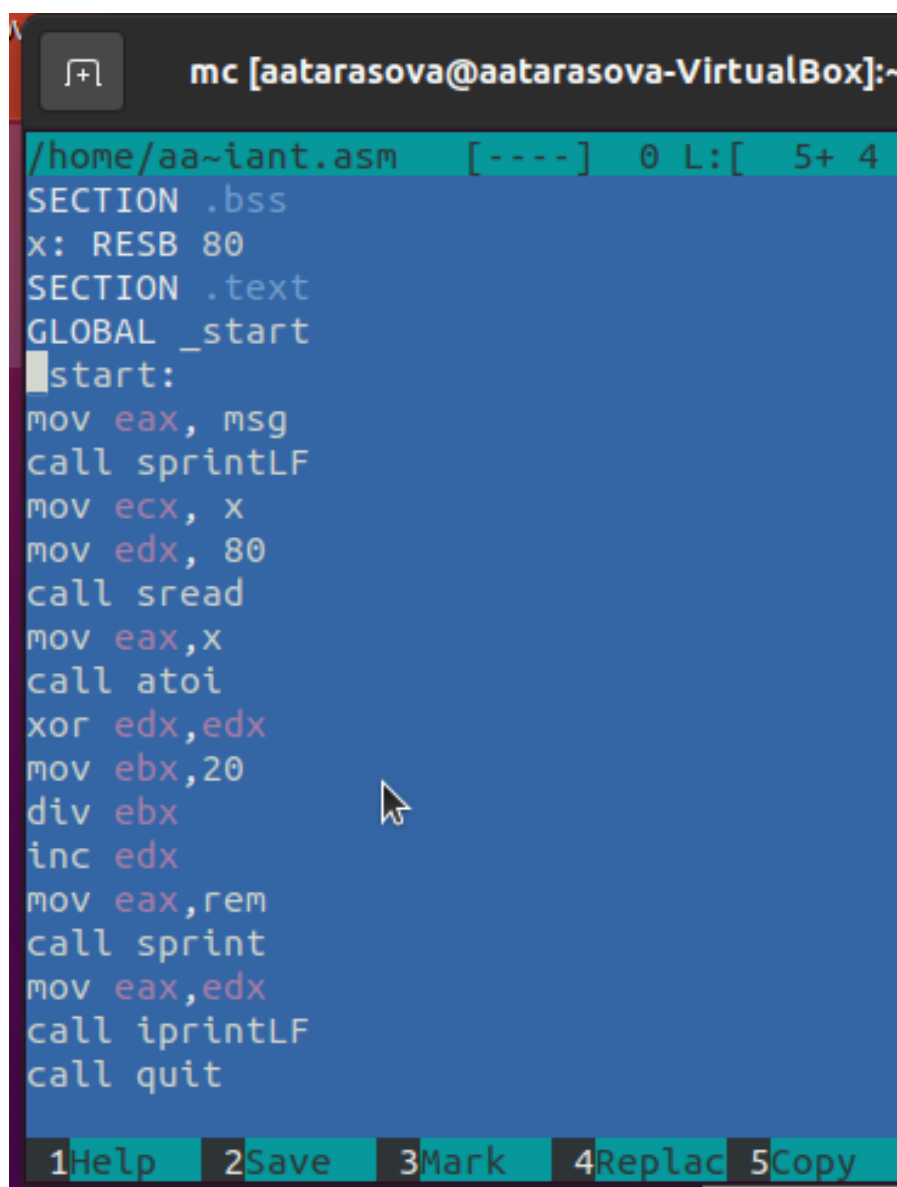
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o
-o lab06-3
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ./lab06-3
Результат: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

```

Рис. 3.13: Запуск программы lab6-3.asm

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. (рис. [3.14]) (рис. [3.15])

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 [aatarasova@aatarasova-VirtualBox]:~  
/home/aa~iant.asm [----] 0 L:[ 5+ 4  
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  
call atoi  
xor edx, edx  
mov ebx, 20  
div ebx  
inc edx  
mov eax, rem  
call sprintf  
mov eax, edx  
call iprintLF  
call quit  
1Help 2Save 3Mark 4Replac 5Copy
```

Рис. 3.14: Программа variant.asm

```

aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ touch variant.asm
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o
-o variant
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132236013
Ваш вариант: 14
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$

```

Рис. 3.15: Запуск программы variant.asm

3.3 Ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения ‘Ваш вариант:’?
 - Инструкция “mov eax, rem” перекладывает значение переменной с фразой ‘Ваш вариант:’ в регистр eax.
 - Инструкция “call sprint” вызывает подпрограмму для вывода строки.
2. Для чего используются следующие инструкции?
 - 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 sread” 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?

- Инструкция “xor edx, edx” обнуляет регистр edx.
- Инструкция “mov ebx, 20” записывает значение 20 в регистр ebx.
- Инструкция “div ebx” выполняет деление номера студенческого билета на 20.
- Инструкция “inc edx” увеличивает значение регистра edx на 1.

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”?

- 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, согласно формуле вычисления варианта.

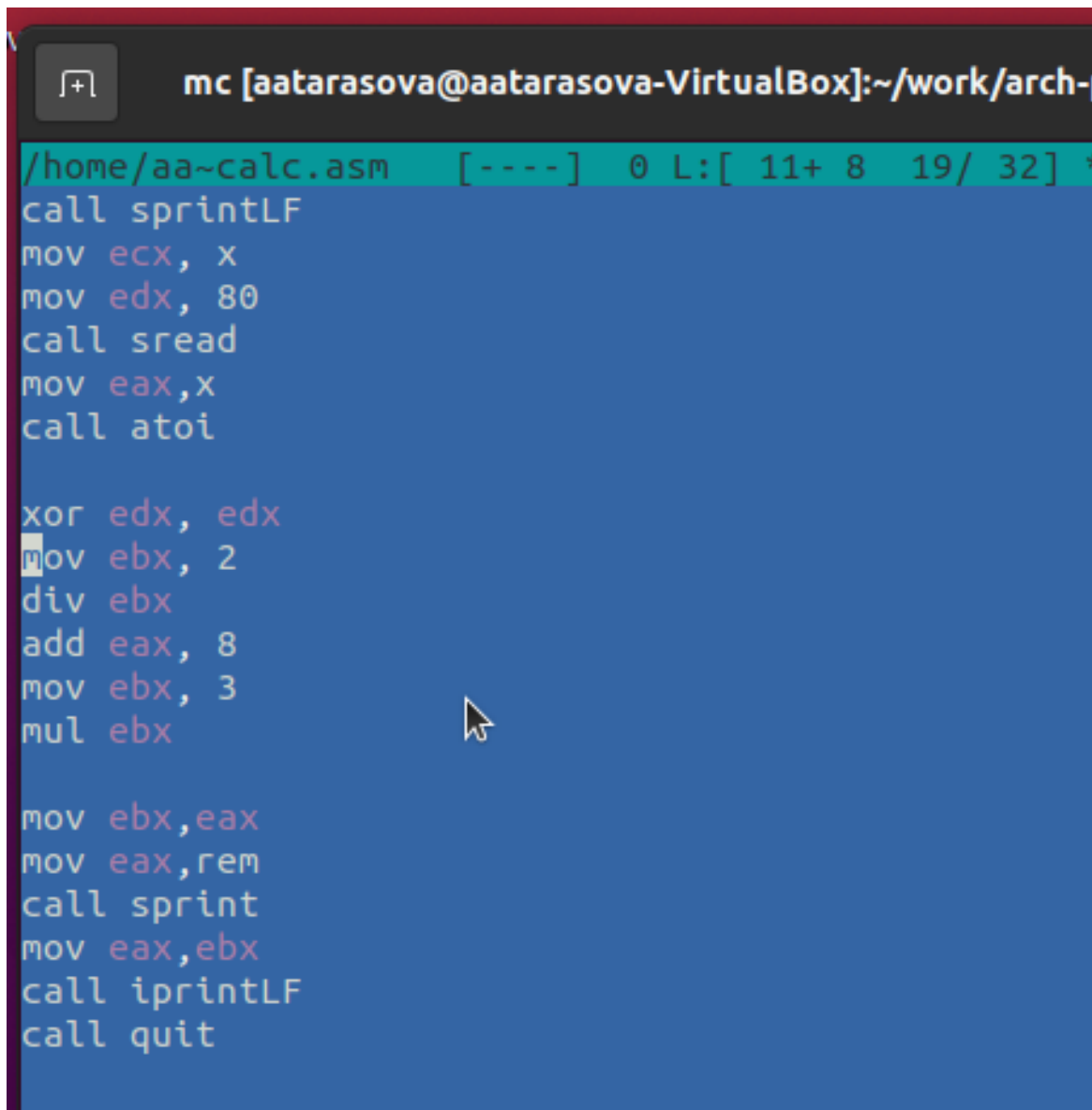
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

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

3.4 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. (рис. [3.16]) (рис. [3.17])

Получили вариант $14 - (x/2 + 8) * 3$ для $x = 1, x = 4$



```
mc [aatarasova@aatarasova-VirtualBox]:~/work/arch-  
/home/aa~calc.asm [----] 0 L:[ 11+ 8 19/ 32]  
call sprintf  
mov ecx, x  
mov edx, 80  
call sread  
mov eax, x  
call atoi  
  
xor edx, edx  
mov ebx, 2  
div ebx  
add eax, 8  
mov ebx, 3  
mul ebx  
  
mov ebx, eax  
mov eax, rem  
call sprintf  
mov eax, ebx  
call iprintLF  
call quit
```

Рис. 3.16: Программа calc.asm

При $x = 1$ получается 24.

При $x = 4$ получается 30.

```
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$  
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf calc.asm  
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 calc.o -o  
calc  
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ./calc  
Введите X  
1  
выражение = : 24  
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$ ./calc  
Введите X  
4  
выражение = : 30  
aatarasova@aatarasova-VirtualBox:~/work/arch-pc/lab06$
```

Рис. 3.17: Запуск программы calc.asm

Программа считает верно.

4 Выводы

Изучили работу с арифметическими операциями.