

Отчёт по лабораторной работе 6

Арифметические операции в NASM

Татьяна Соколова НММбд-03-24

Содержание

| | | |
|----------|---|-----------|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Теоретическое введение | 7 |
| 4 | Выполнение лабораторной работы | 8 |
| 4.1 | Символьные и численные данные в NASM | 8 |
| 4.2 | Выполнение арифметических операций в NASM | 14 |
| 4.3 | Ответы на вопросы | 19 |
| 4.4 | Задание для самостоятельной работы | 20 |
| 5 | Выводы | 23 |

Список иллюстраций

| | | |
|------|--|----|
| 4.1 | Программа lab6-1.asm | 9 |
| 4.2 | Запуск программы lab6-1.asm | 9 |
| 4.3 | Программа lab6-1.asm | 10 |
| 4.4 | Запуск программы lab6-1.asm | 11 |
| 4.5 | Программа lab6-2.asm | 12 |
| 4.6 | Запуск программы lab6-2.asm | 12 |
| 4.7 | Программа lab6-2.asm | 13 |
| 4.8 | Запуск программы lab6-2.asm | 14 |
| 4.9 | Запуск программы lab6-2.asm | 14 |
| 4.10 | Программа lab6-3.asm | 15 |
| 4.11 | Запуск программы lab6-3.asm | 15 |
| 4.12 | Программа lab6-3.asm | 16 |
| 4.13 | Запуск программы lab6-3.asm | 17 |
| 4.14 | Программа variant.asm | 18 |
| 4.15 | Запуск программы variant.asm | 18 |
| 4.16 | Программа calc.asm | 21 |
| 4.17 | Запуск программы calc.asm | 22 |

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

3 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 производятся 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`

4 Выполнение лабораторной работы

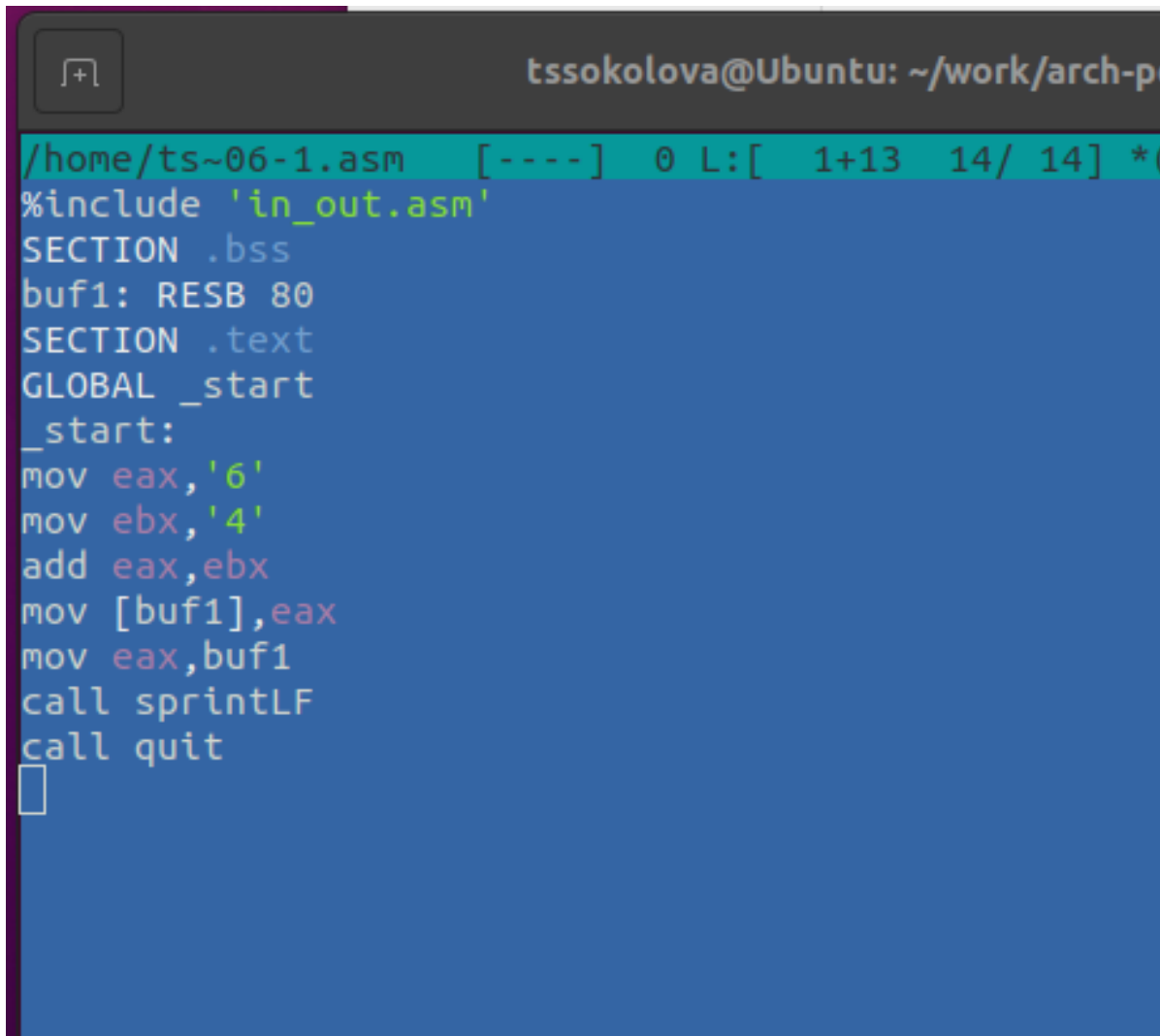
4.1 Символьные и численные данные в NASM

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.

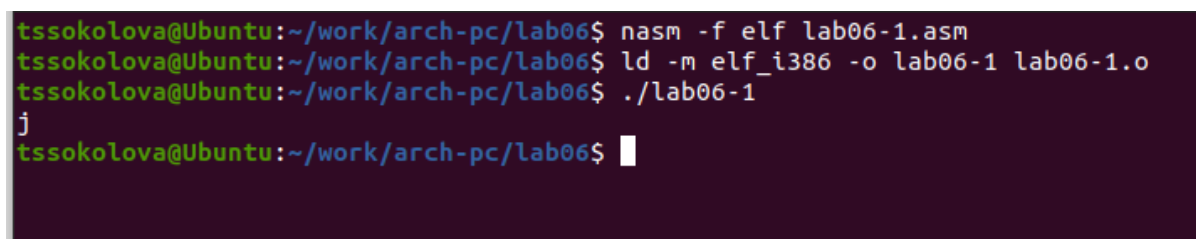
В данной программе в регистр еах записывается символ 6 (`mov еах,'6'`), в регистр ебх символ 4 (`mov ебх,'4'`). Далее к значению в регистре еах прибавляем значение регистра ебх (`add еах,ебх`, результат сложения запишется в регистр еах). Далее выводим результат. (рис. 4.1) (рис. 4.2)

Так как для работы функции `sprintf` 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 `sprintf`.



```
tssokolova@Ubuntu: ~/work/arch-p
/home/ts~06-1.asm [----] 0 L:[ 1+13 14/ 14] *
#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
█
```

Рис. 4.1: Программа lab6-1.asm



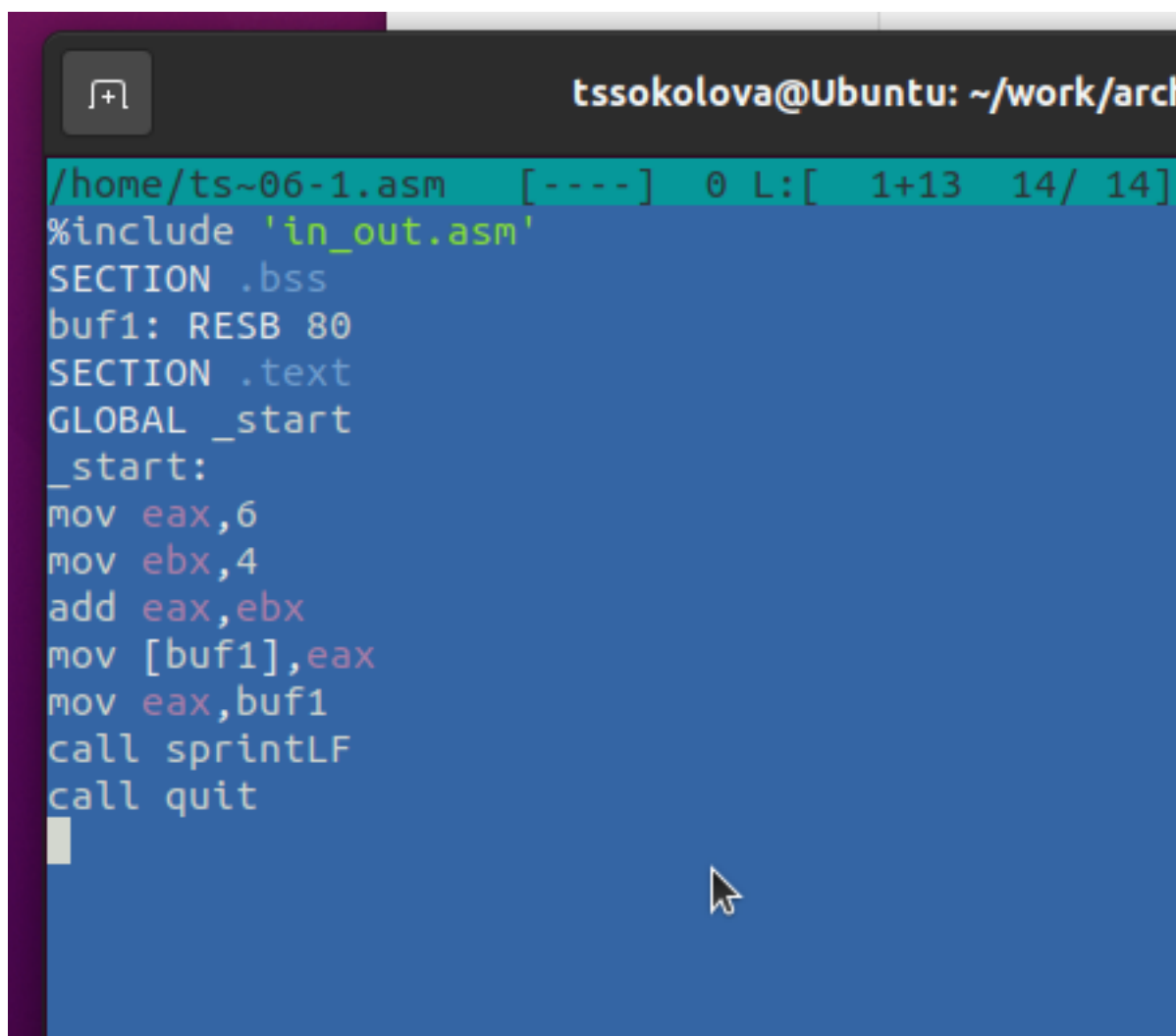
```
tssokolova@Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab06-1 lab06-1.o
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ./lab06-1
j
tssokolova@Ubuntu:~/work/arch-pc/lab06$ █
```

Рис. 4.2: 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`.

Далее изменяю текст программы и вместо символов, запишем в регистры числа. (рис. 4.3) (рис. 4.4)



```
tsokolova@Ubuntu: ~/work/arch
/home/ts~06-1.asm [----] 0 L:[ 1+13 14/ 14]
#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 printf
call _exit
```

Рис. 4.3: Программа lab6-1.asm

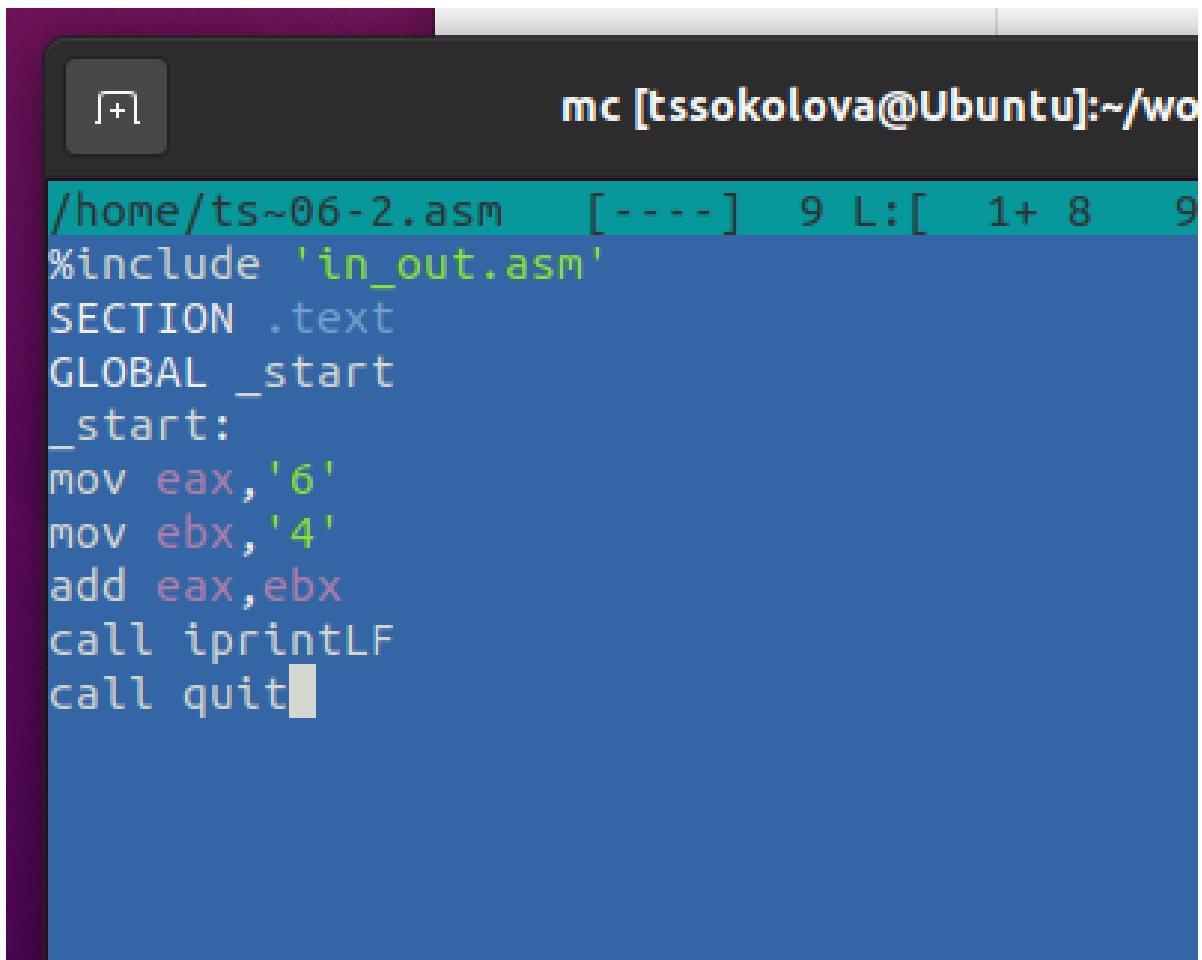
```
tssokolova@Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab06-1 lab06-1.o
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ./lab06-1
j
tssokolova@Ubuntu:~/work/arch-pc/lab06$
tssokolova@Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
tssokolova@Ubuntu:~/work/arch-pc/lab06$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

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

Рис. 4.4: Запуск программы lab6-1.asm

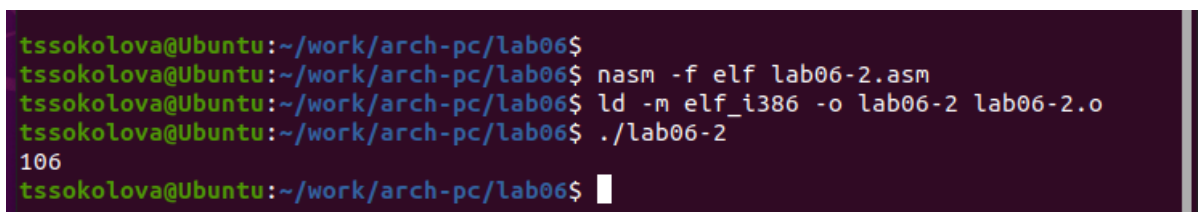
Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Это символ конца строки (возврат каретки). В консоле он не отображается, но добавляет пустую строку.

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. 4.5) (рис. 4.6)



```
mc [tssokolova@Ubuntu]:~/wo
/home/ts~06-2.asm [ - - - - ] 9 L: [ 1+ 8 9
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 4.5: Программа lab6-2.asm



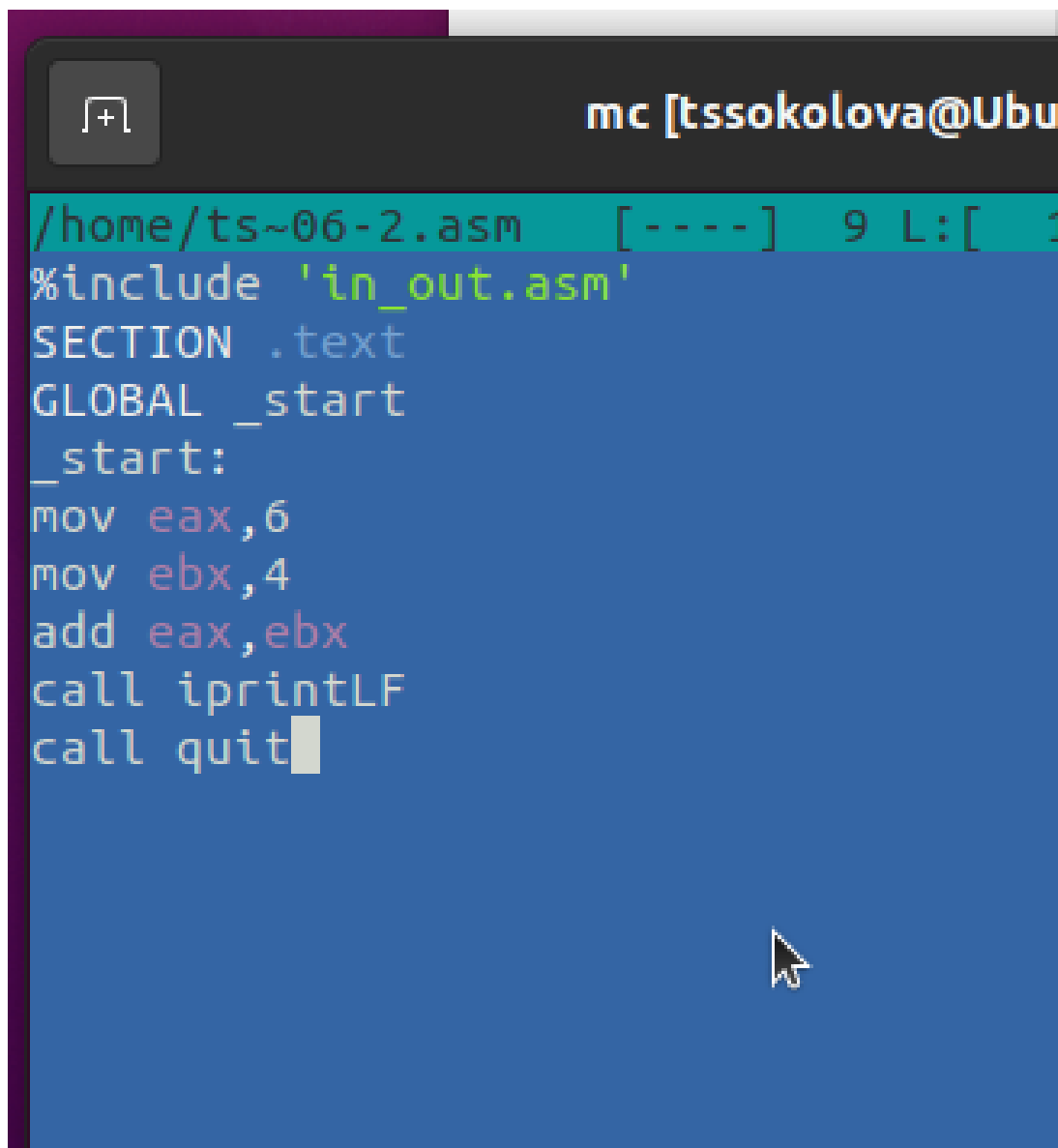
```
tssokolova@Ubuntu:~/work/arch-pc/lab06$
tssokolova@Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab06-2 lab06-2.o
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ./lab06-2
106
tssokolova@Ubuntu:~/work/arch-pc/lab06$
```

Рис. 4.6: 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. 4.7) (рис.

4.8)



```
mc [tssokolova@Ubu
/home/ts~06-2.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. 4.7: Программа lab6-2.asm

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

```

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

```

Рис. 4.8: Запуск программы lab6-2.asm

Заменяла функцию `iprintLF` на `iprint`. Создала исполняемый файл и запустила его. Вывод отличается тем, что нет переноса строки. (рис. 4.9)

```

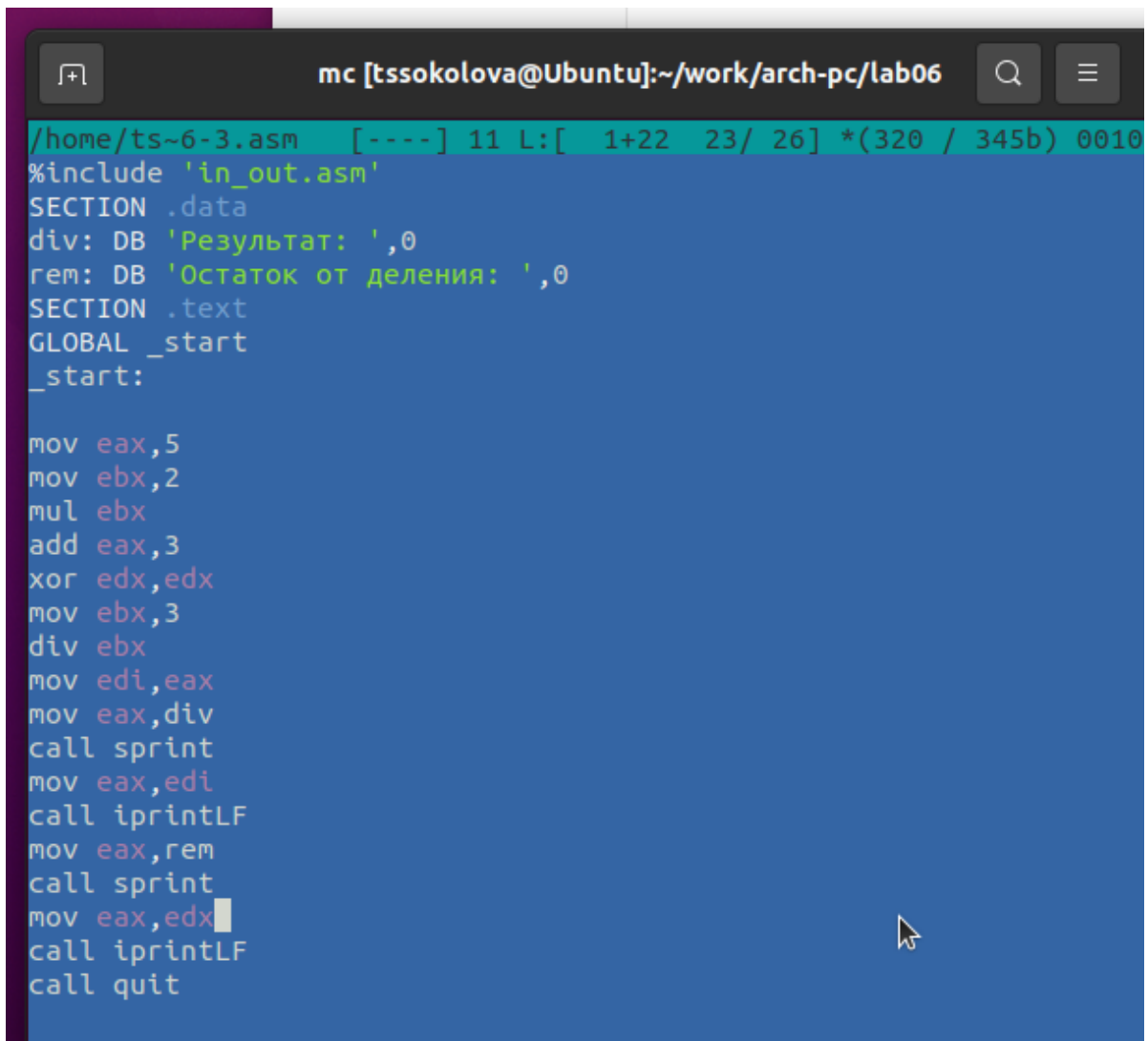
tssokolova@Ubuntu:~/work/arch-pc/lab06$
tssokolova@Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab06-2 lab06-2.o
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ./lab06-2
106
tssokolova@Ubuntu:~/work/arch-pc/lab06$
tssokolova@Ubuntu:~/work/arch-pc/lab06$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

```

Рис. 4.9: Запуск программы lab6-2.asm

4.2 Выполнение арифметических операций в NASM

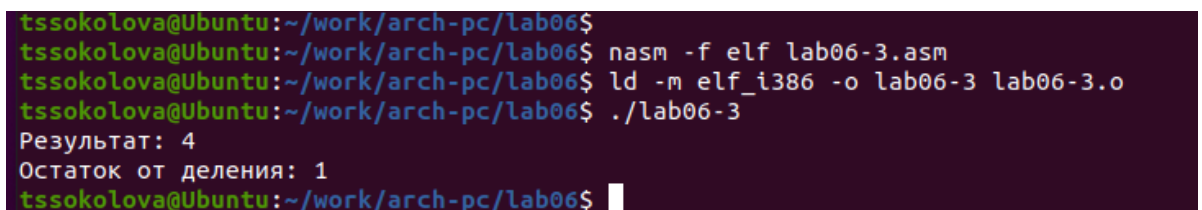
В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3) / 3$. (рис. 4.10) (рис. 4.11)



```
mc [tssokolova@Ubuntu]:~/work/arch-pc/lab06
/home/ts~6-3.asm [----] 11 L:[ 1+22 23/ 26] *(320 / 345b) 0010
#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
```

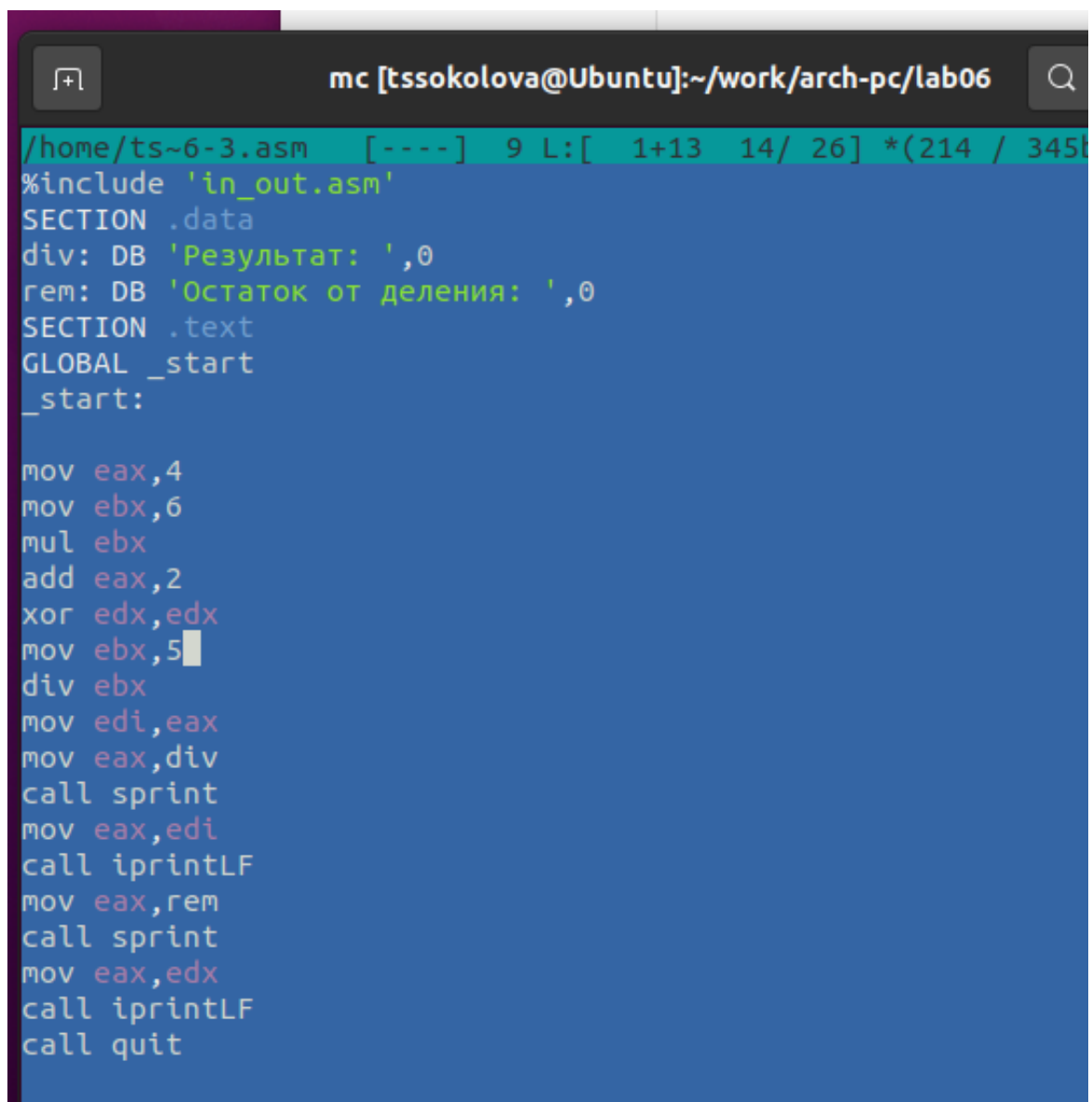
Рис. 4.10: Программа lab6-3.asm



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

Рис. 4.11: 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. 4.12) (рис. 4.13)



```
mc [tssokolova@Ubuntu]:~/work/arch-pc/lab06
/home/ts~6-3.asm  [----]  9 L:[ 1+13 14/ 26] *(214 / 345t
#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
```

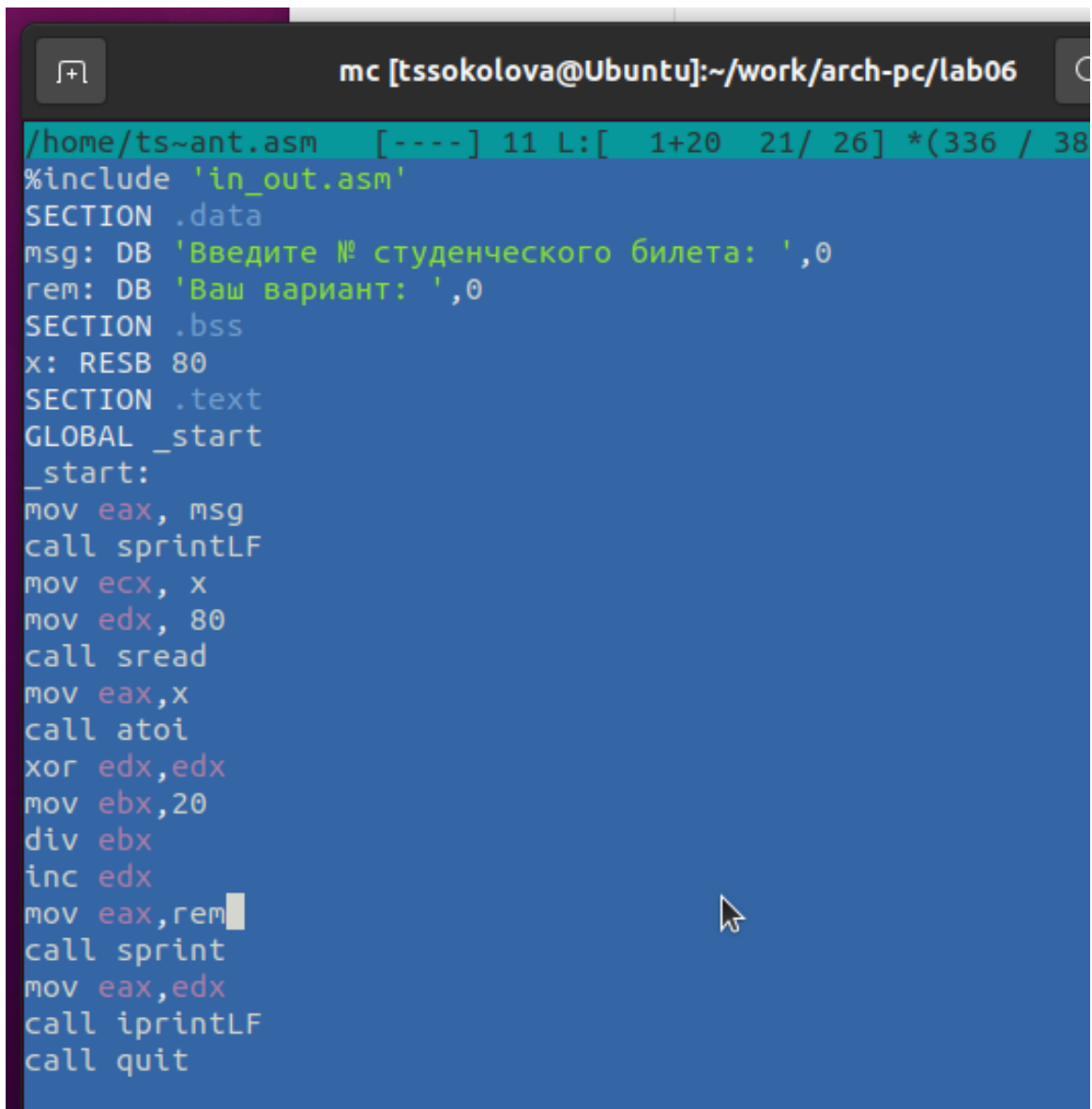
Рис. 4.12: Программа lab6-3.asm


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

Рис. 4.13: Запуск программы lab6-3.asm

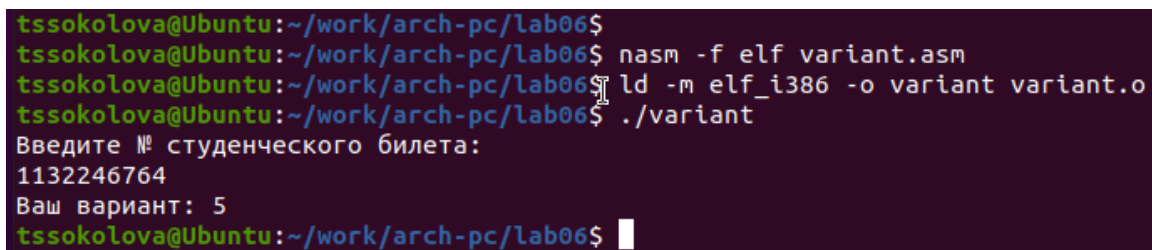
В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. (рис. 4.14) (рис. 4.15)

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 [tssokolova@Ubuntu]:~/work/arch-pc/lab06
/home/ts~ant.asm [----] 11 L:[ 1+20 21/ 26] *(336 / 38
%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 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
```

Рис. 4.14: Программа variant.asm



```
tssokolova@Ubuntu:~/work/arch-pc/lab06$
tssokolova@Ubuntu:~/work/arch-pc/lab06$ nasm -f elf variant.asm
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246764
Ваш вариант: 5
tssokolova@Ubuntu:~/work/arch-pc/lab06$
```

Рис. 4.15: Запуск программы variant.asm

4.3 Ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения ‘Ваш вариант:’?
 - Инструкция “mov eax, ret” 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.

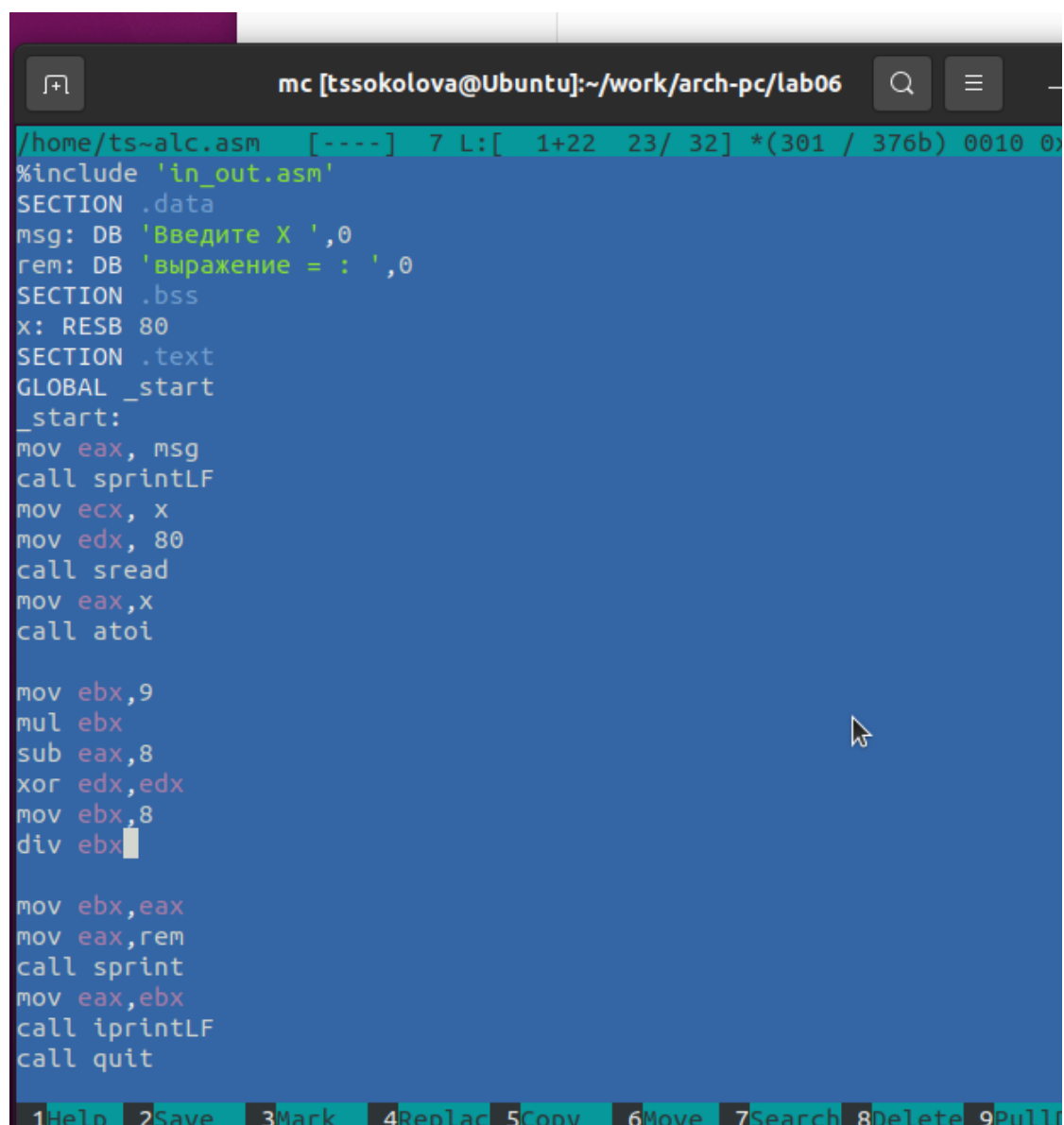
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

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

4.4 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. 4.16) (рис. 4.17)

Получили вариант 5 - $(9x - 8)/8$ для $x = 8, x = 64$



```
/home/ts~alc.asm [----] 7 L:[ 1+22 23/ 32] *(301 / 376b) 0010 00
#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
call atoi

mov ebx, 9
mul ebx
sub eax, 8
xor edx, edx
mov ebx, 8
div ebx

mov ebx, eax
mov eax, rem
call sprintf
mov eax, ebx
call iprintLF
call quit

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9Pull
```

Рис. 4.16: Программа calc.asm

При $x = 8$ получается 8.

При $x = 64$ получается 71.

```
tssokolova@Ubuntu:~/work/arch-pc/lab06$  
tssokolova@Ubuntu:~/work/arch-pc/lab06$ nasm -f elf calc.asm  
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o calc calc.o  
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ./calc  
Введите X  
8  
выражение = : 8  
tssokolova@Ubuntu:~/work/arch-pc/lab06$ ./calc  
Введите X  
64  
выражение = : 71  
tssokolova@Ubuntu:~/work/arch-pc/lab06$
```

Рис. 4.17: Запуск программы calc.asm

Программа считает верно.

5 Выводы

Изучили работу с арифметическими операциями.