

Лабораторная работа №6

Арифметические операции в NASM.

Дмитрий Сергеевич Хохлов

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Ответы на вопросы по программе variant.asm	18
2.2	Самостоятельное задание	19
3	Выводы	22
	Список литературы	23

Список иллюстраций

2.1	Подготовил каталог	6
2.2	Программа в файле lab6-1.asm	7
2.3	Запуск программы lab6-1.asm	8
2.4	Программа в файле lab6-1.asm	9
2.5	Запуск программы lab6-1.asm	9
2.6	Программа в файле lab6-2.asm	10
2.7	Запуск программы lab6-2.asm	11
2.8	Программа в файле lab6-2.asm	12
2.9	Запуск программы lab6-2.asm	13
2.10	Запуск программы lab6-2.asm	13
2.11	Программа в файле lab6-3.asm	14
2.12	Запуск программы lab6-3.asm	14
2.13	Программа в файле lab6-3.asm	15
2.14	Запуск программы lab6-3.asm	16
2.15	Программа в файле variant.asm	17
2.16	Запуск программы variant.asm	17
2.17	Программа в файле calc.asm	20
2.18	Запуск программы calc.asm	21

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

Сформировал директорию для выполнения заданий лабораторной работы №6, перешел в созданную папку и создал файла lab6-1.asm. (рис. [2.1])

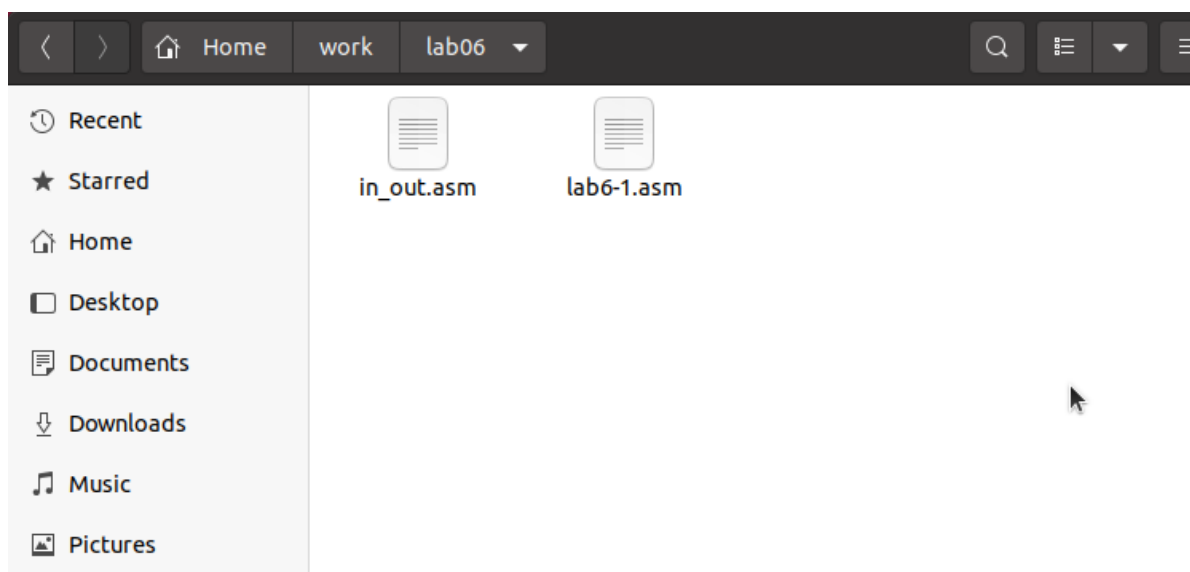
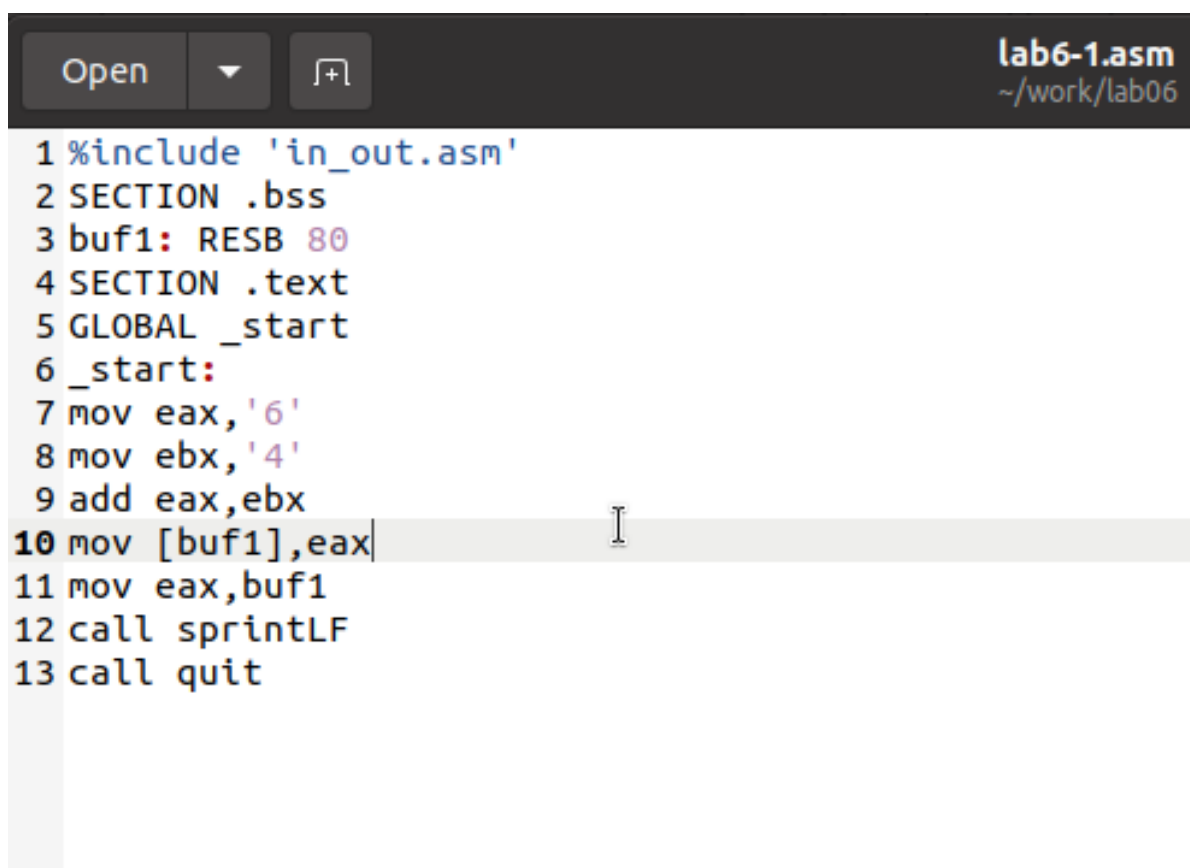


Рис. 2.1: Подготовил каталог

Давайте рассмотрим примеры программ, которые демонстрируют вывод символов и числовых данных. Эти программы будут отображать данные, помещенные в регистр еах.



```
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 printf
13 call quit
```

Рис. 2.2: Программа в файле lab6-1.asm

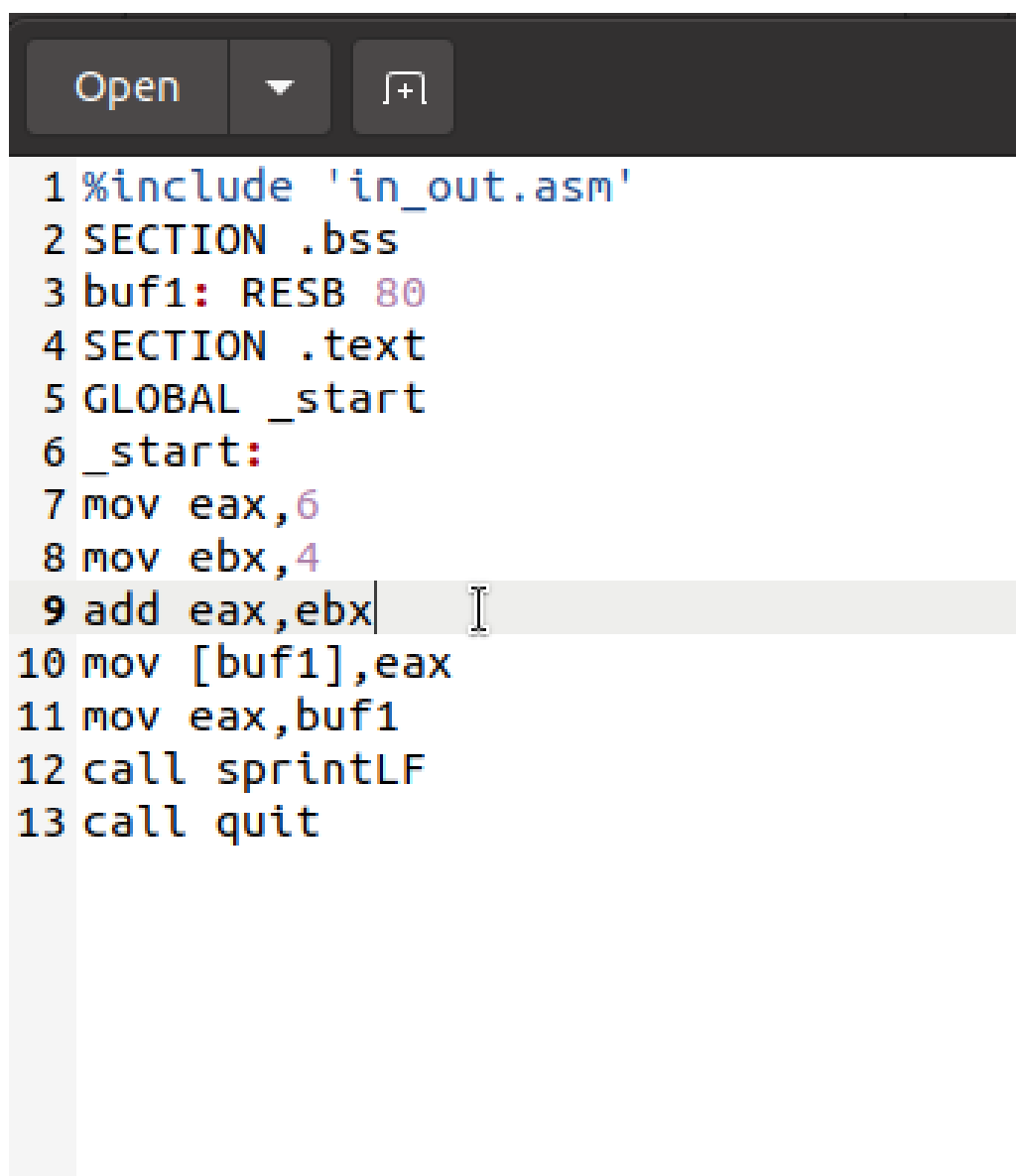
В этом примере программы (см. рисунок [2.2]) мы помещаем символ '6' в регистр `eax` с помощью команды (`mov eax, '6'`), а символ '4' в регистр `ebx` (`mov ebx, '4'`). Далее выполняем сложение значений, содержащихся в регистрах `eax` и `ebx`, с помощью команды (`add eax, ebx`), где результат сложения сохраняется в регистре `eax`. Затем производим вывод полученного результата. Однако для того чтобы функция `sprintf` сработала корректно, нужно, чтобы в регистре `eax` находился адрес. По этой причине мы используем вспомогательную переменную. Значение из регистра `eax` переносим в переменную `buf1` (`mov [buf1], eax`), затем загружаем адрес переменной `buf1` обратно в регистр `eax` (`mov eax, buf1`) и выполняем вызов функции `sprintf`.

```
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ nasm -f elf lab6-1.asm
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ./lab6-1
j
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$
```

Рис. 2.3: Запуск программы lab6-1.asm

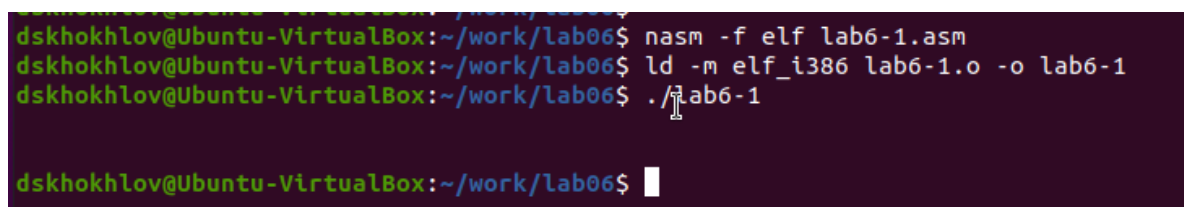
В результате, когда мы ожидаем увидеть число 10 на выводе из регистра `eax`, на деле получим символ 'j'. Это объясняется тем, что символу '6' соответствует двоичный код 00110110 (или 54 в десятичной системе), а символу '4' - двоичный код 00110100 (или 52 в десятичной системе). После выполнения операции сложения `add eax, ebx`, мы получаем сумму кодов - 01101010 (или 106 в десятичной системе), что соответствует символу 'j'. (см. рисунок [2.3])

Затем произвожу корректировку текста программы, заменив символьные значения на числовые в регистрах. (рис. [2.4])



```
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 sprintf
13 call quit
```

Рис. 2.4: Программа в файле lab6-1.asm



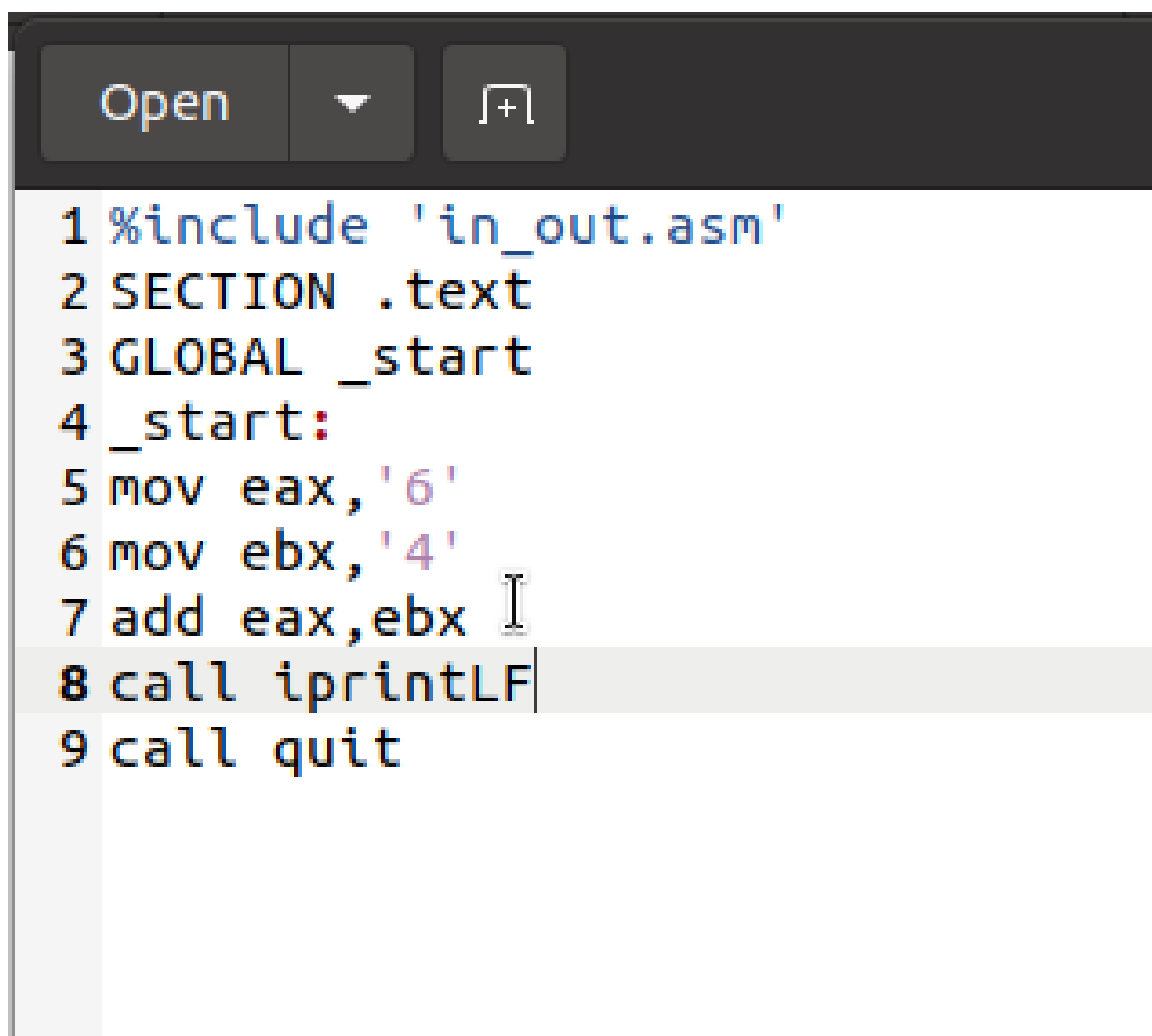
```
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ nasm -f elf lab6-1.asm
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ./lab6-1

dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$
```

Рис. 2.5: Запуск программы lab6-1.asm

Так же, как и в прошлый раз, в результате выполнения программы число 10 не будет выведено. Взамен этого на экране появится символ с кодом 10, который является символом новой строки (перевод строки). Смотрите (рис. [2.5]). Этот символ не виден в окне консоли, но он создаёт пустую строку после вывода.

Как было указано ранее, в файле `in_out.asm` содержатся специальные подпрограммы для того, чтобы преобразовывать символы ASCII в числовые значения и наоборот. Я внес изменения в текст программы, чтобы использовать эти подпрограммы.



```
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
```

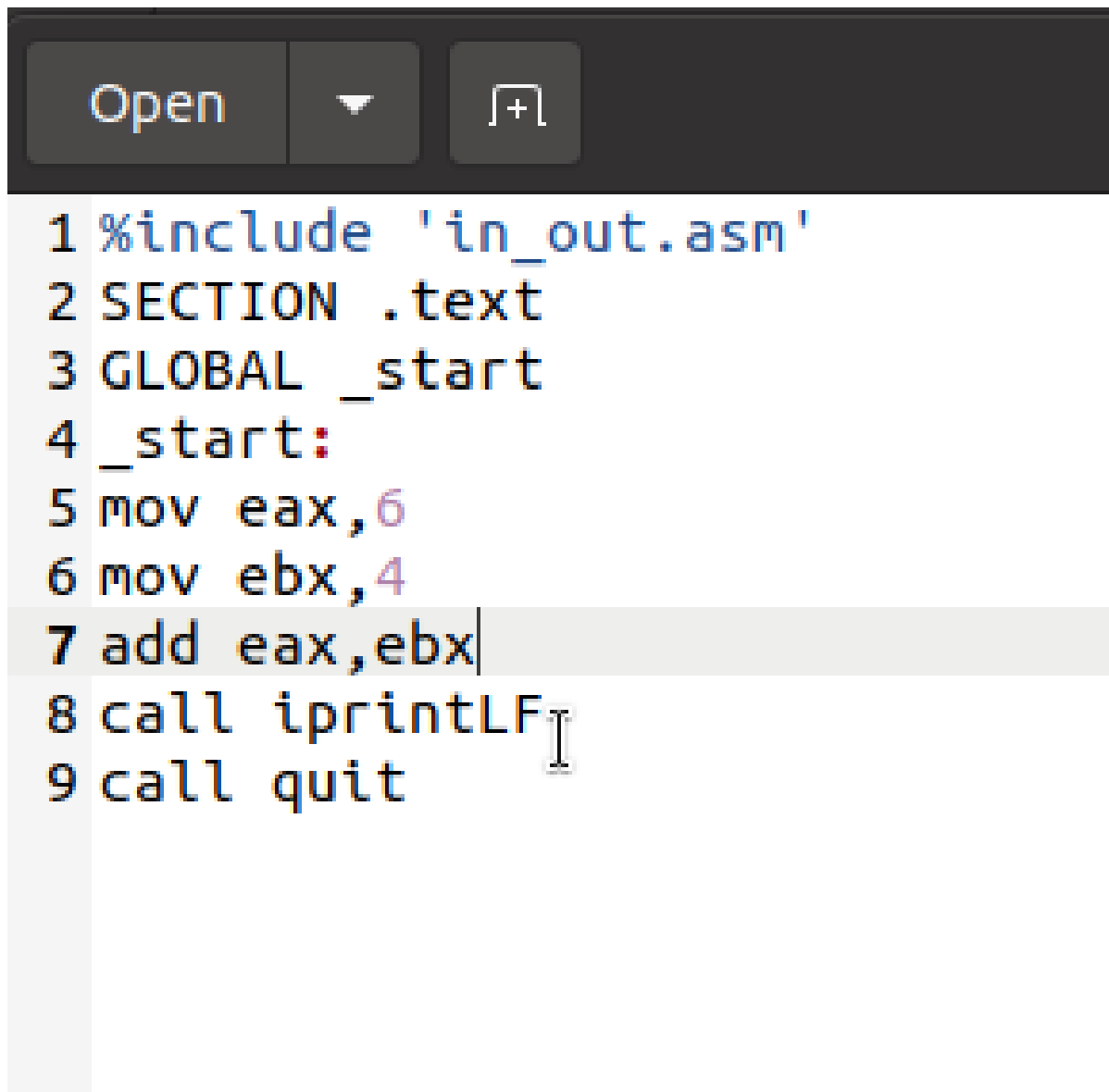
Рис. 2.6: Программа в файле `lab6-2.asm`

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

Рис. 2.7: Запуск программы lab6-2.asm

По завершении работы программы на экран будет выведено число 106. (рис. [2.7]) В этом примере, так же как и в первом, инструкция `add` суммирует коды символов '6' и '4' ($54+52=106$). Но в этот раз, в отличие от предыдущего случая, благодаря функции `iprintLF`, выводится само число, а не символ с соответствующим числовым кодом.

В соответствии с предыдущим примером, мы заменим символы на числа. (рис. [2.8])



```
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. 2.8: Программа в файле lab6-2.asm

Функция `iprintLF` теперь выводит число, так как в качестве операндов использовались числа, а не их символьные коды. В итоге на экране появляется число 10. (рис. [2.9])

```

dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ./lab6-2
10
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$

```

Рис. 2.9: Запуск программы lab6-2.asm

Произвел замену функции `iprintLF` на `iprint`. Скомпилировал и запустил скомпилированный файл. Основное отличие в результате – отсутствие новой строки в конце вывода. (см. рисунок [2.10])

```

dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ./lab6-2
10dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$

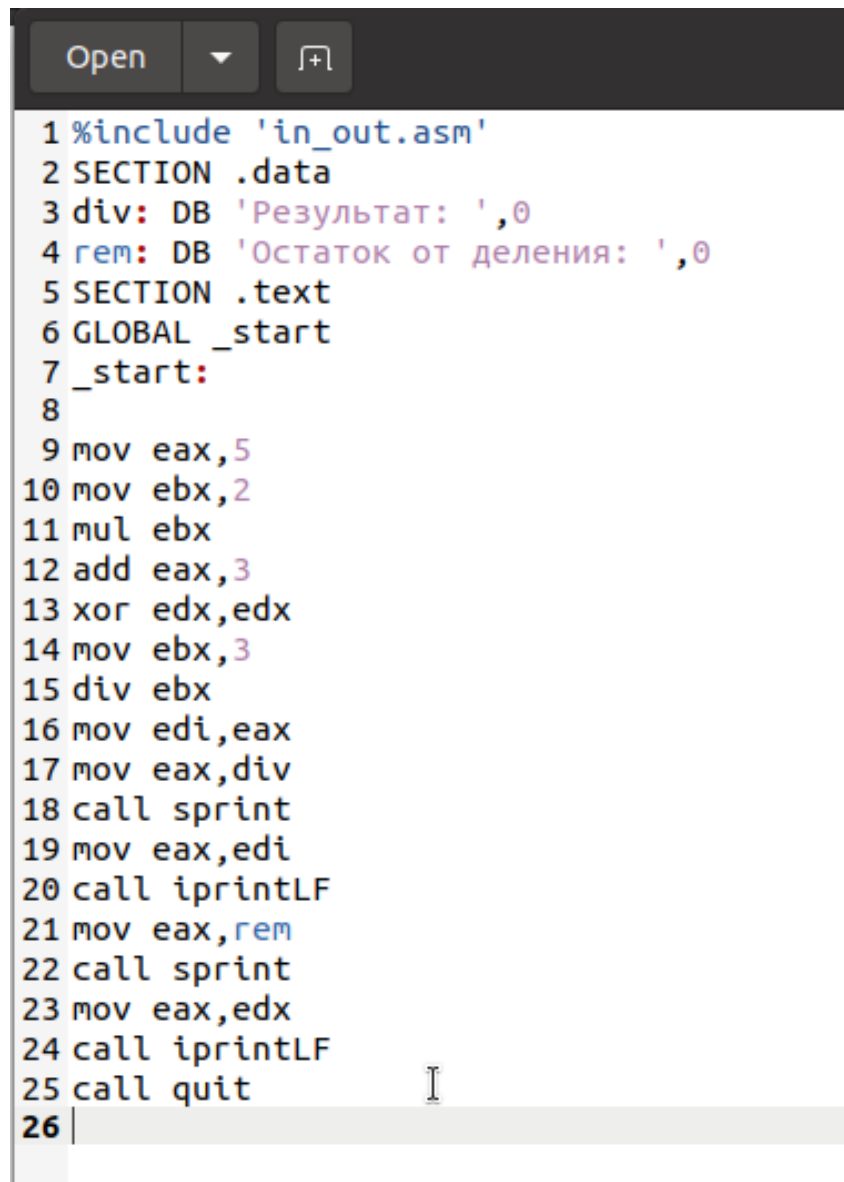
```

Рис. 2.10: Запуск программы lab6-2.asm

Для демонстрации выполнения арифметических операций в NASM, рассмотрим программу, которая вычисляет следующее арифметическое выражение: (см. рисунки [2.11] и [2.12])

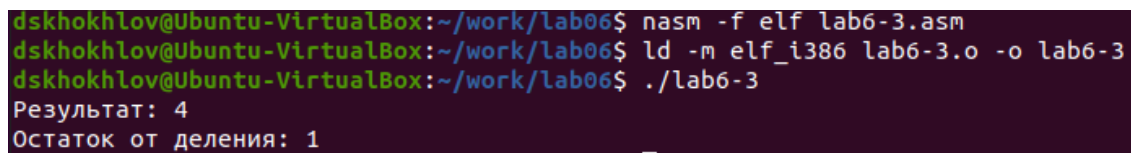
$$f(x) = (5 * 2 + 3) / 3$$

.



```
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
26 |
```

Рис. 2.11: Программа в файле lab6-3.asm



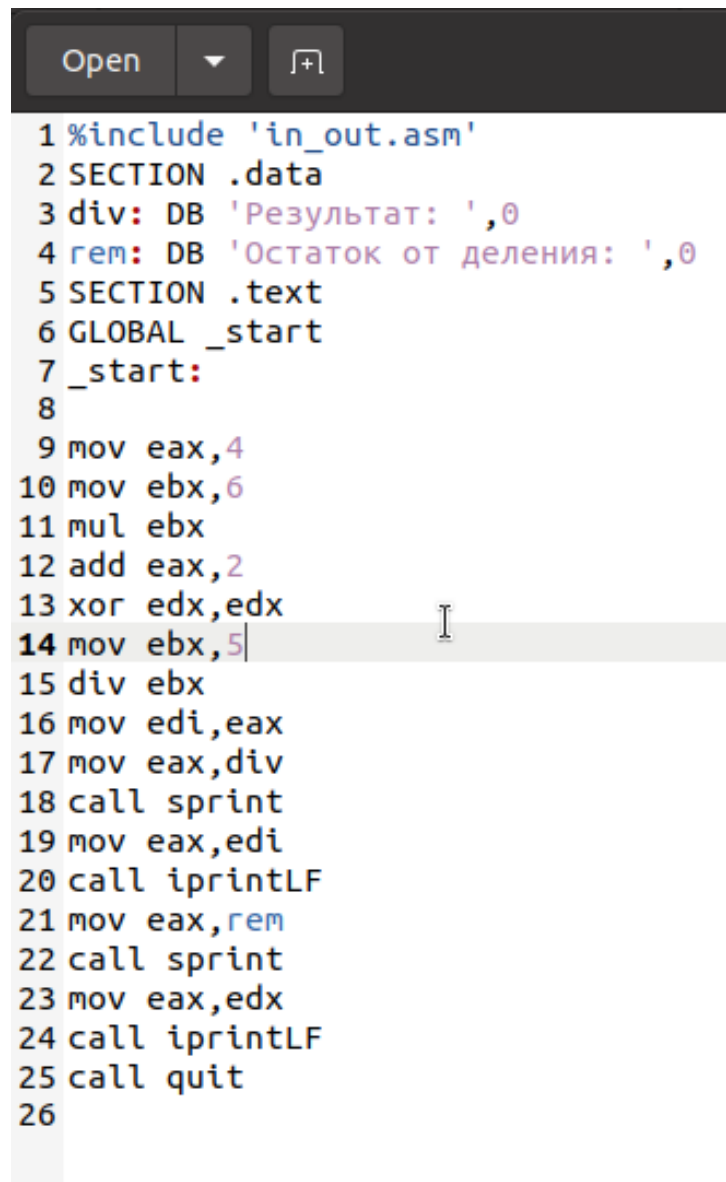
```
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ nasm -f elf lab6-3.asm
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 2.12: Запуск программы lab6-3.asm

Модифицировал код программы для расчета нового выражения:

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

. Собрал исполняемый файл и провел его тестирование. (см. рисунки [2.13] и [2.14])



```
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
26
```

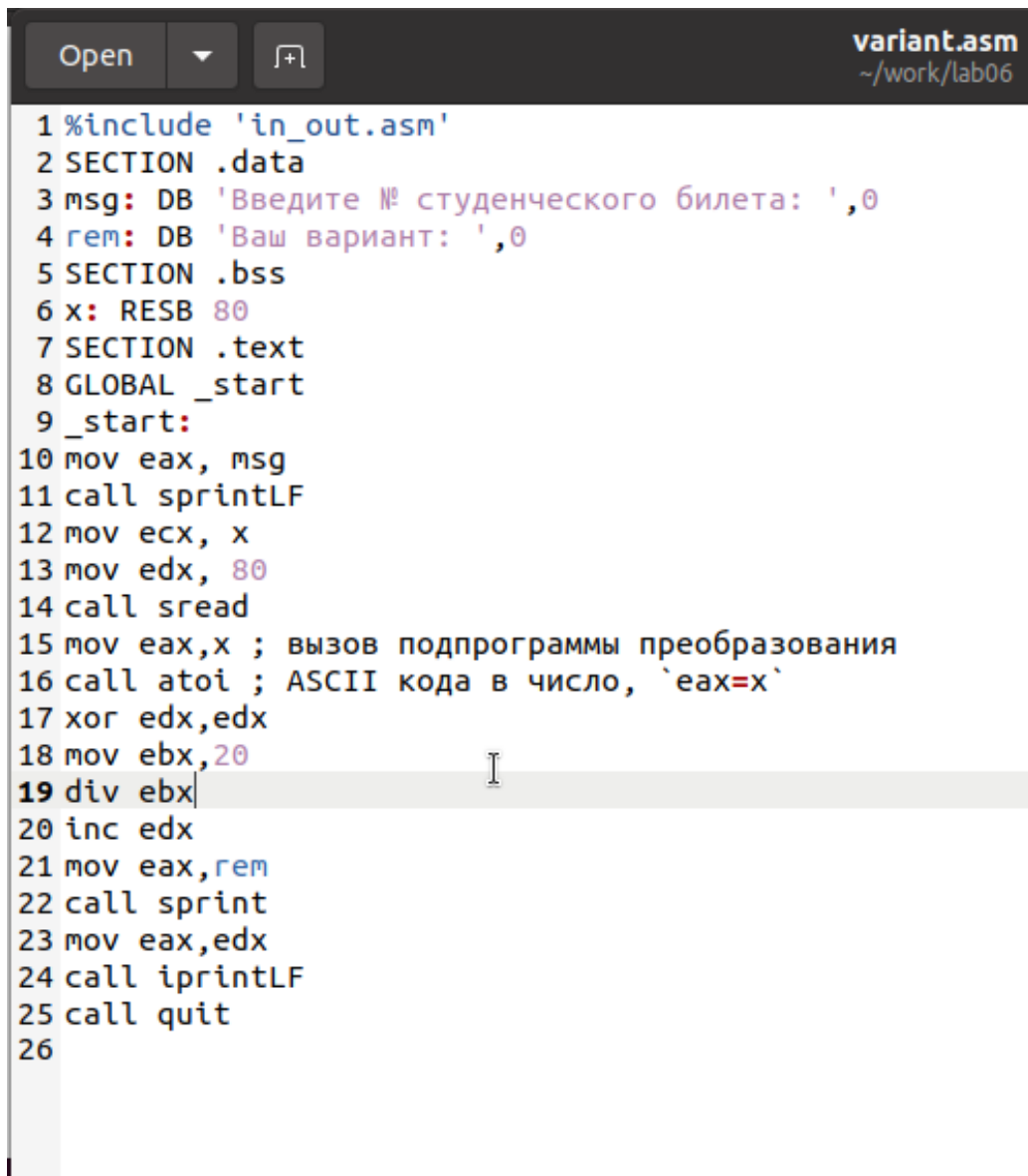
Рис. 2.13: Программа в файле lab6-3.asm

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

Рис. 2.14: Запуск программы lab6-3.asm

В качестве еще одного примера возьмем программу, которая вычисляет вариант задания на основе номера студенческого билета. (см. рисунки [2.15] и [2.16])

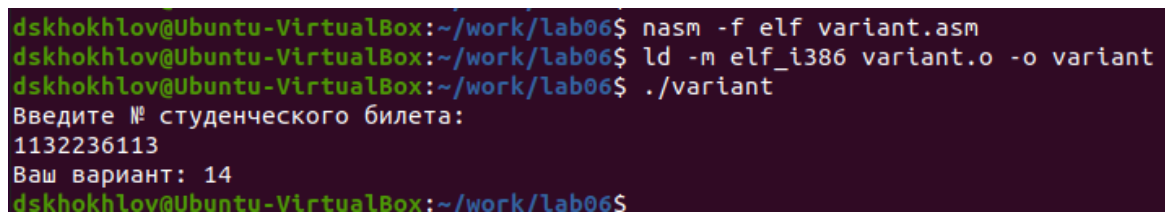
В этом случае число для арифметических операций вводится пользователем через клавиатуру. Как было указано ранее, ввод происходит в символьной форме, и для правильного выполнения арифметических операций в NASM эти символы необходимо преобразовать в числовой формат. Для этого можно использовать функцию `atoi`, которая содержится в файле `in_out.asm`.



```
variant.asm
~/work/lab06

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
```

Рис. 2.15: Программа в файле variant.asm



```
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ nasm -f elf variant.asm
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ld -m elf_i386 variant.o -o variant
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$ ./variant
Введите № студенческого билета:
1132236113
Ваш вариант: 14
dskhokhlov@Ubuntu-VirtualBox:~/work/lab06$
```

Рис. 2.16: Запуск программы variant.asm

2.1 Ответы на вопросы по программе variant.asm

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

Команда “mov eax, rem” загружает в регистр значение, соответствующее строке “Ваш вариант:”

Команда “call sprint” инициирует процедуру, которая выводит строку на экран

2. Для чего используются следующие инструкции?

Команда “mov ecx, x” копирует в регистр ecx значение из переменной x

Команда “mov edx, 80” заносит число 80 в регистр edx

Команда “call sread” активирует процедуру, которая читает вводимые данные, например номер студенческого билета, с консоли

3. Для чего используется инструкция “call atoi”?

Команда “call atoi” служит для конвертации введенного текста в целочисленное значение

4. Какие строки листинга отвечают за вычисления варианта?

Команда “xor edx, edx” очищает содержимое регистра edx

Команда “mov ebx, 20” помещает число 20 в регистр ebx

Команда “div ebx” осуществляет деление числа, находящегося в аккумуляторе, на содержимое регистра ebx

Команда “inc edx” увеличивает на единицу значение в регистре edx

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

Регистр edx получает остаток от операции деления

6. Для чего используется инструкция “inc edx”?

Команда “inc edx” прибавляет единицу к содержимому регистра edx, что необходимо для корректного расчета варианта

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

Команда “mov eax, edx” переносит результат вычислений в регистр eax

Команда “call iprintLF” запускает процедуру, которая выводит результат на экран с новой строки

2.2 Самостоятельное задание

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 14 -

$$(x/2 + 8) * 3$$

для

$$x_1 = 1, x_2 = 4$$

(рис. [2.17]) (рис. [2.18])

Рис. 2.17: Программа в файле calc.asm

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

Рис. 2.18: Запуск программы calc.asm

3 Выводы

Изучили работу с арифметическими операциями.

Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.

13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).