

Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютера

Захаров Кирилл Юрьевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Символьные и численные данные в NASM	6
2.2	Выполнение арифметических операций в NASM	13
2.2.1	Ответы на вопросы	19
2.3	Задание для самостоятельной работы	20
3	Выводы	23

Список иллюстраций

2.1	Создание каталога и файлов	6
2.2	Изменение программы lab6-1.asm	7
2.3	Компиляция и запуск исполняемого файла lab6-1.asm	8
2.4	Изменение программы lab6-1.asm	9
2.5	Компиляция и запуск исполняемого файла lab6-1.asm	10
2.6	Изменение программы lab6-2.asm	11
2.7	Компиляция и запуск исполняемого файла lab6-2.asm	11
2.8	Изменение программы lab6-2.asm	12
2.9	Компиляция и запуск исполняемого файла lab6-2.asm	13
2.10	Компиляция и запуск исполняемого файла lab6-2.asm	13
2.11	Изменение программы lab6-3.asm	14
2.12	Компиляция и запуск исполняемого файла lab6-3.asm	15
2.13	Изменение программы lab6-3.asm	16
2.14	Компиляция и запуск исполняемого файла lab6-3.asm	17
2.15	Изменение программы variant.asm	18
2.16	Компиляция и запуск исполняемого файла variant.asm	18
2.17	Изменение программы prog.asm	21
2.18	Компиляция и запуск исполняемого файла prog.asm	22

Список таблиц

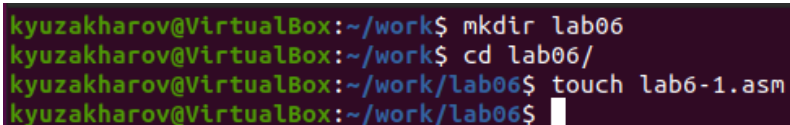
1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

2.1 Символьные и численные данные в NASM

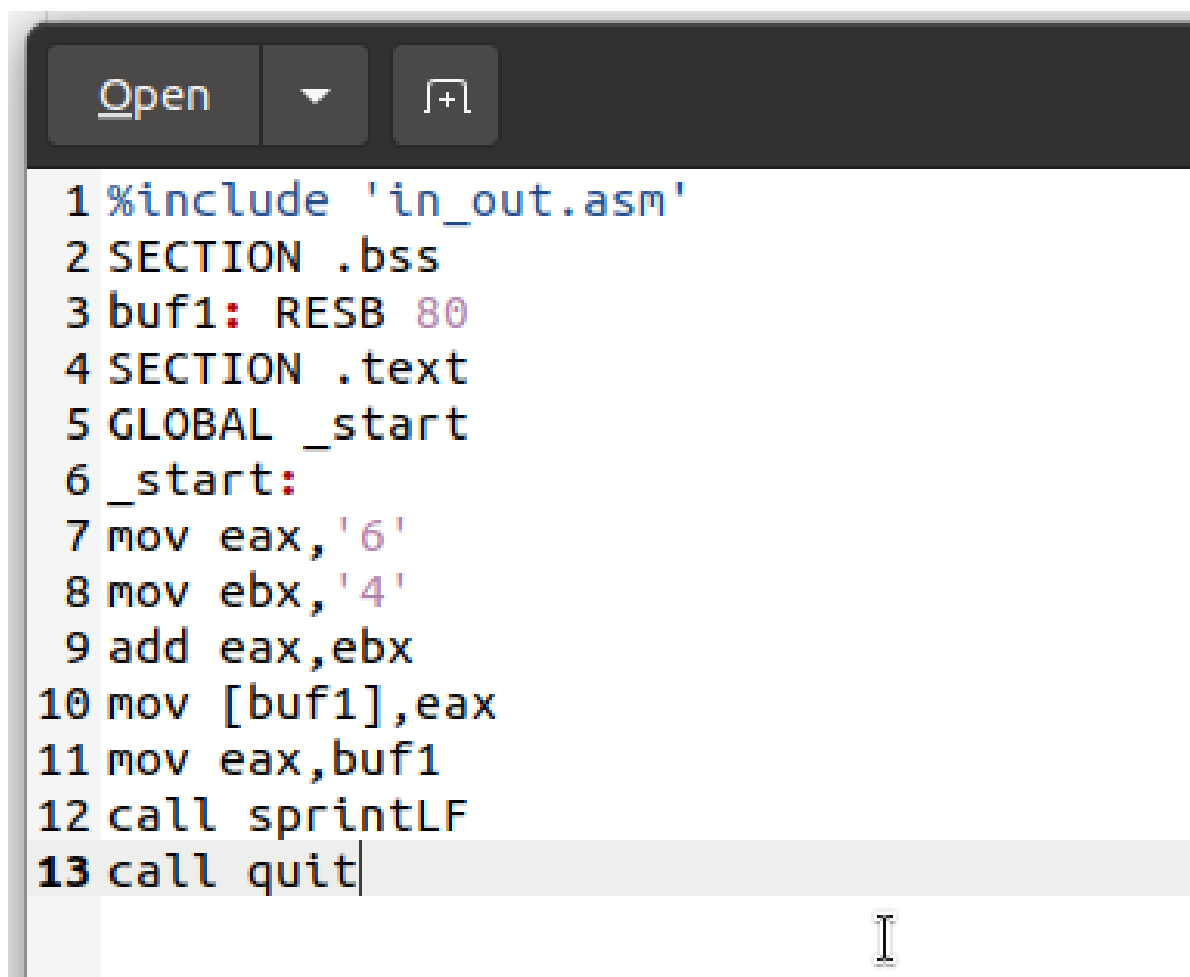
Я создал папку для хранения файлов лабораторной работы номер шесть, перешел в нее и сформировал файл с названием lab6-1.asm. (рис. [2.1])



```
kyuzakharov@VirtualBox:~/work$ mkdir lab06
kyuzakharov@VirtualBox:~/work$ cd lab06/
kyuzakharov@VirtualBox:~/work/lab06$ touch lab6-1.asm
kyuzakharov@VirtualBox:~/work/lab06$
```

Рис. 2.1: Создание каталога и файлов

Давайте изучим примеры программ, которые выполняют вывод символов и числовых данных. Эти программы будут отображать данные, которые хранятся в регистре eax.



```
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. 2.2: Изменение программы lab6-1.asm

В одной из таких программ в регистр `eax` помещается символ '6' (используя команду `mov eax, '6'`), а в регистр `ebx` помещается символ '4' (`mov ebx, '4'`). Затем мы складываем значения регистров `eax` и `ebx` (выполняем команду `add eax, ebx`, и результат сложения сохраняется в регистре `eax`). После этого мы производим вывод результата. (рис. [2.2]) (рис. [2.3])

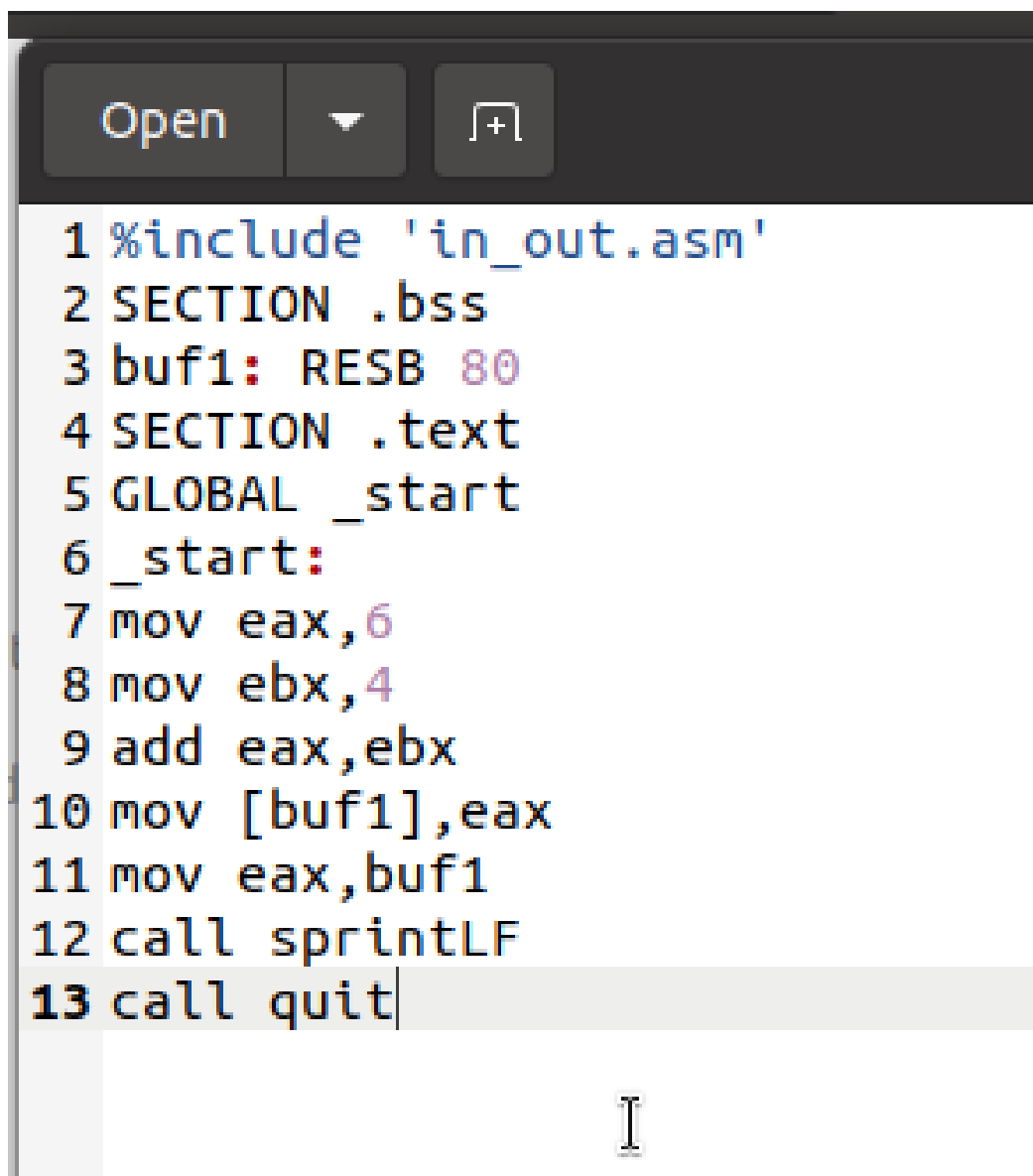
```
kyuzakharov@VirtualBox:~/work/lab06$  
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf lab6-1.asm  
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1  
kyuzakharov@VirtualBox:~/work/lab06$ ./lab6-1  
j  
kyuzakharov@VirtualBox:~/work/lab06$
```

Рис. 2.3: Компиляция и запуск исполняемого файла lab6-1.asm

Так как функция `sprintLF` требует, чтобы в регистре `eax` находился адрес, нам нужно ввести вспомогательную переменную. Мы передадим значение из регистра `eax` в переменную `buf1` (с помощью команды `mov [buf1], eax`), после чего поместим адрес переменной `buf1` обратно в регистр `eax` (команда `mov eax, buf1`) и вызовем функцию `sprintLF`.

При выводе содержимого регистра `eax` мы ожидаем получить значение 10. Тем не менее, на экране появится символ 'j'. Это объясняется тем, что двоичный код для символа '6' составляет 00110110, что в десятичной системе соответствует числу 54, а для символа '4' двоичный код - 00110100, или 52 в десятичном исчислении. Когда выполняется команда `add eax, ebx`, в регистр `eax` записывается их сумма, равная 01101010 в двоичном формате, или 106 в десятичном, что соответствует коду символа 'j'.

Затем я изменил текст программы, заменив символы на числа для записи в регистры. (рис. [2.4]) (рис. [2.5])



```
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
```

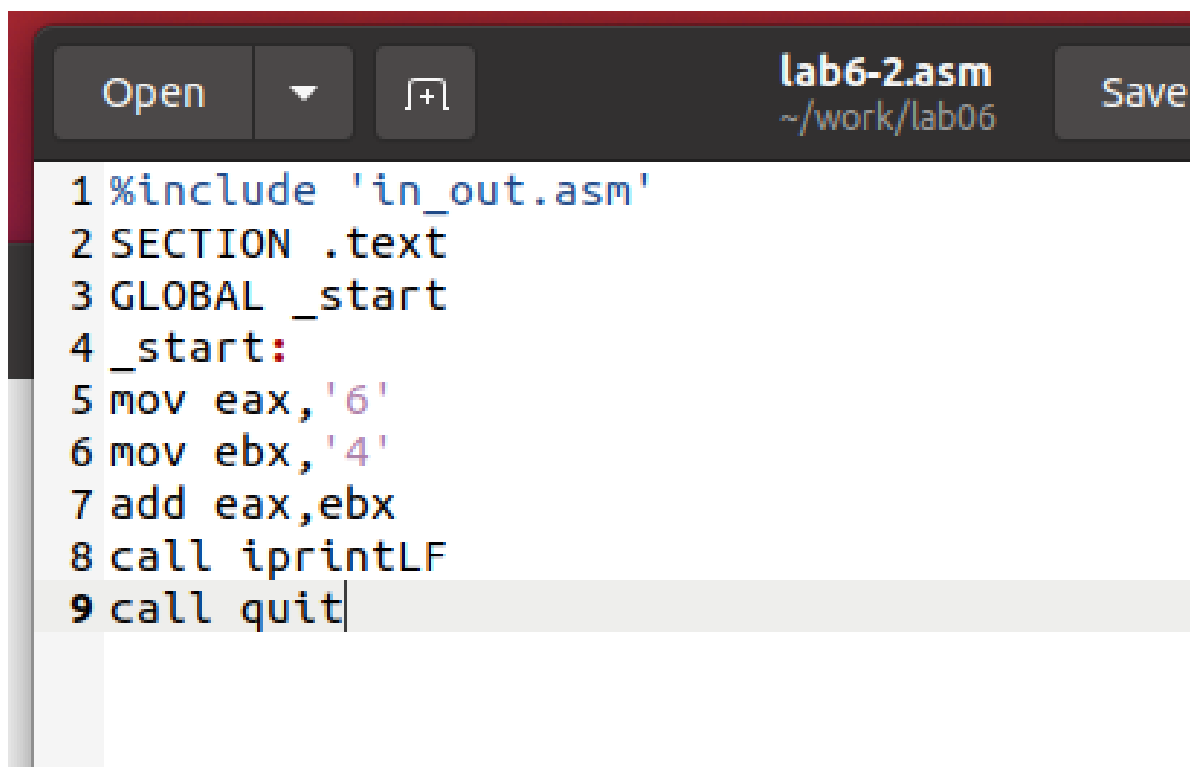
Рис. 2.4: Изменение программы lab6-1.asm

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

Рис. 2.5: Компиляция и запуск исполняемого файла lab6-1.asm

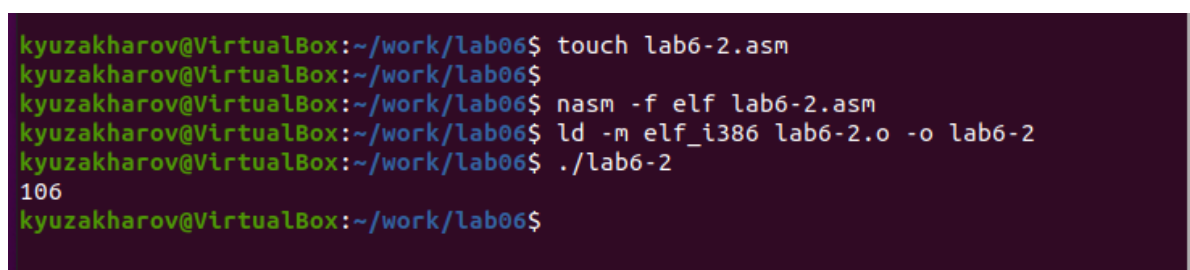
Как и в прошлый раз, при выполнении программы мы не увидим цифру 10. В этом случае на экран выводится символ с кодом 10, который является символом новой строки (или возвратом каретки). В консоли он не виден, но добавляет пустую строку.

Как уже упоминалось, в файле in_out.asm имеются вспомогательные функции для конвертации символов ASCII в числа и наоборот. Используя эти функции, я модифицировал текст программы. (рис. [2.6]) (рис. [2.7])



```
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: Изменение программы lab6-2.asm

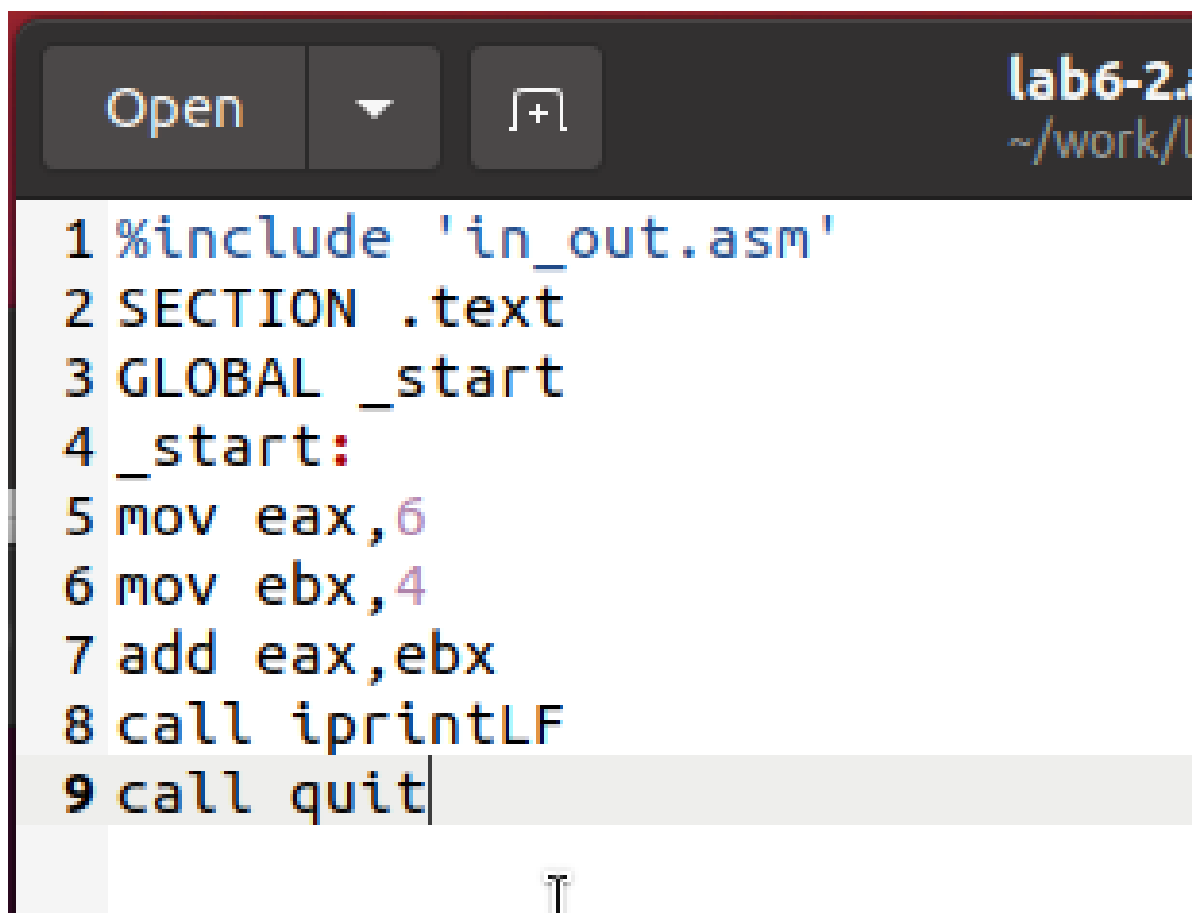


```
kyuzakharov@VirtualBox:~/work/lab06$ touch lab6-2.asm
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
kyuzakharov@VirtualBox:~/work/lab06$ ./lab6-2
106
kyuzakharov@VirtualBox:~/work/lab06$
```

Рис. 2.7: Компиляция и запуск исполняемого файла lab6-2.asm

Теперь программа выдаст число 106. В этом случае, так же как и в первом примере, команда `add` суммирует коды символов '6' и '4' ($54+52=106$). Но в отличие от предыдущей программы, функция `iprintLF` позволяет отобразить именно число, а не символ с соответствующим числовым кодом.

Таким же образом, как и в предыдущем примере, заменяем 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: Изменение программы lab6-2.asm

Функция `iprintLF` дает возможность вывода числа, и поскольку в качестве операндов использовались числа (не коды символов), результатом является число 10. (рис. [2.9])

```

kyuzakharov@VirtualBox:~/work/lab06$ touch lab6-2.asm
kyuzakharov@VirtualBox:~/work/lab06$
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
kyuzakharov@VirtualBox:~/work/lab06$ ./lab6-2
106
kyuzakharov@VirtualBox:~/work/lab06$
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
kyuzakharov@VirtualBox:~/work/lab06$ ./lab6-2
10
kyuzakharov@VirtualBox:~/work/lab06$

```

Рис. 2.9: Компиляция и запуск исполняемого файла lab6-2.asm

Затем я заменил функцию `iprintLF` на `iprint`, создал исполняемый файл и запустил его. Результат отличается отсутствием переноса строки после вывода. (рис. [2.10])

```

kyuzakharov@VirtualBox:~/work/lab06$ touch lab6-2.asm
kyuzakharov@VirtualBox:~/work/lab06$
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
kyuzakharov@VirtualBox:~/work/lab06$ ./lab6-2
106
kyuzakharov@VirtualBox:~/work/lab06$
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
kyuzakharov@VirtualBox:~/work/lab06$ ./lab6-2
10
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf lab6-2.asm
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
kyuzakharov@VirtualBox:~/work/lab06$ ./lab6-2
10kyuzakharov@VirtualBox:~/work/lab06$
kyuzakharov@VirtualBox:~/work/lab06$

```

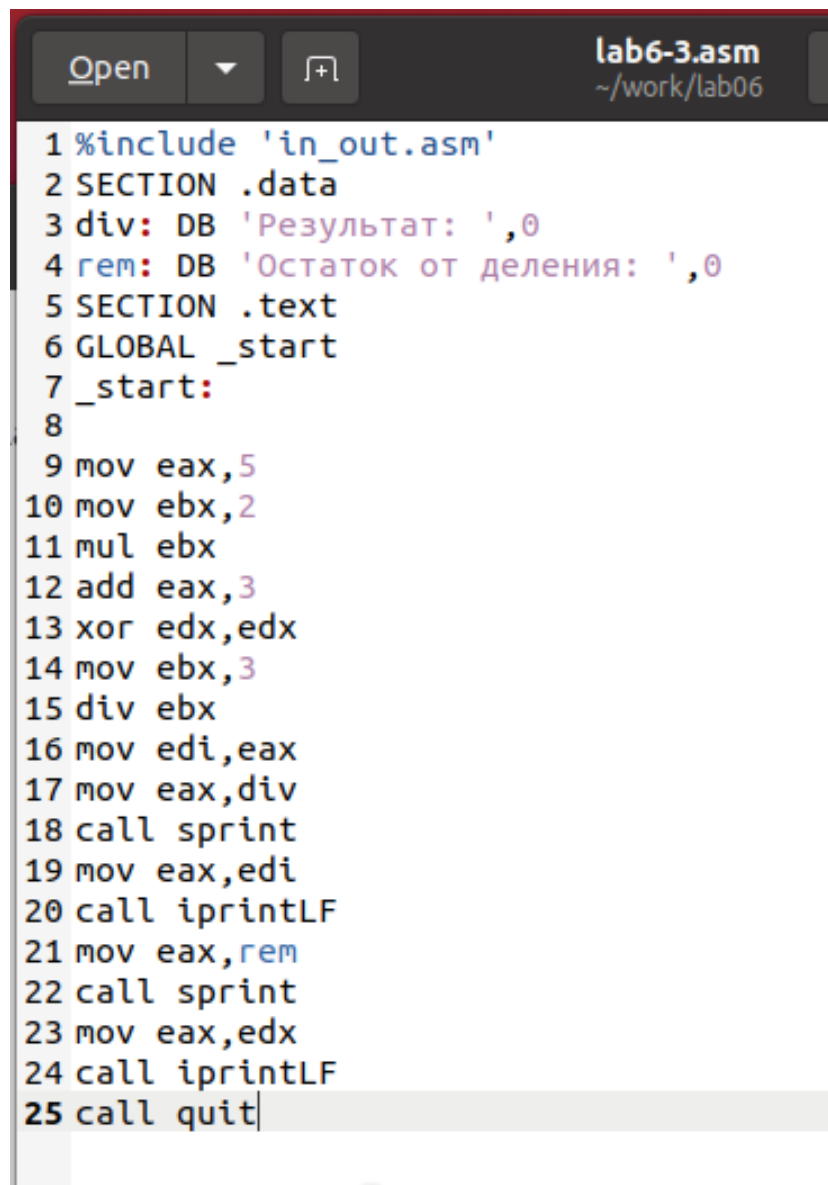
Рис. 2.10: Компиляция и запуск исполняемого файла lab6-2.asm

2.2 Выполнение арифметических операций в NASM

Давайте рассмотрим пример выполнения арифметических действий в NASM на основе программы, которая рассчитывает значение функции

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

. (рис. [2.11]) (рис. [2.12])



```
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
```

Рис. 2.11: Изменение программы lab6-3.asm

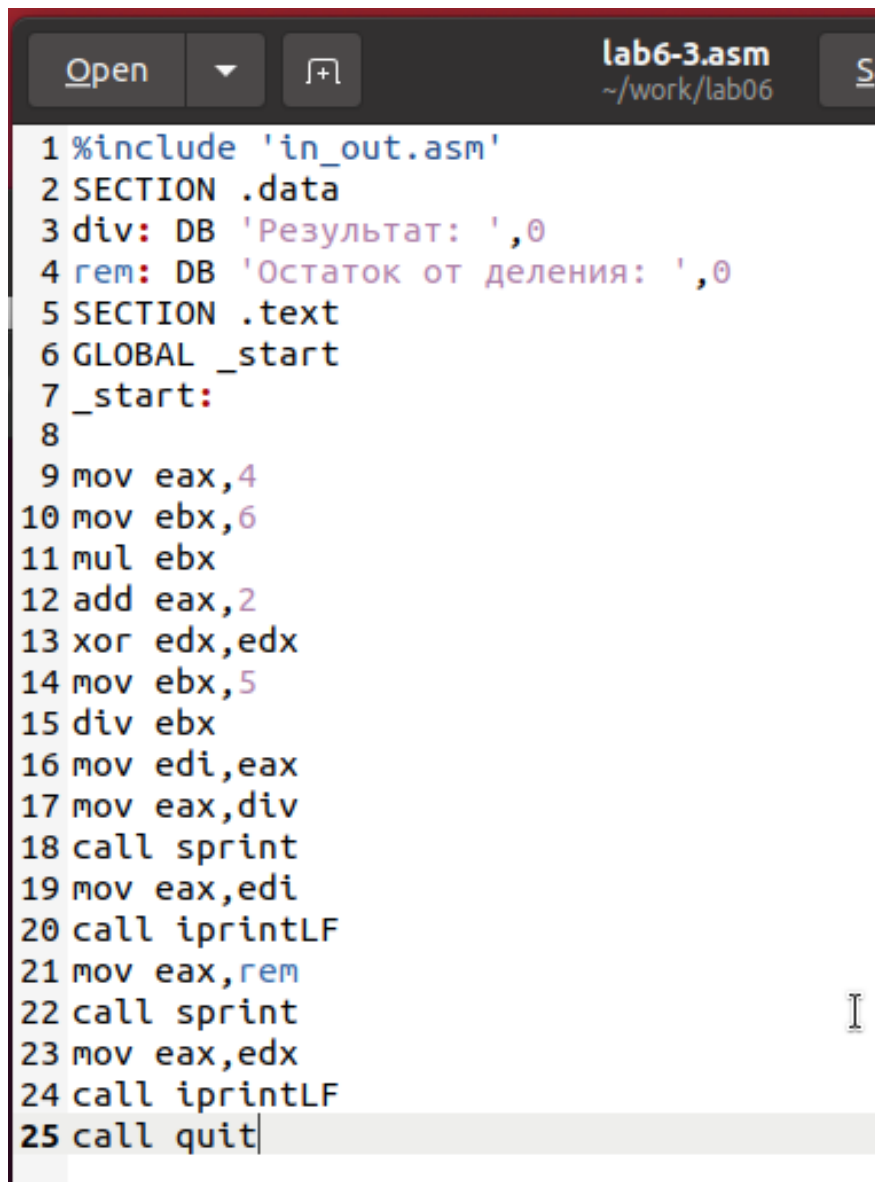
```
kyuzakharov@VirtualBox:~/work/lab06$  
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf lab6-3.asm  
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3  
kyuzakharov@VirtualBox:~/work/lab06$ ./lab6-3  
Результат: 4  
Остаток от деления: 1  
kyuzakharov@VirtualBox:~/work/lab06$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

Рис. 2.12: Компиляция и запуск исполняемого файла lab6-3.asm

Я изменил код программы таким образом, чтобы она теперь вычисляла новое выражение

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

, после чего собрал исполняемый файл и проверил его функционирование. (рис. [2.13]) (рис. [2.14])



```
lab6-3.asm
~/work/lab06

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
```

Рис. 2.13: Изменение программы lab6-3.asm


```

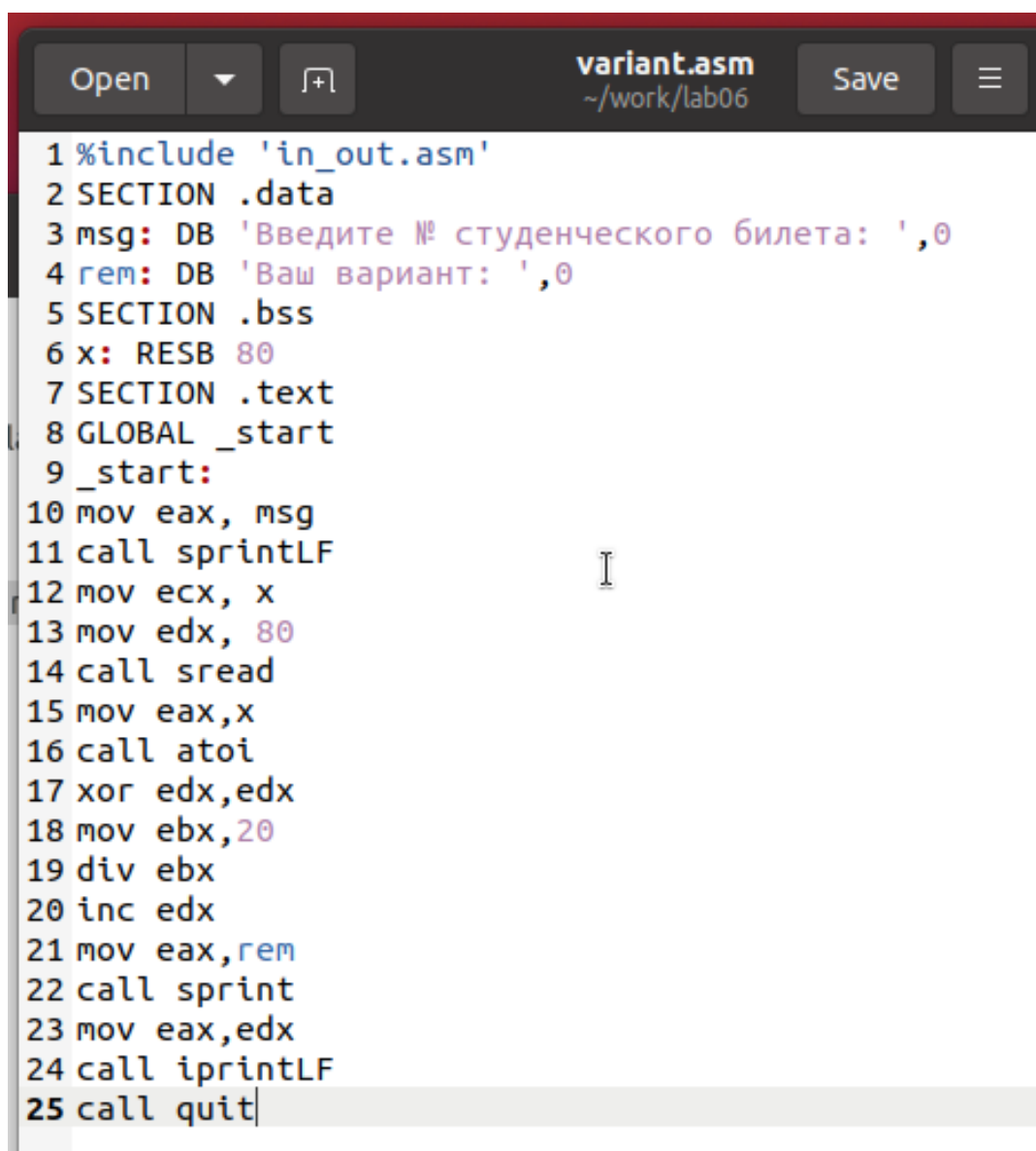
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf lab6-3.asm
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
kyuzakharov@VirtualBox:~/work/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf lab6-3.asm
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
kyuzakharov@VirtualBox:~/work/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
kyuzakharov@VirtualBox:~/work/lab06$

```

Рис. 2.14: Компиляция и запуск исполняемого файла lab6-3.asm

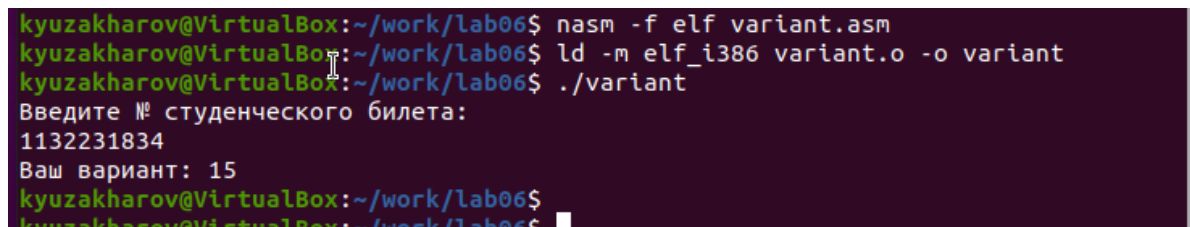
Дополнительный пример демонстрирует программу для расчета индивидуального задания, основанного на номере студенческого билета. (рис. [2.15]) (рис. [2.16])

В этой ситуации число для операций вводится через клавиатуру. Напомним, что ввод осуществляется в текстовом формате, и для того чтобы арифметические операции в NASM работали правильно, текстовые символы необходимо конвертировать в числовые значения. В этом может помочь функция `atoi` из файла `in_out.asm`.



```
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
16 call atoi
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
```

Рис. 2.15: Изменение программы variant.asm



```
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf variant.asm
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 variant.o -o variant
kyuzakharov@VirtualBox:~/work/lab06$ ./variant
Введите № студенческого билета:
1132231834
Ваш вариант: 15
kyuzakharov@VirtualBox:~/work/lab06$
```

Рис. 2.16: Компиляция и запуск исполняемого файла variant.asm

2.2.1 Ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения ‘Ваш вариант:’?
 - Команда `mov eax, tem` загружает в регистр значение, содержащее фразу “Ваш вариант:”.
 - Используя `call sprint`, происходит вызов функции, которая отображает строку на экране.
2. Для чего используются следующие инструкции?
 - `mov ecx, x` помещает значение в регистр `ecx`.
 - `mov edx, 80` устанавливает в регистр `edx` значение 80.
 - `call sread` активирует функцию чтения данных, которая запрашивает ввод студенческого билета и сохраняет его в переменную `X`.
3. Для чего используется инструкция “`call atoi`”?
 - Эта функция конвертирует строку символов в их численное представление.
4. Какие строки листинга отвечают за вычисления варианта?
 - `xor edx, edx` обнуляет регистр `edx`.
 - `mov ebx, 20` устанавливает в регистр `ebx` число 20.
 - `div ebx` выполняет деление, используя значение в `ebx`.
 - `inc edx` увеличивает значение в регистре `edx` на единицу, что соответствует вычислению варианта.
5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”?

- Остаток от деления помещается в регистр `edx`.

6. Для чего используется инструкция `inc edx`?

- Команда увеличивает значение в регистре `edx` на один, что необходимо по алгоритму вычисления варианта.

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

- `mov eax, edx` переносит результат вычислений в регистр `eax`.
- `call iprintLF` вызывает функцию вывода числа на экран с переводом строки.

2.3 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. (рис. [2.17]) (рис. [2.18]) Создайте исполняемый файл и проверьте его работу для значений x_1 и x_2 из 6.3.

Получили вариант 15 -

$$(5 + x)^2 - 3$$

для

$$x = 5, x = 1$$

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите X ',0
4 rem: DB 'выражение = : ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintLF
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax,x
16 call atoi
17 add eax,5
18 mov ebx,eax
19 mul ebx
20 sub eax,3
21 mov ebx,eax
22 mov eax,rem
23 call sprint
24 mov eax,ebx
25 call iprintLF
26 call quit
```

Рис. 2.17: Изменение программы prog.asm

```
kyuzakharov@VirtualBox:~/work/lab06$ touch prog.asm
kyuzakharov@VirtualBox:~/work/lab06$ nasm -f elf prog.asm
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 prog.o -o prog.
kyuzakharov@VirtualBox:~/work/lab06$ ld -m elf_i386 prog.o -o prog
kyuzakharov@VirtualBox:~/work/lab06$ ./prog
Введите X
5
выражение = : 97
kyuzakharov@VirtualBox:~/work/lab06$ ./prog
Введите X
1
выражение = : 33
kyuzakharov@VirtualBox:~/work/lab06$
```

Рис. 2.18: Компиляция и запуск исполняемого файла prog.asm

3 Выводы

Изучили работу с арифметическими операциями.