

Отчёт по лабораторной работе 6

Архитектура компьютеров

Иван Горбунов

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
3 Выводы	20

Список иллюстраций

2.1 Программа в файле lab6-1.asm	7
2.2 Запуск программы lab6-1.asm	7
2.3 Программа в файле lab6-1.asm	8
2.4 Запуск программы lab6-1.asm	9
2.5 Программа в файле lab6-2.asm	10
2.6 Запуск программы lab6-2.asm	10
2.7 Программа в файле lab6-2.asm	11
2.8 Запуск программы lab6-2.asm	11
2.9 Запуск программы lab6-2.asm	12
2.10 Программа в файле lab6-3.asm	13
2.11 Запуск программы lab6-3.asm	13
2.12 Программа в файле lab6-3.asm	14
2.13 Запуск программы lab6-3.asm	14
2.14 Программа в файле variant.asm	15
2.15 Запуск программы variant.asm	16
2.16 Программа в файле work.asm	18
2.17 Запуск программы work.asm	19

Список таблиц

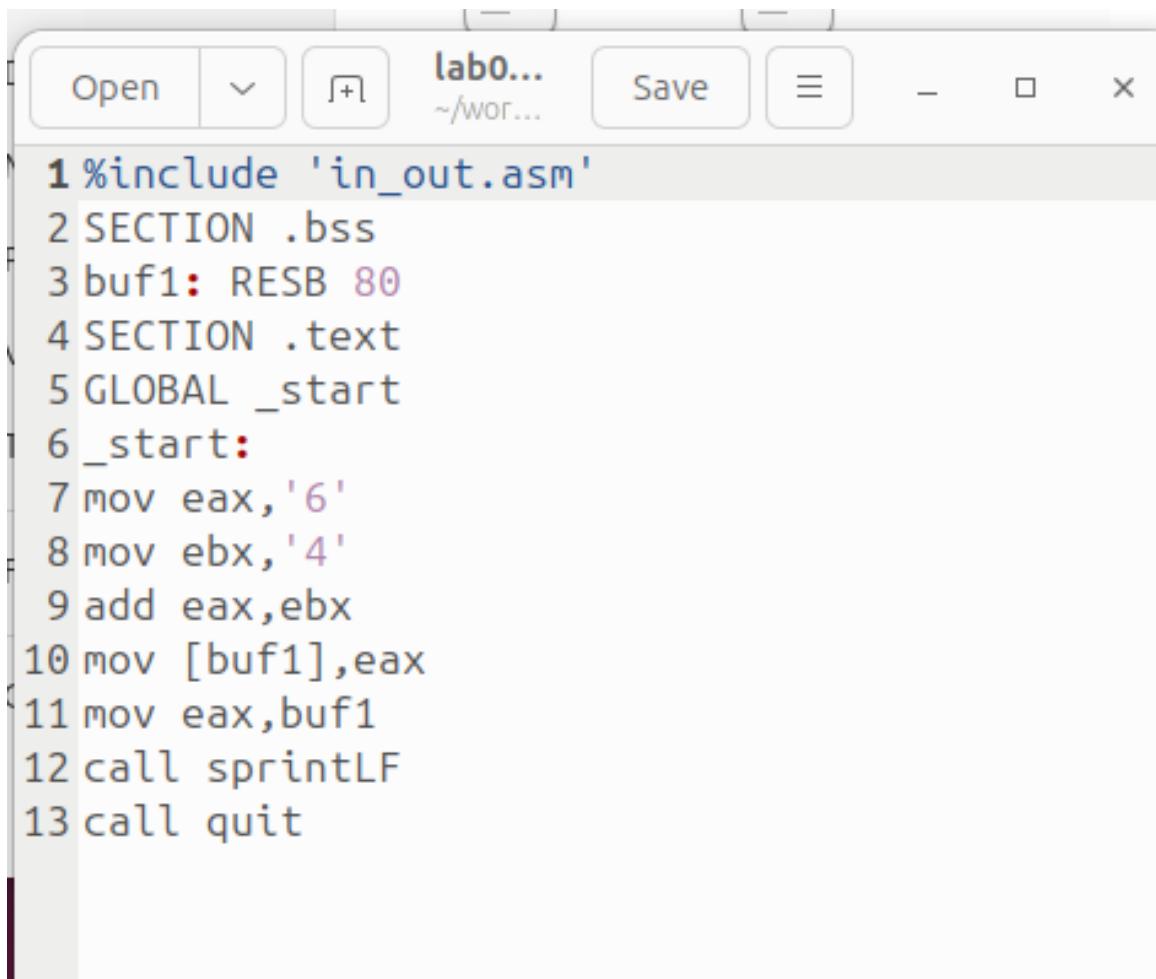
1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

1. Создал каталог для программам 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 eax.

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



The screenshot shows a text editor window with the following assembly code:

```
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 sprintLF
13 call quit
```

Рисунок 2.1: Программа в файле lab6-1.asm

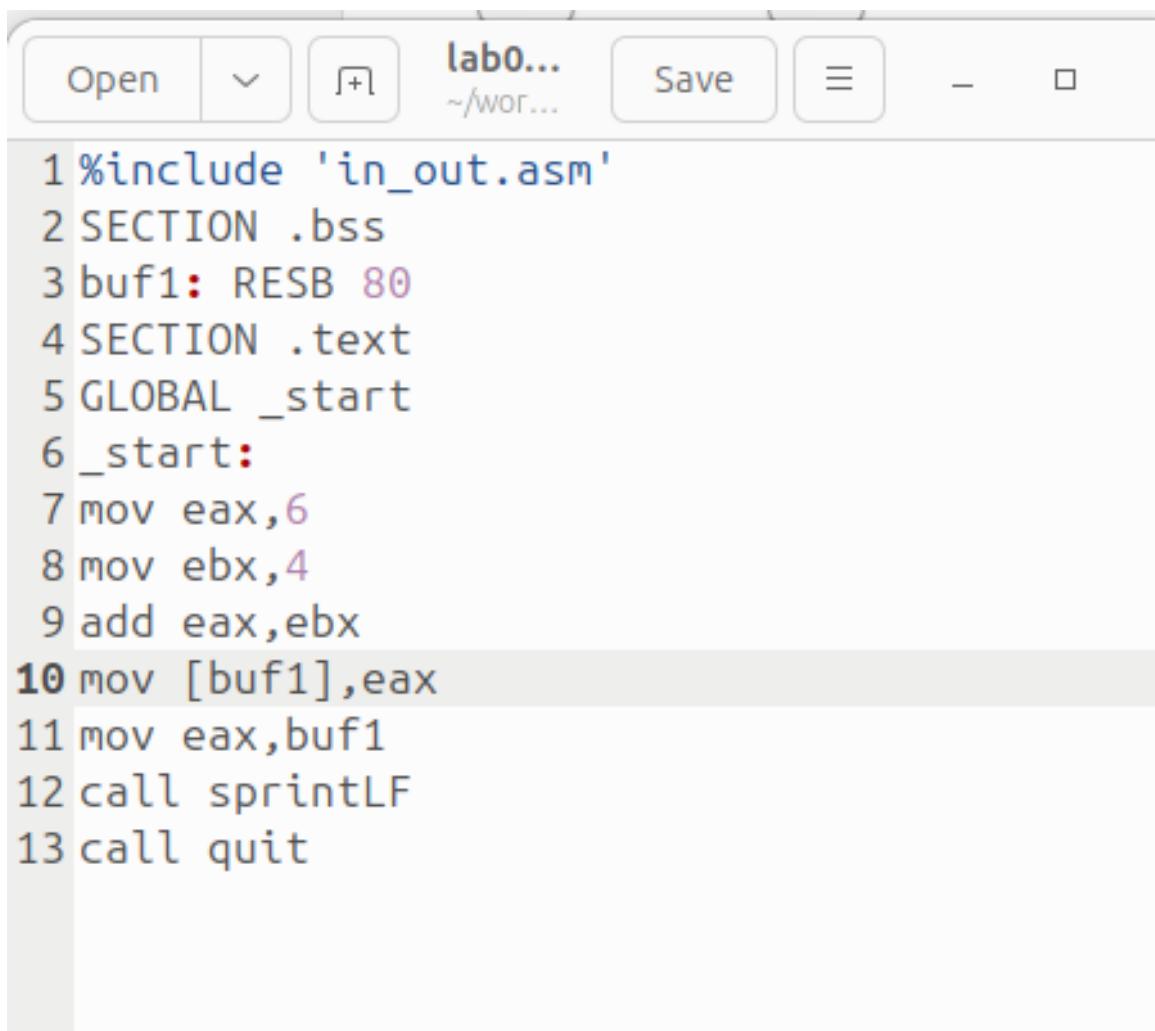
```
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ ./lab6-1
j
imgorbunov@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 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 запи-

шет 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.



The screenshot shows a text editor window with the following assembly code:

```
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 sprintLF
13 call quit
```

The file is titled "lab0..." and has tabs for Open, Save, and other file operations. The assembly code is numbered from 1 to 13.

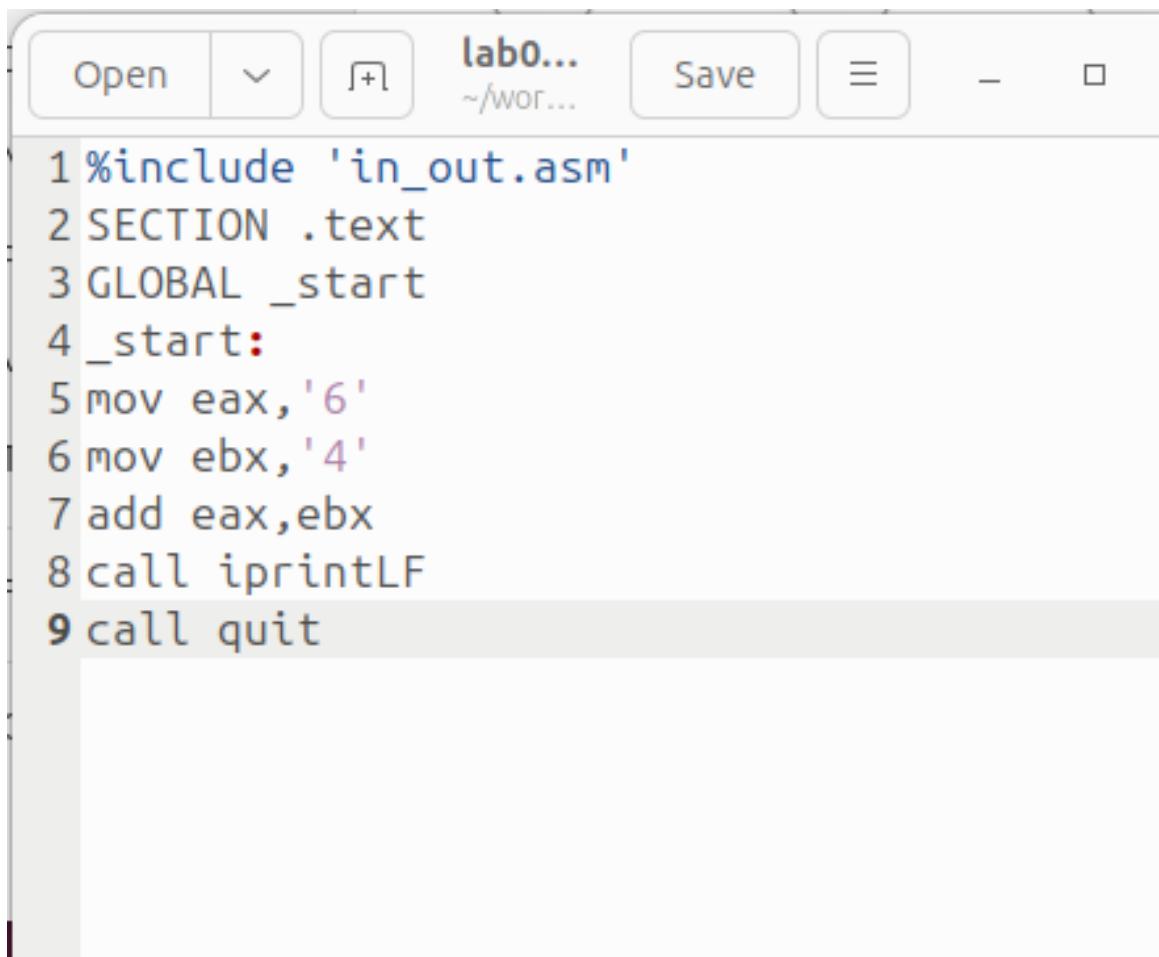
Рисунок 2.3: Программа в файле lab6-1.asm

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

Рисунок 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.

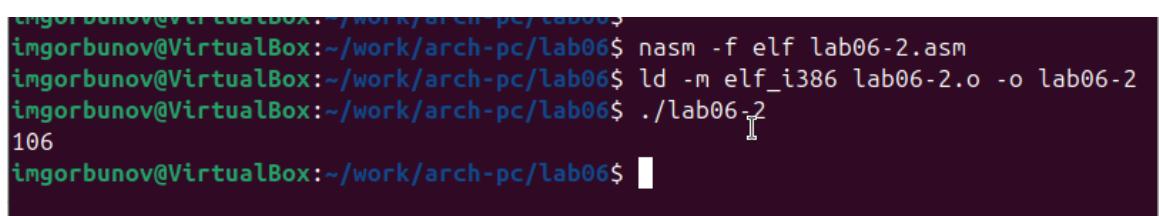
4. Как отмечалось выше, для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовал текст программы с использованием этих функций.



The screenshot shows a text editor window with the following assembly code:

```
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
```

Рисунок 2.5: Программа в файле lab6-2.asm

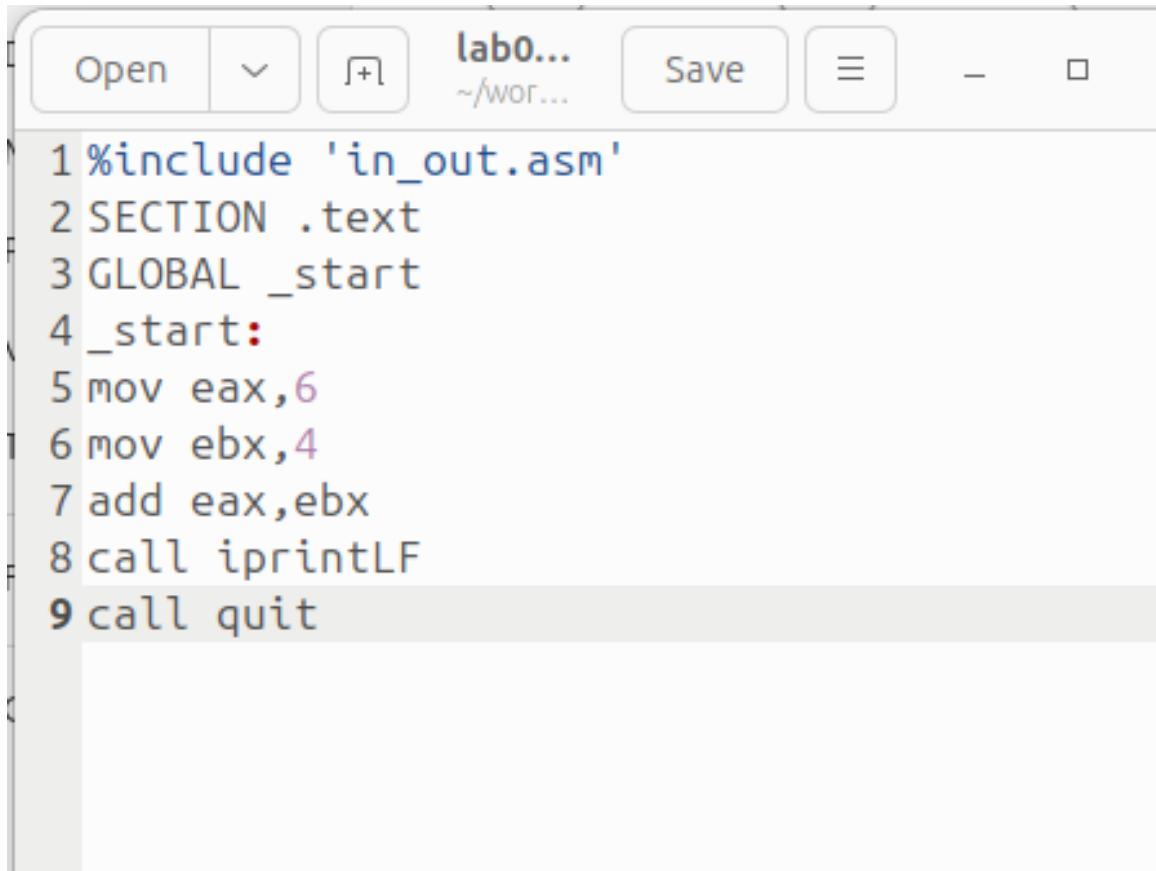


```
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ ./lab06-2
106
imgorbunov@VirtualBox:~/work/arch-pc/lab06$
```

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

5. Аналогично предыдущему примеру изменим символы на числа.

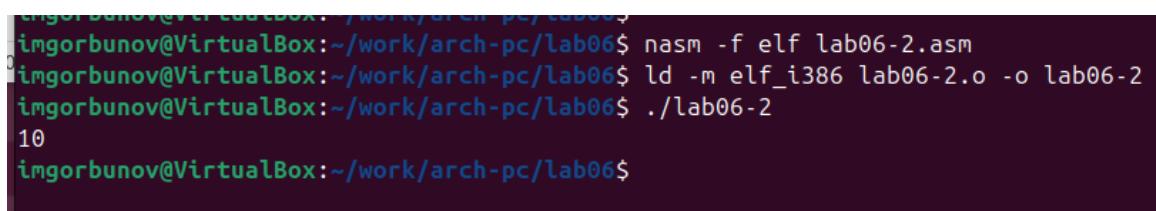


The screenshot shows a text editor window with the following assembly code:

```
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
```

Рисунок 2.7: Программа в файле lab6-2.asm

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



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

Рисунок 2.8: Запуск программы lab6-2.asm

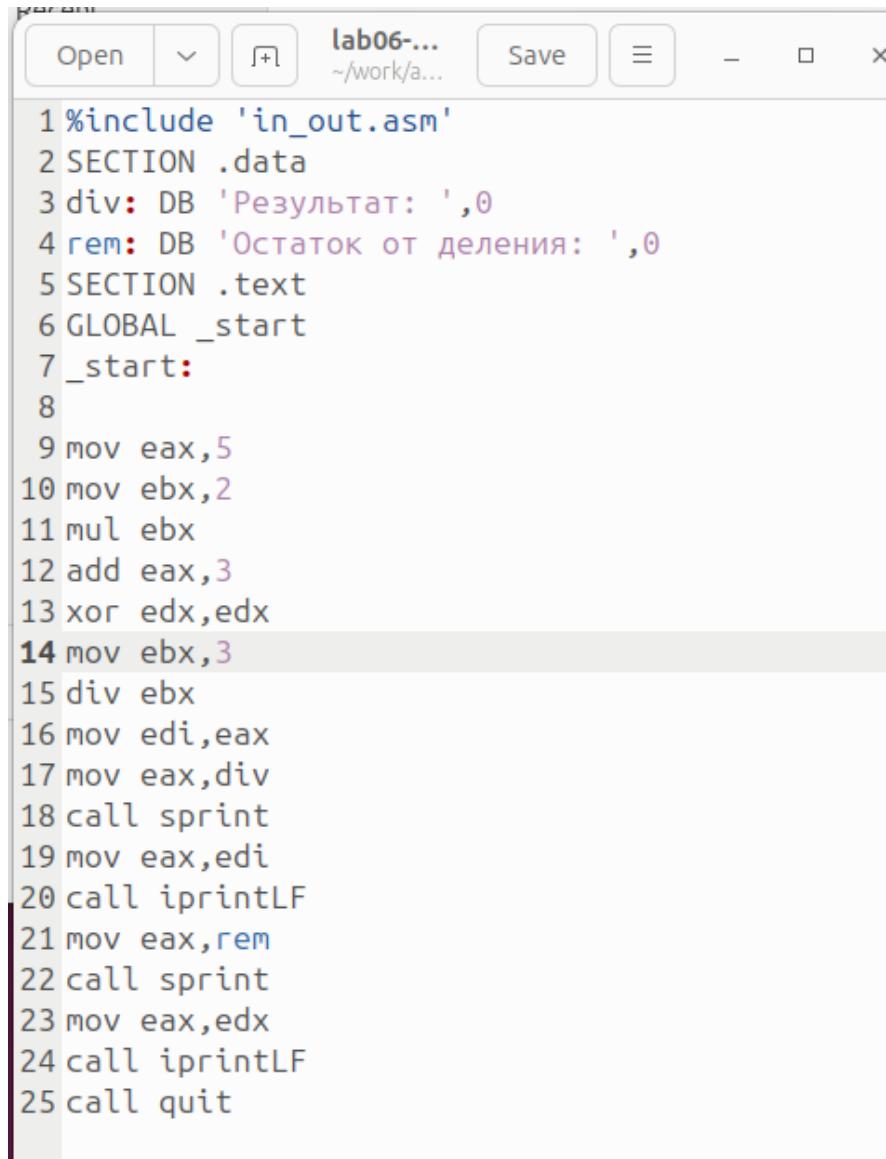
Заменил функцию iprintLF на iprint. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки.

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

Рисунок 2.9: Запуск программы lab6-2.asm

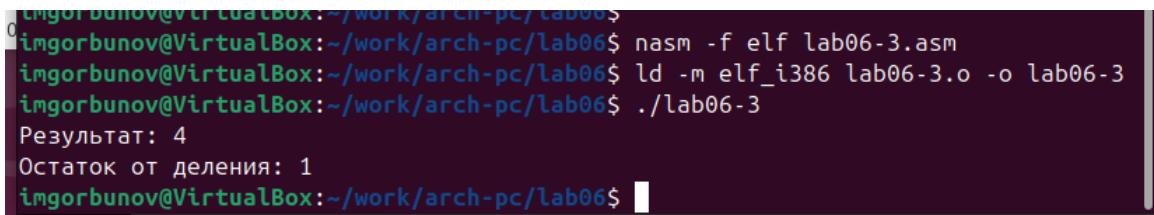
6. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения

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



```
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
```

Рисунок 2.10: Программа в файле lab6-3.asm

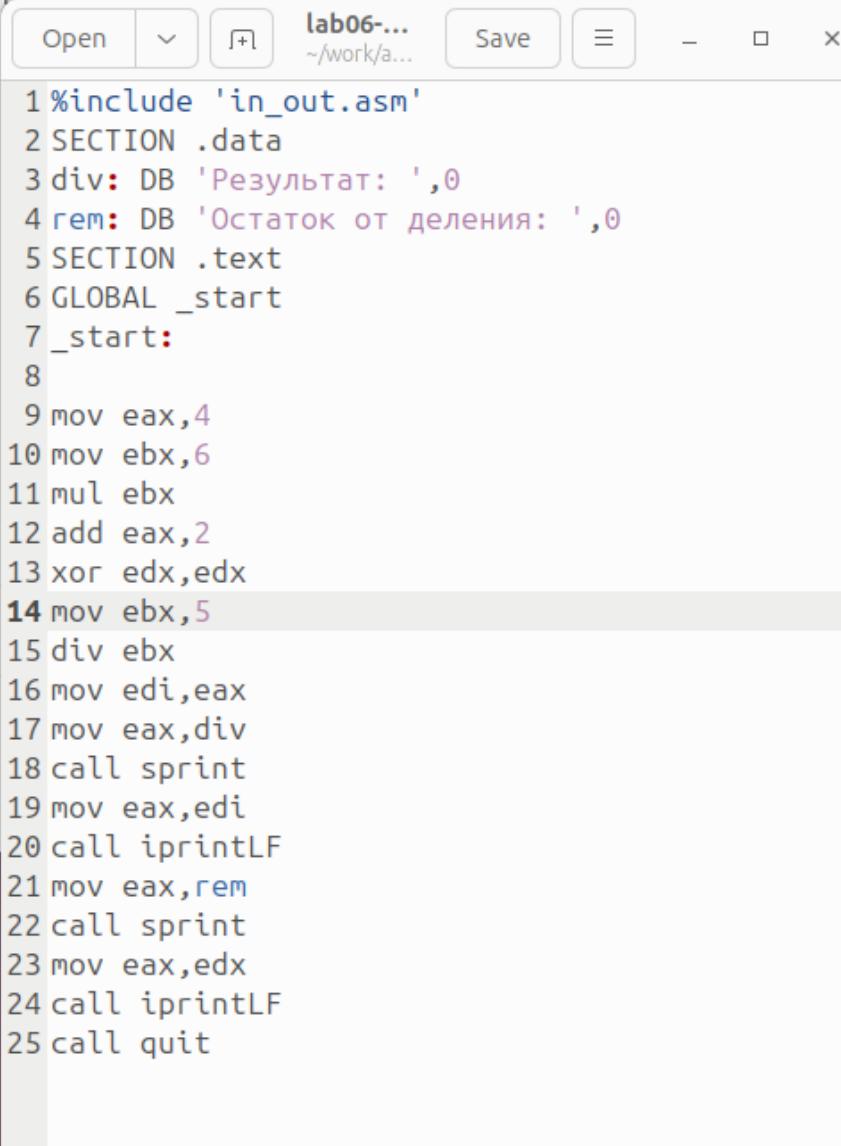


```
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ ./lab06-3
Результат: 4
Остаток от деления: 1
imgorbunov@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 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.



```
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
```

Рисунок 2.12: Программа в файле lab6-3.asm

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

Рисунок 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.



The screenshot shows a window titled 'variant' with the following assembly code:

```
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 sprintLF
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 sprint
23 mov eax,edx
24 call iprintLF
25 call quit
```

Рисунок 2.14: Программа в файле variant.asm

```
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
imgorbunov@VirtualBox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032254762
Ваш вариант: 3
imgorbunov@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 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 используется следующие инструкции?

```
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 iprintfLF – 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 3 -

$$(2 + x)^2$$

для

$$x_1 = 2, x_2 = 8$$

The screenshot shows a text editor window with the following interface elements:

- Top bar: "Recent", "Open" (dropdown), "New", "work.a...", "Save", "≡", "-", and a close button.
- Code area:

```
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,2
18 mov ebx,eax
19 mul ebx
20 mov ebx, eax
21 mov eax,rem
22 call sprint
23 mov eax,ebx
24 call iprintLF
25 call quit
26
```

Рисунок 2.16: Программа в файле work.asm

```
igorbunov@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf work.asm
igorbunov@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 work.o -o work
igorbunov@VirtualBox:~/work/arch-pc/lab06$ ./work
Введите X
2
выражение = : 16
igorbunov@VirtualBox:~/work/arch-pc/lab06$ ../work
Введите X
8
выражение = : 100
igorbunov@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 2.17: Запуск программы work.asm

3 Выводы

Изучили работу с арифметическими операциями.