

Отчёт по лабораторной работе 6

Архитектура компьютеров

Алхоев Абдулмалик-Салим Гапурович

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	11
2.3	Задание для самостоятельной работы	16
3	Выводы	19

Список иллюстраций

2.1	Код программы lab6-1.asm	7
2.2	Запуск программы lab6-1.asm	7
2.3	Код программы lab6-1.asm с числами	8
2.4	Запуск программы lab6-1.asm с числами	8
2.5	Код программы lab6-2.asm	9
2.6	Запуск программы lab6-2.asm	10
2.7	Код программы lab6-2.asm с числами	10
2.8	Запуск программы lab6-2.asm с числами	11
2.9	Запуск программы lab6-2.asm без переноса строки	11
2.10	Код программы lab6-3.asm	12
2.11	Запуск программы lab6-3.asm	12
2.12	Код программы lab6-3.asm с новым выражением	13
2.13	Запуск программы lab6-3.asm с новым выражением	13
2.14	Код программы variant.asm	14
2.15	Запуск программы variant.asm	15
2.16	Код программы task.asm	17
2.17	Запуск программы task.asm	18

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

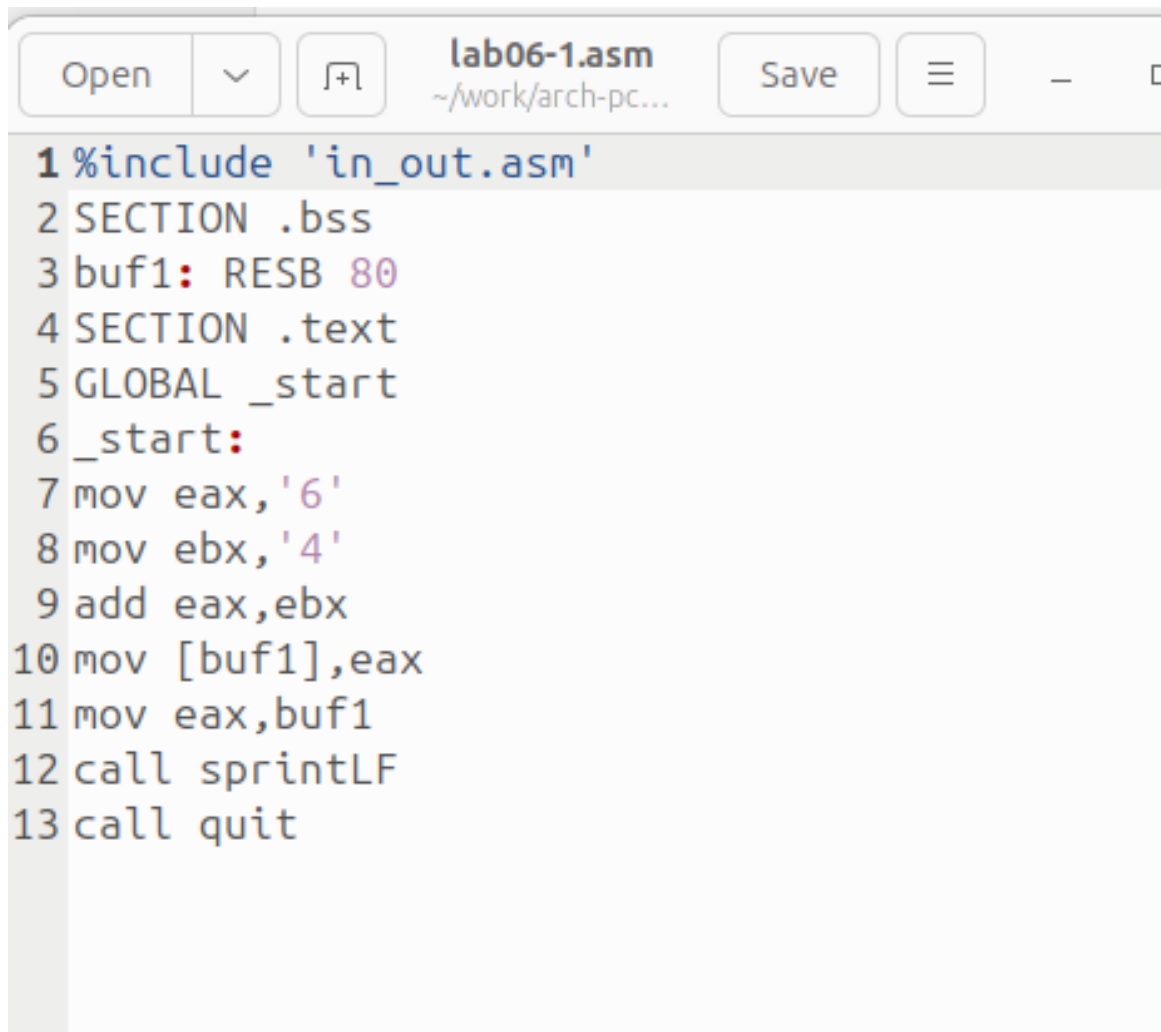
2 Выполнение лабораторной работы

2.1 Символьные и численные данные в NASM

Создаю папку для программ лабораторной работы № 6, перехожу в неё и создаю файл lab6-1.asm.

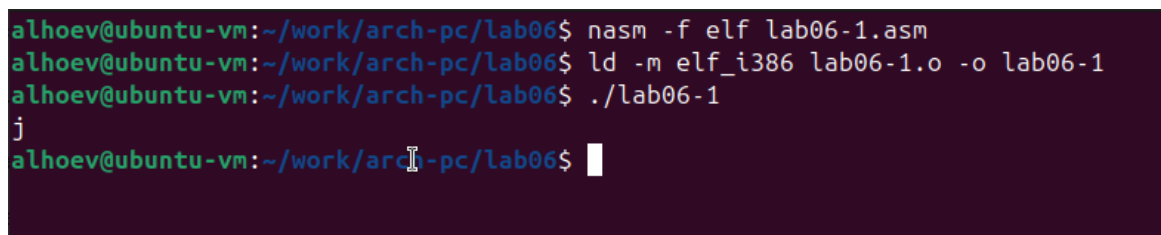
Рассмотрим примеры программ, которые выводят символы и числа. Программы будут выводить значения из регистра еах.

В этой программе в регистр еах записывается символ „6“ (инструкция `mov еах,„6“`), а в регистр ебх — символ „4“ (инструкция `mov ебх,„4“`). Затем к значению в еах добавляется значение из ебх (инструкция `add еах,ебх`), и результат записывается обратно в еах. После этого выводим результат.



```
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: Код программы lab6-1.asm



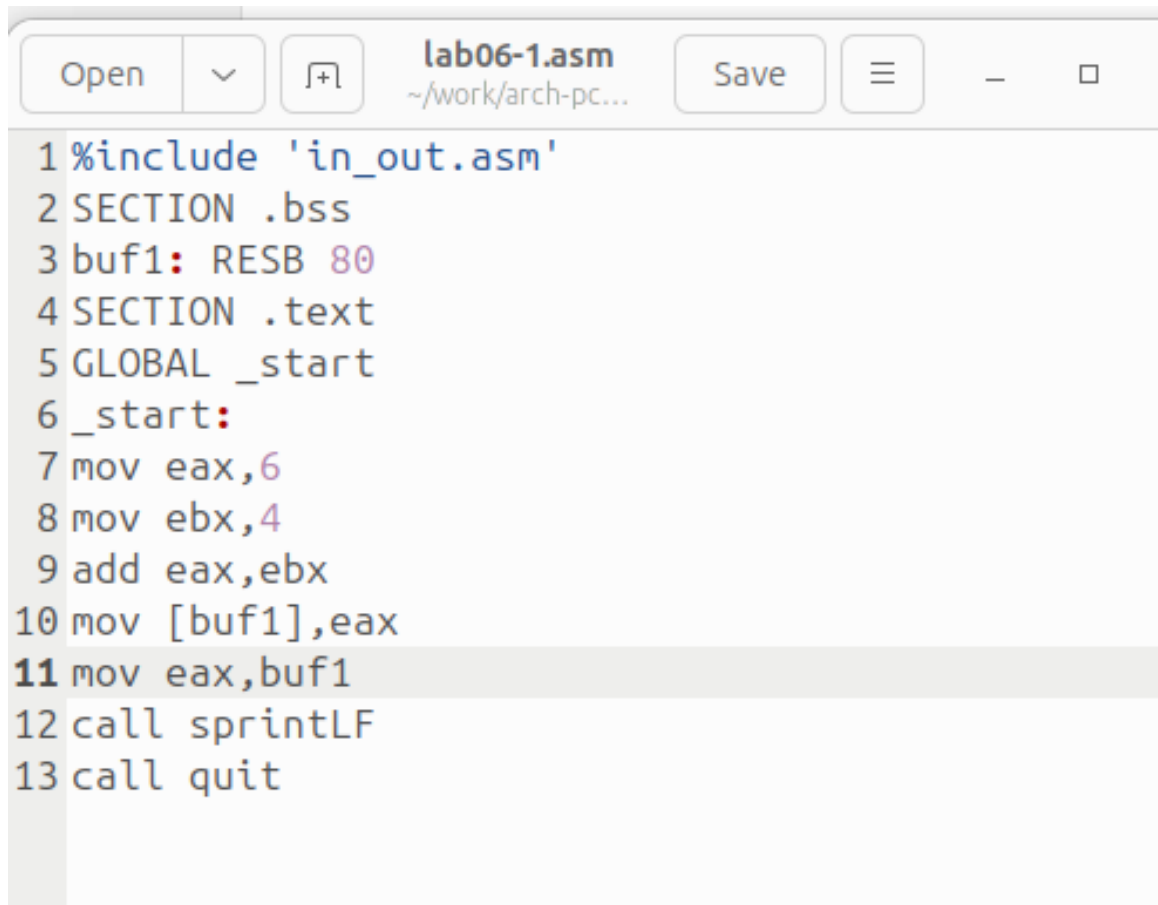
```
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ ./lab06-1
j
alhoev@ubuntu-vm:~/work/arch-pc/lab06$
```

Рисунок 2.2: Запуск программы lab6-1.asm

При выводе значения из `eax` ожидаем увидеть число 10. Однако вместо этого выводится символ „j“. Это связано с тем, что код символа „6“ в двоичном

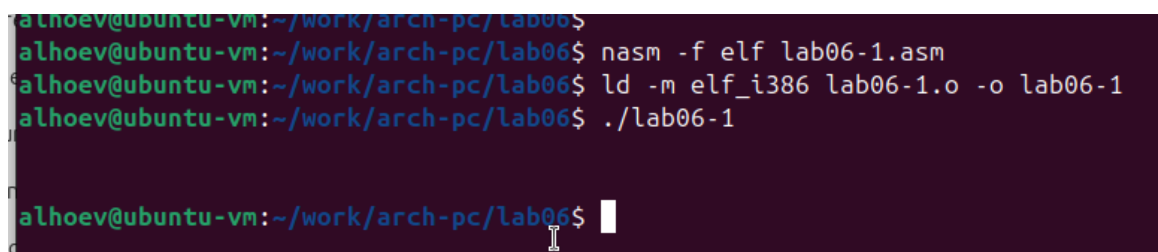
формате — 00110110 (54 в десятичной системе), а код символа „4“ — 00110100 (52). После сложения в еах получаем 01101010 (106), что соответствует символу „j“.

Теперь изменим программу и вместо символов запишем в регистры числа.



```
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
```

Рисунок 2.3: Код программы lab6-1.asm с числами



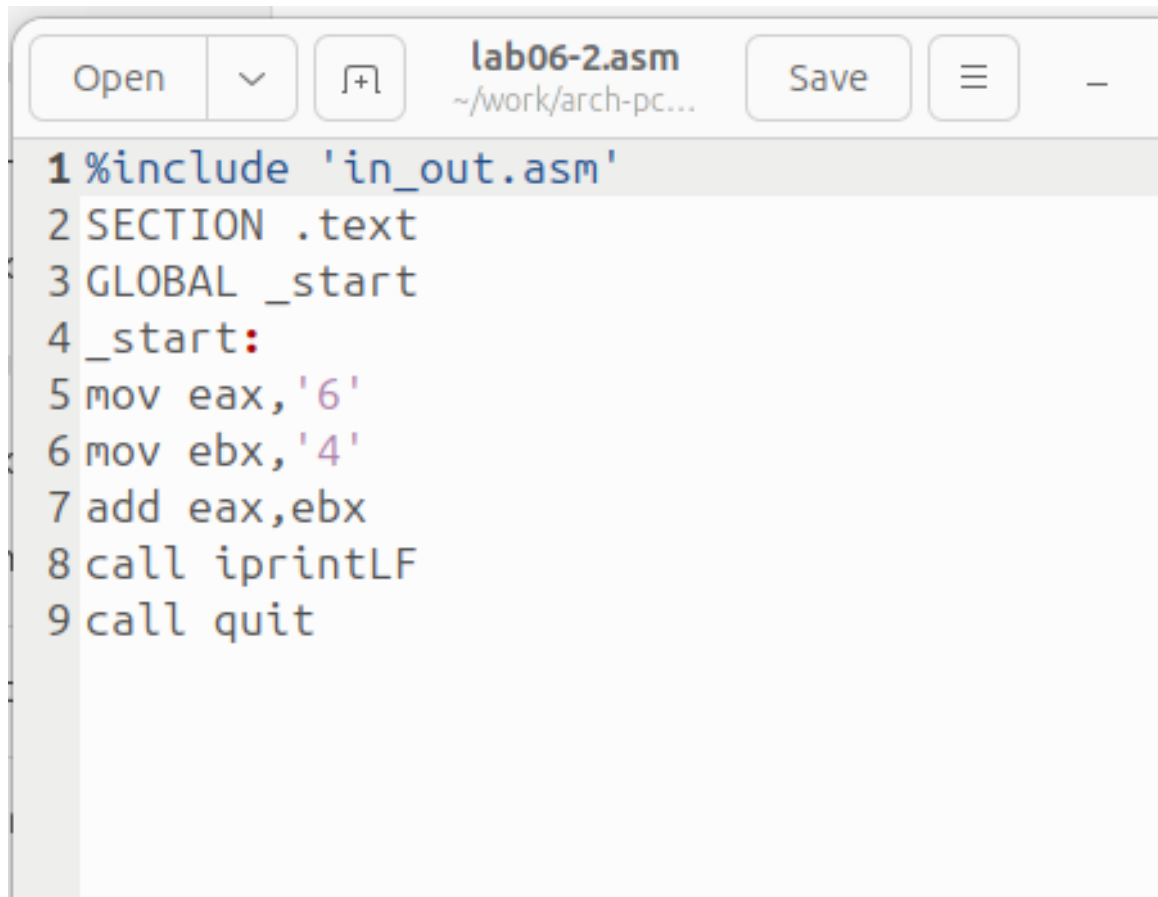
```
alhoev@ubuntu-vm:~/work/arch-pc/lab06$
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ ./lab06-1

alhoev@ubuntu-vm:~/work/arch-pc/lab06$
```

Рисунок 2.4: Запуск программы lab6-1.asm с числами

Как и в предыдущем случае, при выполнении программы не получаем число 10. В этот раз выводится символ с кодом 10, что означает конец строки (возврат каретки). В консоли он не отображается, но добавляет пустую строку.

Для работы с числами в файле `in_out.asm` есть подпрограммы, которые преобразуют символы ASCII в числа и обратно. Изменяем программу, используя эти функции.



```
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
```

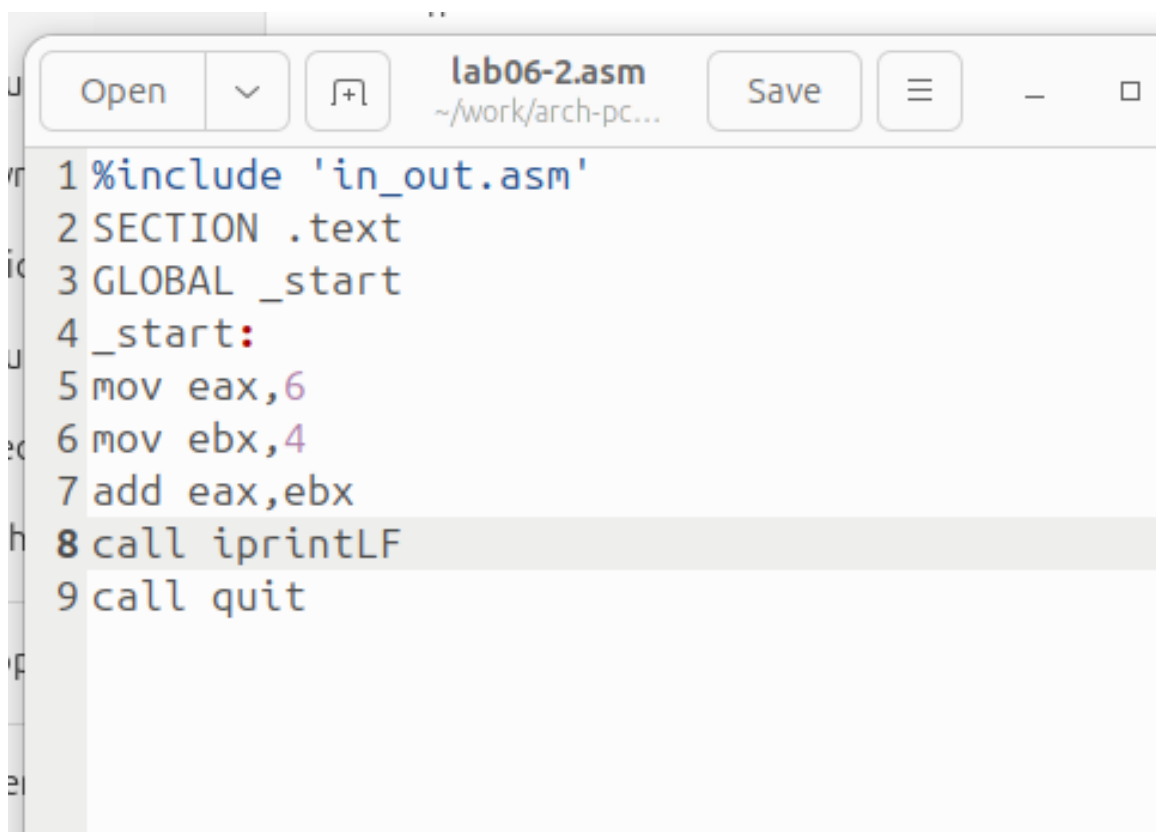
Рисунок 2.5: Код программы `lab6-2.asm`

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

Рисунок 2.6: Запуск программы lab6-2.asm

В результате выполнения программы мы получим число 106. Здесь команда `add` складывает коды символов „6“ и „4“ ($54 + 52 = 106$). Но, в отличие от предыдущей программы, функция `iprintLF` выводит число, а не соответствующий ему символ.

Теперь снова изменим символы на числа.



```
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: Код программы lab6-2.asm с числами

Функция `iprintLF` позволяет вывести число, так как операндами являются

числа. Поэтому получаем число 10.

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

Рисунок 2.8: Запуск программы lab6-2.asm с числами

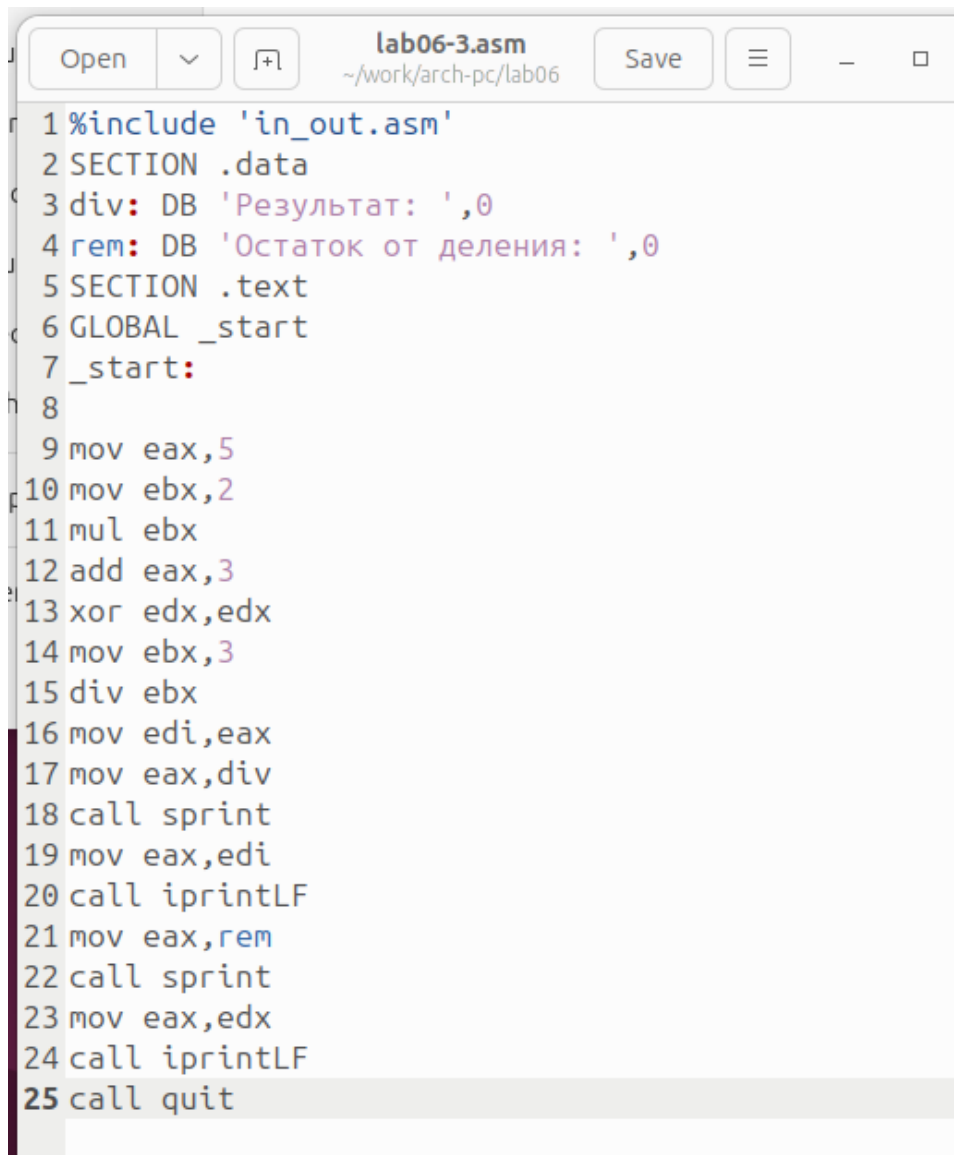
Заменяю функцию `iprintLF` на `iprint`. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки.

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

Рисунок 2.9: Запуск программы lab6-2.asm без переноса строки

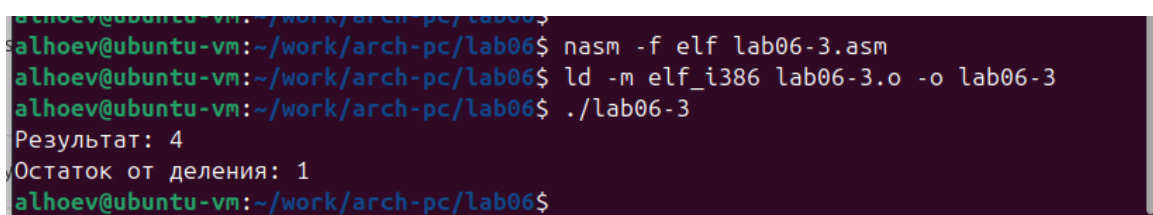
2.2 Выполнение арифметических операций в NASM

Для примера выполнения арифметических операций в NASM рассмотрим программу, вычисляющую выражение $f(x) = (5 * 2 + 3)/3$.



```
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: Код программы lab6-3.asm

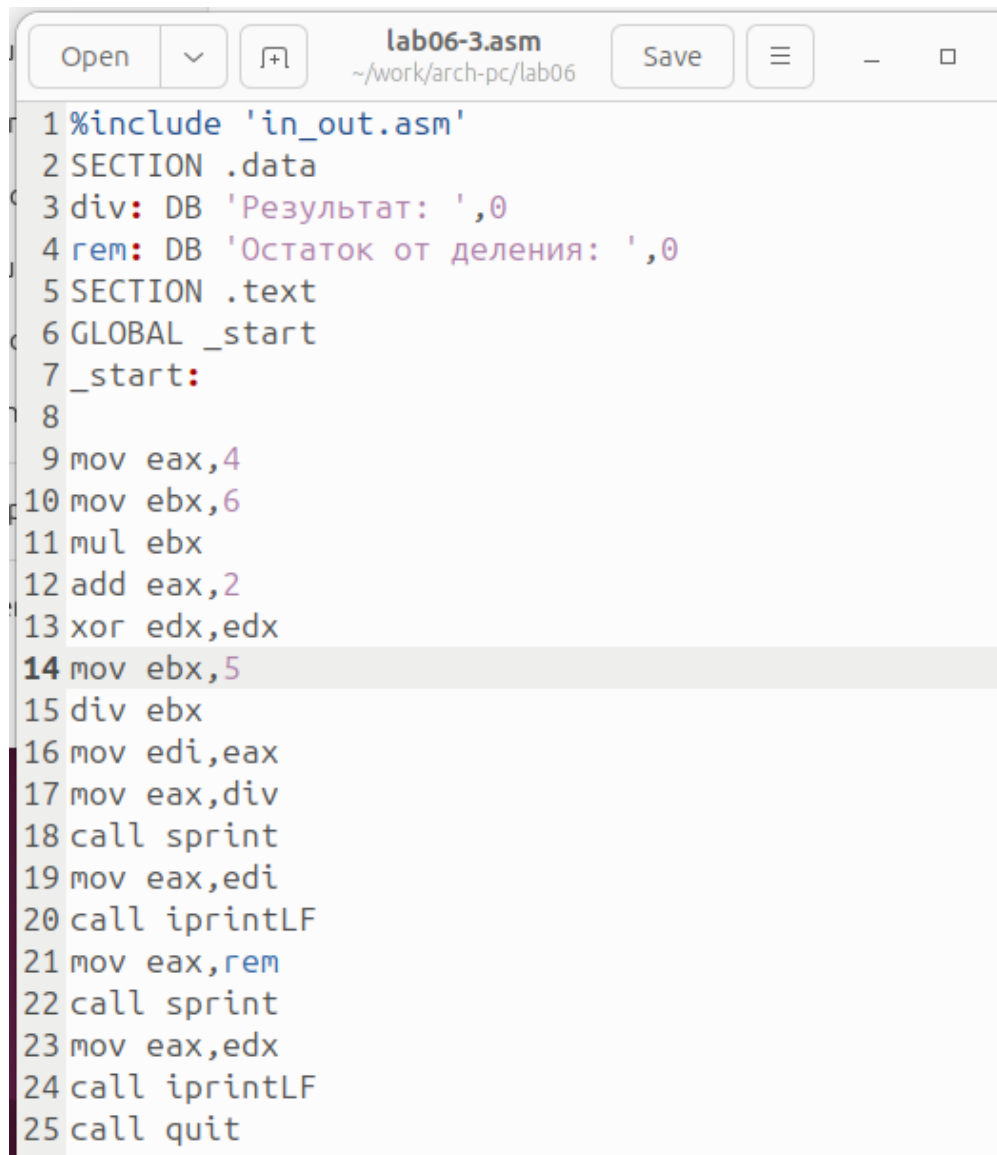


```
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ ./lab06-3
Результат: 4
Остаток от деления: 1
alhoev@ubuntu-vm:~/work/arch-pc/lab06$
```

Рисунок 2.11: Запуск программы lab6-3.asm

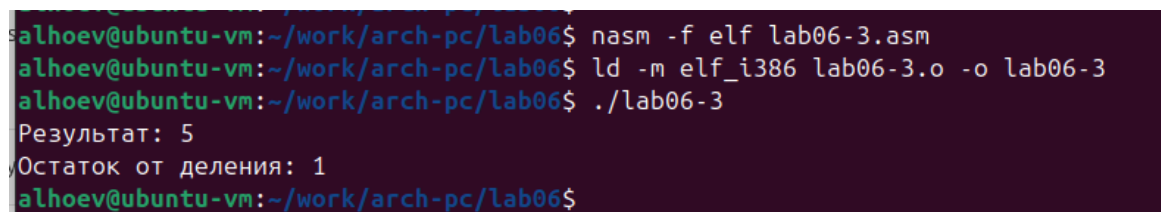
Изменяю программу для вычисления выражения $f(x) = (4 * 6 + 2) / 5$. Создаю

исполняемый файл и проверяю 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
```

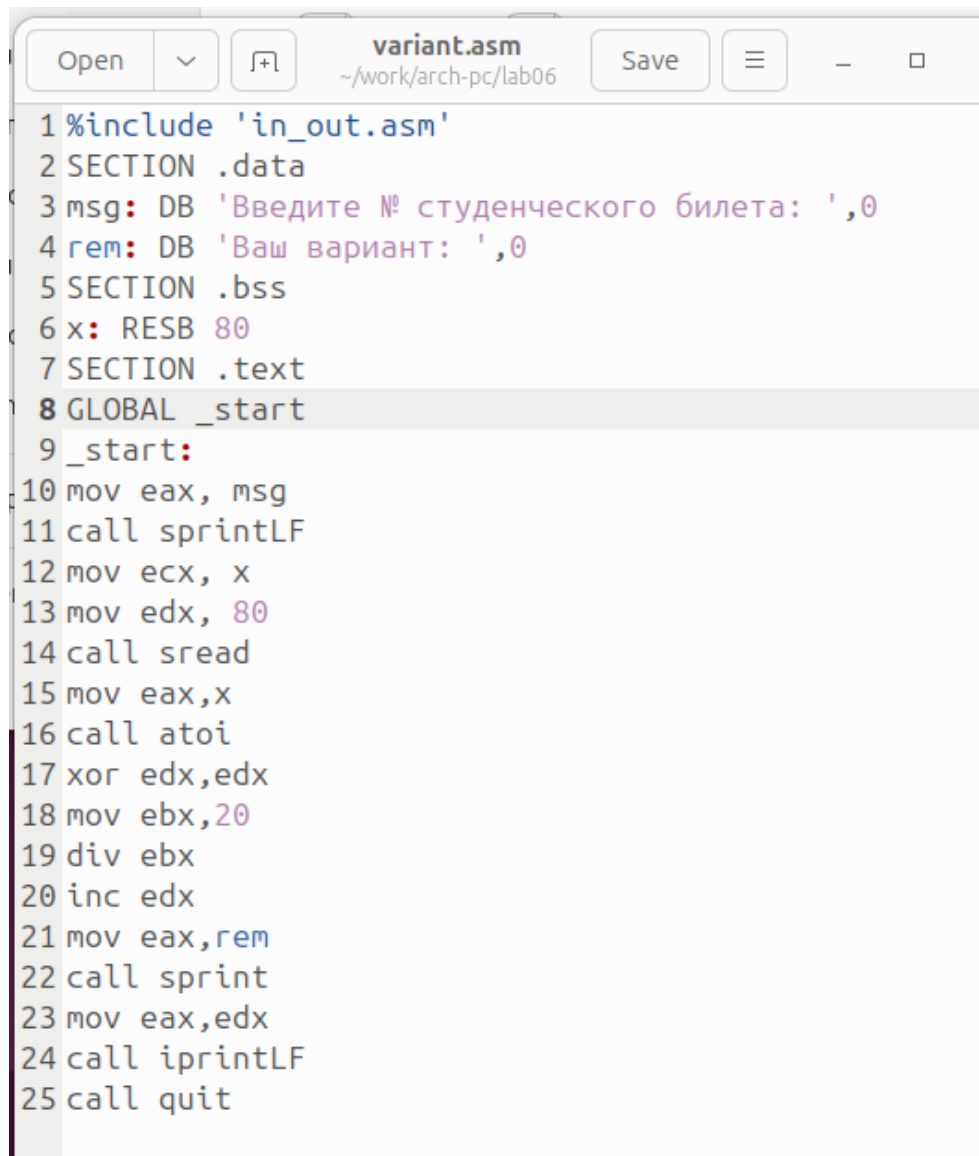
Рисунок 2.12: Код программы lab6-3.asm с новым выражением



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

Рисунок 2.13: Запуск программы lab6-3.asm с новым выражением

Рассмотрим ещё одну программу, вычисляющую вариант задания по номеру студенческого билета.



```
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: Код программы variant.asm

```
alhoev@ubuntu-vm:~/work/arch-pc/lab06$  
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ nasm -f elf variant.asm  
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant  
alhoev@ubuntu-vm:~/work/arch-pc/lab06$ ./variant  
Введите № студенческого билета:  
1132250453  
Ваш вариант: 14  
alhoev@ubuntu-vm:~/work/arch-pc/lab06$
```

Рисунок 2.15: Запуск программы variant.asm

Здесь число, над которым нужно выполнять арифметические операции, вводится с клавиатуры. Поскольку ввод осуществляется в символьном виде, символы нужно преобразовать в числа. Для этого можно использовать функцию `atoi` из файла `in_out.asm`.

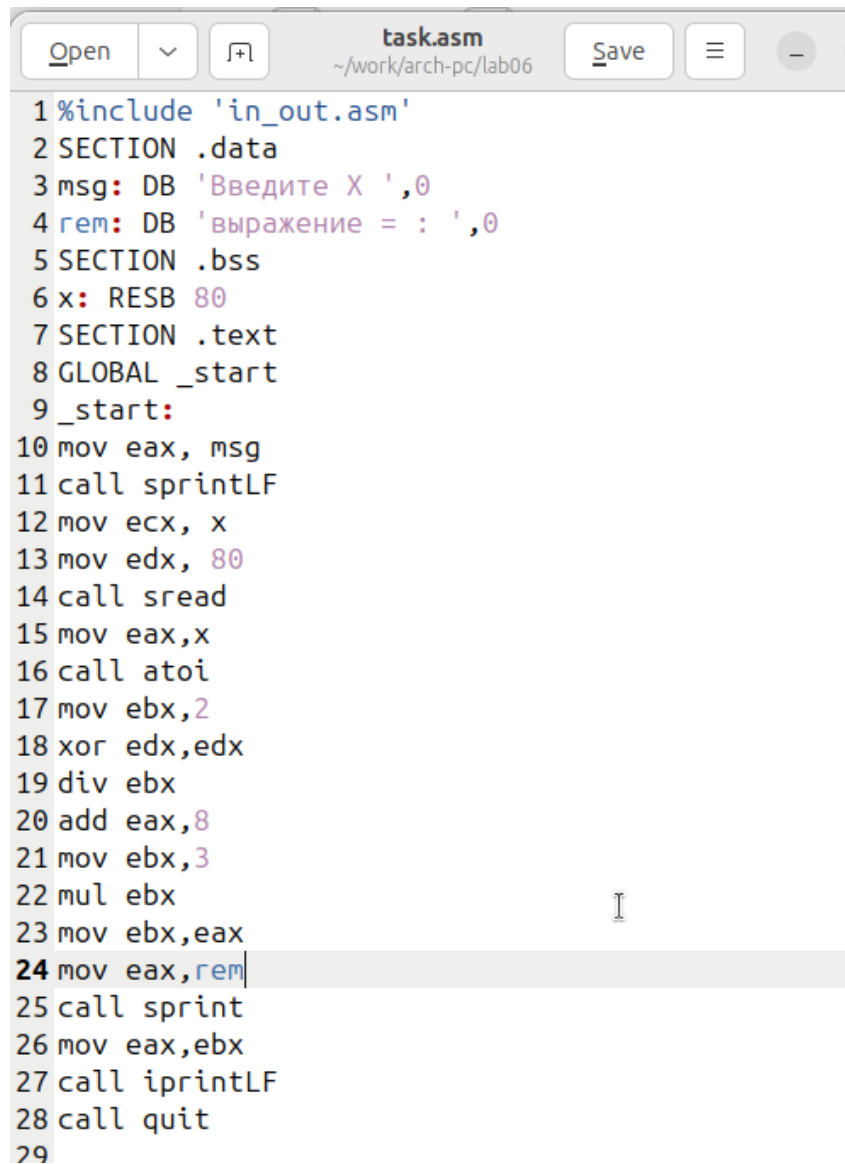
2.2.1 Ответы на вопросы

1. Какие строки отвечают за вывод сообщения „Ваш вариант:“?
 - Инструкция `mov eax, hex` загружает значение переменной с фразой „Ваш вариант:“ в регистр `eax`.
 - Инструкция `call sprint` вызывает подпрограмму для вывода строки.
2. Для чего нужны следующие инструкции?
 - Инструкция `mov ecx, x` перемещает значение переменной `x` в регистр `ecx`.
 - Инструкция `mov edx, 80` перемещает значение `80` в регистр `edx`.
 - Инструкция `call sread` вызывает подпрограмму для считывания номера студенческого билета из консоли.
3. Для чего нужна инструкция `call atoi`?
 - Инструкция `call atoi` используется для преобразования введенных символов в числовой формат.
4. Какие строки отвечают за вычисления варианта?

- Инструкция `xor edx, edx` обнуляет регистр `edx`.
 - Инструкция `mov ebx, 20` загружает значение 20 в регистр `ebx`.
 - Инструкция `div ebx` делит номер студенческого билета на 20.
 - Инструкция `inc edx` увеличивает значение регистра `edx` на 1. Здесь происходит деление номера студенческого билета на 20, а в регистре `edx` хранится остаток, к которому прибавляется 1.
5. В какой регистр записывается остаток от деления при выполнении `div ebx`?
- Остаток от деления записывается в регистр `edx`.
6. Для чего нужна инструкция `inc edx`?
- Инструкция `inc edx` увеличивает значение в регистре `edx` на 1, как это предусмотрено формулой для вычисления варианта.
7. Какие строки отвечают за вывод результата вычислений на экран?
- Инструкция `mov eax, edx` помещает результат вычислений в регистр `eax`.
 - Инструкция `call iprintLF` вызывает подпрограмму для вывода значения на экран.

2.3 Задание для самостоятельной работы

Напишите программу для вычисления выражения $y = f(x)$. Программа должна выводить формулу, запрашивать ввод значения x , вычислять выражение в зависимости от введенного x и выводить результат. Форму функции $f(x)$ выберите из таблицы 6.3 вариантов заданий в соответствии с номером, полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его для значений x_1 и x_2 из 6.3. Получили вариант 11 - $(x/2 + 8) * 3$ для $x = 1, x = 4$.



```
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 sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x
16 call atoi
17 mov ebx, 2
18 xor edx, edx
19 div ebx
20 add eax, 8
21 mov ebx, 3
22 mul ebx
23 mov ebx, eax
24 mov eax, rem
25 call sprintf
26 mov eax, ebx
27 call iprintLF
28 call quit
29
```

Рисунок 2.16: Код программы task.asm

При $x = 1$ результат — 24.

При $x = 4$ результат — 30.

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

Рисунок 2.17: Запуск программы task.asm

Программа работает корректно.

3 Выводы

Изучили работу с арифметическими операциями.