

Отчёт по лабораторной работе 6

дисциплина: Архитектура компьютера

Тяпкова Альбина НММбд-04-24

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
2.2.1	Ответы на вопросы	15
2.3	Задание для самостоятельной работы	16
3	Выводы	19

Список иллюстраций

2.1	Код программы в lab6-1.asm	7
2.2	Запуск программы lab6-1.asm	7
2.3	Измененный код в lab6-1.asm	8
2.4	Запуск измененной программы	8
2.5	Код программы в lab6-2.asm	9
2.6	Запуск программы lab6-2.asm	9
2.7	Измененный код программы в lab6-2.asm	10
2.8	Запуск программы lab6-2.asm	10
2.9	Запуск программы lab6-2.asm без переноса строки	11
2.10	Код программы в lab6-3.asm	12
2.11	Запуск программы lab6-3.asm	12
2.12	Измененный код в lab6-3.asm	13
2.13	Запуск программы lab6-3.asm	13
2.14	Код программы variant.asm	14
2.15	Запуск программы variant.asm	14
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.

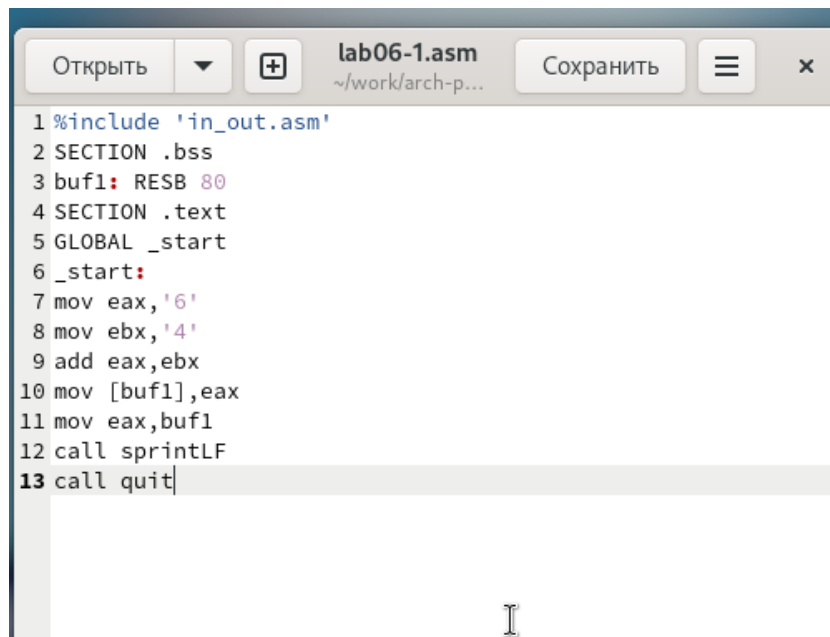
2 Выполнение лабораторной работы

2.1 Символьные и численные данные в NASM

Создала папку для лабораторной работы №6, перешла в неё и создала файл `lab6-1.asm`.

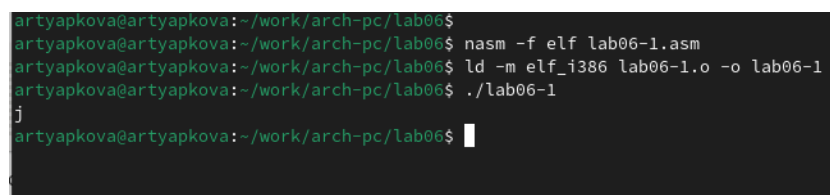
Давайте посмотрим примеры программ, которые выводят символы и числа. Они выводят значения из регистра `eax`.

В программе записываем символ 6 в регистр `eax` (с помощью `mov eax, '6'`), а символ 4 — в `ebx` (`mov ebx, '4'`). Сложим `eax` и `ebx` (командой `add eax, ebx`, результат окажется в `eax`) и выведем его. Так как для `sprintf` в `eax` нужен адрес, создаём переменную: сохраняем значение `eax` в `buf1`, записываем адрес `buf1` в `eax` и вызываем `sprintf`.



```
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
```

Рис. 2.1: Код программы в lab6-1.asm

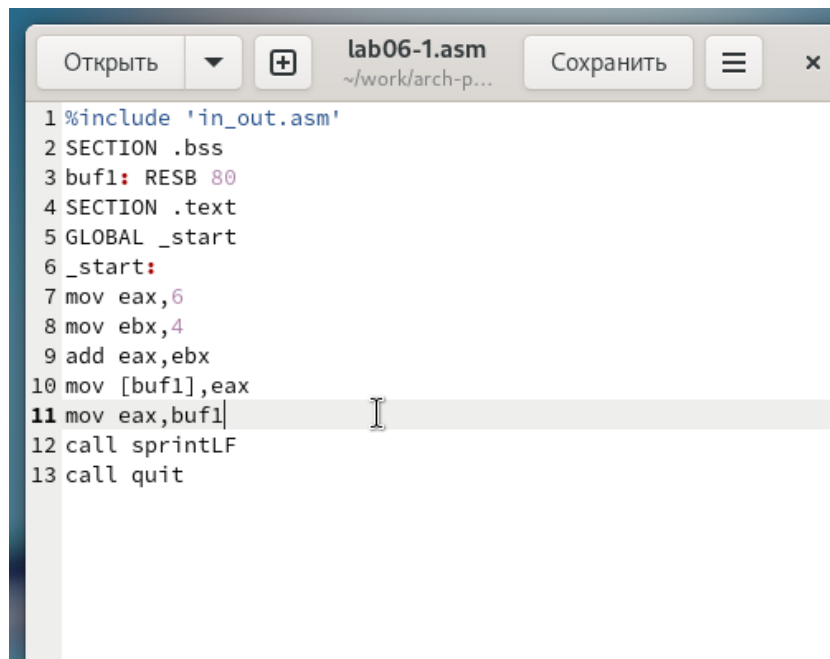


```
artyapkova@artyapkova:~/work/arch-pc/lab06$
artyapkova@artyapkova:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
artyapkova@artyapkova:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
artyapkova@artyapkova:~/work/arch-pc/lab06$ ./lab06-1
j
artyapkova@artyapkova:~/work/arch-pc/lab06$
```

Рис. 2.2: Запуск программы lab6-1.asm

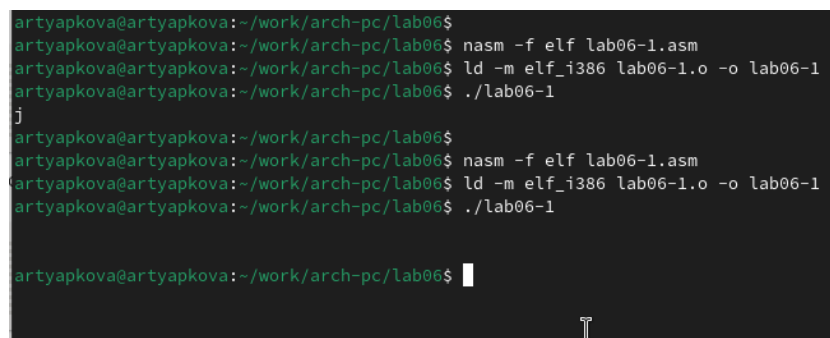
Мы ожидаем увидеть число 10, но вместо этого получаем символ j. Это потому, что `add eax, ebx` сложил коды символов 6 и 4 (106, что соответствует j).

Теперь изменяю программу, чтобы в регистрах были числа, а не символы.



```
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
```

Рис. 2.3: Измененный код в lab6-1.asm



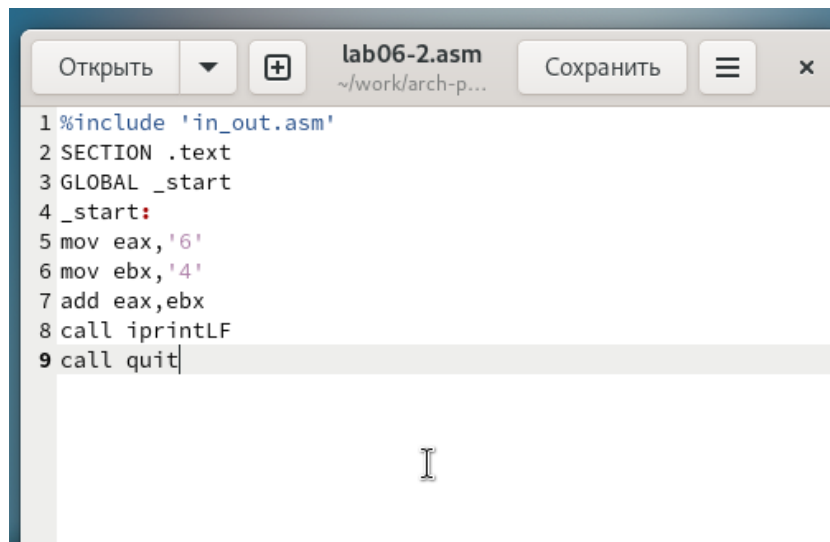
```
artyapkova@artyapkova:~/work/arch-pc/lab06$
artyapkova@artyapkova:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
artyapkova@artyapkova:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
artyapkova@artyapkova:~/work/arch-pc/lab06$ ./lab06-1
j
artyapkova@artyapkova:~/work/arch-pc/lab06$
artyapkova@artyapkova:~/work/arch-pc/lab06$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

artyapkova@artyapkova:~/work/arch-pc/lab06$
```

Рис. 2.4: Запуск измененной программы

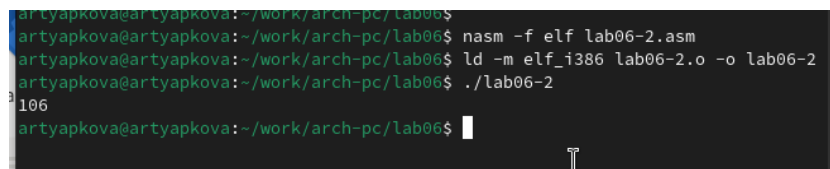
Теперь результатом будет символ с кодом 10, который обозначает конец строки и выводит пустую строку.

В `in_out.asm` есть подпрограммы для преобразования ASCII-символов в числа. Применяю их в новой версии программы.



```
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
```

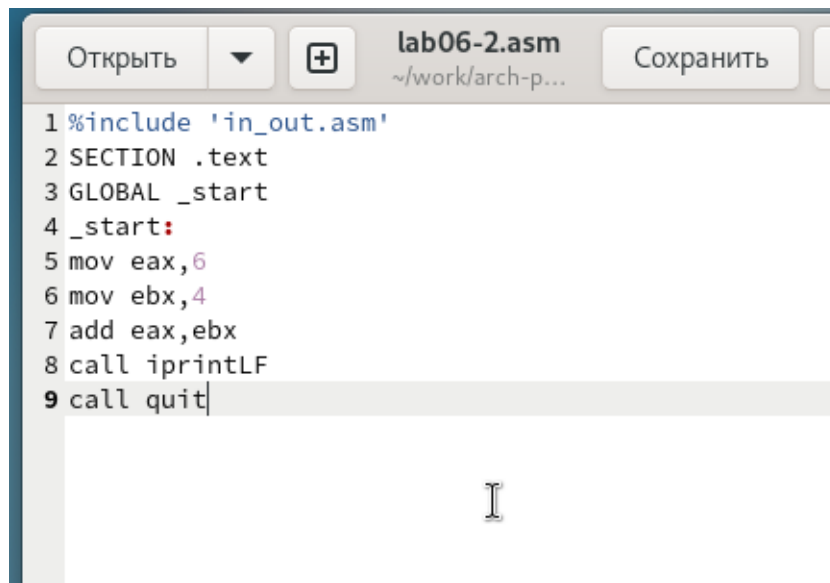
Рис. 2.5: Код программы в lab6-2.asm



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

Рис. 2.6: Запуск программы lab6-2.asm

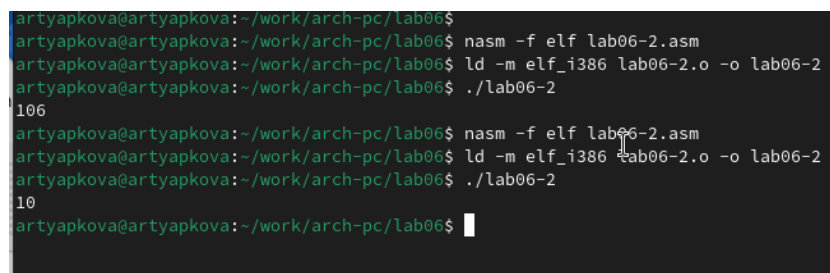
В итоге программа выдаёт число 106, так как add складывает коды символов. Функция iprintLF выводит число, а не символ. Аналогично, заменяю символы на числа.



```
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
```

Рис. 2.7: Измененный код программы в lab6-2.asm

Теперь iprintLF выводит число 10, так как операндами были числа, а не символы.



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

Рис. 2.8: Запуск программы lab6-2.asm

Меняю iprintLF на iprint, собираю и запускаю файл. Результат выводится без переноса строки.

```

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

```

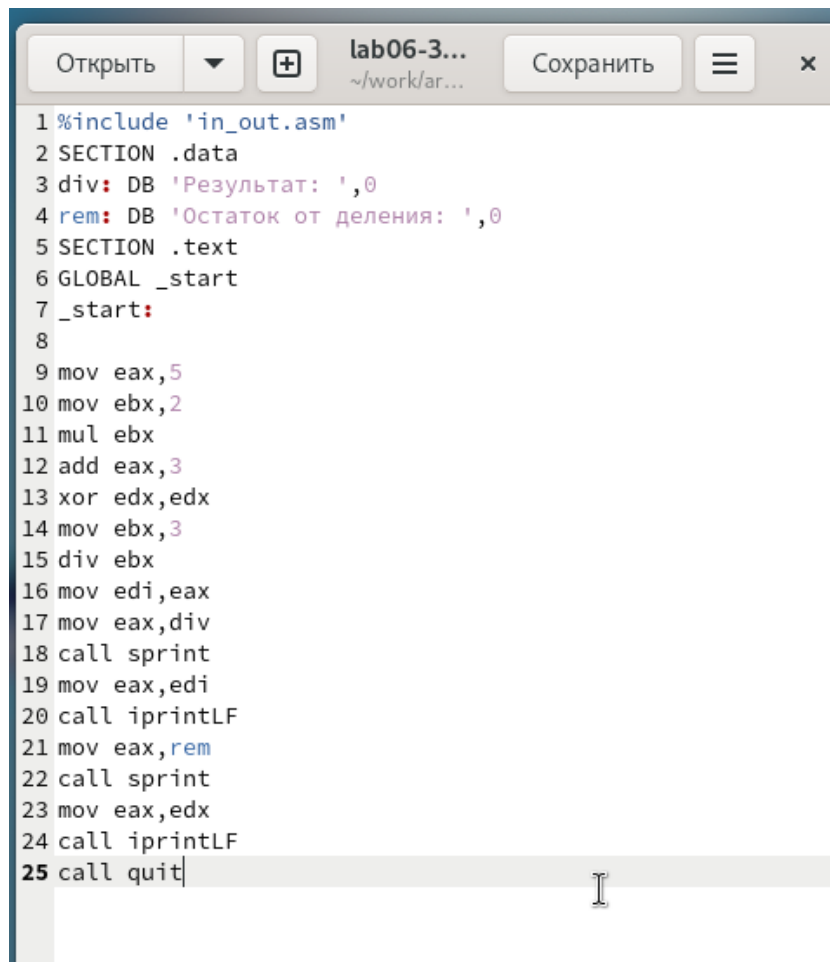
Рис. 2.9: Запуск программы lab6-2.asm без переноса строки

2.2 Выполнение арифметических операций в NASM

Рассмотрим программу для выражения

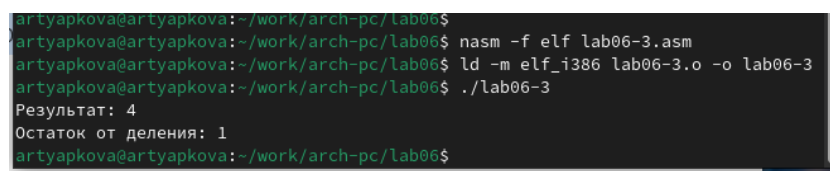
$$f(x) = (5 * 2 + 3) / 3$$

.



```
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
```

Рис. 2.10: Код программы в lab6-3.asm



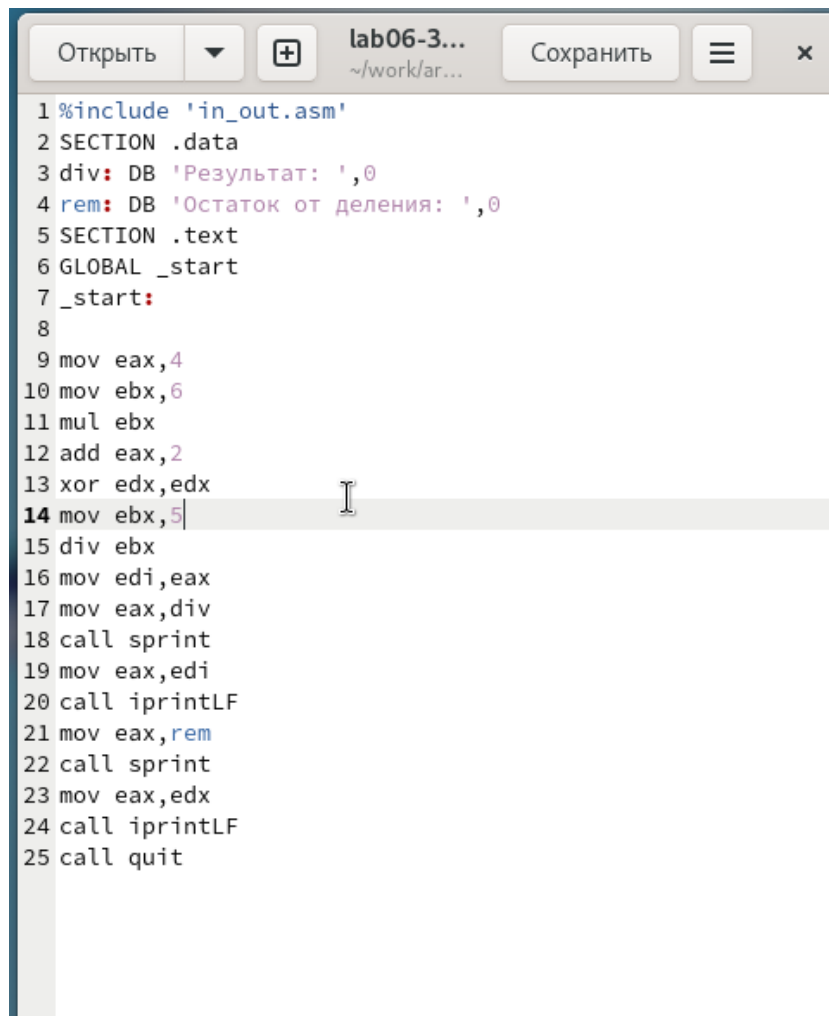
```
artyapkova@artyapkova:~/work/arch-pc/lab06$
artyapkova@artyapkova:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
artyapkova@artyapkova:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
artyapkova@artyapkova:~/work/arch-pc/lab06$ ./lab06-3
Результат: 4
Остаток от деления: 1
artyapkova@artyapkova:~/work/arch-pc/lab06$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

Рис. 2.11: Запуск программы lab6-3.asm

Изменяю программу для вычисления

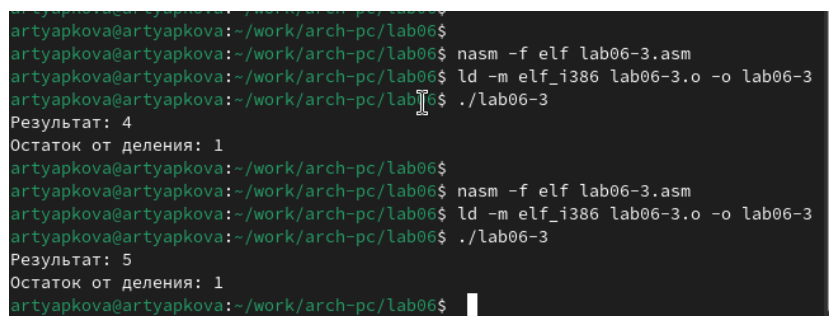
$$f(x) = (4 * 6 + 2)/5$$

, собираю и запускаю.



```
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. 2.12: Измененный код в lab6-3.asm

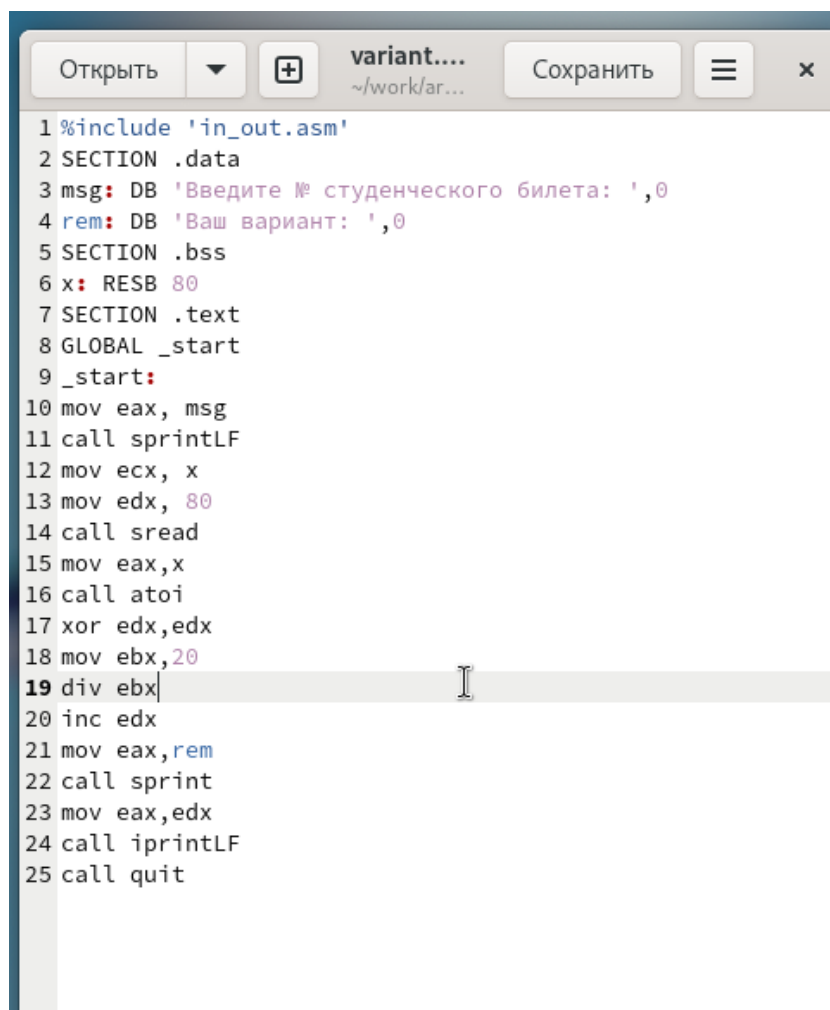


```
artyapkova@artyapkova:~/work/arch-pc/lab06$
artyapkova@artyapkova:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
artyapkova@artyapkova:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
artyapkova@artyapkova:~/work/arch-pc/lab06$ ./lab06-3
Результат: 4
Остаток от деления: 1
artyapkova@artyapkova:~/work/arch-pc/lab06$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

Рис. 2.13: Запуск программы lab6-3.asm

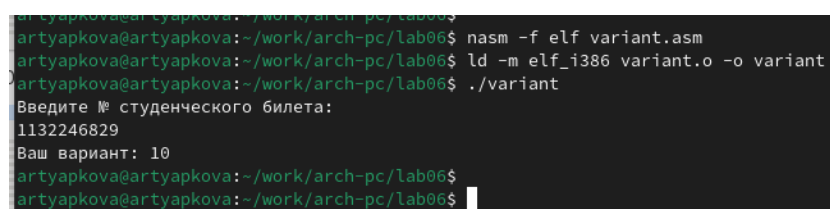
Также есть программа для вычисления значения по номеру студенческого билета.

Ввод производится с клавиатуры в символьном виде и преобразуется в число с помощью `atoi` из `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 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
```

Рис. 2.14: Код программы `variant.asm`



```
artyapkova@artyapkova: ~/work/arch-pc/lab06$ nasm -f elf variant.asm
artyapkova@artyapkova: ~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
artyapkova@artyapkova: ~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246829
Ваш вариант: 10
artyapkova@artyapkova: ~/work/arch-pc/lab06$
artyapkova@artyapkova: ~/work/arch-pc/lab06$
```

Рис. 2.15: Запуск программы `variant.asm`

2.2.1 Ответы на вопросы

1. Какие строки отвечают за вывод сообщения 'Ваш вариант'?

- `mov eax, rem` — значение переменной с фразой 'Ваш вариант' переносится в регистр.
- `call sprint` — вывод сообщения.

2. Для чего используются инструкции?

- `mov ecx, x`
- `mov edx, 80`
- `call sread`

Они считывают номер студенческого билета в переменную `x`.

3. Для чего используется `call atoi`?

Преобразует введённые символы в числовой формат.

4. Какие строки отвечают за вычисления варианта?

- `xor edx, edx`
- `mov ebx, 20`
- `div ebx`
- `inc edx`

Здесь номер билета делится на 20, и к остатку, сохранённому в `edx`, прибавляется 1.

5. В какой регистр записывается остаток от деления при `div ebx`?

В `edx`.

6. Для чего используется `inc edx`?

По формуле добавляем единицу.

7. Какие строки отвечают за вывод результата?

- `mov eax, edx` — результат записывается в `eax`.
- `call iprintLF` — вывод результата.

2.3 Задание для самостоятельной работы

Написать программу для вычисления выражения $y = f(x)$. Программа должна запрашивать значение x , вычислять результат и выводить его. Вид функции $f(x)$ определить по таблице 6.3 в соответствии с номером, полученным в лабораторной.

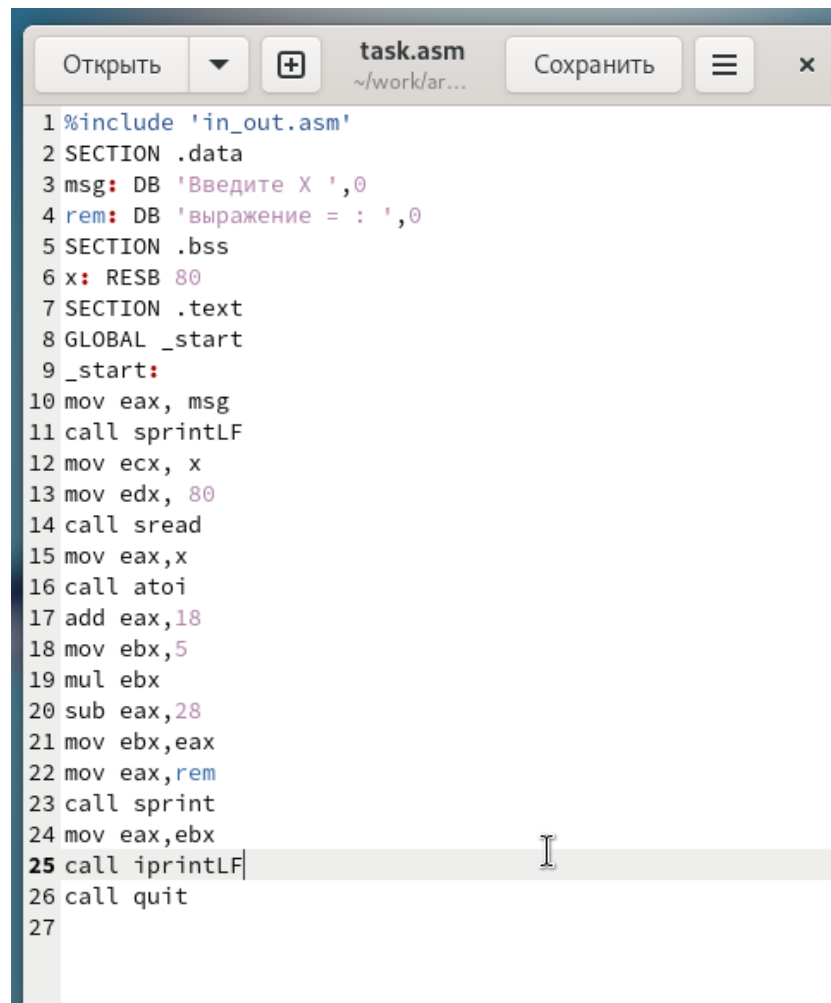
Полученный вариант — 10, выражение:

$$5(x + 18) - 28$$

для

$$x_1 = 2, x_2 = 3$$

.



```
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 add eax, 18
18 mov ebx, 5
19 mul ebx
20 sub eax, 28
21 mov ebx, eax
22 mov eax, rem
23 call sprintf
24 mov eax, ebx
25 call iprintLF
26 call quit
27
```

Рис. 2.16: Код программы task.asm

Результаты: - При $x = 2$:

$$5(2 + 18) - 28 = 72$$

. - При $x = 3$:

$$5(3 + 18) - 28 = 77$$

.

```
artyapkova@artyapkova:~/work/arch-pc/lab06$  
artyapkova@artyapkova:~/work/arch-pc/lab06$ nasm -f elf task.asm  
artyapkova@artyapkova:~/work/arch-pc/lab06$ ld -m elf_i386 task.o -o task  
artyapkova@artyapkova:~/work/arch-pc/lab06$ ./task  
Введите X  
2  
выражение = : 72  
artyapkova@artyapkova:~/work/arch-pc/lab06$ ./task  
Введите X  
3  
выражение = : 77  
artyapkova@artyapkova:~/work/arch-pc/lab06$
```

Рис. 2.17: Запуск программы task.asm

Программа работает верно.

3 Выводы

Освоила работу с арифметическими операциями в NASM.