

Отчёт по лабораторной работе 6

Архитектура компьютера

Чермашенцев Павел Андреевич НБИбд-03-24

Содержание

| | | |
|----------|--|-----------|
| 1 | Цель работы | 5 |
| 2 | Выполнение лабораторной работы | 6 |
| 2.1 | Ответы на вопросы по программе variant.asm | 15 |
| 2.2 | Самостоятельное задание | 16 |
| 3 | Выводы | 19 |

Список иллюстраций

| | | |
|------|---|----|
| 2.1 | Программа в файле lab6-1.asm | 6 |
| 2.2 | Запуск программы lab6-1.asm | 7 |
| 2.3 | Программа в файле lab6-1.asm | 8 |
| 2.4 | Запуск программы lab6-1.asm | 8 |
| 2.5 | Программа в файле lab6-2.asm | 9 |
| 2.6 | Запуск программы lab6-2.asm | 9 |
| 2.7 | Программа в файле lab6-2.asm | 10 |
| 2.8 | Запуск программы lab6-2.asm | 10 |
| 2.9 | Запуск программы lab6-2.asm | 10 |
| 2.10 | Программа в файле lab6-3.asm | 11 |
| 2.11 | Запуск программы lab6-3.asm | 12 |
| 2.12 | Программа в файле lab6-3.asm | 13 |
| 2.13 | Запуск программы lab6-3.asm | 13 |
| 2.14 | Программа в файле variant.asm | 14 |
| 2.15 | Запуск программы variant.asm | 15 |
| 2.16 | Программа в файле task.asm | 17 |
| 2.17 | Запуск программы task.asm | 18 |

Список таблиц

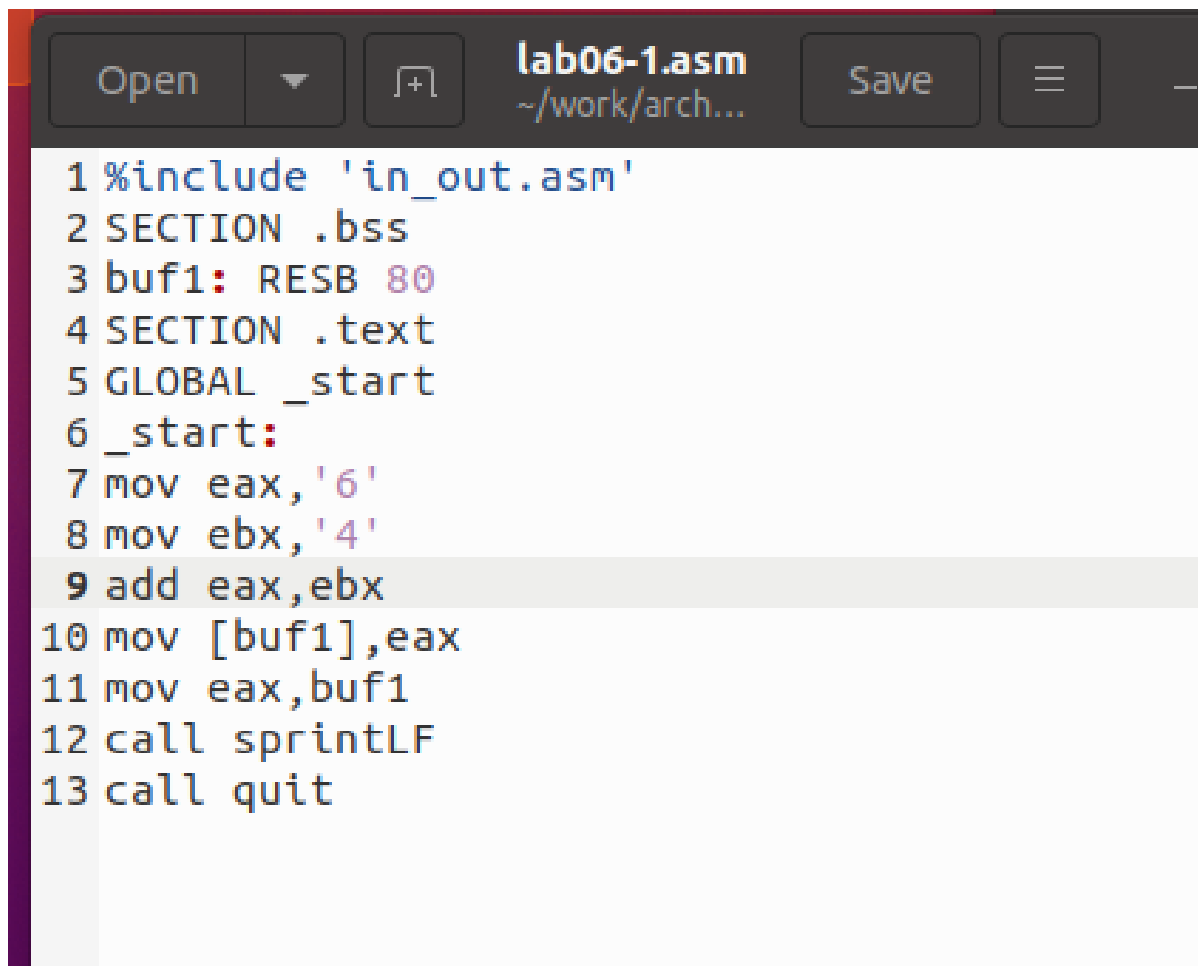
1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

Создан каталог для программ лабораторной работы № 6, в который был добавлен файл `lab6-1.asm`.

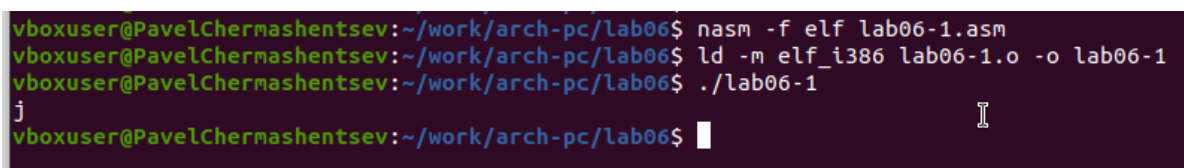
Рассмотрим примеры программ, выводящих символьные и числовые значения. Эти программы будут выводить данные, записанные в регистр `eax`.



```
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 sprintLF
13 call quit
```

Рис. 2.1: Программа в файле `lab6-1.asm`

В данной программе (рис. 2.1) в регистр `eax` записывается символ '6' (команда `mov eax, '6'`), а в регистр `ebx` — символ '4' (команда `mov ebx, '4'`). Затем происходит сложение значений регистров `eax` и `ebx` (команда `add eax, ebx`), результат операции записывается в регистр `eax`. После этого выводится результат. Для использования функции `sprintf` необходимо, чтобы в регистре `eax` находился адрес, поэтому используется дополнительная переменная. Значение регистра `eax` записывается в переменную `buf1` (команда `mov [buf1], eax`), а затем в регистр `eax` записывается адрес переменной `buf1` (команда `mov eax, buf1`), после чего вызывается функция `sprintf`.

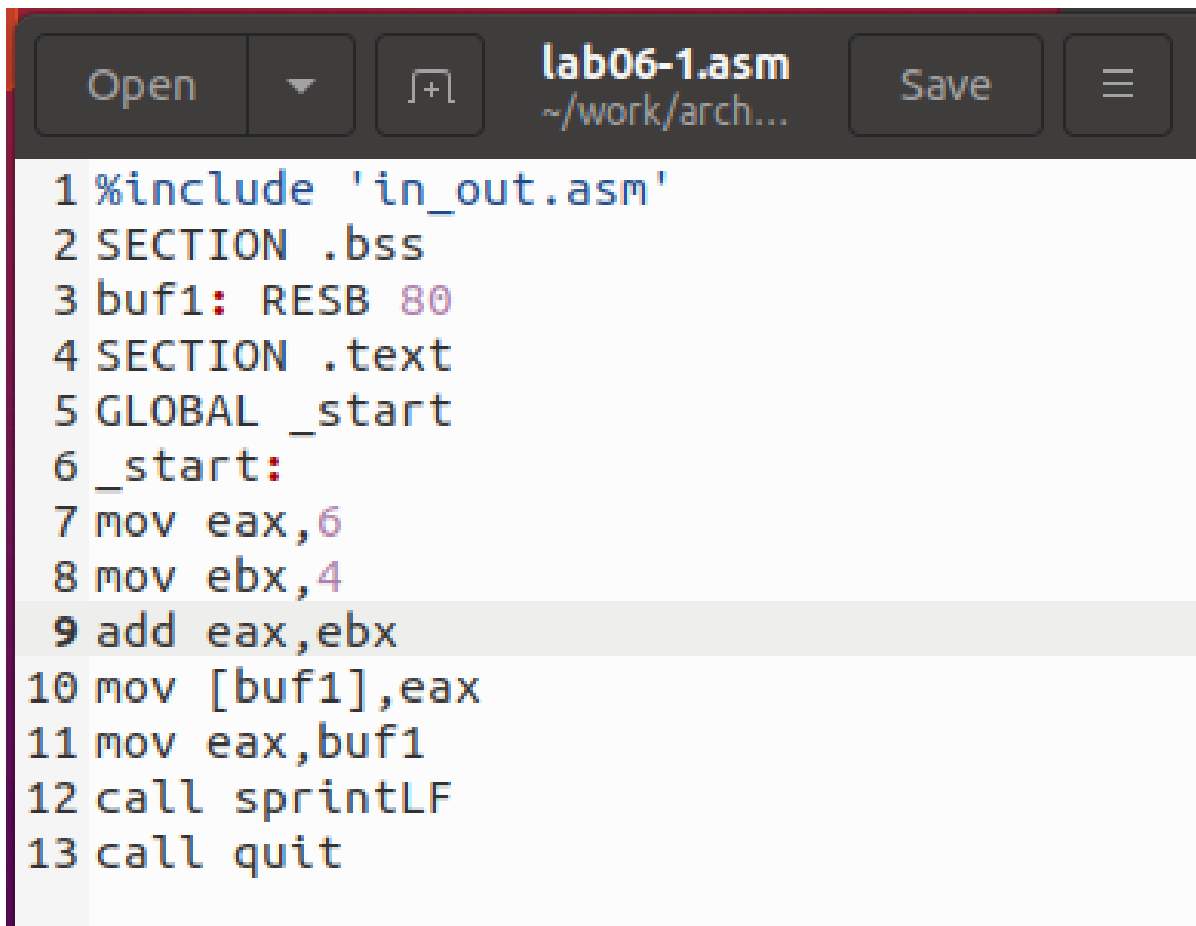


```
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ nasm -f elf lab06-1.asm
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-1.o -o lab06-1
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ./lab06-1
j
```

Рис. 2.2: Запуск программы lab6-1.asm

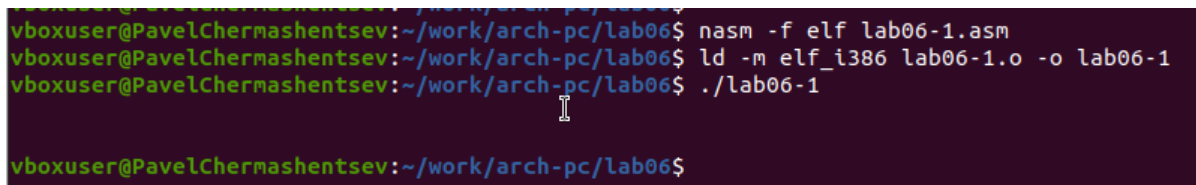
В случае, когда ожидаем получить число 10 при выводе содержимого регистра `eax`, фактический результат будет символ 'j'. Это объясняется тем, что код символа '6' в двоичном представлении равен 00110110 (54 в десятичном), а код символа '4' — 00110100 (52 в десятичном). При выполнении команды `add eax, ebx` результатом будет сумма этих кодов — 01101010 (106 в десятичном), что соответствует символу 'j' (рис. 2.2).

Далее изменяем программу, заменяя символы на числа (рис. 2.3).



```
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. 2.3: Программа в файле lab6-1.asm



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

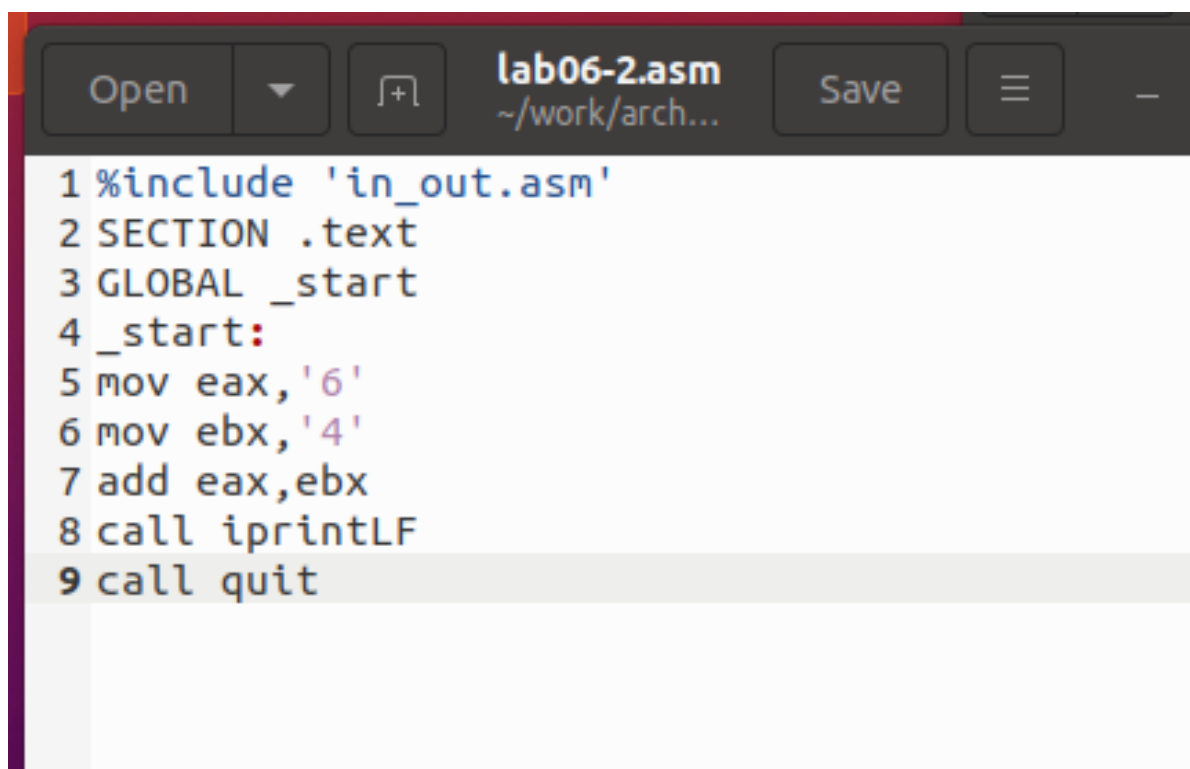
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$
```

Рис. 2.4: Запуск программы lab6-1.asm

Как и в предыдущем примере, при выполнении программы мы не получаем число 10. Вместо этого выводится символ с кодом 10, который представляет собой символ конца строки (возврат каретки). Этот символ не отображается в консоли, но добавляет пустую строку.

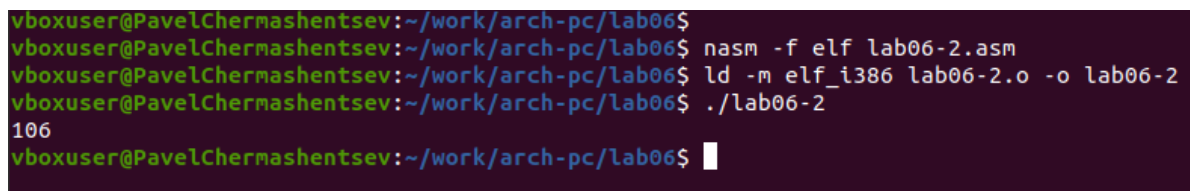
Как упоминалось ранее, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно.

Преобразовал программу с использованием этих функций (рис. 2.5).



```
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: Программа в файле lab6-2.asm

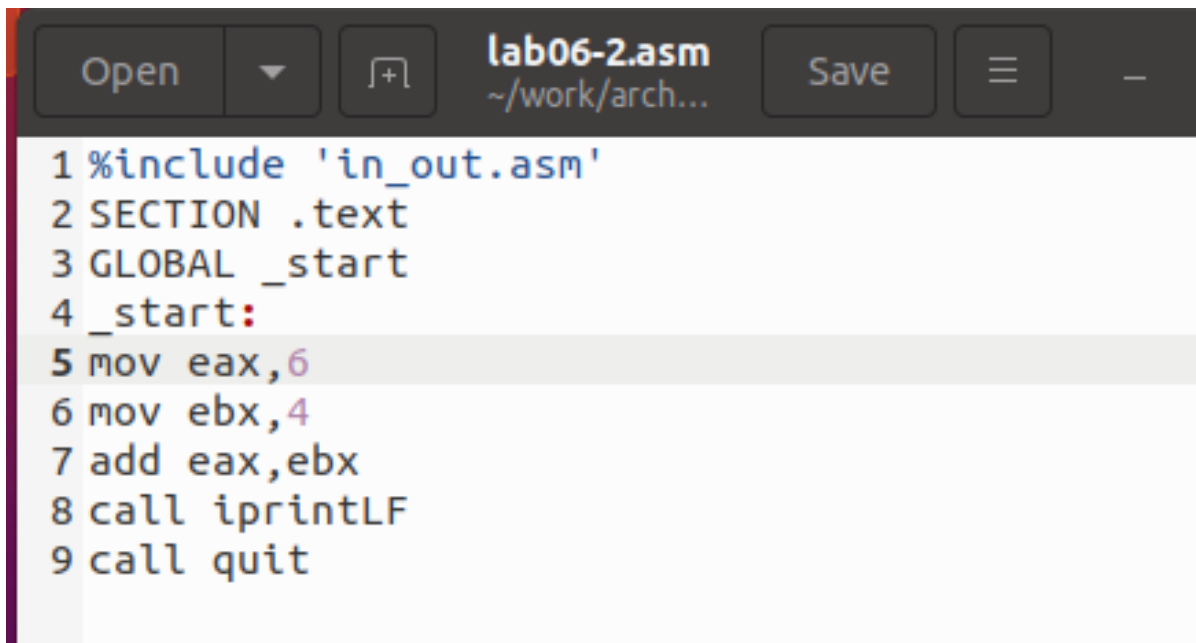


```
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ./lab06-2
106
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$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

Рис. 2.6: Запуск программы lab6-2.asm

В результате выполнения программы выводится число 106 (рис. 2.6). В этом случае команда `add` суммирует коды символов '6' и '4' ($54 + 52 = 106$). Однако, в отличие от предыдущей программы, функция `iprintLF` позволяет вывести число, а не символ, который соответствует этому числу.

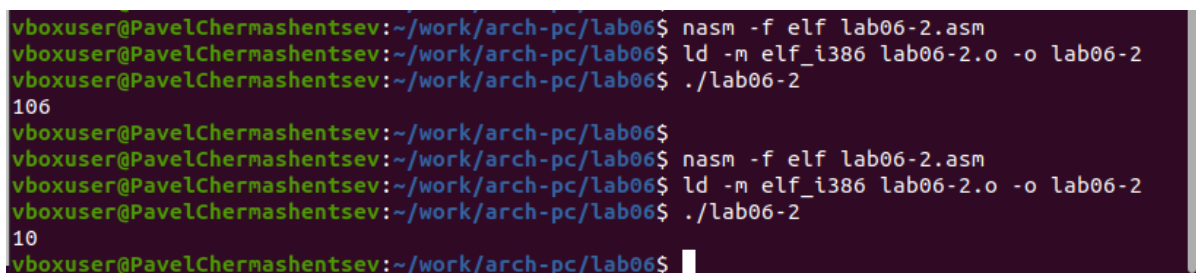
Аналогично предыдущему примеру изменяем символы на числа (рис. 2.7).



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

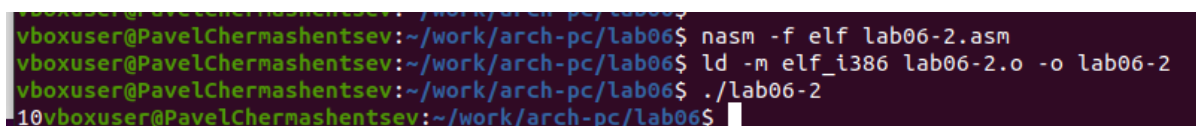
Функция `iprintLF` позволяет вывести число, и операндами являются числа, а не коды символов. Поэтому на экране будет выведено число 10 (рис. 2.8).



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

Рис. 2.8: Запуск программы lab6-2.asm

Заменяю функцию `iprintLF` на `iprint`, создаю исполняемый файл и запускаю его. Вывод отличается тем, что теперь нет переноса строки (рис. 2.9).

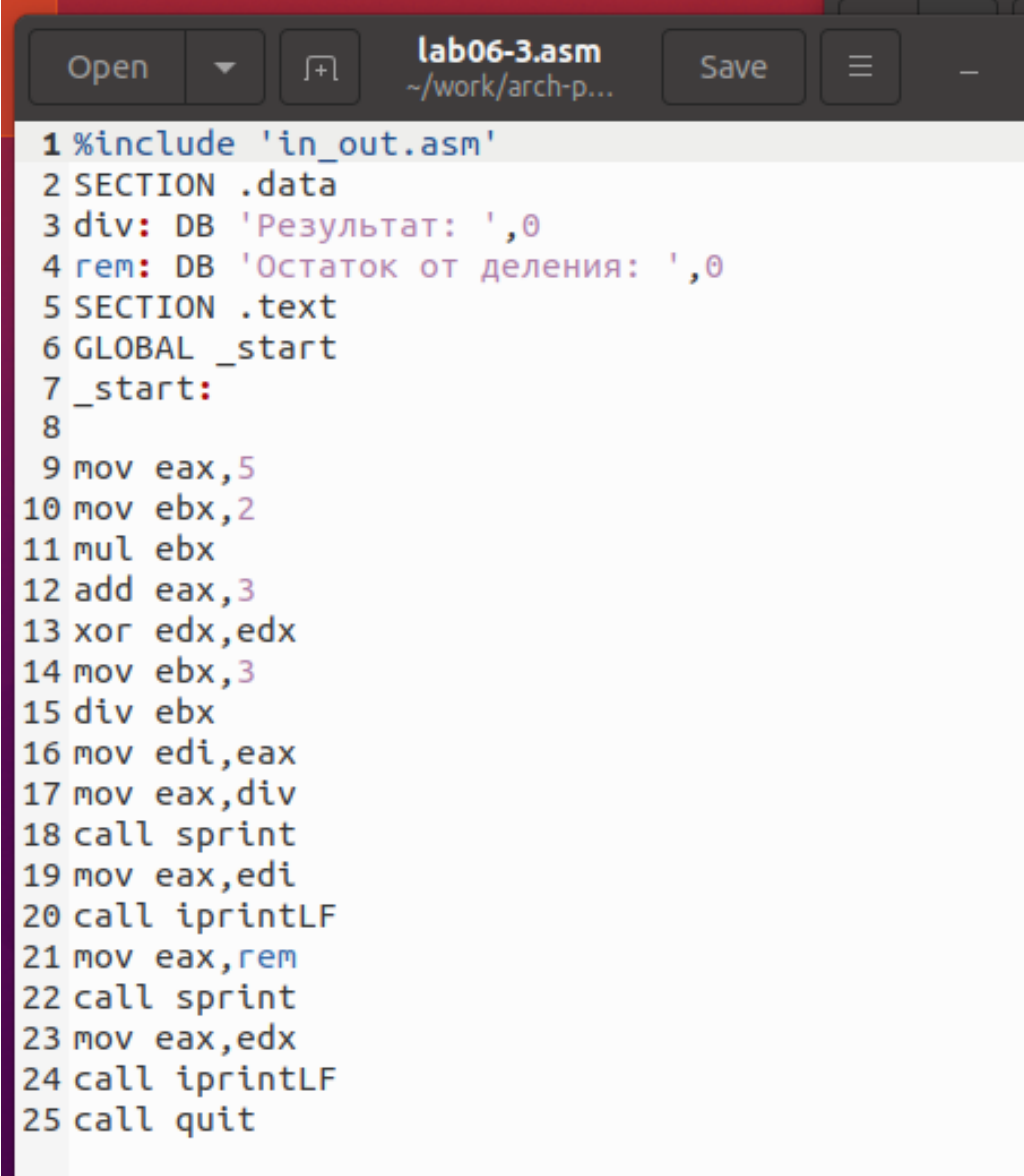


```
vboxuser@PavelChernashentsev:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
vboxuser@PavelChernashentsev:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
vboxuser@PavelChernashentsev:~/work/arch-pc/lab06$ ./lab06-2
10vboxuser@PavelChernashentsev:~/work/arch-pc/lab06$
```

Рис. 2.9: Запуск программы lab6-2.asm

Для примера арифметических операций в NASM привожу программу для вычисления выражения (рис. 2.10) (рис. 2.11):

$$f(x) = \frac{5 \times 2 + 3}{3}.$$



```
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: Программа в файле lab6-3.asm

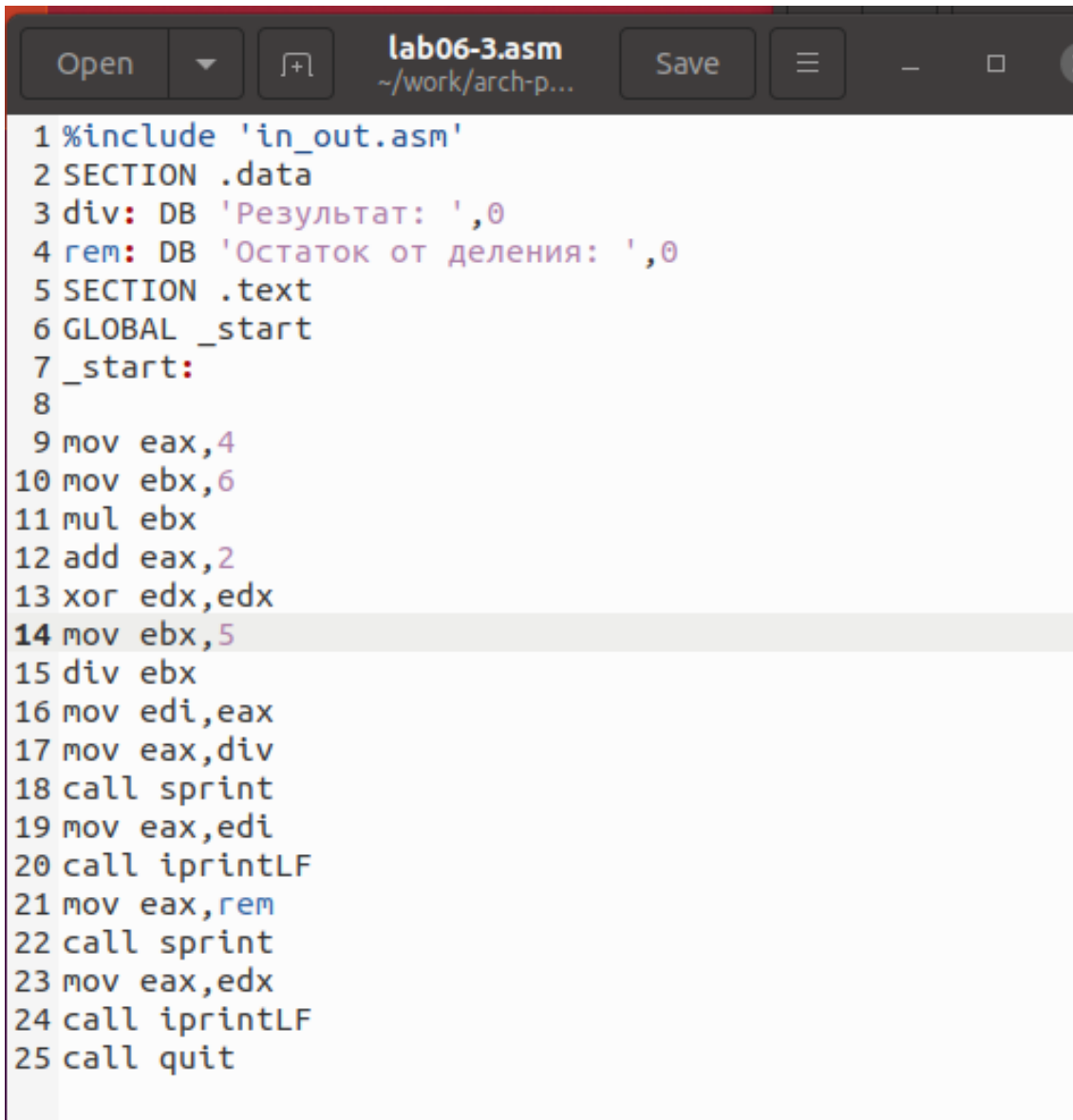
```
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ./lab06-3
Результат: 4
Остаток от деления: 1
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$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

Рис. 2.11: Запуск программы lab6-3.asm

Изменил программу для вычисления выражения:

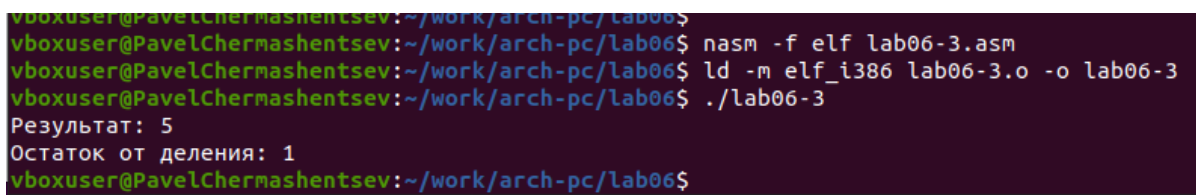
$$f(x) = \frac{4 \times 6 + 2}{5}.$$

Создал исполняемый файл и проверил его работу (рис. 2.12) (рис. 2.13).



```
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: Программа в файле lab6-3.asm



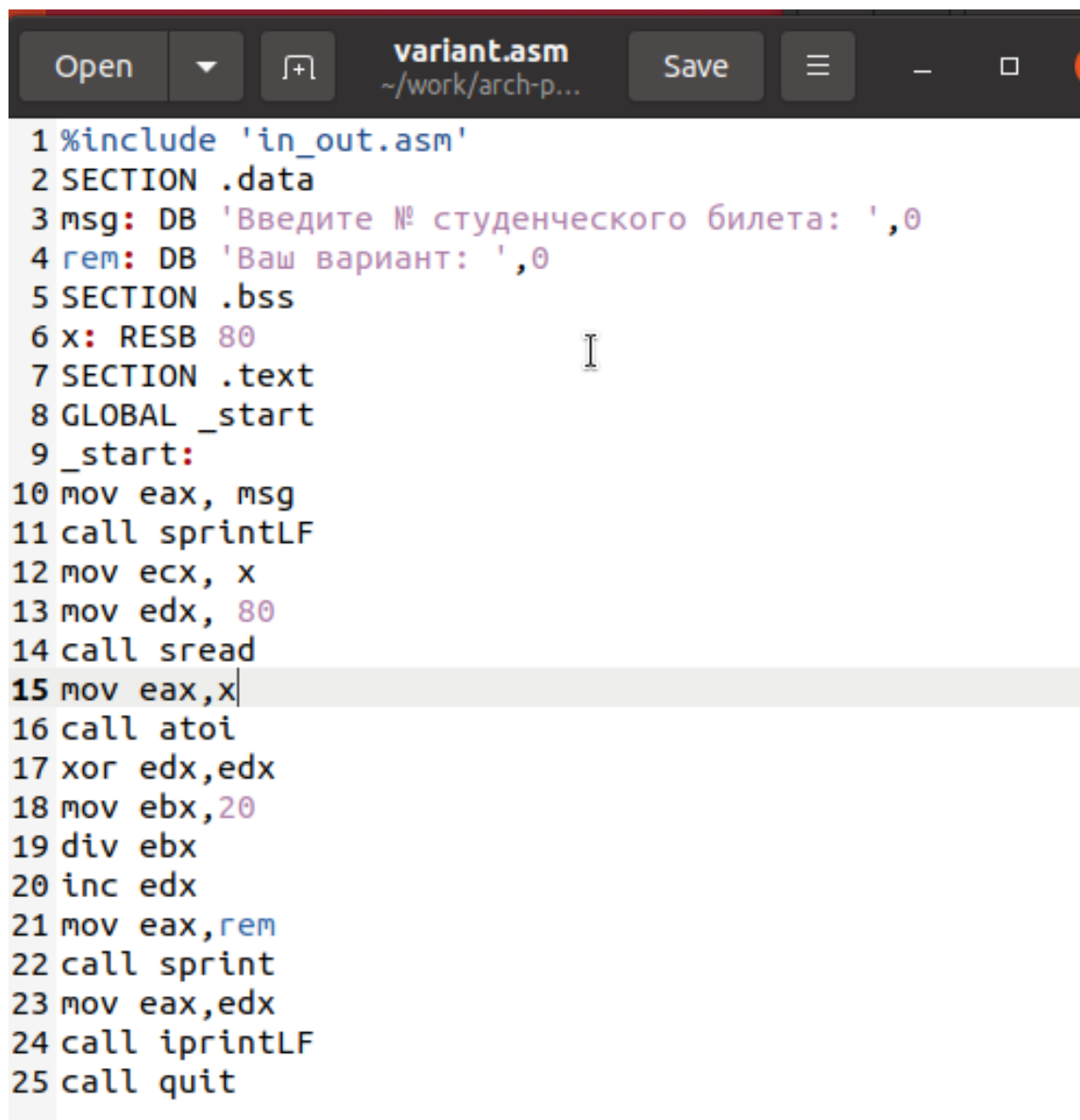
```
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ nasm -f elf lab06-3.asm
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-3.o -o lab06-3
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ./lab06-3
Результат: 5
Остаток от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

Рис. 2.13: Запуск программы lab6-3.asm

Рассмотрим еще один пример программы для вычисления варианта задания

по номеру студенческого билета (рис. 2.14) (рис. 2.15).

В этом примере число, с которым нужно проводить арифметические операции, вводится с клавиатуры. Так как ввод осуществляется в виде символов, для корректной работы арифметических операций символы необходимо преобразовать в числа. Для этого используется 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 iprintf
25 call quit
```

Рис. 2.14: Программа в файле `variant.asm`

```
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ nasm -f elf variant.asm
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246450
Ваш вариант: 11
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$
```

Рис. 2.15: 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:»?

Строка `mov eax, rem` записывает в регистр значение переменной с фразой «Ваш вариант:», а строка `call sprint` вызывает подпрограмму для вывода этой строки на экран.

2. Для чего используются следующие инструкции?

- `nasm`: используется для компиляции кода на языке ассемблера NASM.
- `mov ecx, x`: перемещает значение переменной `x` в регистр `ecx`.
- `mov edx, 80`: перемещает значение 80 в регистр `edx`.
- `call sread`: вызывает подпрограмму для считывания значения студенческого билета с консоли.

3. Для чего используется инструкция `call atoi`?

Инструкция `call atoi` используется для преобразования введенных символов в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?

- `xor edx, edx`: обнуляет регистр `edx`.
- `mov ebx, 20`: записывает значение 20 в регистр `ebx`.
- `div ebx`: выполняет деление номера студенческого билета на 20.

- `inc edx`: увеличивает значение регистра `edx` на 1.

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

Остаток от деления записывается в регистр `edx`.

6. Для чего используется инструкция `inc edx`?

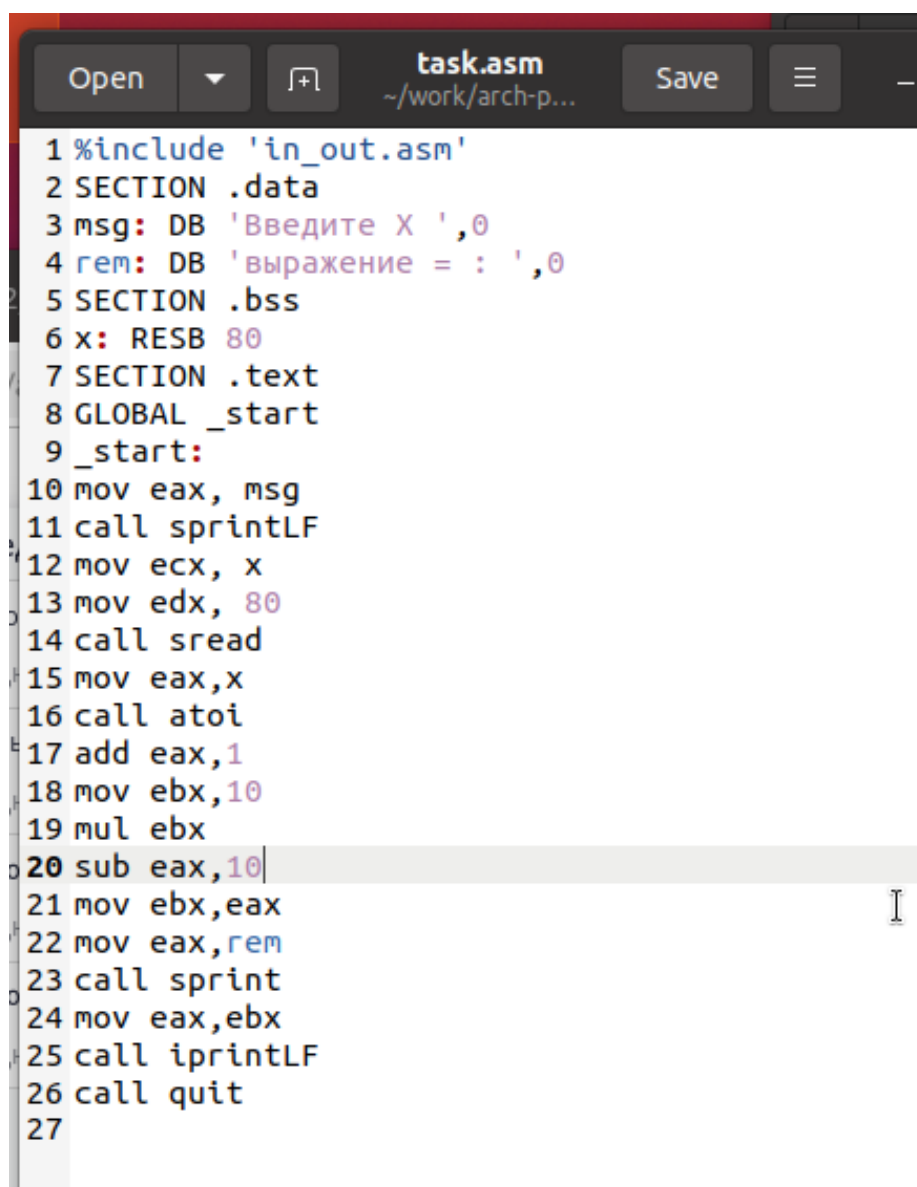
Инструкция `inc edx` увеличивает значение в регистре `edx` на 1, что соответствует формуле вычисления варианта.

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

Строка `mov eax, edx` перекладывает результат вычислений в регистр `eax`, а строка `call iprintLF` вызывает подпрограмму для вывода этого результата на экран.

2.2 Самостоятельное задание

Написана программа для вычисления выражения $y = f(x)$. Программа выводит формулу для вычисления, запрашивает ввод значения x , вычисляет выражение в зависимости от введенного x и выводит результат. В зависимости от лабораторного задания, был выбран вариант 11 — $10(x + 1) - 10$ для $x_1 = 1$, $x_2 = 7$ (рис. 2.16) (рис. 2.17).



```
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, 1
18 mov ebx, 10
19 mul ebx
20 sub eax, 10
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: Программа в файле task.asm

```
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$  
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ nasm -f elf task.asm  
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ld -m elf_i386 task.o -o task  
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ./task  
Введите X  
1  
выражение = : 10  
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$ ./task  
Введите X  
7  
выражение = : 70  
vboxuser@PavelChermashentsev:~/work/arch-pc/lab06$
```

Рис. 2.17: Запуск программы task.asm

3 Выводы

Изучили работу с арифметическими операциями.