

# Отчёта по лабораторной работе 6

## Освоение арифметических инструкций языка ассемблера NASM.

Останин Владислав Александрович НПМбв-01-21

### Содержание

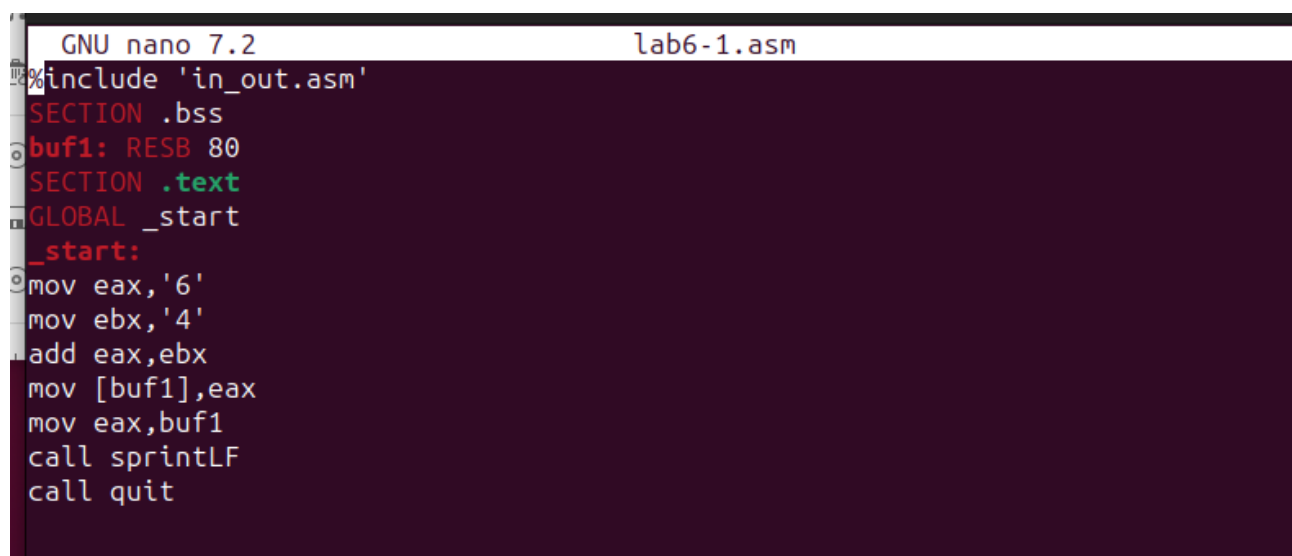
1	Цель работы .....	1
2	Выполнение лабораторной работы.....	1
3	Выводы .....	11

## 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

1. Создаем каталог для программ лабораторной работы № 6, переходим в него и создаем файл lab6-1.asm:
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр eax.



```
GNU nano 7.2 lab6-1.asm
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintLF
call quit
```

Figure 1: Пример программы

```
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./lab6-1

vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nasm -f elf lab6-1.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./lab6-1
j
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$
```

Figure 2: Работа программы

3. Далее изменим текст программы и вместо символов, запишем в регистры числа.

```
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintLF
call quit
```

Figure 3: Пример программы

```
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nano lab6-1.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nasm -f elf lab6-1.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./lab6-1

vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$
```

Figure 4: Работа программы

Никакой символ не виден, но он есть. Это возврат каретки LF.

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

```

GNU nano 7.2                                lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit

```

Figure 5: Пример программы

```

vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nano lab6-2.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nasm -f elf lab6-2.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./lab6-2
106
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$

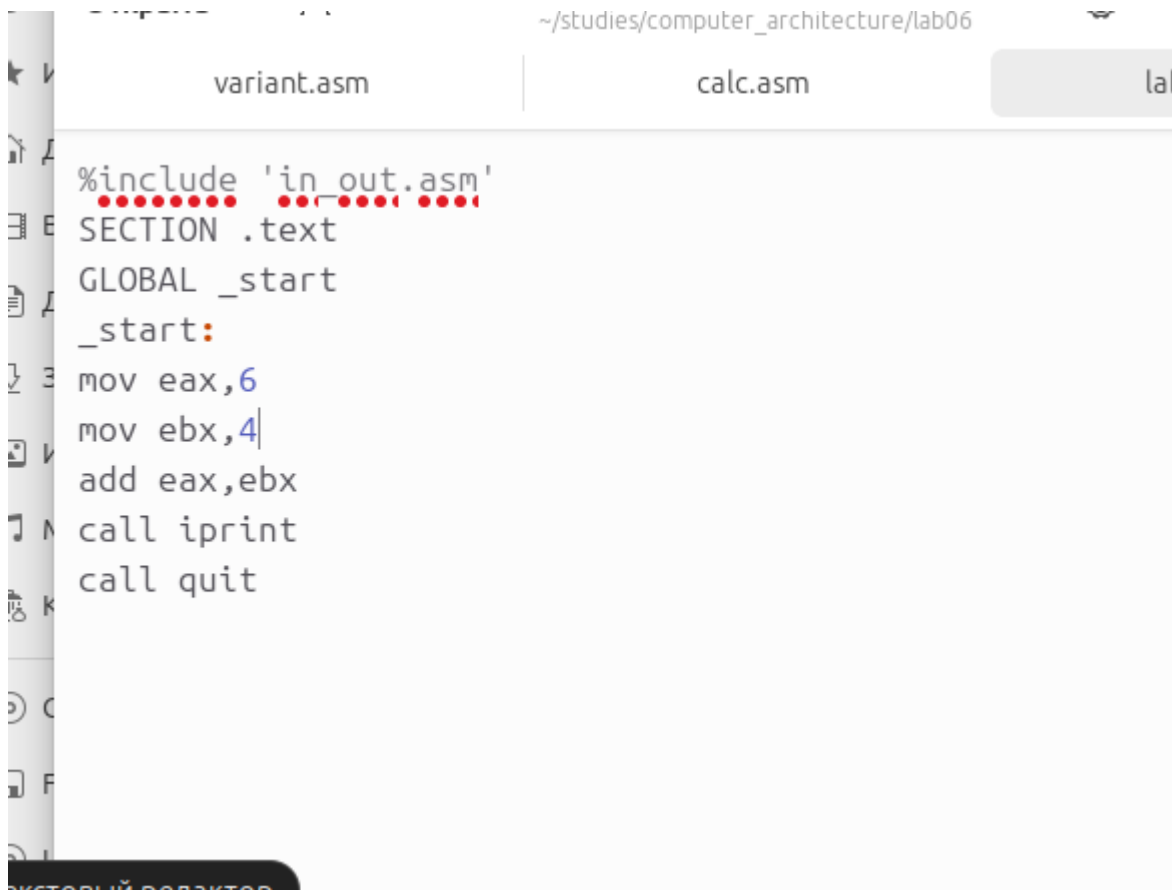
```

Figure 6: Работа программы

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

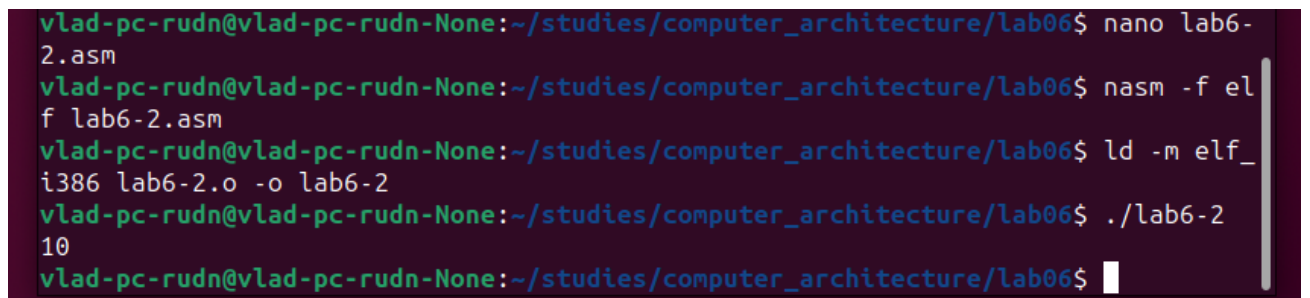
5. Аналогично предыдущему примеру изменим символы на числа.

Создайте исполняемый файл и запустите его. Какой результат будет получен при исполнении программы? – получили число 10



```
~/studies/computer_architecture/lab06  
variant.asm | calc.asm | lab6-2.asm  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprint  
call quit
```

Figure 7: Пример программы



```
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nano lab6-2.asm  
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nasm -f elf lab6-2.asm  
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2  
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./lab6-2  
10  
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$
```

Figure 8: Работа программы

Замените функцию `iprintLF` на `iprint`. Создайте исполняемый файл и запустите его. Чем отличается вывод функций `iprintLF` и `iprint`? - Вывод отличается что нет переноса строки. (рис. [9])

```
1000 1000 210 0 1000 1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./lab6-2
10
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nano lab6-
2.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nasm -f el
f lab6-2.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ld -m elf_
i386 lab6-2.o -o lab6-2
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./lab6-2
10vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$
```

Figure 9: Работа программы

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

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

.

```
vlad-pc-rudn@vlad-pc-rudn-None: ~/studies/computer_architecture/lab06
GNU nano 7.2 lab6-3.asm
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:

mov eax,5
mov ebx,2
mul ebx
add eax,3
xor edx,edx
mov ebx,3
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Figure 10: Пример программы

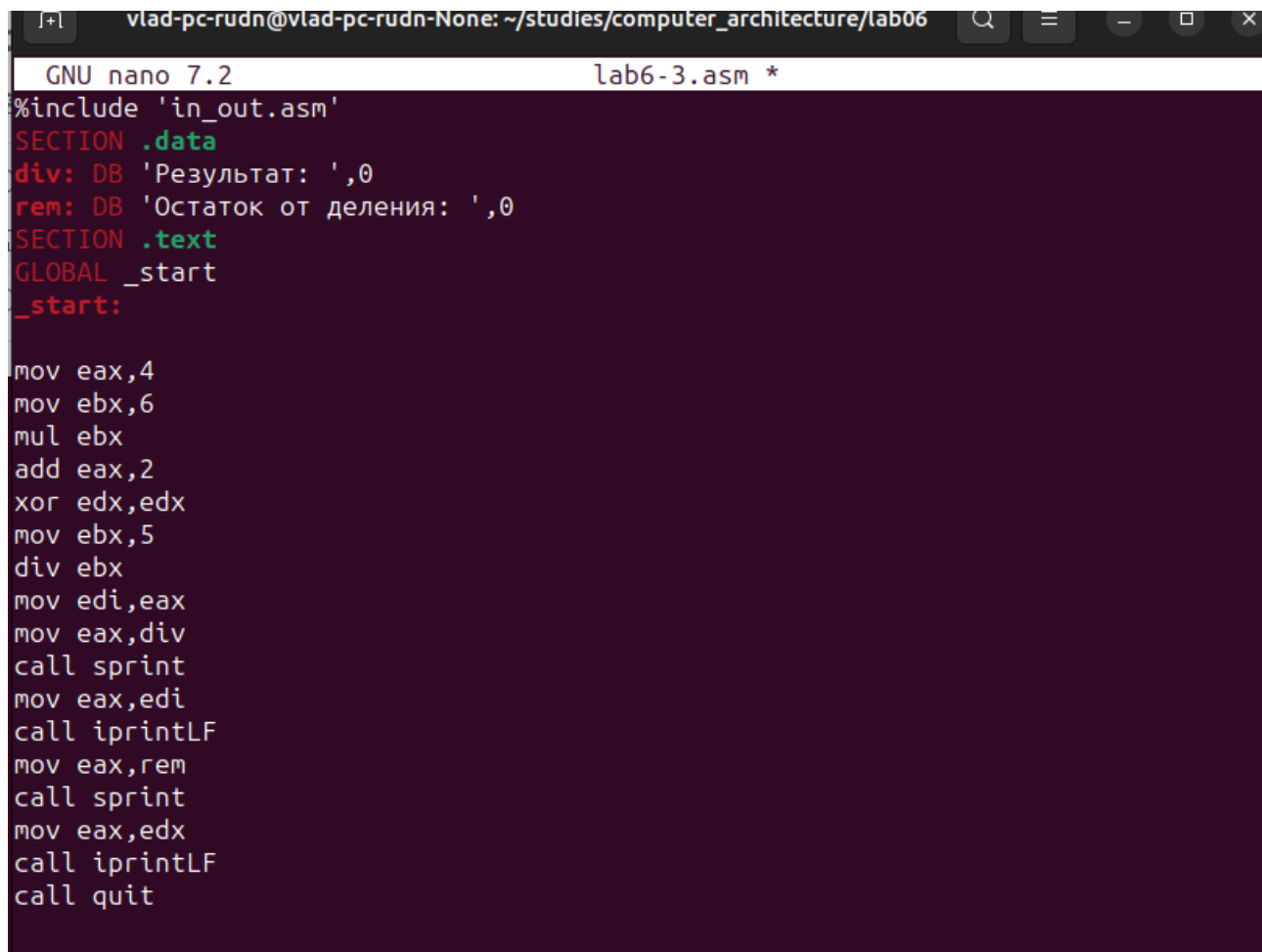
```
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./lab6-2
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nano lab6-3.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nasm -f elf lab6-3.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$
```

Figure 11: Работа программы

Измените текст программы для вычисления выражения

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

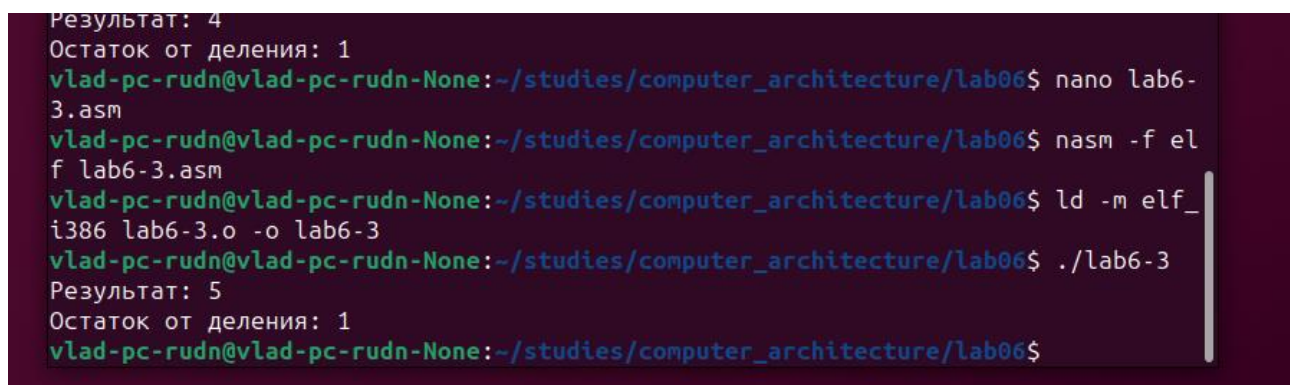
. Создайте исполняемый файл и проверьте его работу.



```
GNU nano 7.2 lab6-3.asm *
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:

mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Figure 12: Пример программы



```
Результат: 4
Остаток от деления: 1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nano lab6-3.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nasm -f elf lab6-3.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$
```

Figure 13: Работа программы

- В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета:

```

GNU nano 7.2                                variant.asm
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprintf
mov eax, edx
call iprintLF
call quit

```

Figure 14: Пример программы

```

vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nano varia
nt.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nasm -f el
f variant.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ld -m elf_
i386 variant.o -o variant
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./variant
Введите № студенческого билета:
1032210512
Ваш вариант: 13
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$

```

Figure 15: Работа программы

- Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант:?' – `mov eax,rem` – перекладывает в регистр значение переменной с фразой 'Ваш вариант:' `call sprintf` – вызов подпрограммы вывода строки



- Для чего используются следующие инструкции? `mov ecx, x` `mov edx, 80`  
`call sread`

Считывает значение студбилета в переменную X из консоли

- Для чего используется инструкция “`call atoi`”? - эта подпрограмма переводит введенные символы в числовой формат
- Какие строки листинга 6.4 отвечают за вычисления варианта?

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

- В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”?

1 байт AH 2 байта DX 4 байта EDX – наш случай

- Для чего используется инструкция “`inc edx`”? по формуле вычисления варианта нужно прибавить единицу
- Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений

`mov eax,edx` – результат перекладывается в регистр `eax` `call iprintLF` – 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 13 -  $(8x + 6) \cdot 10$

для  $x=1$  и 4

```
%include 'in_out.asm'
SECTION .data
msg db 'Введите X: ', 0
rem db 'Результат выражения (8x + 6) * 10 = : ', 0

SECTION .bss
x resb 80

SECTION .text
global _start

_start:
    mov eax, msg
    call sprintf
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi

    mov ebx, 8
    mul ebx
    add eax, 6
    mov ebx, 10
    mul ebx

    mov ebx, eax
    mov eax, rem
    call sprintf
    mov eax, ebx
    call iprintLF
    call quit
```

Figure 16: Пример программы

```
Выражение = : 1
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nano calc.
asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ nasm -f elf
f calc.asm
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ld -m elf_
i386 calc.o -o calc
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./calc
Введите X:
1
Результат выражения (8x + 6) * 10 = : 140
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$ ./calc
Введите X:
4
Результат выражения (8x + 6) * 10 = : 380
vlad-pc-rudn@vlad-pc-rudn-None:~/studies/computer_architecture/lab06$
```

add lab 2594dbb · 4 days ago Histor

Figure 17: Работа программы

### 3 Выводы

Изучили работу с арифметическими операциями