Отчёта по лабораторной работе № 6

Арифметические операции в NASM

Гущина Екатерина Антоновна

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

3 Задание для самостоятельной работы **19**

4 Выводы **22**

Список иллюстраций

2.1 Создание файла lab6-1.asm 6
2.2 Текст программы lab6-1.asm 7
2.3 Сборка и запуск программы lab6-1 7
2.4 Текст измененной программы lab6-1.asm 8
2.5 Сборка и запуск измененной программы lab6-1 8
2.6 Текст программы lab6-2.asm 9
2.7 Сборка и запуск программы lab6-2 9
2.8 Текст измененной программы lab6-2.asm 10
2.9 Сборка и запуск измененной программы lab6-2 10
2.10 Замена iprintLF на iprint 11
2.11 Текст программы lab6-3.asm 12
2.12 Сборка и запуск программы lab6-3 13
2.13 Текст измененной программы lab6-3.asm 14
2.14 Сборка и запуск программы lab6-3 15
2.15 Текст программы variant.asm 16
2.16 Сборка и запуск программы variant 17

3.1 Текст программы lab6-4.asm 20
3.2 Сборка и запуск программы lab6-4 21

Список таблиц

3.1 Вариант задания 19

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

2.1 Символьные и численные данные в NASM

Создадим каталог ~/work/arch-pc/lab06 для программам лабораторной работы № 6, перейдем в него и создадим файл lab6-1.asm:

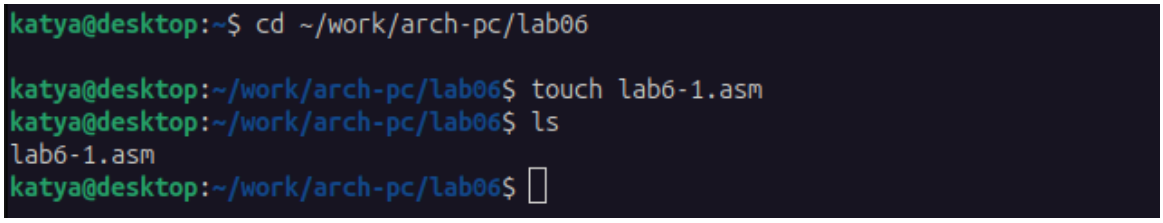

Рисунок 2.1: Создание файла lab6-1.asm

Введем в файл lab6-1.asm следующий текст программы:

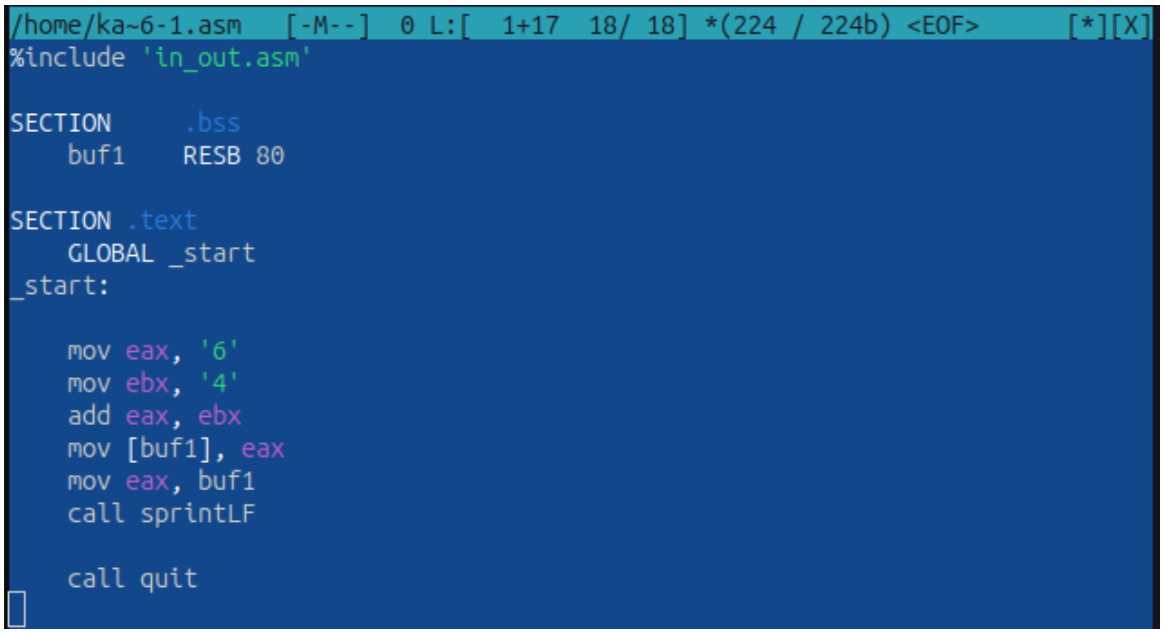

Рисунок 2.2: Текст программы lab6-1.asm

Программа включает файл in_out.asm, который мы использовали в лабораторной работе № 5. Скопируем его в текущую рабочую папку, создадим исполняемый файл lab6-1 и запустим:

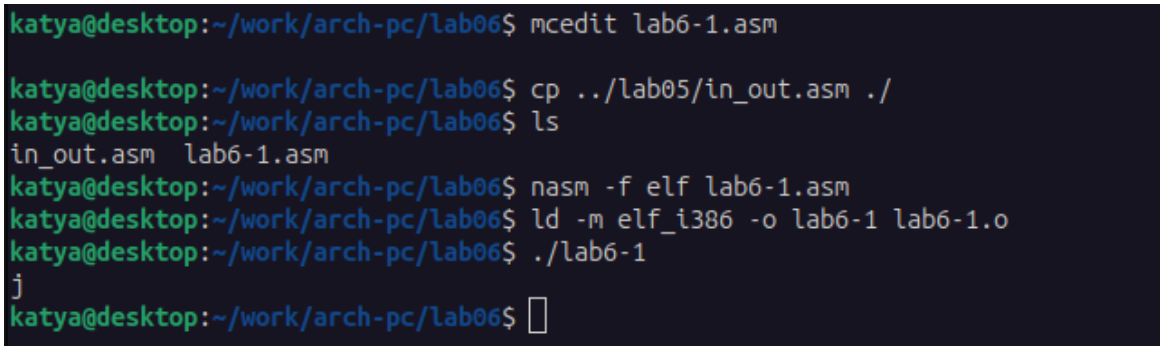

Рисунок 2.3: Сборка и запуск программы lab6-1

Результат вывода - символ „j“, так как в программе складывались значения ascii символов „6“ и „4“ (54 и 53 в десятичном представлении) и получили результат, равный 106 (ascii „j“).

Изменим текст программы lab6-1.asm и вместо символов, запишем в регистры числа:

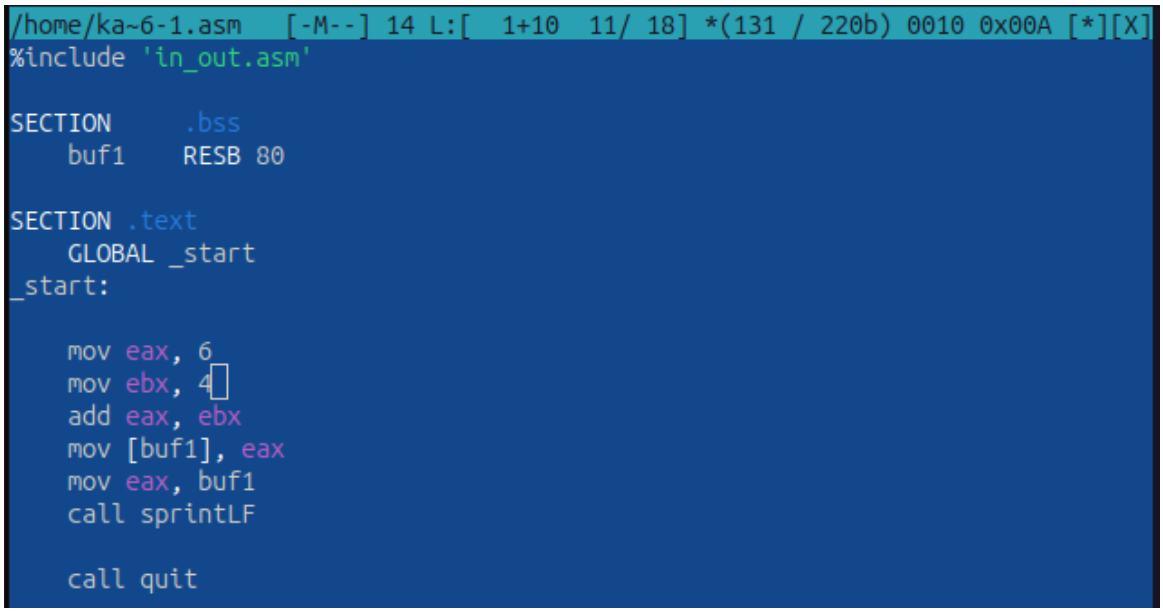

Рисунок 2.4: Текст измененной программы lab6-1.asm

Создадим исполняемый файл lab6-1 и запустим его:

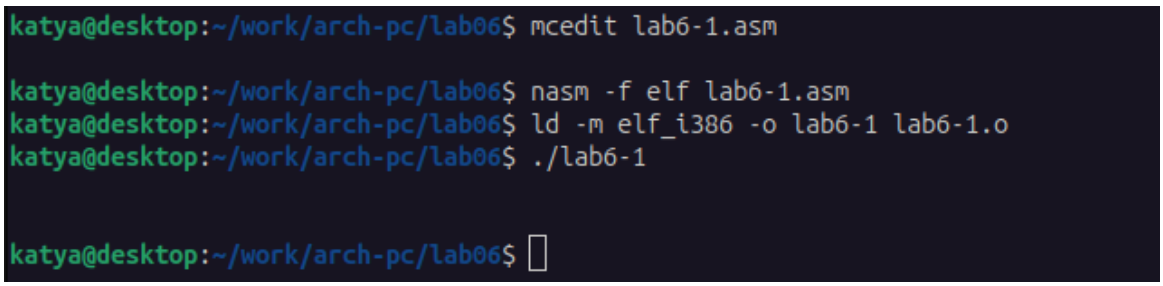

Рисунок 2.5: Сборка и запуск измененной программы lab6-1

В данном случае выводится символ с кодом 10 - это отображается при выводе на экран в виде перевода строки (добавлена новая строка).

Изменим текст программы с использованием подпрограмм из файла in_out.asm для преобразования ASCII символов в числа и обратно. Для этого создадим файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введите в него следующий текст программы:

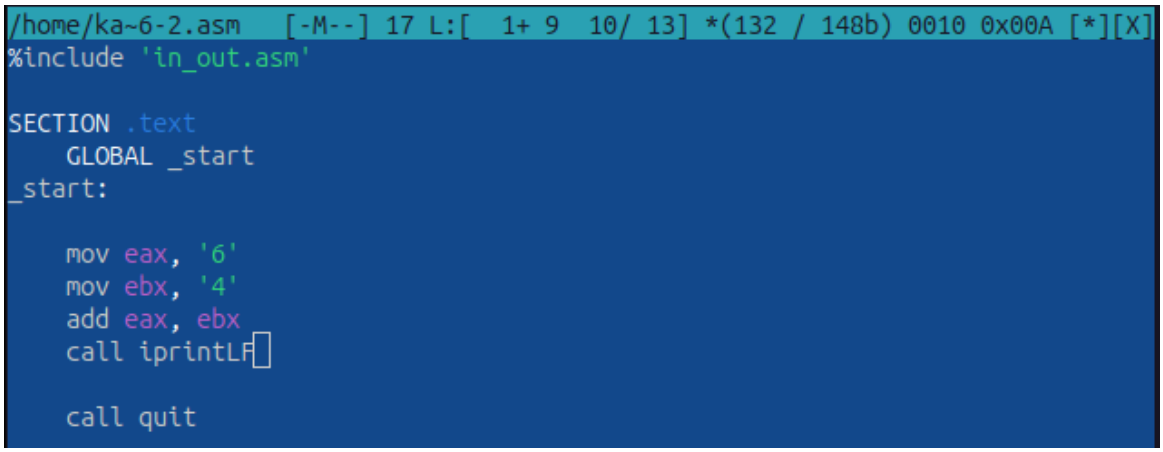

Рисунок 2.6: Текст программы lab6-2.asm

Создадим исполняемый файл и запустим его:

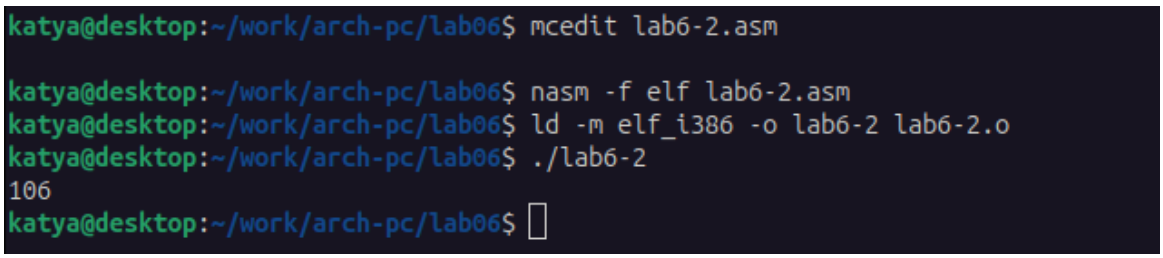

Рисунок 2.7: Сборка и запуск программы lab6-2

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 предыдущей версии программы, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим в файле lab6-2.asm символы на числа:

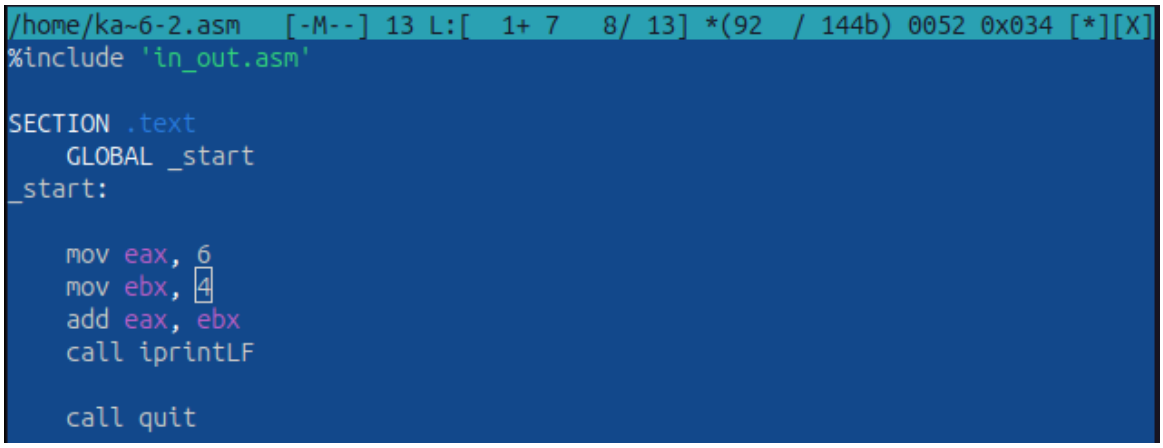

Рисунок 2.8: Текст измененной программы lab6-2.asm

Создадим исполняемый файл lab6-2 и запустим его:

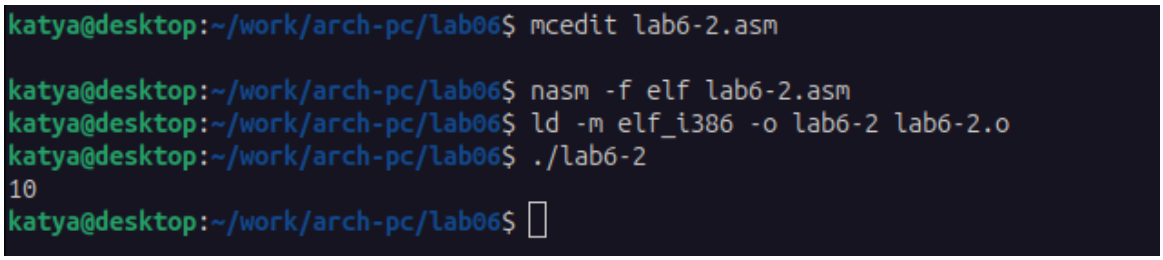

Рисунок 2.9: Сборка и запуск измененной программы lab6-2

В результате работы программы мы получим число 10. В данном случае программа складывает числа 4 и 6. Функция `iprintLF` выводит результат сложения в виде числа.

Заменим функцию `iprintLF` на `iprint`, создадим исполняемый файл lab6-2 и запустим:

```
katya@desktop:~/work/arch-pc/lab06$ mcedit lab6-2.asm

katya@desktop:~/work/arch-pc/lab06$ cat 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
katya@desktop:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
katya@desktop:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
katya@desktop:~/work/arch-pc/lab06$ ./lab6-2
10katya@desktop:~/work/arch-pc/lab06$
```

Рисунок 2.10: Замена iprintLF на iprint

В данном случае программа также выведет число 10, но без перевода на новую строку.

2.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3)/3$.

Создадим файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 и введем текст программы:

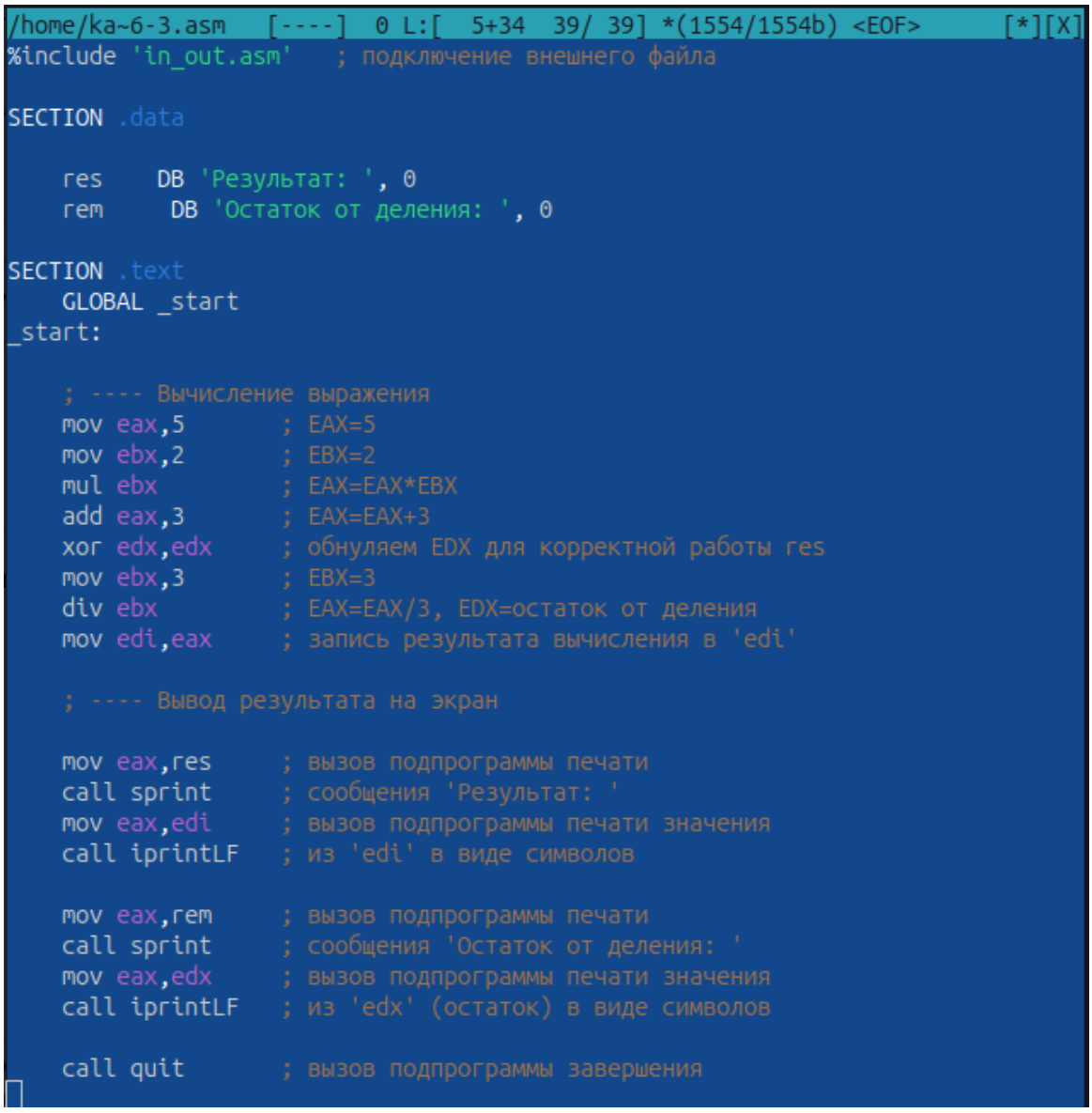

Рисунок 2.11: Текст программы lab6-3.asm

Создадим исполняемый файл lab6-3 и запустим его:

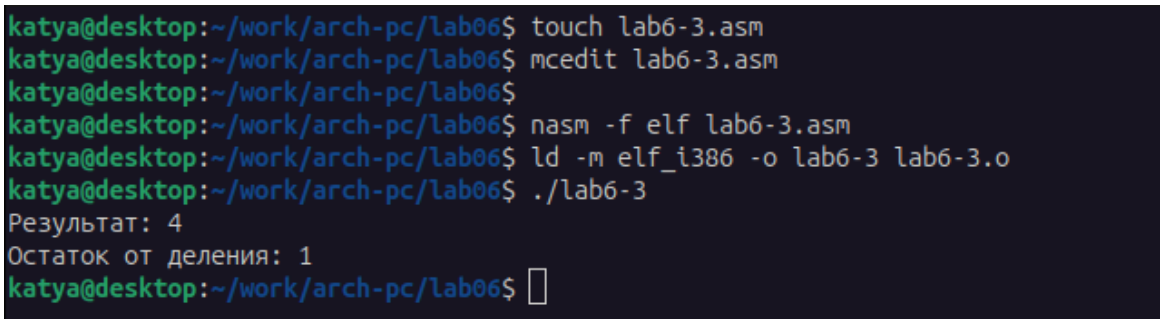

Рисунок 2.12: Сборка и запуск программы lab6-3

Результат деления равен 4, а остаток равен 1, что и ожидалось.

Измените текст программы lab6-3.asm для вычисления выражения $f(x) = (4 * 6 + 2)/5$:

```
/home/ka~6-3.asm   [-M--] 27 L:[  5+29  34/ 39] *(1259/1554b) 1077 0x435 [*][X]
%include 'in_out.asm'   ; подключение внешнего файла

SECTION .data

    res    DB 'Результат: ', 0
    rem     DB 'Остаток от деления: ', 0

SECTION .text
    GLOBAL _start
_start:

    ; ---- Вычисление выражения
    mov eax,4       ; EAX=4
    mov ebx,6       ; EBX=6
    mul ebx         ; EAX=EAX*EBX
    add eax,2       ; EAX=EAX+2
    xor edx,edx     ; обнуляем EDX для корректной работы res
    mov ebx,5       ; EBX=5
    div ebx         ; EAX=EAX/3, EDX=остаток от деления
    mov edi,eax     ; запись результата вычисления в 'edi'

    ; ---- Вывод результата на экран

    mov eax,res     ; вызов подпрограммы печати
    call sprint     ; сообщения 'Результат: '
    mov eax,edi     ; вызов подпрограммы печати значения
    call iprintLF   ; из 'edi' в виде символов

    mov eax,rem     ; вызов подпрограммы печати
    call sprint     ; сообщения 'Остаток от деления: '
    mov eax,edx     ; вызов подпрограммы печати значения
    call iprintLF   ; из 'edx' (остаток) в виде символов

    call quit       ; вызов подпрограммы завершения
```

Рисунок 2.13: Текст измененной программы lab6-3.asm

Создадим исполняемый файл lab6-3 и запустим его:

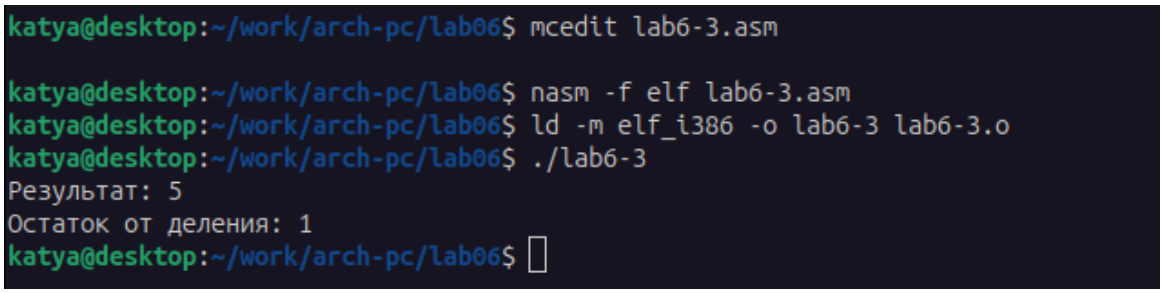

Рисунок 2.14: Сборка и запуск программы lab6-3

Результат деления равен 5, а остаток равен 1, что и ожидалось.

Напишем программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму: - вывести запрос на введение № студенческого билета; - вычислить номер варианта по формуле: $(S_n \mod 20) + 1$, где S_n – номер студенческого билета (В данном случае $a \mod b$ – это остаток от деления a на b); - вывести на экран номер варианта.

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. 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.

Создадим файл variant.asm в каталоге ~/work/arch-pc/lab06 и введем текст программы:

```
/home/ka~ant.asm   [----] 13 L:[  5+33  38/ 39] *(722 / 723b) 0010 0x00A [*][X]
%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 sprintLF

    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 sprint
    mov eax, edx
    call iprintLF

    call quit
```

Рисунок 2.15: Текст программы variant.asm

Создадим исполняемый файл variant и запустим его:

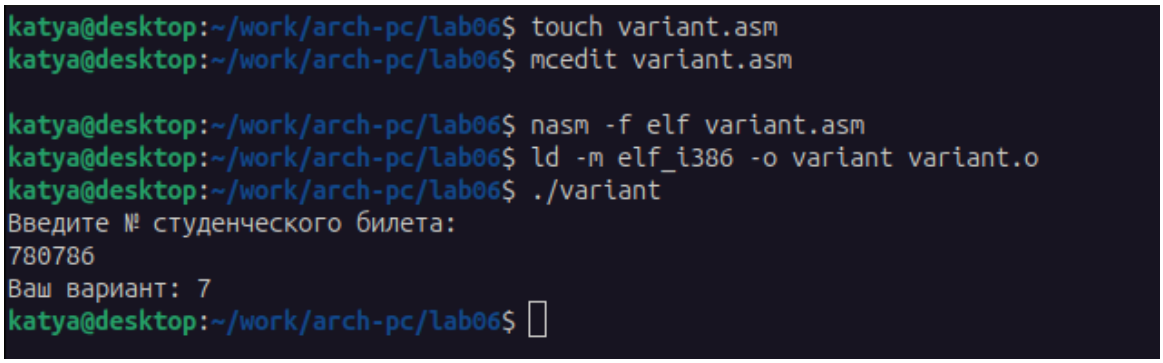

Рисунок 2.16: Сборка и запуск программы variant

При вводе номера студенческого билета получили номер варианта 7. Номер студенческого билета 780786: $(780786 \mod 20) + 1 = 6 + 1 = 7$, что совпадает с выводом программы.

Ответим на следующие вопросы:

1. Какие строки отвечают за вывод на экран сообщения „Ваш вариант:“?

За вывод на экран сообщения отвечают следующие строки:

```
mov eax, msg
call sprintfLF
```

2. Для чего используется следующие инструкции?

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

Данные инструкции используются для ввода строки (с номером студенческого билета).

3. Для чего используется инструкция `call atoi`?

Инструкция `call atoi` используется для преобразования строки, адрес начала которой передается в регистре eax, в число, которое возвращается в регистре eax.

4. Какие строки отвечают за вычисления варианта?

За вычисления варианта отвечают следующие строки:

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

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`?

Данная инструкция используется для инкремента (увеличения на 1) значения в регистре edx.

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

За вывод на экран результата вычислений отвечают следующие строки:

```
mov eax, rem
call sprint
mov eax, edx
call iprintLF
```

Первые две строки выводят поясняющее сообщение („Ваш вариант:“), последние две строки выводят число - результат вычисления (вариант).

3 Задание для самостоятельной работы

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 в соответствии с номером полученным при выполнении лабораторной работы.

Таблица 3.1: Вариант задания

Номер варианта	Выражение для $f(x)$	x_1	x_2
7	$5(x-1)^2$	3	5

Создадим новый файл lab6-4.asm в каталоге ~/work/arch-pc/lab06 и введем текст программы, вычисляющей значение функции из табл. 3.1:

```
/home/ka~6-4.asm   [-M--]  0 L:[  5+39  44/ 45] *(1532/1533b) 0010 0x00A [*][X]
%include 'in_out.asm'   ; подключение внешнего файла

SECTION .data

    msg     DB 'Введите x: ', 0
    res     DB 'Результат: ', 0

SECTION .bss
    buf     RESB 10

SECTION .text
    GLOBAL _start
_start:

    mov eax, msg    ; вызов подпрограммы печати
    call sprint     ; сообщения 'Введите x: '

    mov ecx, buf    ; вызов подпрограммы чтения
    mov edx, 10     ; пользовательского ввода
    call sread      ; в буфер buf

    mov eax, buf    ; вызов подпрограммы преобразования
    call atoi       ; ASCII кода в число (в EAX)

    ; ---- Вычисление выражения
    dec eax         ; EAX=EAX-1
    mul eax         ; EAX=EAX*EAX
    mov ebx,5       ; EBX=5
    mul ebx         ; EAX=EAX*EBX
    mov edi,eax     ; запись результата вычисления в 'edi'

    ; ---- Вывод результата на экран

    mov eax,res     ; вызов подпрограммы печати
    call sprint     ; сообщения 'Результат: '
    mov eax,edi     ; вызов подпрограммы печати значения
    call iprintLF   ; из 'edi' в виде символов

    call quit       ; вызов подпрограммы завершения
```

Рисунок 3.1: Текст программы lab6-4.asm

Создадим исполняемый файл lab6-4 и запустим его со значениями x_1 и x_2 из табл. 3.1:

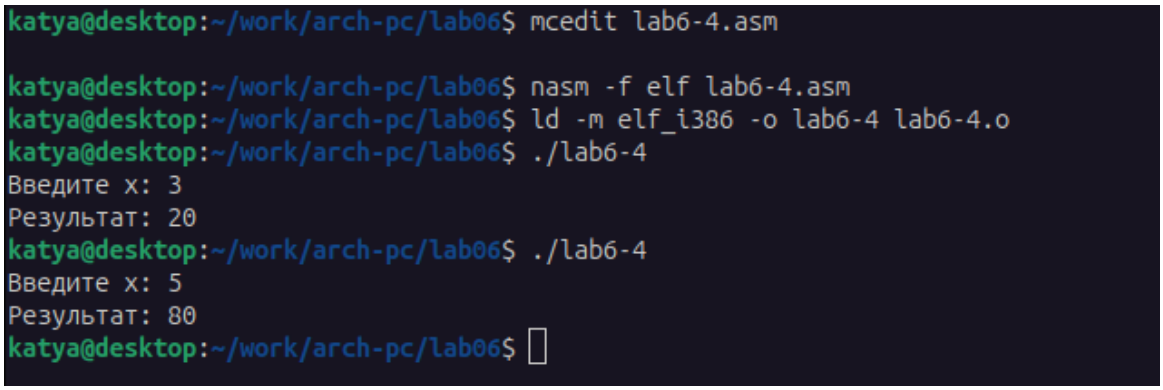

Рисунок 3.2: Сборка и запуск программы lab6-4

Для $x = 3$ имеем $5*(3-1)^2 = 5*4 = 20$. Для $x = 5$ имеем $5*(5-1)^2 = 5*16 = 80$. Аналитические результаты совпадают с результатами работы программы.

4 Выводы

Освоили арифметические инструкции языка ассемблера NASM.