**Лабораторная Работа 7**

### Освоение арифметических инструкций языка ассемблера NASM.

## Герра Гарсия Максимиано Антонио

**Содержание**

1. [Цель работы](#_bookmark0) 5
2. [Задание](#_bookmark1) 6
3. [Теоретическое введение](#_bookmark2) 7
4. [Выполнение лабораторной работы](#_bookmark3) 9
5. [Выводы](#_bookmark19) 21

[Список литературы](#_bookmark20) 22

**Список иллюстраций**

* 1. [Пример программы](#_bookmark4) 10
  2. [Работа программы](#_bookmark5) 11
  3. [Пример программы](#_bookmark6) 12
  4. [Работа программы](#_bookmark7) 12
  5. [Пример программы](#_bookmark8) 13
  6. [Работа программы](#_bookmark9) 13
  7. [Пример программы](#_bookmark10) 14
  8. [Работа программы](#_bookmark11) 14
  9. [Работа программы](#_bookmark12) 15
  10. [Пример программы](#_bookmark13) 16
  11. [Работа программы](#_bookmark14) 16
  12. [Пример программы](#_bookmark15) 17
  13. [Работа программы](#_bookmark16) 18
  14. [Пример программы](#_bookmark17) 20
  15. [Работа программы](#_bookmark18) 2

# Цель работы

Целью работы является освоение арифметических инструкций языка ассем- блера NASM.

# Задание

* 1. Изучите примеры программ.
  2. Напишите программу вычисления выражения в соответсвии с вариантом.
  3. Загрузите файлы на GitHub.

# Теоретическое введение

В основном наборе инструкций входят разные вариации четырех арифмети- ческих действий: сложение, вычитание, умножение, деление. Важно помнить, что в результате арифметических действий меняются некоторые биты регистра флагов, что позволяет выполнять команду условного перехода, т.е. разветвлять программу на основе результат операции. Замечу, что для команд с ложения и вычитания справедливыми являются отмеченное выше для операндов коман- ды mov . К командам сложения можно отнести: add – обычное сложение, adc – сложение с добавлением результату флага переноса в качестве единицы (если флаг равен нулю, то команда эквивалентна команде add ), xadd – сложение, с предварительным обменом данных между операндами, inc – прибавление еди- ницы к содержимому операнда. Несколько примеров: add %rbx , dt (или addq, dt, где четко указано, что складываются 64-битовые величины) – к содержимому области памяти dt добавляется содержимое регистра rbx и результат помещается в dt ; adc %rdx , %rdx – удвоение содержимого регистра rdx плюс добавление значения флага переноса; incl ll – увеличение на единицу содержимого памяти по адресу ll. При этом явно указывается, что операнд имеет размер 32 бита (d - dword).

К командам вычитания можно отнести следующие инструкции процессора x86-64: sub – обычное вычитание, sbb - вычитание из результата флага переноса в качестве единицы (если флаг равен нулю, то команда эквивалентна sub ), dec

* вычитание единицы из результата, neg – вычитание значения операнда из 0 . Несколько примеров: sub %rax , ll - из содержимого ll вычитается содержимое

регистра rax (или явно subq %rax , ll, где указывается, что операнды имеют 64- размер), и результат помещается в ll; subw go, %ax – вычитание из содержимого ax числа по адресу go, результат помещается в ax ; sbb %rdx , %rax – вычитание с дополнительным вычитанием флага переноса (из числа в rax вычитается число в rdx и результат в rax); decb l – вычитание единицы из байта, расположенного по адресу l . Следует отметить еще специальную команду cmp , которая во всем похо- жа на команду sub, кроме одного – результат вычитания никуда не помещается. Инструкция используется специально, для сравнения операндов.

Две основные команды умножения: mul – умножение беззнаковых чисел, imul

* умножение знаковых чисел. Команда содержит один операнд – регистр или адрес памяти. В зависимости от размера операнда данные помещаются: в ax , dx : ax , edx : eax , rdx : rax . Например: mull ll – содержимое памяти с адресом ll будет умножено на содержимое eax (не забываем о суффиксе l), а результат отправлен в пару регистров edx : eax; mul %dl – умножить содержимое регистра dl на содержимое регистра al , а результат положить в ax ; mul %r8 – умножить содержимое регистра r8 на содержимое регистра rax , а результат положить в пару регистров rdx : rax.

Для деления (целого) также предусмотрены две команды: div – беззнаковое деление, idiv – знаковое деление. Инструкция также имеет один операнд - дели- тель. В зависимости от его размера результат помещается: al – результат деления, ah – остаток от деления; ax – результат деления, dx – остаток от деления; eax

* результат деления, edx – остаток от деления; rax – результат деления, rdx – остаток от деления. Приведем примеры: divl dv – содержимое edx : eax делится на делитель, находящийся в памяти по адресу dv и результат деления помещается в eax , остаток в edx ; div %rsi – содержимое rdx : rax делится на содержимое rsi , результат помещается в rax , остаток в rdx .

# Выполнение лабораторной работы

* 1. Создайте каталог для программам лабораторной работы № 6, перейдите в него и создайте файл lab7-1.asm:
  2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр eax. (рис. [4.1,](#_bookmark4) [4.2)](#_bookmark5)

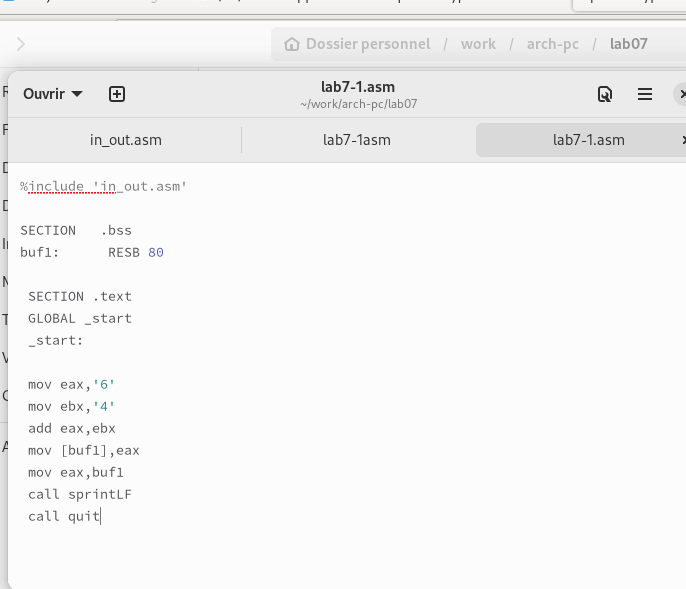


Рис. 4.1: Пример программы

Logo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

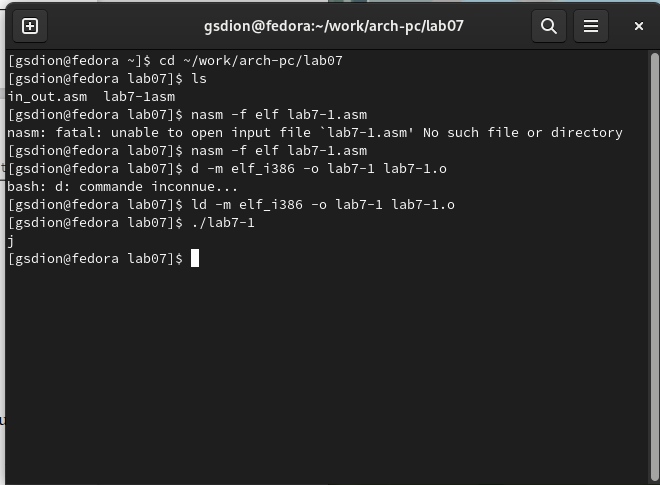
Description automatically generated

Рис. 4.2: Работа программы

* 1. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы (Листинг 1) следующим образом: (рис. [4.3,](#_bookmark6) [4.4)](#_bookmark7)

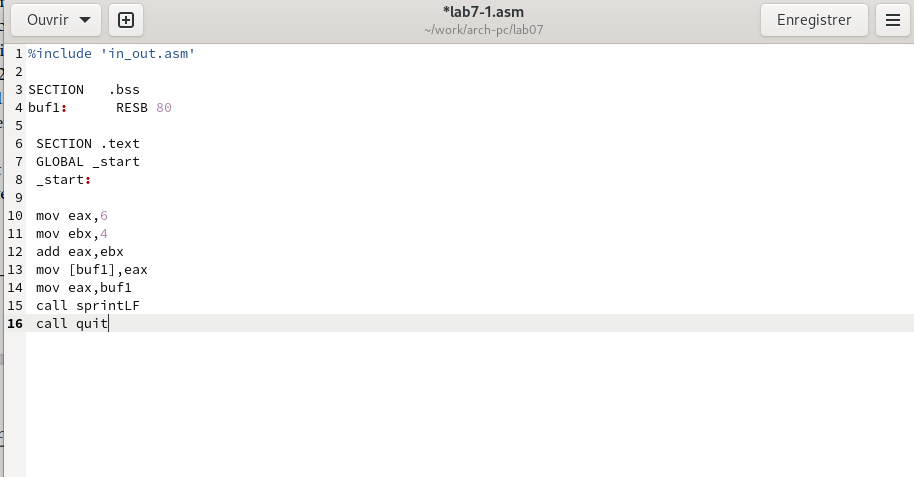


Рис. 4.3: Пример программы

Logo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

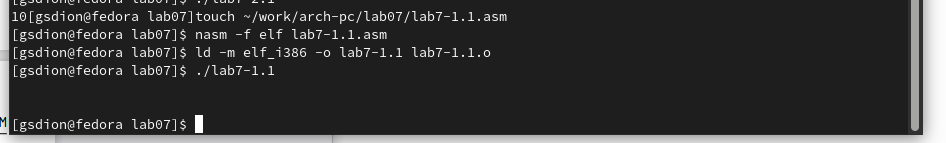
Description automatically generated

Рис. 4.4: Работа программы

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

* 1. Как отмечалось выше,для работы с числами в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Пре- образуем текст программы из Листинга 7.1 с использованием этих функций. (рис. [4.5,](#_bookmark8) [4.6)](#_bookmark9)

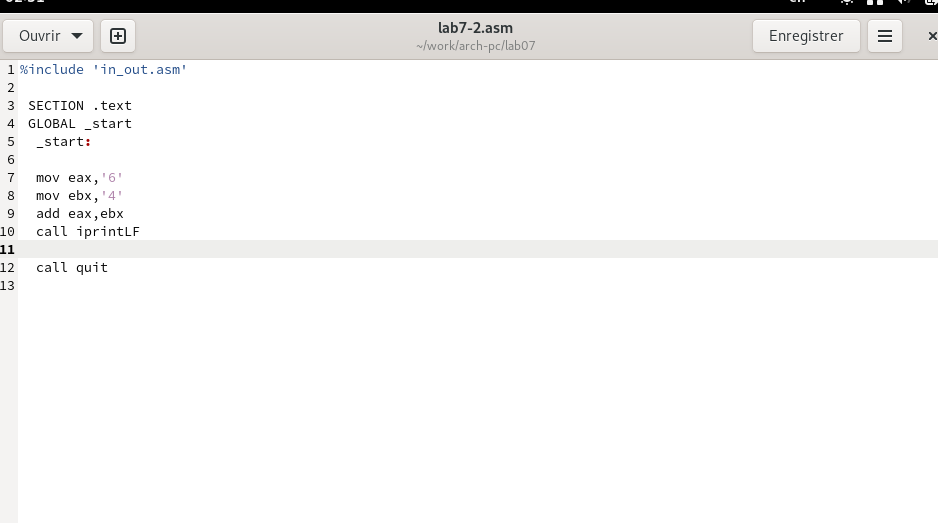


Рис. 4.5: Пример программы

Logo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

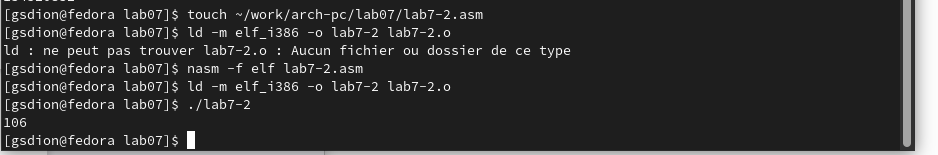
Description automatically generated

Рис. 4.6: 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 отличии от программы из листинга 7.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

* 1. Аналогично предыдущему примеру изменим символы на числа. (рис. [4.7,](#_bookmark10) [4.8)](#_bookmark11)

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

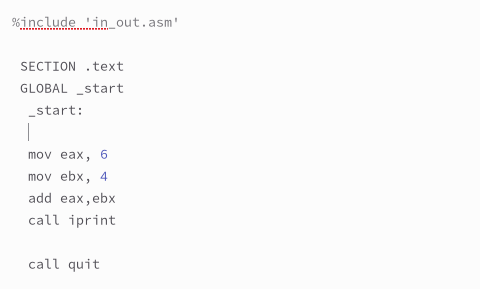


Рис. 4.7: Пример программы

Logo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

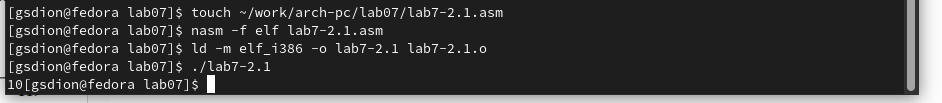
Description automatically generated

Рис. 4.8: 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. **??**)

Работа программы

* 1. В качестве примера выполнения арифметических операций в NASM приве- дем программу вычисления арифметического выражения

#### 𝑓(𝑥) = (5 ∗ 2 + 3)/3.

. (рис. **??**, рис. [4.9)](#_bookmark12)

Пример программы

Logo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

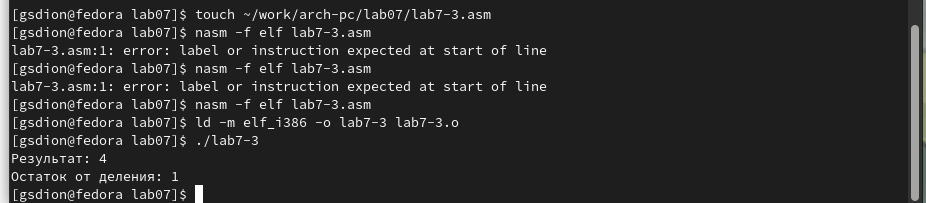
Description automatically generated

Рис. 4.9: Работа программы Измените текст программы для вычисления выражения

#### 𝑓(𝑥) = (4 ∗ 6 + 2)/5

. Создайте исполняемый файл и проверьте его работу. (рис. [4.10,](#_bookmark13) рис. [4.11)](#_bookmark14)

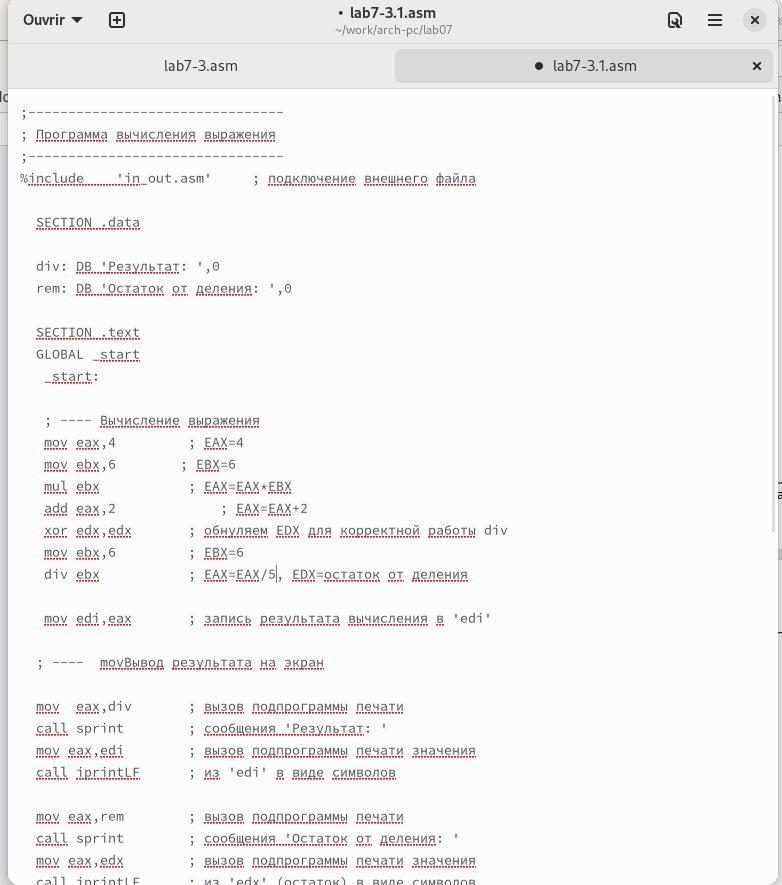


Рис. 4.10: Пример программы

Logo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

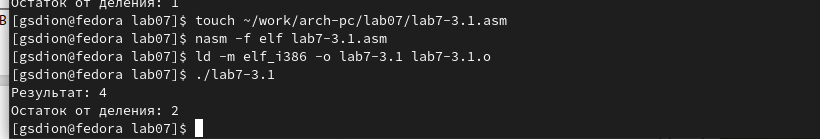
Description automatically generated

Рис. 4.11: Работа программы

* 1. В качестве другого примера рассмотрим программу вычисления варианта

задания по номеру студенческого билета, работающую по следующему алгоритму: (рис. [4.12,](#_bookmark15) рис. [4.13)](#_bookmark16)

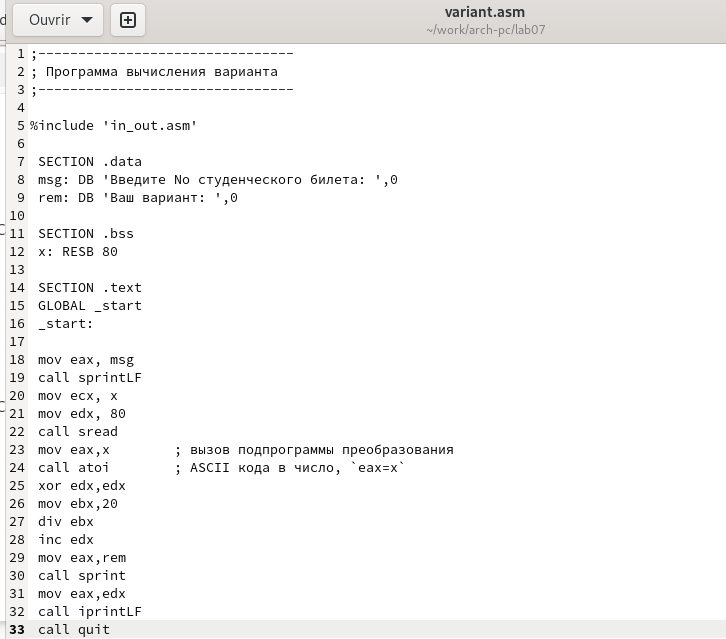


Рис. 4.12: Пример программы

Logo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

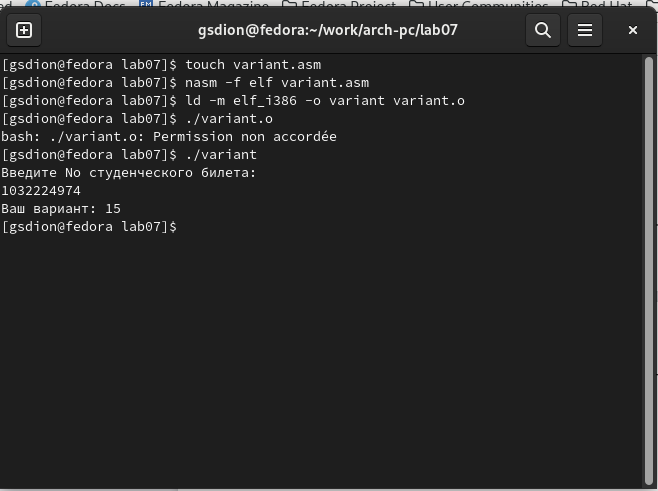
Description automatically generated

Рис. 4.13: Работа программы

* Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? – mov eax,rem – перекладывает в регистр значение переменной с фразой ‘Ваш вариант:’ call sprint – вызов подпрограммы вывода строки
* Для чего используется следующие инструкции? nasm mov ecx, x mov edx, 80 call sread

Считывает значение студбилета в переменную Х из консоли

* Для чего используется инструкция “call atoi”? - эта подпрограмма переводит введенные символы в числовой формат
* Какие строки листинга 7.4 отвечают за вычисления варианта? xor edx,edx mov ebx,20 div ebx
* В какой регистр записывается остаток от деления при выполнении инструк- ции “div ebx”? 1 байт AH 2 байта DX 4 байта EDX – наш случай
* Для чего используется инструкция “inc edx”? по формуле вычисления вари- анта нужно прибавить единицу
* Какие строки листинга 7.4 отвечают за вывод на экран результата вычисле- ний mov eax,edx – результат перекладывается в регистр eax call iprintLF – вызов подпрограммы вывода
  1. 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 лабора- торной работы. Создайте исполняемый файл и проверьте его работу для значений x1 и x2 из 6.3. (рис. [4.14,](#_bookmark17) рис. [4.15)](#_bookmark18)

Получили вариант 15 -

#### (5 + 𝑥)2 − 3

для х=5 и 1

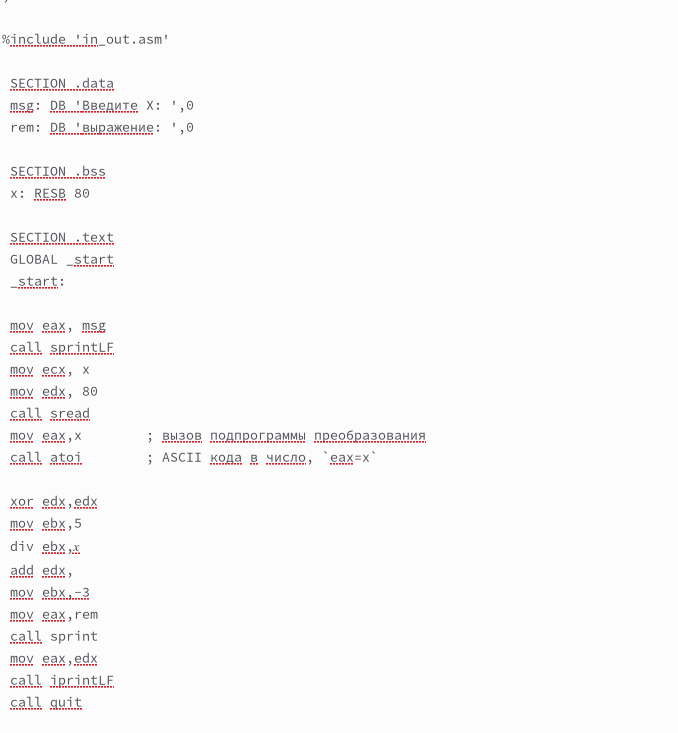


Рис. 4.14: Пример программы

Logo

Description automatically generatedLogo

Description automatically generatedLogo

Description automatically generatedLogo

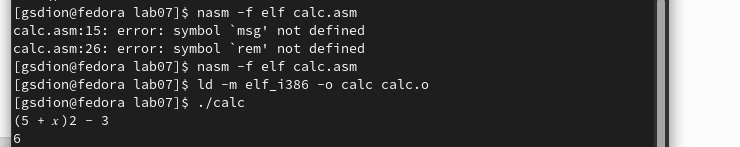
Description automatically generated

Рис. 4.15: Работа программы

# Выводы

Изучили работу с арифметическими операциями

# Список литературы

* 1. [Расширенный ассемблер: NASM](https://www.opennet.ru/docs/RUS/nasm/)
  2. [MASM, TASM, FASM, NASM под Windows и Linux](https://habr.com/ru/post/326078/)