Лабораторная работа №6

Седохин Даниил Алексеевич

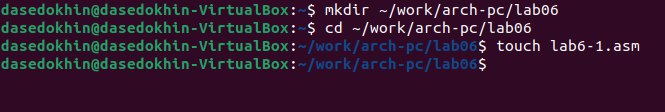
Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

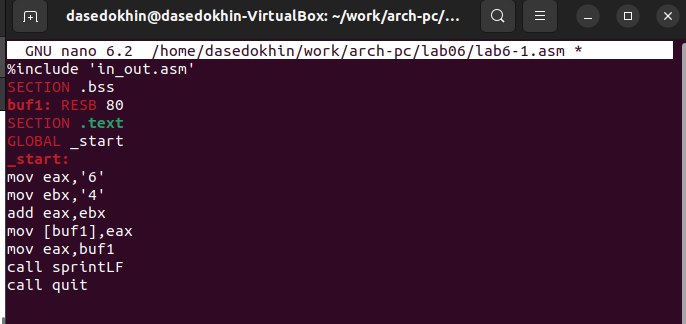
# 2 Выполнение лабораторной работы

1. Создаём каталог для программам лабораторной работы № 6, перейдём в него и создадим файл lab6-1.asm (рис. ??).



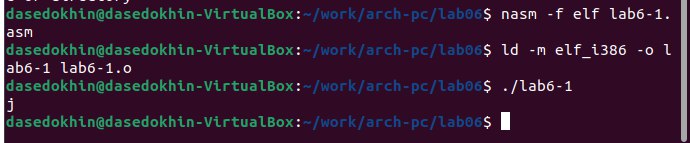
Создание каталога для лабораторной работы №6 и файла lab6-1.asm

1. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax. Введём в файл lab6-1.asm текст программы из листинга 6.1. (рис. ??).  
   В данной программе в регистр eax записывается символ 6 (mov eax,‘6’), в регистр ebx символ 4 (mov ebx,‘4’). Далее к значению в регистре eax прибавляем значение регистра ebx (add eax,ebx, результат сложения запишется в регистр eax). Далее выводим результат. Так как для работы функции sprintLF в регистр eax должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра eax в переменную buf1 (mov [buf1],eax), а затем запишем адрес переменной buf1 в регистр eax (mov eax,buf1) и вызовем функцию sprintLF.



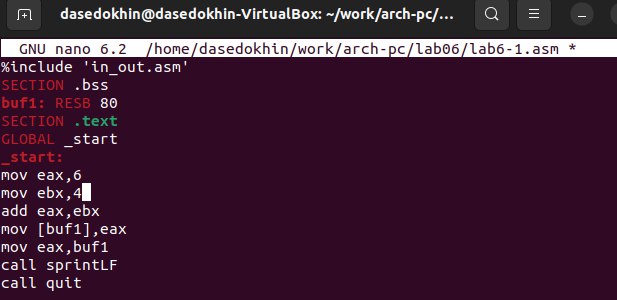
Ввод в файл lab6-1.asm текст программы из листинга 6.1

1. Создадим исполняемый файл и проверим его. (рис. ??).  
   Пояснение: В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда add eax,ebx запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j.



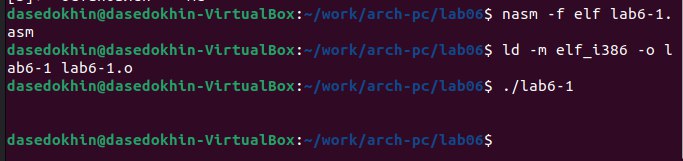
Создание и проверка исполняемого файла

1. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы (Листинг 6.1). (рис. ??).



Редактирование текста программы из листинга 6.1

Создадим исполняемый файл и запустим его. (рис. ??).



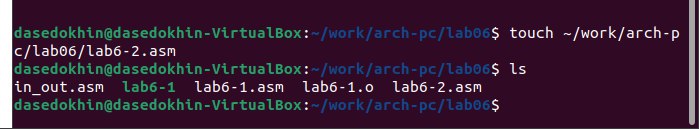
Создание и проверка исполняемого файла

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определим какому символу соответствует код 10. (рис. ??).  
Данный символ не отображается на экране.

Таблица ASCII

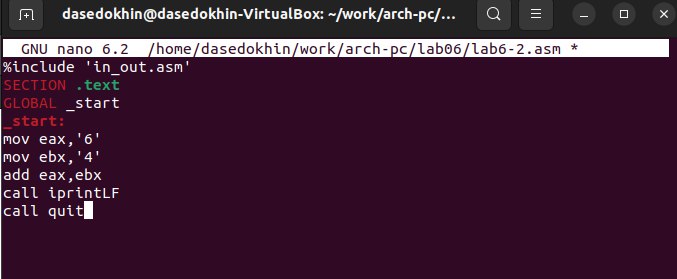
Таблица ASCII

1. 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 из Листинга 6.1 с использованием этих функций. Создадим файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 . (рис. ??).



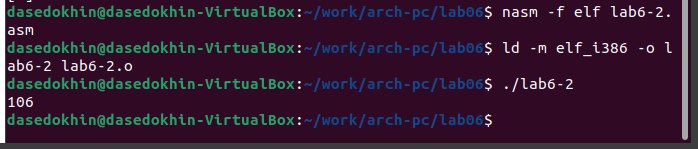
Создание файла lab6-2.asm

Введём в него текст программы из листинга 6.2 (рис. ??).



Редактирование файла lab6-2.asm из листинга 6.2

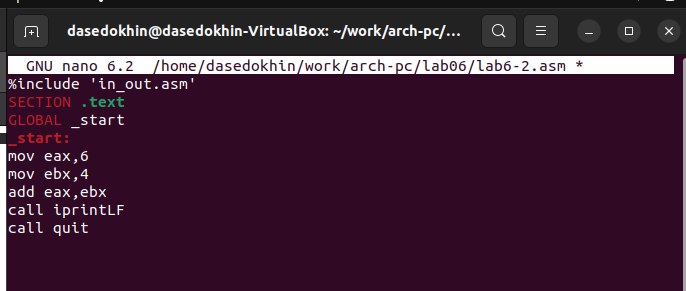
Создадим исполняемый файл и запустим его. (рис. ??)



Создание и проверка исполняемого файла

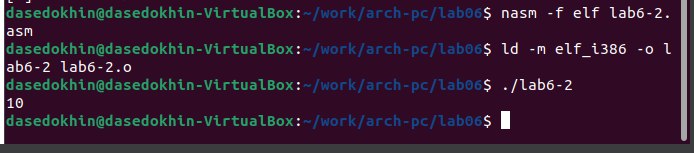
В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличии от программы из листинга 6.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

1. Аналогично предыдущему примеру изменим символы на числа, Заменив строки. (рис. ??).



Редактирование файла lab6-2.asm

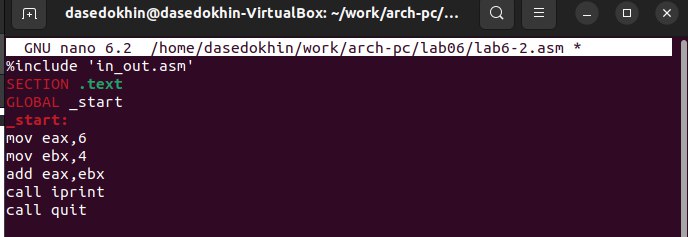
Создадим исполняемый файл и запустим его. (рис. ??).



Создание и проверка исполняемого файла

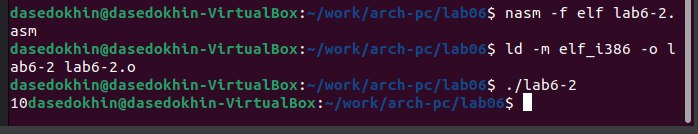
В результате при запуске исполняемого файла мы получаем число 10.

Заменим функцию iprintLF на iprint. (рис. ??).



Замена функции iprintLF на iprint

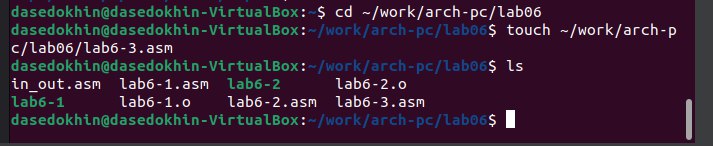
Создадим исполняемый файл и запустим его. (рис. ??).



Создание и проверка файла исполняемого файла

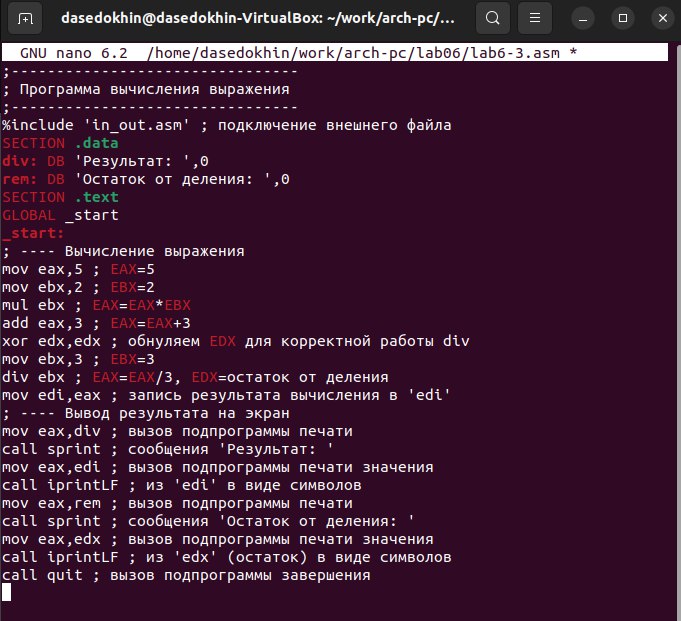
Вывод функции iprintLF отличается от функции inprint тем, что выводимое число не записывается на отдельной строке при использовании функции inprint.

1. 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 (рис. ??).



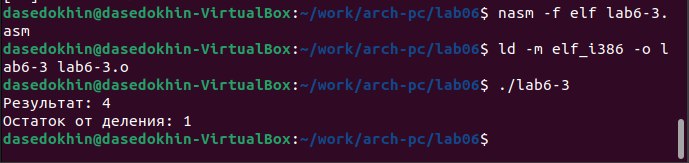
Создание файла lab6-3.asm

1. Введём в lab6-3.asm текст из листинга 6.3 (рис. ??).



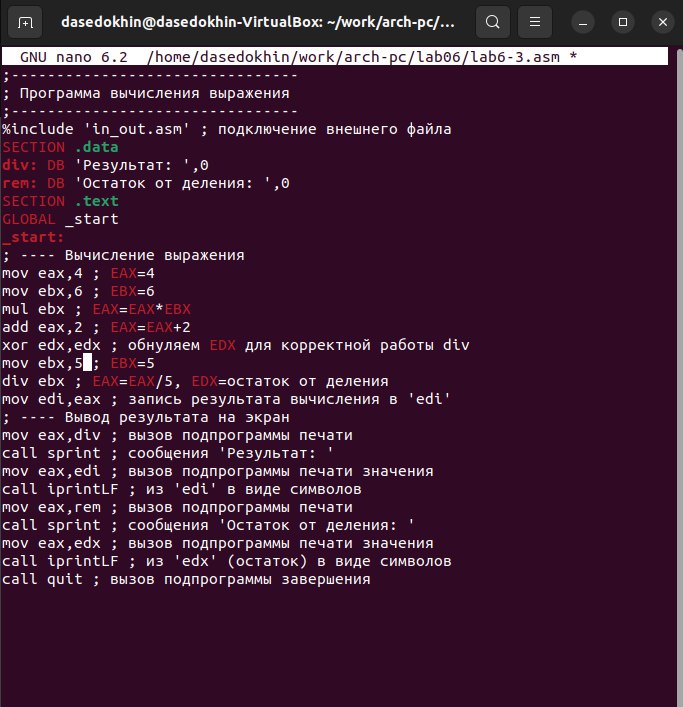
Ввод листинга 6.3 в файл lab6-3.asm

1. Создадим исполняемый файл и запустим его. (рис. ??).



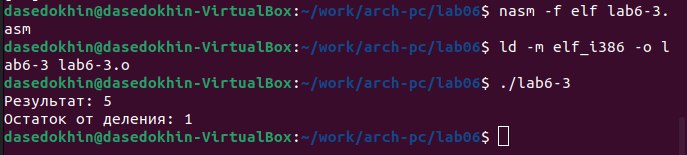
Создание и проверка исполняемого файла

1. Изменим текст программы для вычисления выражения 𝑓(𝑥) = (4 ∗ 6 + 2)/5. (рис. ??).



Редактирование текста программы lab6-3.asm

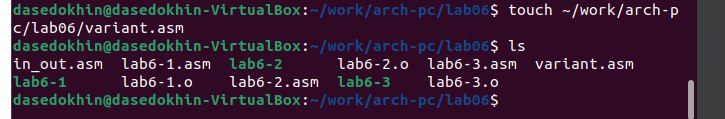
Создадим исполняемый файл и проверим его работу. (рис. ??).



Создание и проверка исполняемого файла

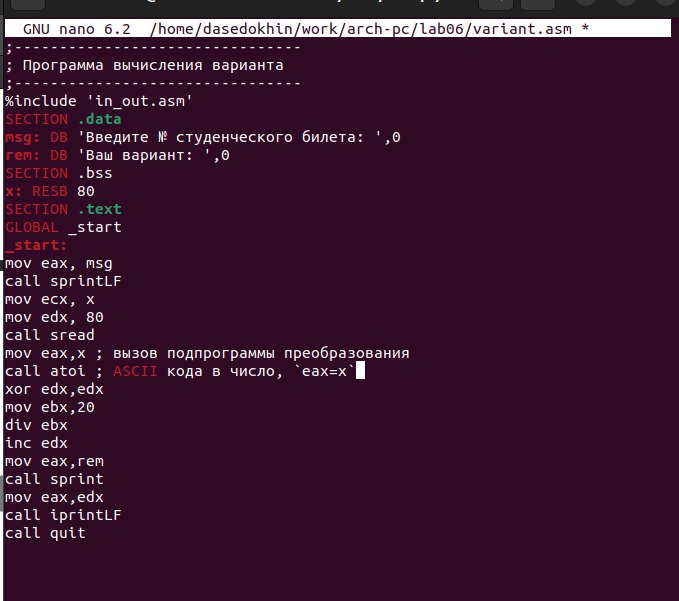
1. В качестве другого примера рассмотрим программу вычисления варианта задания пономеру студенческого билета, работающую по следующему алгоритму:

* вывести запрос на введение № студенческого билета
* вычислить номер варианта по формуле: (𝑆𝑛 mod 20) + 1, где 𝑆𝑛 – 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. Как отмечалось выше 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 (рис. ??).



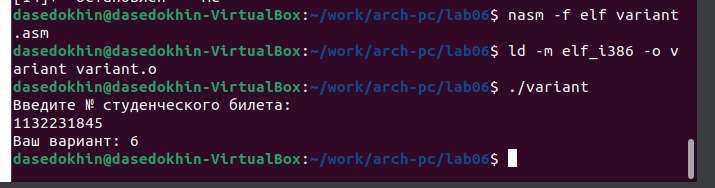
Создание файла variant.asm

Введём в файл variant.asm текст из листинга 6.4 (рис. ??).



Ввод листинга 6.4 в файл variant.asm

Создадим исполняемый файл и проверим его. (рис. ??).



Создание и проверка исполняемого файла

При аналитическом вычислении номера варианта результат совпал с результатом работы программы.

## 2.1 Ответы на вопросы:

1. За вывод строки ***Ваш вариант:*** отвечают 2 строки: **mov eax,rem** **call sprint**
2. Объявляется перемнная, под неё выделяется 80 байт, а затем вызывается функция которая отвечает за ввод строки пользователем в ранее обяъявленную переменную.
3. Это функция преобразования ascii-код символа в целое число
4. За вычисление варианта отвечает следующая последовательность строк - **xor edx,edx** **mov ebx,20** **div ebx** **inc edx**
5. Запись происходит в регистр edx.
6. Данная инструкция пребовляет 1 к значению регистра edx.
7. Строки - **mov eax,edx** **call iprintLF**

отвечают за вывод результата вычеслений в консоль

# 3 Задание для самостоятельной работы

Загрузим файлы на github. (рис. ??).

|  |
| --- |
| Загрузка файлов на github |

Загрузка файлов на github

# 4 Выводы

Я освоил арифметические инструкции языка ассемблера NASM.