Отчёт по лабораторной работе №7

Уткина Алина Дмитриевна

Содержание

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Выполнение лабораторной работы

## 2.1 Символьные строки и численные данные в NASM

Создадим каталог для программ лабораторной работы №7, перейдем в него и создадим файл lab7-1.asm. (рис. 1)



Рис. 1: Создание файла для выполнения работы

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax.

Введем в файл lab7-1.asm текст программы из листинга 7.1 (рис. 2). 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.

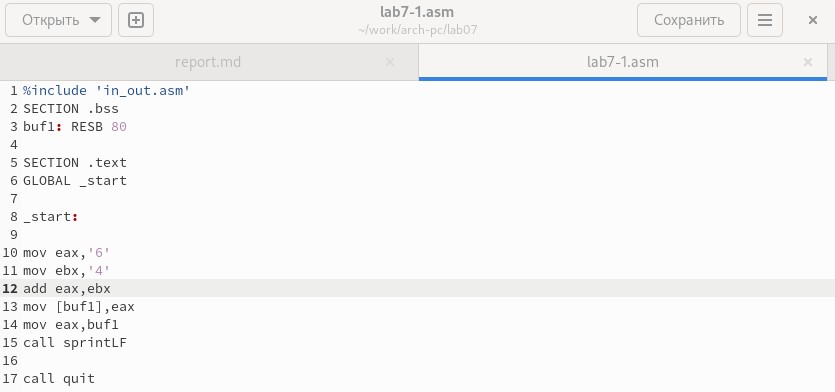


Рис. 2: Программа вывода значения регистра eax

Создадим исполняемый файл и запустим его (рис. 3).

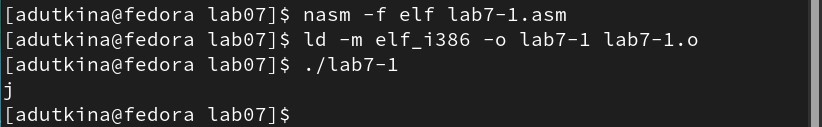


Рис. 3: Создание исполняемого файла

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.

Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы следующим образом (рис. 4):

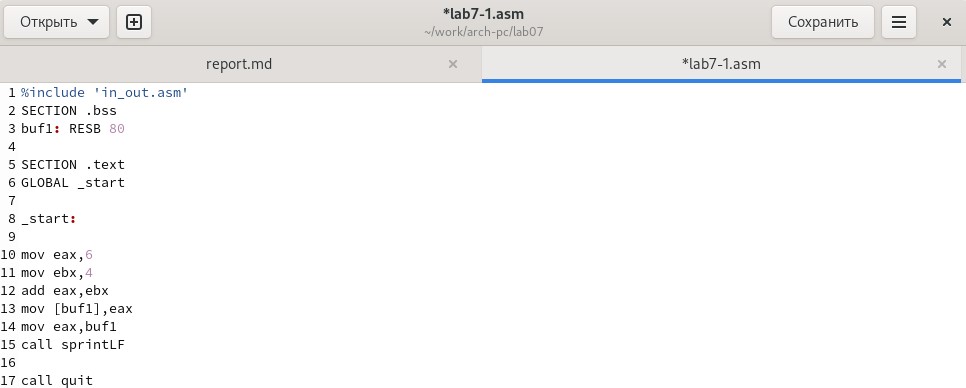


Рис. 4: Измененный текст программы

Как и в предыдущем случае мы не получаем число 10. В данном случае выводится символ с кодом 10. По таблице ASCII можно определить, что код 10 соответствует символу “”. При запуске программы видно, что на экран выводится только пустая строка (рис. 5).

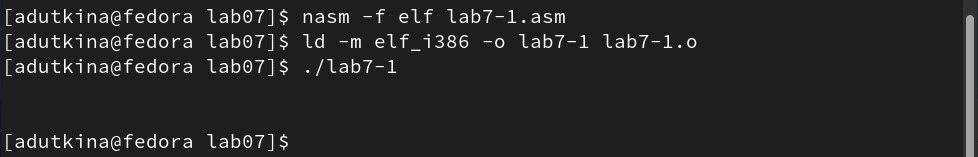


Рис. 5: Результат измененной программы

Для работы с числами в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 7.1 с использованием этих функций. Создадим файл lab7-2.asm и введем в него текст программы из листинга 7.2 (рис. 6).

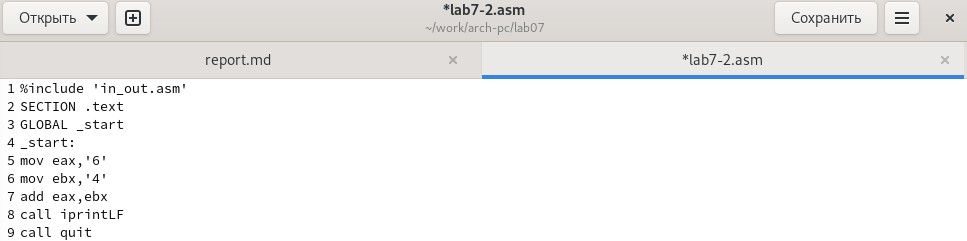


Рис. 6: Использование функций в программе

В результате работы программы мы получили 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 (рис. 7)

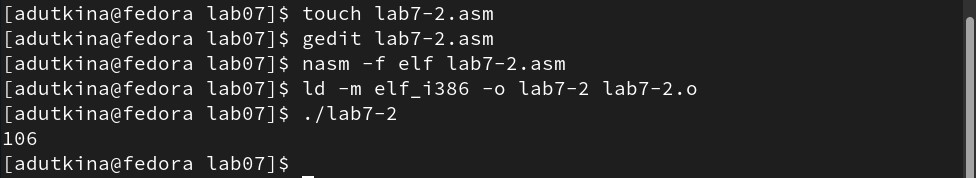


Рис. 7: Результат работы программы с функциями

Аналогично предыдущему примеру изменим символы на числа. При исполнении программы мы получим 10 (рис. 8).

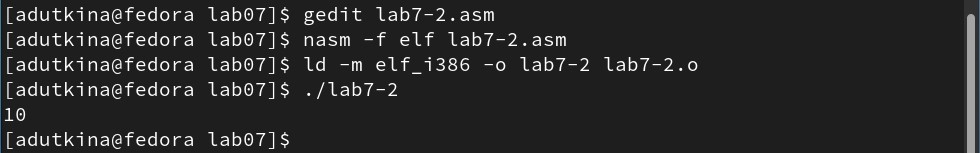


Рис. 8: Результат измененной программы

Заменим функцию iprintLF на iprint. Создадим исполняемый файл и запустим его. Чем отличается вывод функций iprintLF и iprint? В этом случае после вывода результата нет отступа сроки (рис. 9)

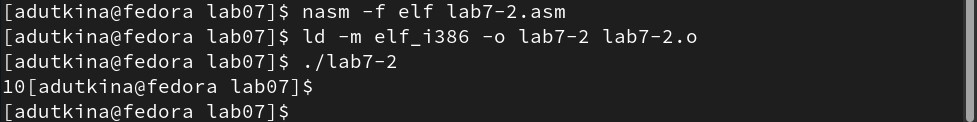


Рис. 9: Изменение функции в программе

## 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 lab7-3.asm и введем в него текст из листинга 7.3 (рис. 10).

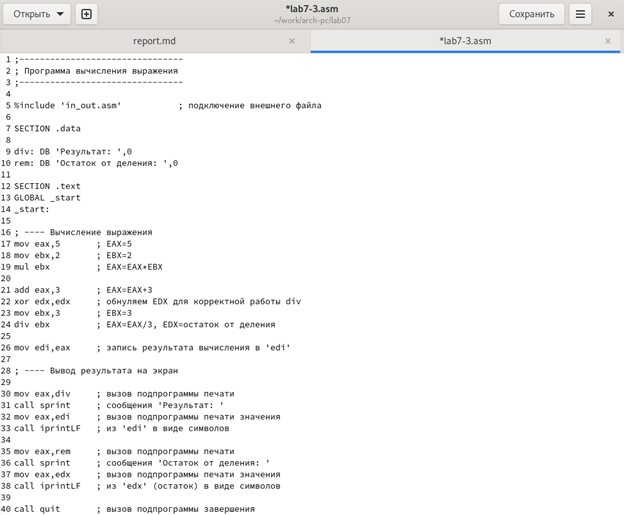


Рис. 10: Программа вычисления выражения

При запуске программы выводятся результат и остаток от деления (рис. 11).

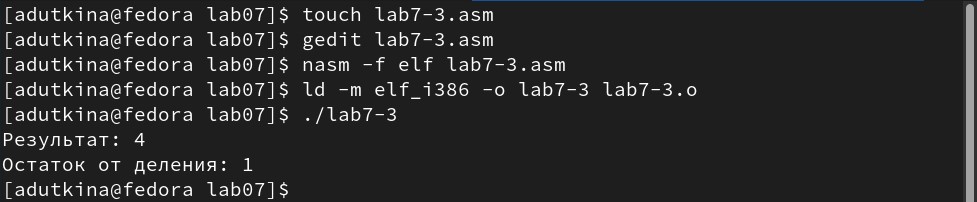


Рис. 11: Резулььтат вычисления выражения

Изменим текст программы для вычисления выражения 𝑓(𝑥) = (4 ∗ 6 + 2)/5 (рис. 12). Запускаем исполняемый файл (рис. 13).

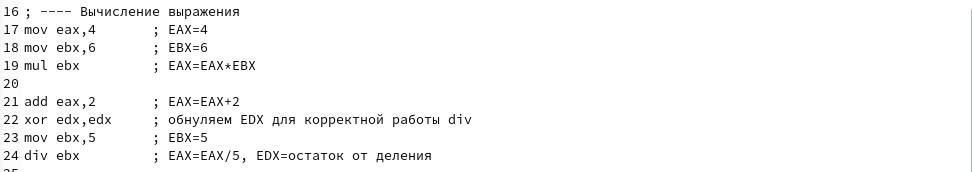


Рис. 12: Изменение значений в программе

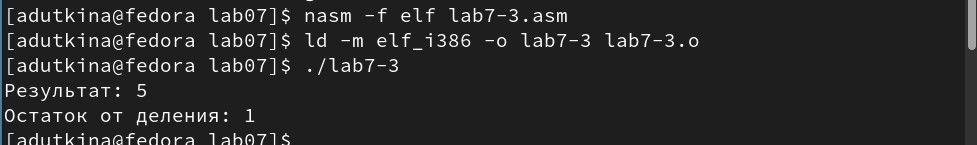


Рис. 13: Проверка правильности работы программы

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру 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 и введем в него текст из листинга 7.4 (рис. 14).

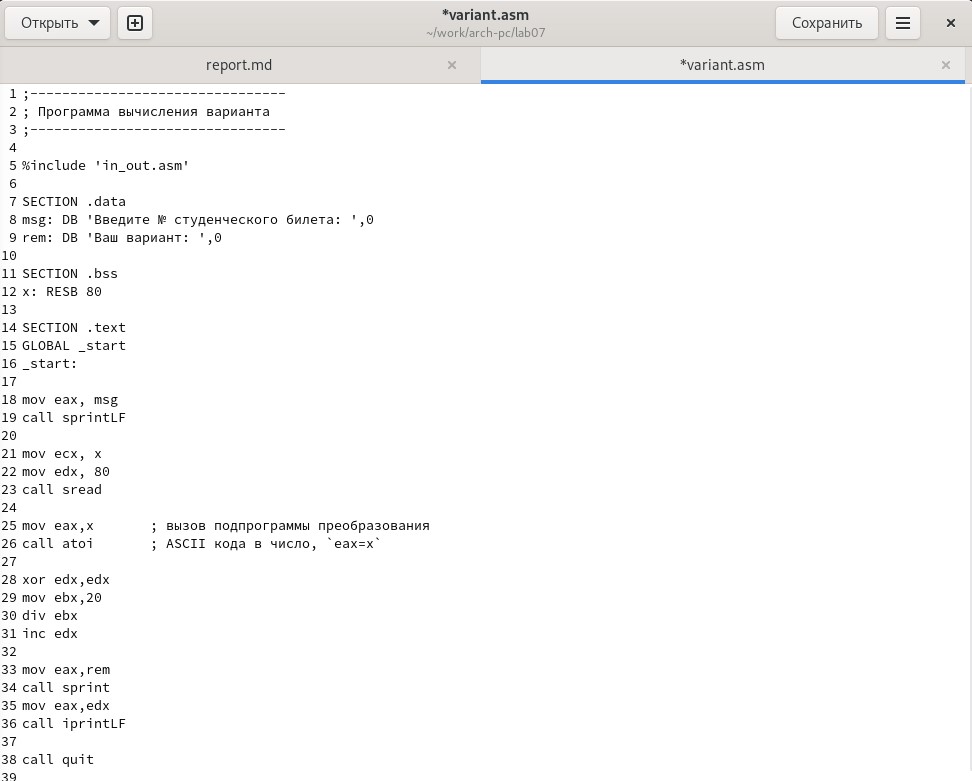


Рис. 14: Вычисление варианта задания по номеру студенческого билета

Создадим исполняемый файл и запустим его (рис. 15). Вычислив номер варианта аналитически мы получаем, что 1132226534 % 20 = 14, 14 + 1 = 15. Значит, программа работает правильно.

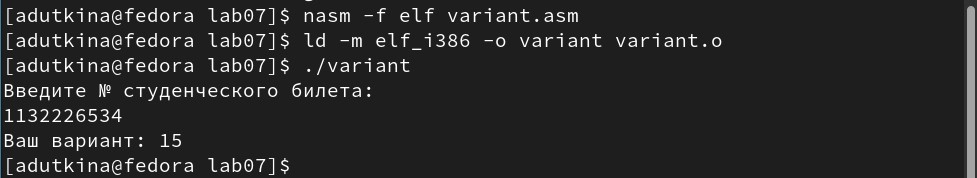


Рис. 15: Запуск программы

Таким образом: 1. За вывод на экран сообщения “Ваш вариант” отвечают строки 33-34 kbcnbyuf 7.4; 2. Конструкция mov eсx,x отвечает за запись значения х в регистр ecx; mov edx, 80 определяет длину вводимой строки; call sread вызыват подпрограмму ввода сообщения; 3. Call atoi используется для преобразования ASCII кода в число; 4. За вычисление варианта отвечают строки 28-31 листинга 7.4; 5. Остаток от деления при выполнении инструкции “div ebx” запишется в регистр “edx”; 6. Инструкция “inc edx” используется для уменьшения значения в регистре “edx”; 7. За вывод на экран результата вычислений отвечают строки 35-36.

## 2.3 Самостоятельная работа

Напишем программу для работы с функцией f(x)=(5+x)^2 - 3 в файл variant2.asm. Проверим работу программы для значений х1=5, х2=1 (рис. 16).

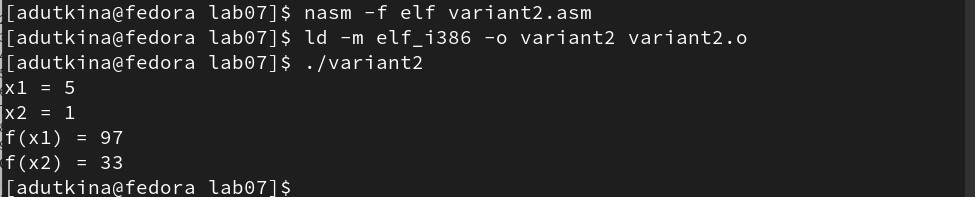


Рис. 16: Работа программы вычисления значения функции

# 3 Выводы

В ходе лабораторной работы были изучены арифметические инструкции языка ассемблера NASM.