Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Батова Ирина Сергеевна, НММбд-01-22

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Выполнение лабораторной работы

1. Сначала создаем каталог для программ данной лабораторной работы, используя команду ‘mkdir’. Далее переходим в этот каталог с помощью команды ‘cd’ и создаем файл ‘lab7-1.asm’ (рис. 1).

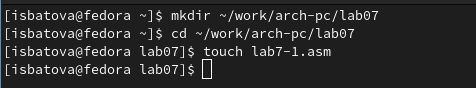


Рис. 1: Создание необходимых для работы каталогов и файлов

1. Открываем файл ‘lab7-1.asm’ и вводим в него программу из листинга 7.1 из лабораторной работы (рис. 2).



Рис. 2: Ввод программы листинга 7.1

Перед запуском файла копируем файл ‘in\_out.asm’ из каталога lab06 в каталог lab07 для корректной работы программы.

После этого создаем исполняемый файл и запускаем его (рис. 3). Наша программа выводит ‘j’ - это происходит потому, что программа складывает код символа ‘6’ и код символа ‘4’, и выводит символ, соответствующий сумме этих кодов.

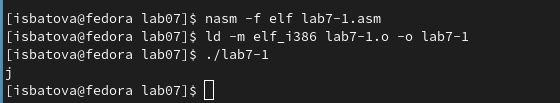


Рис. 3: Запуск программы lab7-1

1. Открываем файл ‘lab7-1.asm’ и немного видоизменяем программу: вместо символов ‘6’ и ‘4’ записываем числа 6 и 4 соответственно (рис. 4).

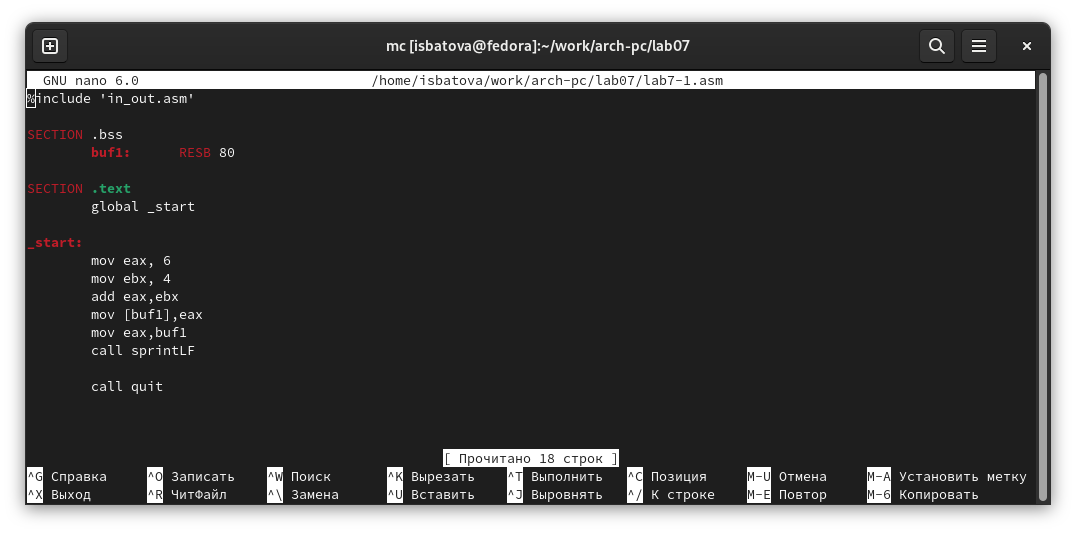


Рис. 4: Внесение изменений в файл с программой lab7-1

Создаем исполняемый файл и запускаем его (рис. 5). В данном случае у нас будет выводится символ с кодом 10. По таблице определяем, что это символ перевода строки, поэтому программа ничего не выводит, а только переходит на следующую строку.

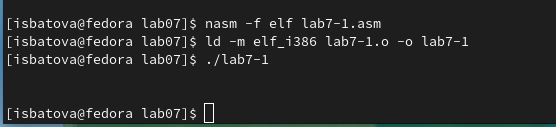


Рис. 5: Запуск измененной программы lab7-1

1. В том же каталоге создаем файл ‘lab7-2.asm’, используя команду ‘touch’ (рис. 6).

Рис. 6: Создание файла ‘lab7-2.asm’

Рис. 6: Создание файла ‘lab7-2.asm’

Далее открываем этот файл и вводим в него программу из листинга 7.2 из лабораторной работы (рис. 7).

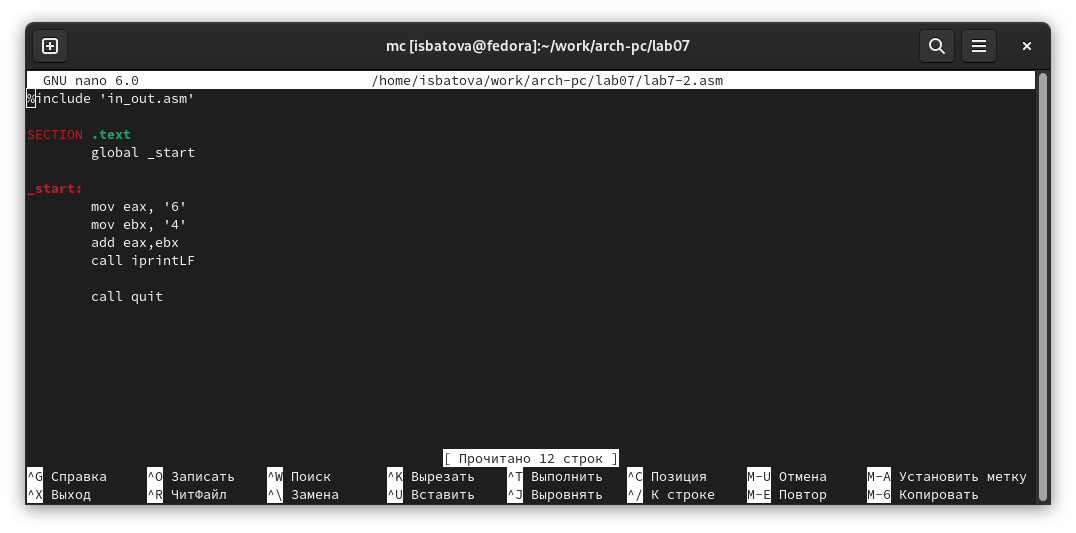


Рис. 7: Ввод программы листинга 7.2

Создаем исполняемый файл и запускаем его (рис. 8). Программа выводит нам число 106, так как аналогично складываются коды символов ‘6’ и ‘4’, но благодаря функции iprintLF выводится само число (а не символ, которому соответствует данное число).

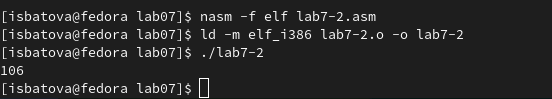


Рис. 8: Запуск программы lab7-2

1. Далее открываем файл ‘lab7-2.asm’ и видоизменяем его также, как изменяли первый файл: вместо символов ‘6’ и ‘4’ записываем числа 6 и 4 соответственно (рис. 9).

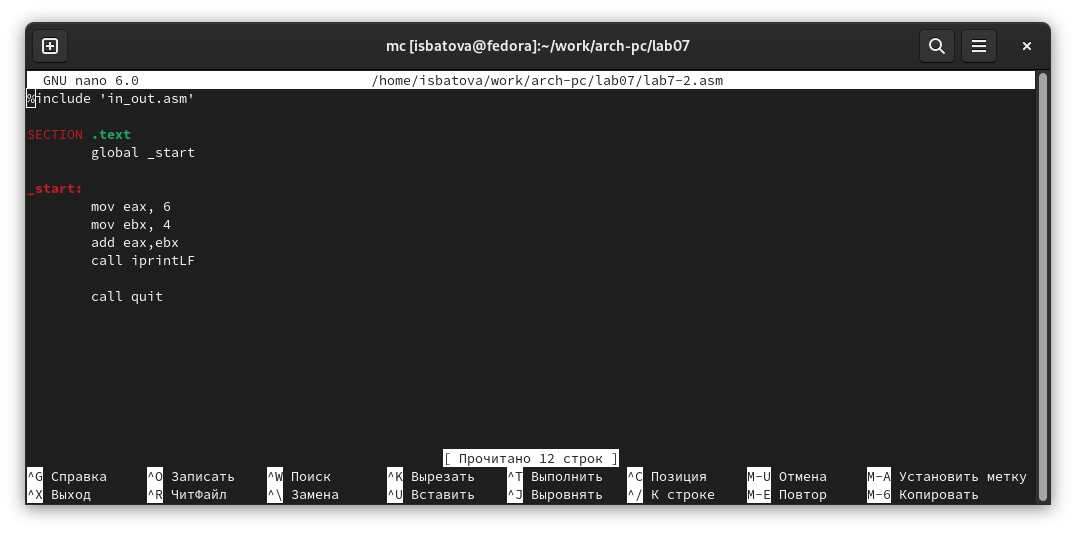


Рис. 9: Внесение изменений в файл с программой lab7-2

Создаем исполняемый файл и запускаем его (рис. 10). Наша программа выводит ‘10’.

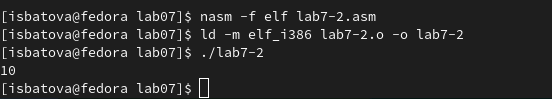


Рис. 10: Запуск измененной программы lab7-2

Далее вновь открываем файл и вносим изменения - вместо iprintLF вводим iprint (рис. 11).

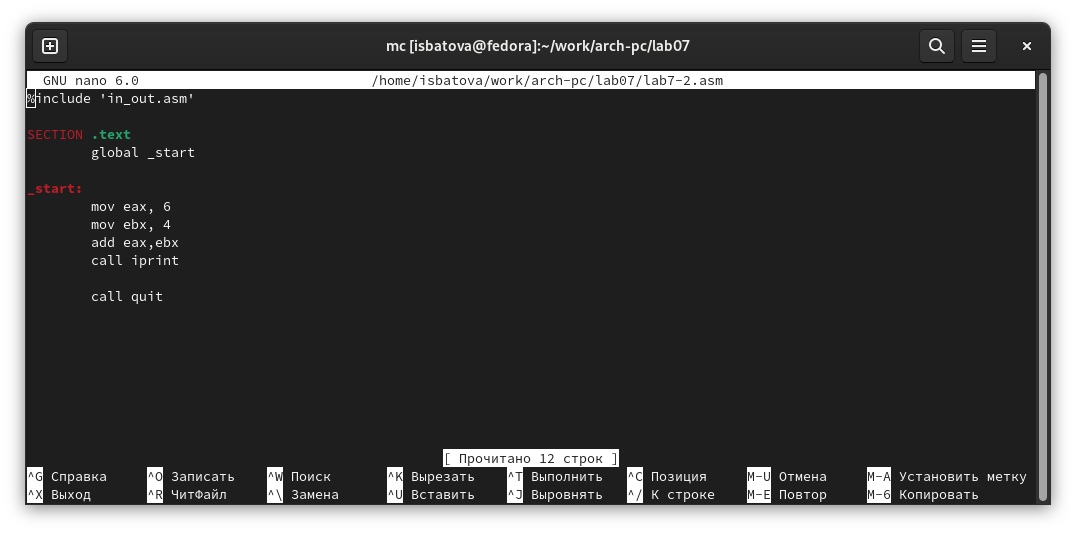


Рис. 11: Замена команды irpintLF на iprint

Вновь создаем исполняемый файл и запускаем его (рис. 12). Наша программа также выводит ‘10’, но, в отличии от использования ‘iprintLF’, ‘iprint’ не делает переход на новую строку.

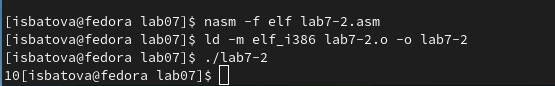


Рис. 12: Запуск дважды измененной программы lab7-2

1. С помощью команды ‘touch’ в каталоге lab07 создаем файл ‘lab7-3.asm’ (рис. 13).

Рис. 13: Создание файла ‘lab7-3.asm’

Рис. 13: Создание файла ‘lab7-3.asm’

Открываем его и вводим программу из листинга 7.3 из лабораторной работы (рис. 14).

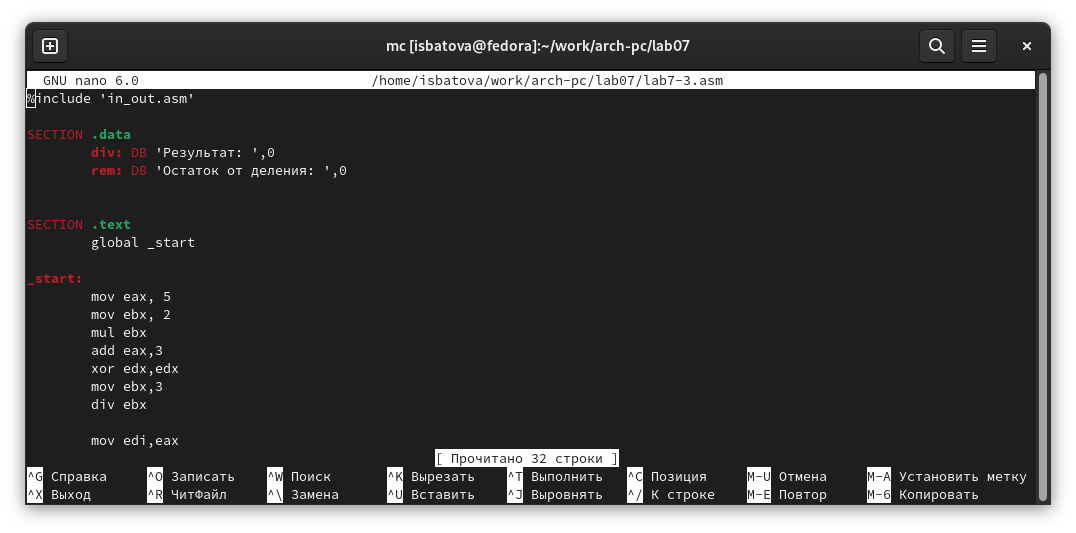


Рис. 14: Ввод программы листинга 7.3

Создаем исполняемый файл и запускаем его (рис. 15). Видим, что программа работает корректно.

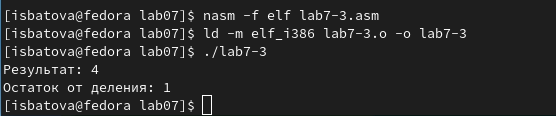


Рис. 15: Запуск программы lab7-3

Далее вновь открываем этот файл и вносим изменения так, чтобы у нас вычислялось выражение f(x)=(4\*6+2)/5 (рис. 16).

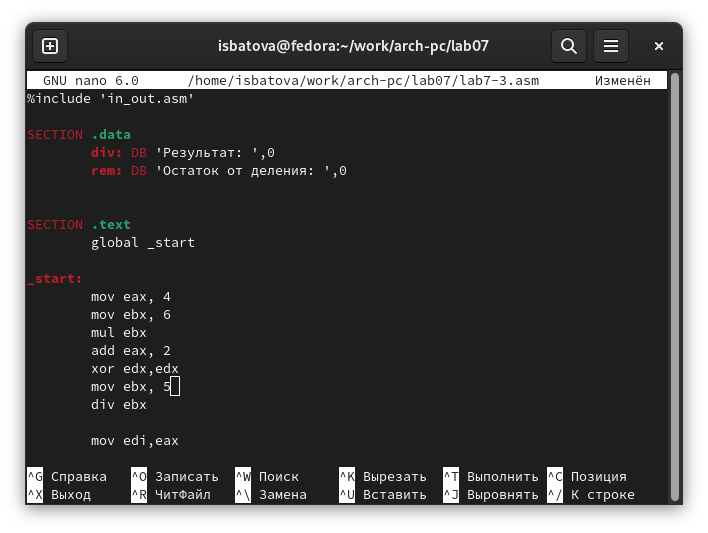


Рис. 16: Внесение изменений в файл с программой lab7-3

Создаем исполняемый файл и запускаем его (рис. 17). Программа работает корректно: действительно, если мы устно посчитаем уравнение, то получим 5 с остатком 1.

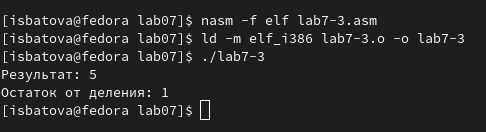


Рис. 17: Запуск измененной программы lab7-3

1. В каталоге lab07 создаем файл ‘variant.asm’, используя команду ‘touch’ (рис. 18).

Рис. 18: Создание файла ‘variant.asm’

Рис. 18: Создание файла ‘variant.asm’

Открываем данный файл, вводим в него программу из листинга 7.4 (рис. 19).

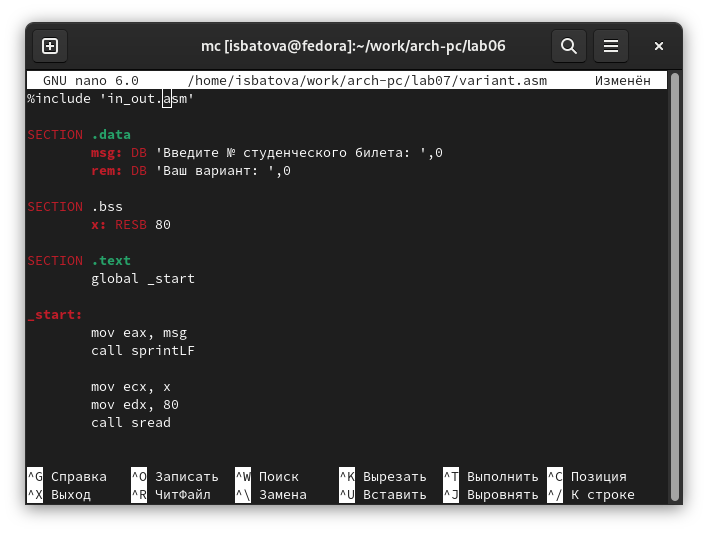


Рис. 19: Ввод программы листинга 7.4

Создаем исполняемый файл и запускаем его (рис. 20). На предложение ввести номер студенческого билета вводим его и получаем свой вариант. Проверяем аналитически - вариант выведен верно.

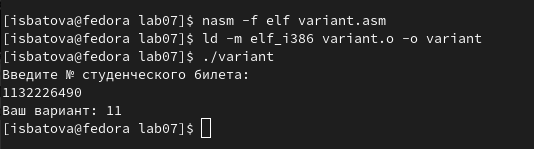


Рис. 20: Запуск программы variant

# 3 Ответы на вопросы

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

За это отвечают строки:

mov eax,rem call sprint

1. Для чего используется следующие инструкции? nasm mov ecx, x mov edx, 80 call sread

Данные инструкции используются для ввода сообщения (в данном случае х) и запись введенных данных в ‘ecx’.

1. Для чего используется инструкция “call atoi”?

Данная инструкция используется для преобразования символьного вида в число.

1. Какие строки листинга 7.4 отвечают за вычисления варианта?

За вычисление варианта отвечают строки:

mov ebx,20 div ebx inc edx

1. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

Остаток от деления записывается в регистр ‘edx’.

1. Для чего используется инструкция “inc edx”?

Эта инструкция используется для прибавления к значению регистра edx единицу.

1. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений?

За вывод на экран результату вычислений отвечают строки:

mov eax,edx call iprintLF

# 4 Задание для самостоятельной работы

Создаем файл с именем ‘sam.asm’. В последнем пункте лабораторной работы мне был выдан 11 вариант, поэтому программа была написана для функции f(x)=10(x+1)-10 (рис. 21).



Рис. 21: Ввод программы для вычисления значений функции

Создаем исполняемый файл и запускаем его (рис. 22). На предложение ввести значение х вводим значения из таблицы: сначала 1, потом 7. При подсчете аналитически получаются такие же ответы, поэтому программа написана правильно.

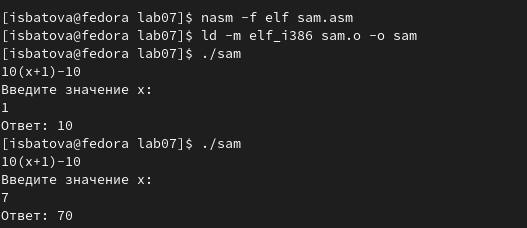


Рис. 22: Запуск программы sam

# 5 Выводы

В данной лабораторной работе мной были освоены арифметические инструкции языка ассемблера NASM.