Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Луцкая Алиса Витальевна

Содержание

# 1 Цель работы

Цель данной лабораторной работы - освоение арифметческих инструкций языка ассемблера NASM.

# 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

# 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx. - Непосредственная адресация – значение операнда задается непосредственно в команде, Например: mov ax,2. - Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно

# 4 Выполнение лабораторной работы

## 4.1 Символьные и численные данные в NASM

1. Создаю каталог для программам лабораторной работы № 6 и перехожу в него (рис. 1).

Рис. 1: Создание директории

Рис. 1: Создание директории

Создаю файл lab6-1.asm (рис. 2).

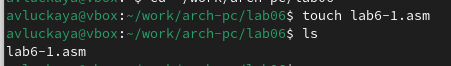


Рис. 2: Создание файла

1. Перед созданием исполняемого файла создаю копию файла in\_out.asm в каталоге ~/work/arch-pc/lab06 (рис. 3).

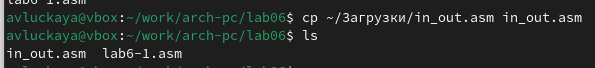


Рис. 3: Создание копии файла

Открываю созданный файл lab6-1.asm, вставляю в него программу (рис. 4).

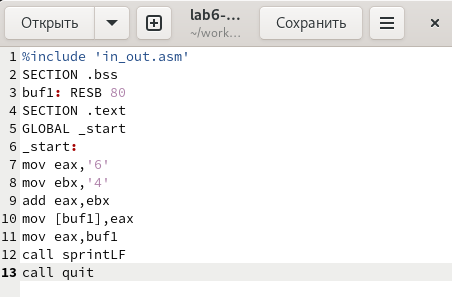


Рис. 4: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 5). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

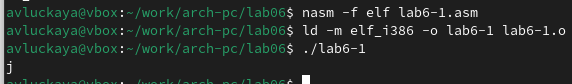


Рис. 5: Запуск исполняемого файла

1. Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 6).

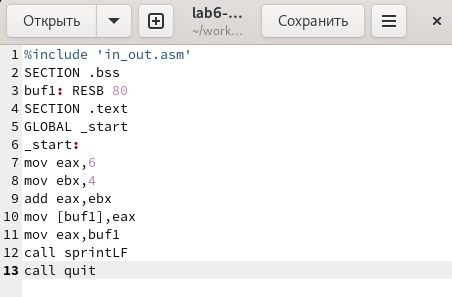


Рис. 6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 7). Выводится символ с кодом 10, это символ перевода строки, он не отображается при выводе на экран.

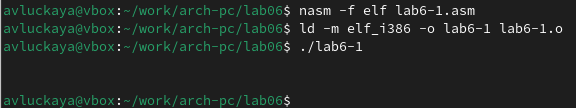


Рис. 7: Запуск исполняемого файла

1. Создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввожу в него текст про- граммы из листинга 6.2 (рис. 8).

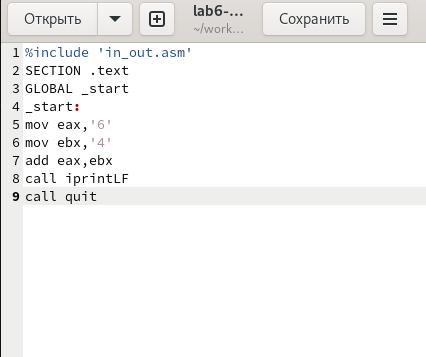


Рис. 8: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 9). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

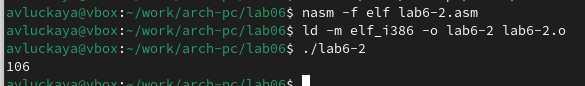


Рис. 9: Запуск исполняемого файла

1. Аналогично предыдущему примеру заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 10).

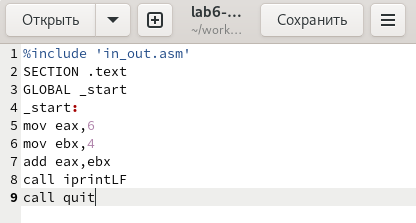


Рис. 10: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 11).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

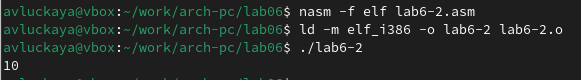


Рис. 11: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 12).

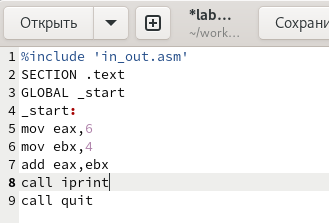


Рис. 12: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 13). Вывод не изменился, единственное iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF, поэтому командная строка отображается на той же строке, что и результат программы.

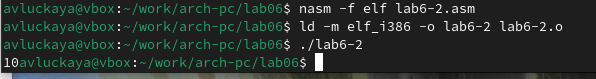


Рис. 13: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

1. Создаю файл lab6-3.asm в каталоге ~/work/arch-pc/lab06, внимательно изучаю текст программы из листинга 6.3 и ввожу в lab6-3.asm (рис. 14).

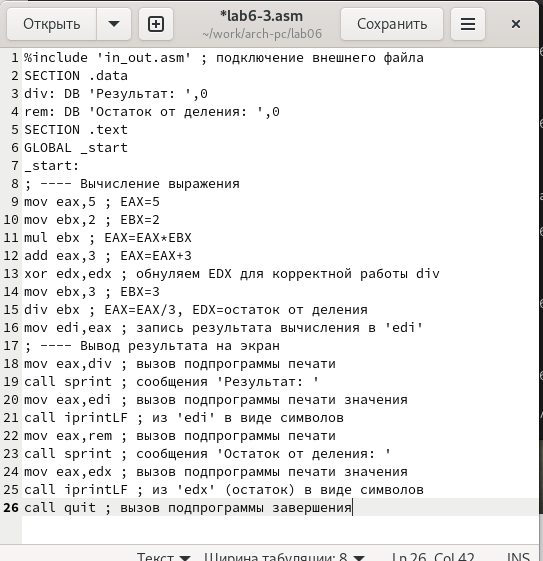


Рис. 14: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 15).

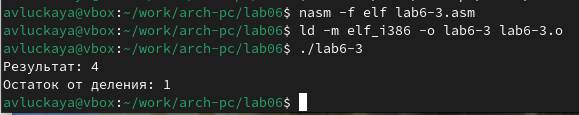


Рис. 15: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. 16).

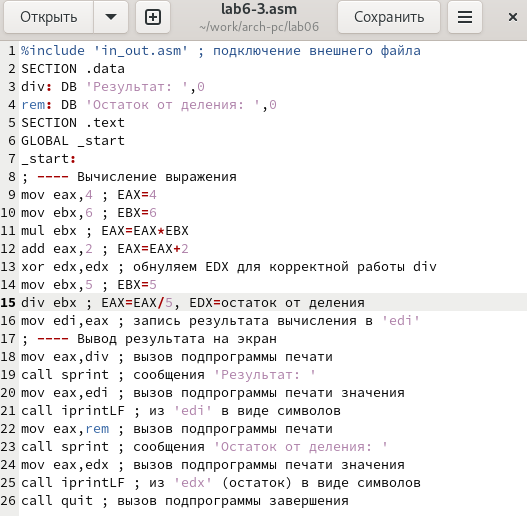


Рис. 16: Изменение программы

1. Создаю и запускаю новый исполняемый файл (рис. 17). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

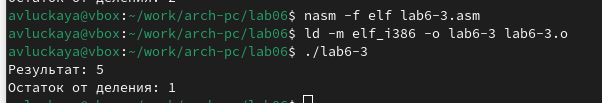


Рис. 17: Запуск исполняемого файла

Создаю файл variant.asm и ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 18).

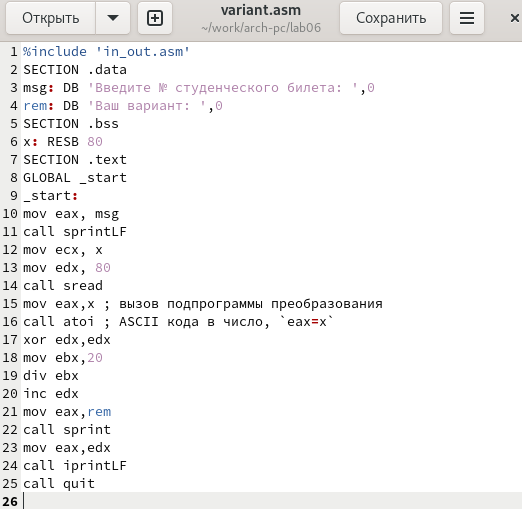


Рис. 18: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 19). Ввожу номер своего студенческого билета, программа вывела мой вариант - 11.

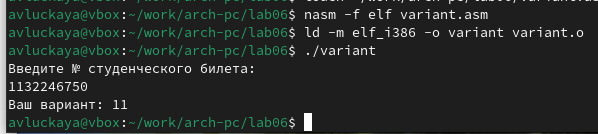


Рис. 19: Запуск исполняемого файла

### 4.2.1 Ответы на вопросы по программе

1. За вывод на экран сообщения “Ваш вариант” отвечают строки:

mov eax,rem  
call sprint

1. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
2. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
3. За вычисления варианта отвечают строки:

xor edx,edx ; обнуление edx для корректной работы div  
mov ebx,20 ; ebx = 20  
div ebx ; eax = eax/20, edx - остаток от деления  
inc edx ; edx = edx + 1

1. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
2. Инструкция inc edx увеличивает значение регистра edx на 1
3. За вывод на экран результатов вычислений отвечают строки:

mov eax,edx  
call iprintLF

## 4.3 Выполнение заданий для самостоятельной работы

С помощью touch Создаю файл lab6-4.asm, открываю в редакторе gedit, ввожу в него текст программы для вычисления значения выражения 10\*(1 + x) - 10. Это выражение было под вариантом 11. (рис. 20).

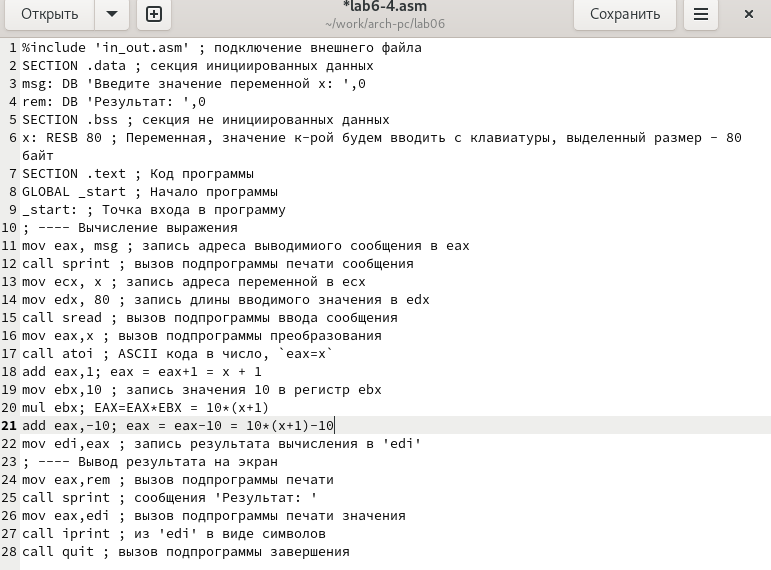


Рис. 20: Написание программы

Создаю и запускаю исполняемый файл (рис. 21). При вводе значения 1, вывод - 10.

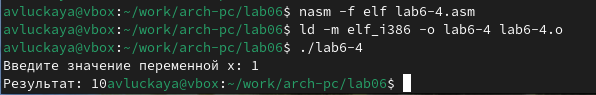


Рис. 21: Запуск исполняемого файла

Проверяю коректность работы программы с другим значением на вводе. Провожу запуск исполняемого файла, на вводе значение 5, вывод- 50. Программы работает коректно (рис. 22).

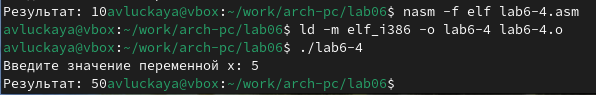


Рис. 22: Повторный запуск исполняемого файла

# 5 Выводы

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

# 6 Список литературы

1. https://esystem.rudn.ru/course/view.php?id=112