Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Алиева Милена Арифовна

Содержание

1 Цель работы

2 Задания

3 Теоретическое введение

4 Выполнение лабораторной работы

5 Вывод

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Задание 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.

Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: inc (от англ. increment) и dec (от англ. decrement), которые увеличивают и уменьшают на 1 свой операнд.

Команда neg рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.

# 4 Выполнение лабораторной работы

1. Создаю каталог для программам лабораторной работы № 7, перейхожу в него и создаю файл lab7-1.asm: (рис. 1)

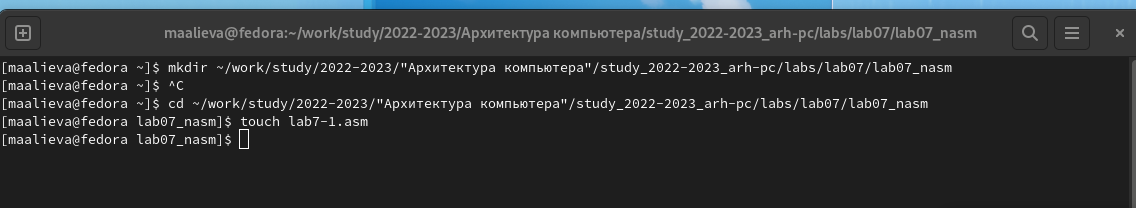


Рис. 1: Создание каталога lab7-1.asm и файла для работы

1. Открываю файл lab7-1.asm и вставляю в него программу вывода значений, записанных в регистр eax (рис. 2)



Рис. 2: Текст программы

1. Создаю исполняемый файл и запускаю его, предварительно добавив в каталог подключаемый файл in\_out.asm. Результатом будет символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении, , а код символа 4 – 00110100. Команда add eax,ebx запишет в регистр eax сумму кодов – 01101010 (106 в десятичном представлении), что в свою очередь является кодом символа j по таблице АSCII (рис. 3)

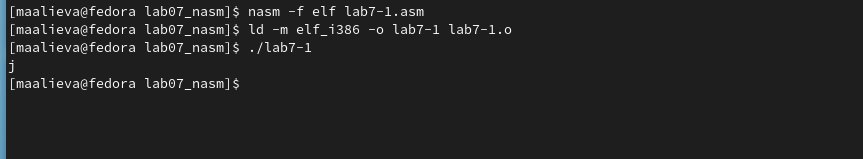


Рис. 3: Создание исполняемого файла и запуск программы

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

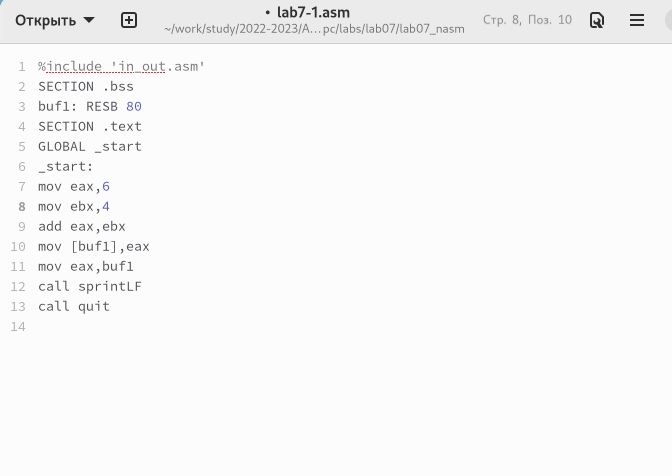


Рис. 4: Изменённый текст программы

1. Создаю новый исполняемый файл программы и запускаю его. Результатом является пустой вывод(так как вывелся символ с кодом 10, а это символ перевода строки, этот символ не отображается при выводе на экран (рис. 5)

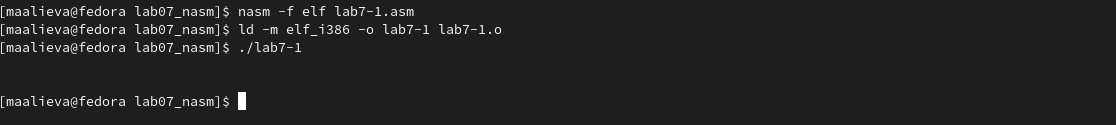


Рис. 5: Вывод изменённой программы

1. Создаю файл lab7-2.asm с помощью утилиты touch. Ввожу в файл текст программы для вывода значения регистра eax (рис. 6)



Рис. 6: Программа для вывода значения регистра eax

1. Создаю исполняемый файл и запускаю его. Результатом является число 106(это происходит потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”) (рис. 7)

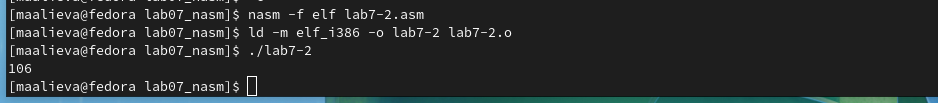


Рис. 7: Вывод программы(файл lab7-2)

1. Заменяю в тексте программы в файле lab7-2.asm символы “6” и “4” на числа 6 и 4 (рис. 8)



Рис. 8: Замена в тексте программы “6” и “4” на числа 6 и 4

1. Создаю исполняемый файл и запускаю его. Результатом является число 106(это происходит потому что теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10)(рис. 9)

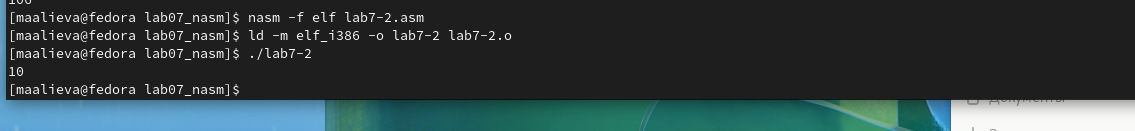


Рис. 9: Запуск изменённый программы в файле lab7-2

1. Заменяю в тексте программы функцию iprintLF на iprint. Создаю исполняемый файл и запускаю его. Результатом является исчезновение переноса строки (рис. 10)

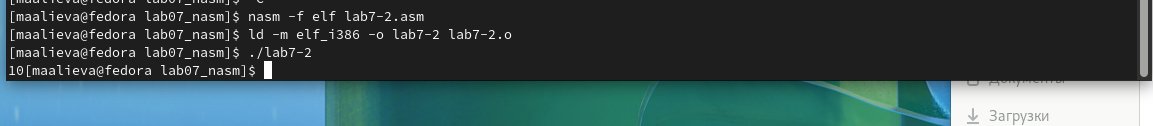


Рис. 10: Замена функции iprintLF на iprint

1. Создаю файл lab7-3.asm с помощью утилиты touch. Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3. Создаю исполняемый файл и запускаю его. Получаю корректный результат (рис. 11)

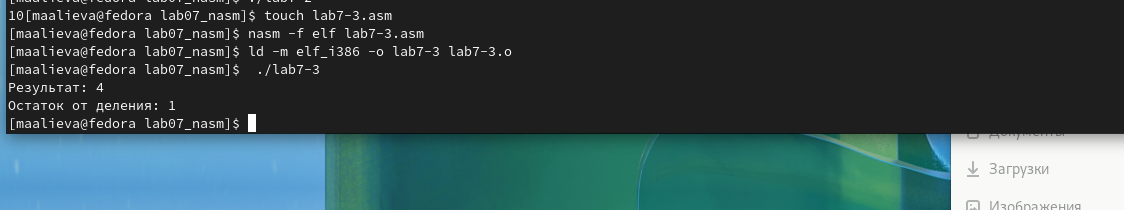


Рис. 11: Программа по вычислению выражения f(x) = (5 \* 2 + 3)/3

1. Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. 12)

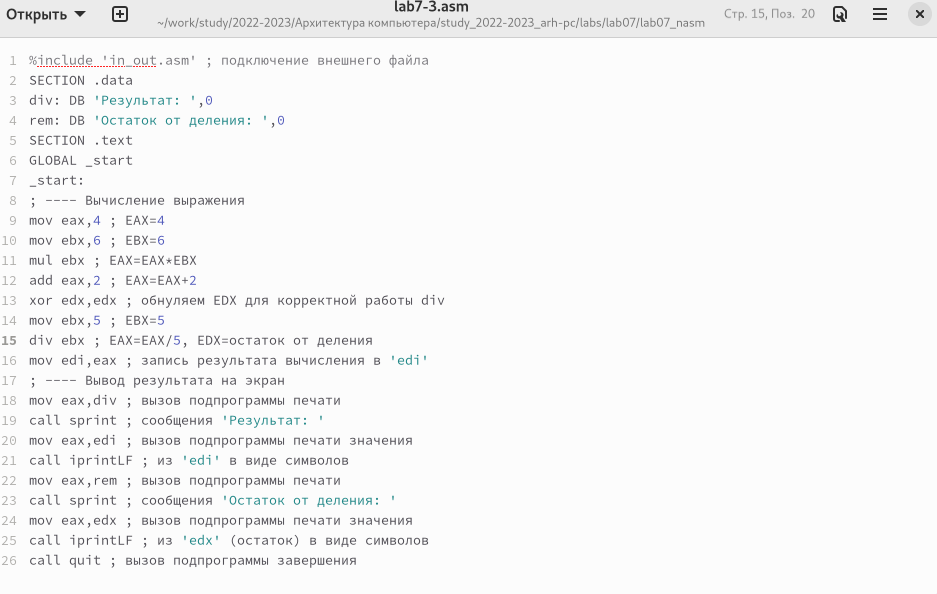


Рис. 12: Программа по вычислению выражения f(x) = (4 \* 6 + 2)/5

1. Создаю исполняемый файл и запускаю его. Вывод результата и остатка от деления. Программа отработала верно (рис. 13)

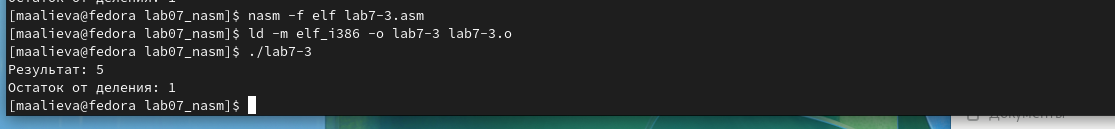


Рис. 13: Создание исполняемого файла и вывод программы

1. Создаю файл variant.asm с помощью утилиты touch. Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 14)

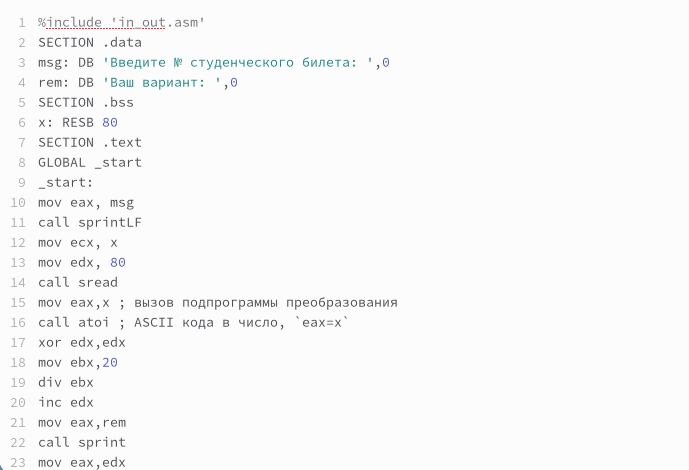


Рис. 14: Текст программы по вычислению варианта

Ответы на вопросы:

1. За вывод на экран сообщения ‘Ваш вариант:’ отвечают строки mov eax,rem call sprint
2. mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx. mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
3. Инструкция call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
4. За вычисление варианта отвечают строки xor edx,edx mov ebx,20 div ebx inc edx
5. Остаток от деления при выполнении инструкции “div ebx” записывается в регистр edx
6. Инструкция “inc edx” используется для увеличения значения регистра edx на 1
7. За вывод на экран результата вычислений отвечают строки mov eax,edx call iprintLF
8. Создаю исполняемый файл и запускаю его. Получаю номер варианта, при проверке всё сходится (рис. 15)

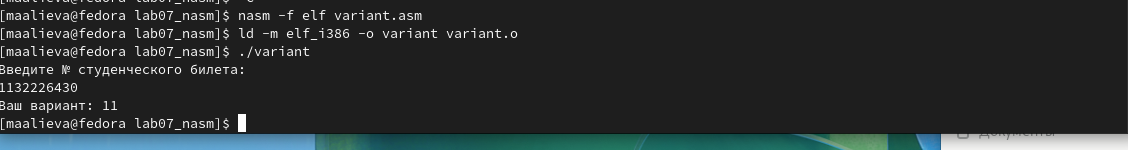


Рис. 15: Вывод программы из файла variant.asm

1. Приступаю к выполнению заданий для самостоятельной работы. Создаю файл lab7-4.asm с помощью утилиты touch. Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения под номером 10: (x + 18) \* 5 - 28 (рис. 16)

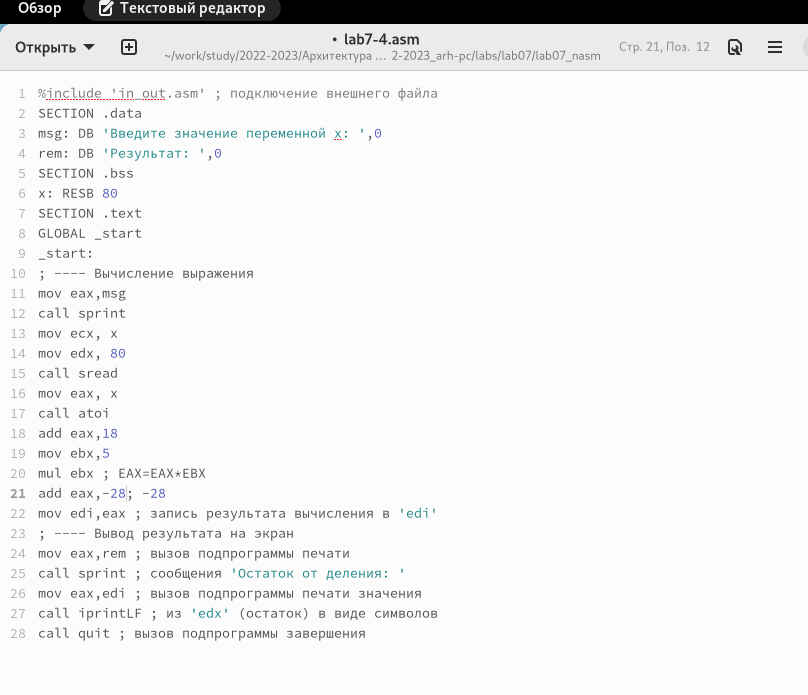


Рис. 16: Текст программы для вычисления (x + 18) \* 5 - 28

1. Создаю исполняемый файл и запускаю его. При вводе x = 2 получаю ответ 72, при вводе x = 3 получаю ответ 77. Делаю проверку вручную. Оба полученных значения являются корректными (рис. 17)

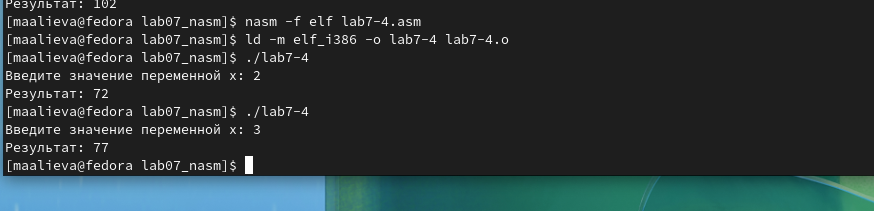


Рис. 17: Вывод программы по вычислению (x + 18) \* 5 - 28

# 5 Выводы

При выполнении лабораторной работы освоила арифметические инструкции языка ассемблера NASM.

# Список литературы