Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Закиров Нурислам Дамирович

Содержание

# 1 Цель работы

Цель данной лабораторной работы состоит в том, чтобы вы научились использовать арифметические инструкции 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

С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. -[1](#fig:001)). Перехожу в созданный каталог с помощью утилиты cd.

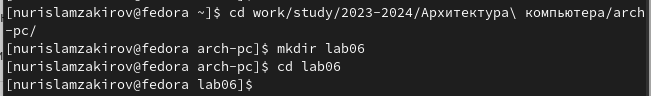


Figure 1: Создание директории

Создаю файл lab6-1.asm, c помощью утилиты touch (рис. -[2](#fig:002)).

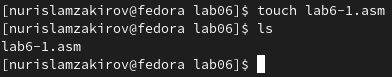


Figure 2: Создание файла

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. -[3](#fig:003)).

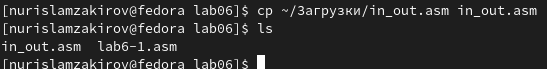


Figure 3: Создание копии файла

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. -[4](#fig:004)).

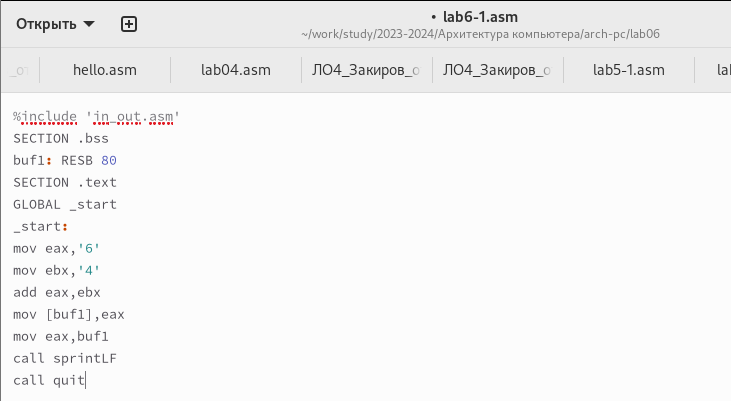


Figure 4: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. -[5](#fig:005)). Программа вывела символ, соответствующий системе ASCII сумме двоичных кодов символов 4 и 6.

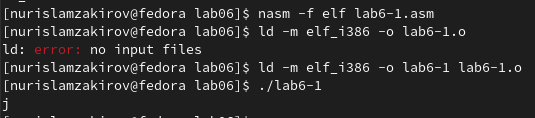


Figure 5: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. -[6](#fig:006)).

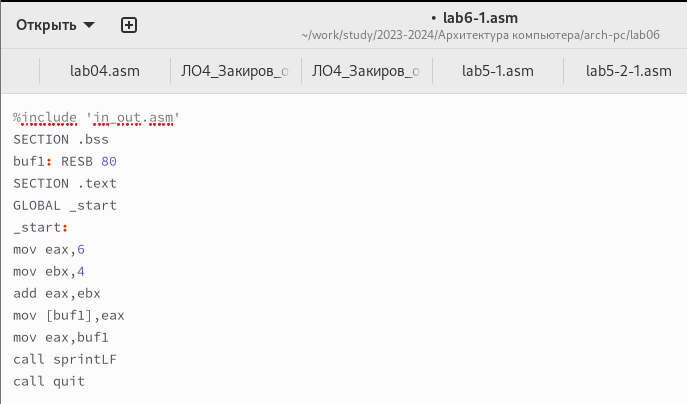


Figure 6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. -[7](#fig:007)). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

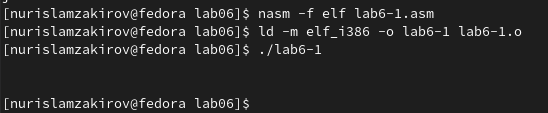


Figure 7: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. -[8](#fig:008)).

Figure 8: Создание файла

Figure 8: Создание файла

Ввожу текст другой программы в файл, чтобы получить значения регистра eax. (рис. -[9](#fig:009)).



Figure 9: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. -[10](#fig:010)). Хотя сложение кодов символов «6» и «4” продолжается, программа позволяет теперь выводить число 106, а не символ.

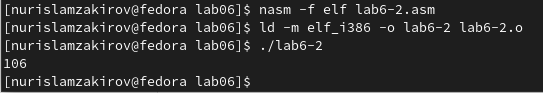


Figure 10: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. -[11](#fig:011)).



Figure 11: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. -[12](#fig:012)).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

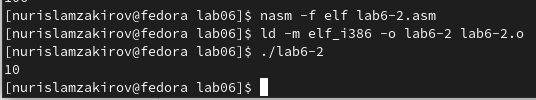


Figure 12: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. -[13](#fig:013)).



Figure 13: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. -[14](#fig:014)). Когда программа выполнялась с функцией iprintLF, символ переноса строки не отображался. Кроме того, iprint, в отличие от iprintLF, не добавляет символа переноса строки к выводу. Таким образом, вывод не изменился.

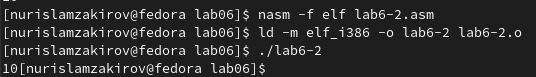


Figure 14: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. -[15](#fig:015)).

Figure 15: Создание файла

Figure 15: Создание файла

В созданный файл ввожу текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3 (рис. -[16](#fig:016)).

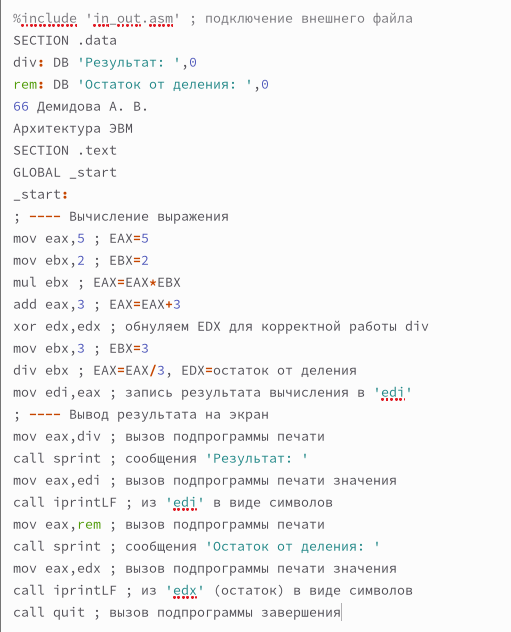


Figure 16: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. -[17](#fig:017)).

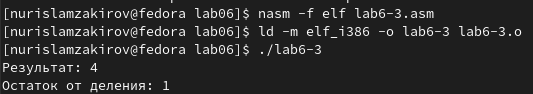


Figure 17: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. -[18](#fig:018)).

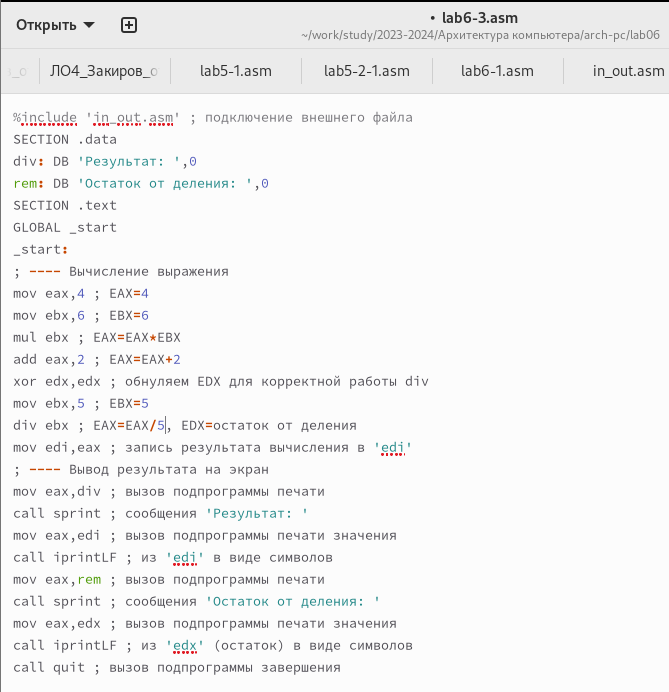


Figure 18: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. -[19](#fig:019)).

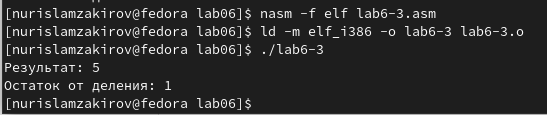


Figure 19: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. -[20](#fig:020)).

Figure 20: Создание файла

Figure 20: Создание файла

Ввожу текст в файл программы для вычисления варианта задания по номеру студенческого билета (рис. -[21](#fig:021)).

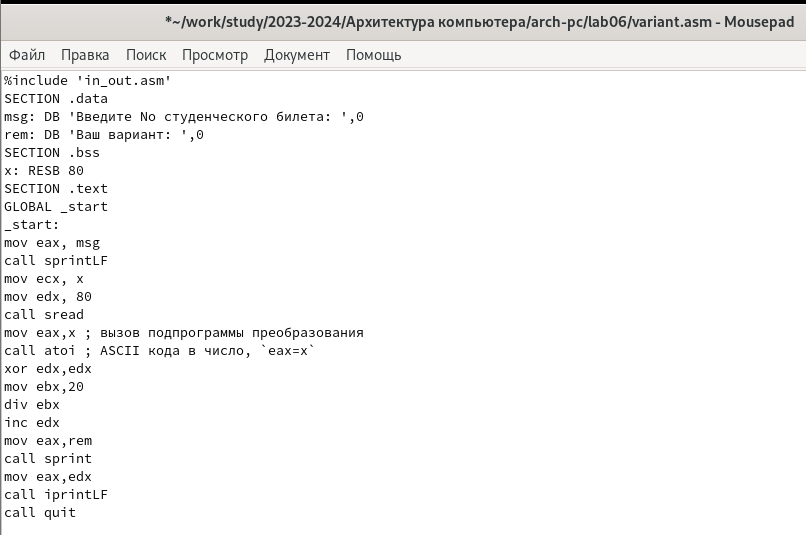


Figure 21: Редактирование файла

Создаю и запускаю исполняемый файл (рис. -[22](#fig:022)). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 1.

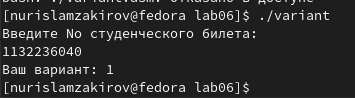


Figure 22: Запуск исполняемого файла

### 4.2.1 Ответы на вопросы по программе

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

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

Создаю файл lab6-4.asm с помощью утилиты touch (рис. -[23](#fig:023)).

Figure 23: Создание файла

Figure 23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения (10 + 2 \* x) / 3 (рис. -[24](#fig:024)). Это выражение было под вариантом 1.

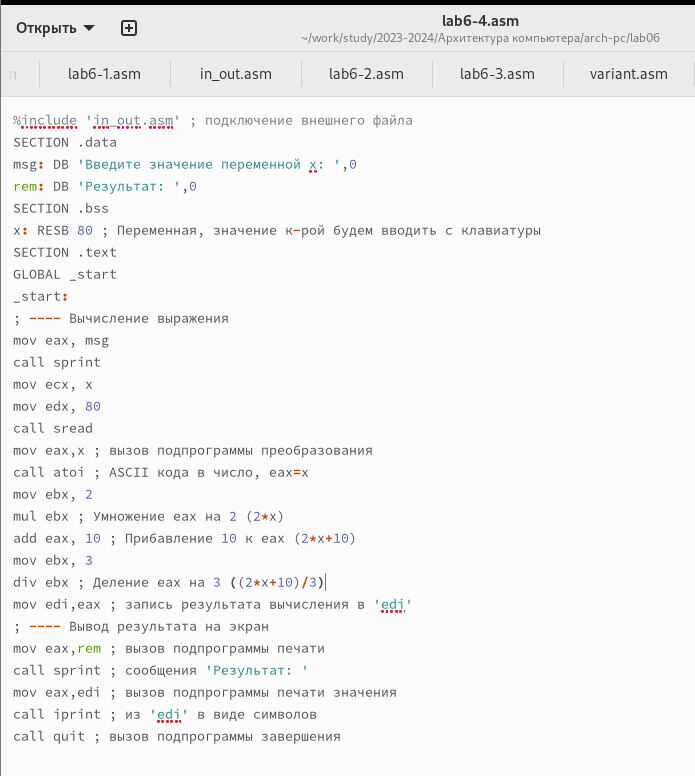


Figure 24: Написание программы

Создаю и запускаю исполняемый файл (рис. -[25](#fig:025)). При вводе значения 1, вывод 4.

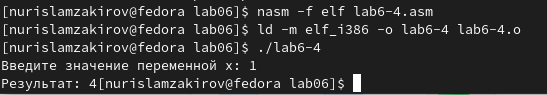


Figure 25: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. -[26](#fig:026)). Программа отработала верно.

Figure 26: Запуск исполняемого файла

Figure 26: Запуск исполняемого файла

**Листинг 4.1. Программа для вычисления значения выражения (10 + 2 \* x) / 3.**

msg: DB 'Введите значение переменной х: ',0  
rem: DB 'Результат: ',0  
SECTION .bss  
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры  
SECTION .text  
GLOBAL \_start  
\_start:  
; ---- Вычисление выражения  
mov eax, msg  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax,x ; вызов подпрограммы преобразования  
call atoi ; ASCII кода в число, eax=x  
mov ebx, 2  
mul ebx ; Умножение eax на 2 (2\*x)  
add eax, 10 ; Прибавление 10 к eax (2\*x+10)  
mov ebx, 3  
div ebx ; Деление eax на 3 ((2\*x+10)/3)  
mov edi,eax ; запись результата вычисления в 'edi'  
; ---- Вывод результата на экран  
mov eax,rem ; вызов подпрограммы печати  
call sprint ; сообщения 'Результат: '  
mov eax,edi ; вызов подпрограммы печати значения  
call iprint ; из 'edi' в виде символов  
call quit ; вызов подпрограммы завершения

# 5 Выводы

Выполняя данную лабораторную работу, я освоил арифметические инструкции языка ассемблера NASM.

# 6 Список литературы

1. [Лабораторная работа №6](https://esystem.rudn.ru/pluginfile.php/2089086/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%966.%20%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5%20%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8%20%D0%B2%20NASM..pdf)
2. [Таблица ASCII](https://www.rapidtables.com/code/text/ascii-table.html)