Отчёт по лабораторной работе №6

Дисциплина: Архитектура Компьютера

Дарина Андреевна Куокконен

Содержание

# 1 Цель работы

Целью данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM и приобретение навыков работы с ними.

# 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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации: • 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. Расширенная таблица ASCII состоит из двух частей. Первая является универсальной, а вторая предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться. 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 lab06 для выполнения соответствующей лабораторной работы. Перехожу в созданный каталог с помощью утилиты cd. С помощью touch создаю файл lab6-1.asm. (рис. [1](#fig:001)).

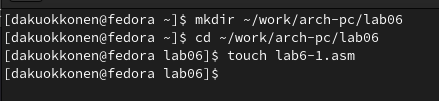


Figure 1: Работа с директориями и создание файла

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, ибо он будет использоваться в дальнейшем. (рис. [2](#fig:002)).

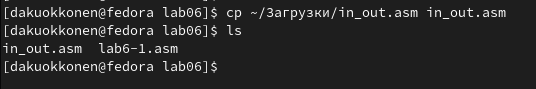


Figure 2: Создание копии фала и проверка

Открываю созданный файл lab6-1.asm, вставляю в него следующую программу: (рис. [3](#fig:003)).

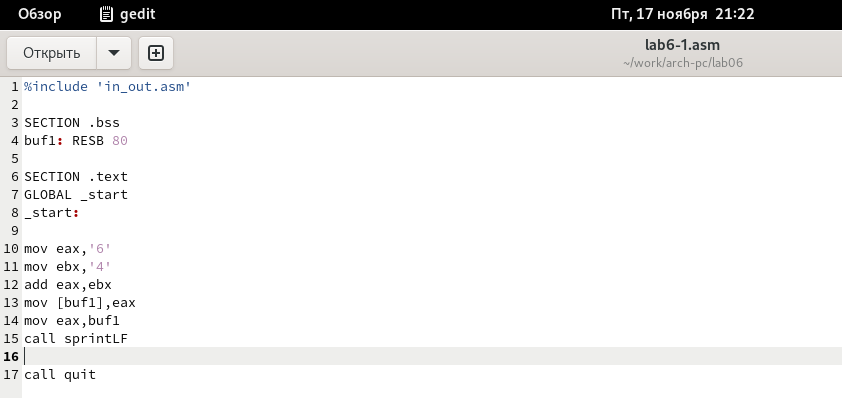


Figure 3: Редактирование файла

Выполняю компиляцию, компоновку файла и запускаю его. Программа выводит символ j, потому что он соответствует сумме двоичных кодов символов 4 и 6 по системе ASCII. (рис. [4](#fig:004)).

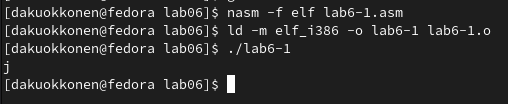


Figure 4: Подготовка и исполнение файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. [5](#fig:005)).

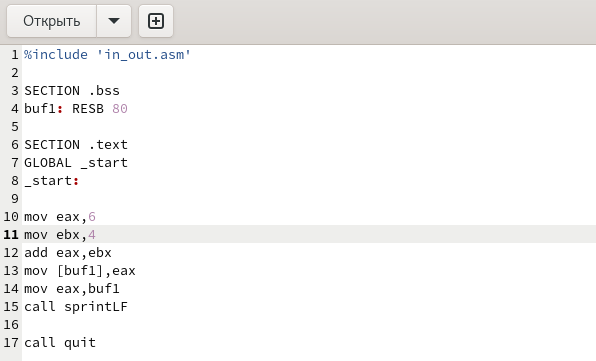


Figure 5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его. На экран вывелся символ с кодом 10 и он не отображается, т.к. это символ перевода строки. (рис. [6](#fig:006)).

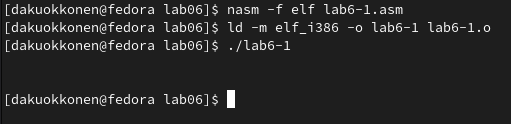


Figure 6: Трансляция, компоновка, запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch. Открываю файл lab6-2.asm для редактирования, ввожу следующую программу. (рис. [7](#fig:007)).

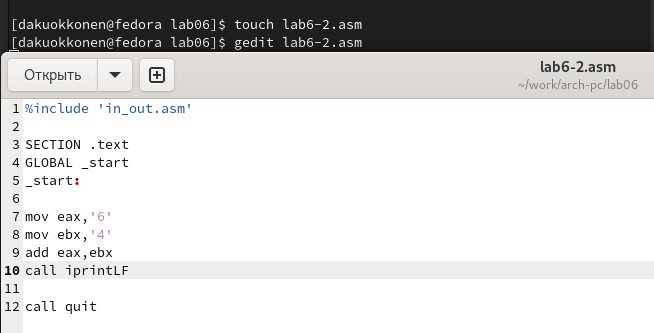


Figure 7: Создание, редактирование файла

Выполняю компиляцию и компоновку, и запускаю исполняемый файл lab6-2. Теперь на экран выводится число 106, т.к. происходит сложение кодов символов “6” и “4”, но при этом программа позволяет вывести именно число, а не символ, кодом которого является это число (рис. [8](#fig:008)).

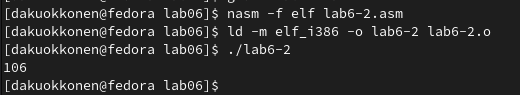


Figure 8: Компиляция, обработка и запуск исполняемого файла

Следом также заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4. Затем провожу компиляцию и компоновку, после этого запускаю исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому на экране мы видим число 10. (рис. [9](#fig:009)).

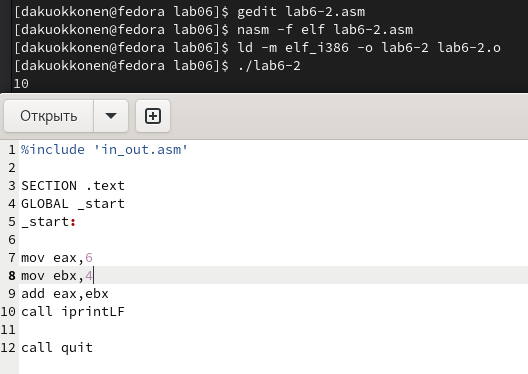


Figure 9: Редактирование, компиляция, обработка и запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint. (рис. [10](#fig:010)).

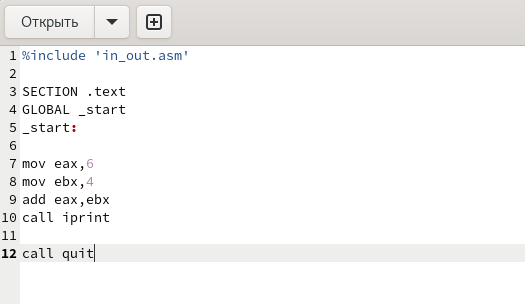


Figure 10: Редактирование файла

Проводим с файлом привычные операции. Вывод изменился (см. рисунок), потому что функция iprint не добавляет к выводу символ переноса строки в отличие от функции iprintLF. (рис. [11](#fig:011)).

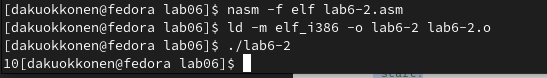


Figure 11: Компиляция, обработка и запуск исполняемого файла

**4.2) Выполнение арифметических операций в NASM**

Создаю файл lab6-3.asm с помощью утилиты touch. Ввожу в созданный файл текст программы для вычисления значения выражения: f(x) = (5\*2+3)/3 (рис. [12](#fig:012)).

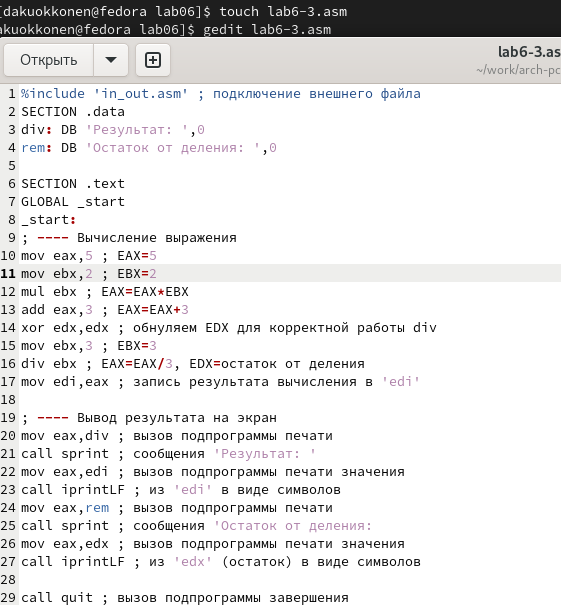


Figure 12: Создание и редактирование файла

Реализовываю исполняемый файл и запускаю его (рис. [13](#fig:013)).

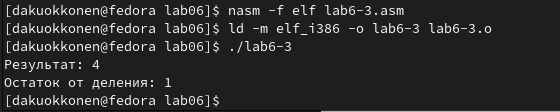


Figure 13: Компиляция, обработка и запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4\*6+2)/5. Выполняю компиляцию и компоновку файла, далее запускаю новый исполняемый файл. Если проведем устную проверку, то убедимся в правильности работы программы. (рис. [14](#fig:014)).

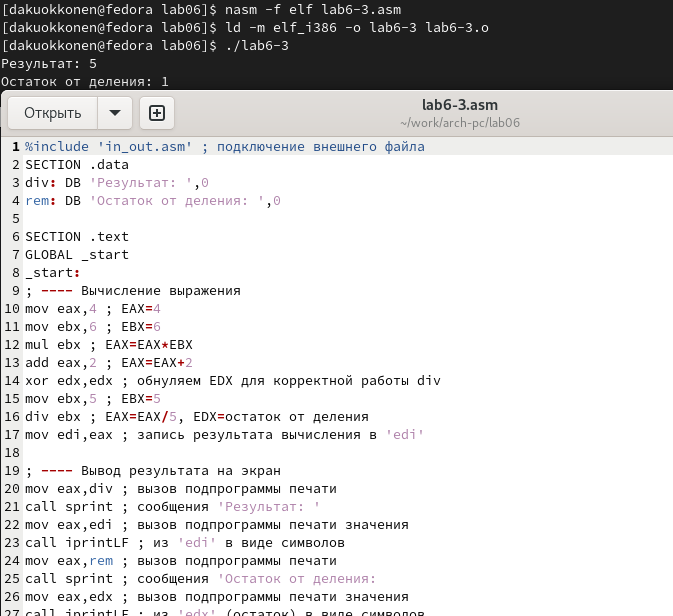


Figure 14: Редактирование, компиляция, обработка и запуск исполняемого файла

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

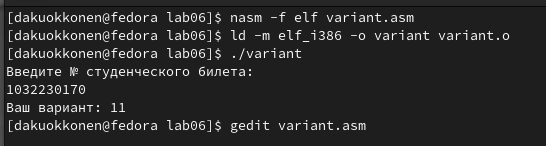


Figure 15: Работа с файлом variant.asm

Провожу компиляцию и компоновку, запускаю исполняемый файл. Ввожу номер своего студенческого билета. Программа выводит, что мой вариант - 11 (рис. [15](#fig:015)).

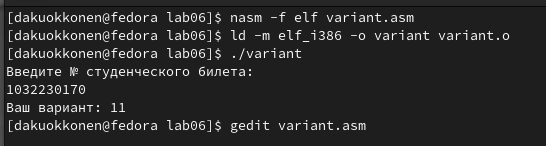


Figure 16: Работа с файлом variant.asm

**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  
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. ??)

Figure 17: Создание файла

Figure 17: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения своего, 11-го варианта, 10\*(x+1)-10. (рис. [18](#fig:018))

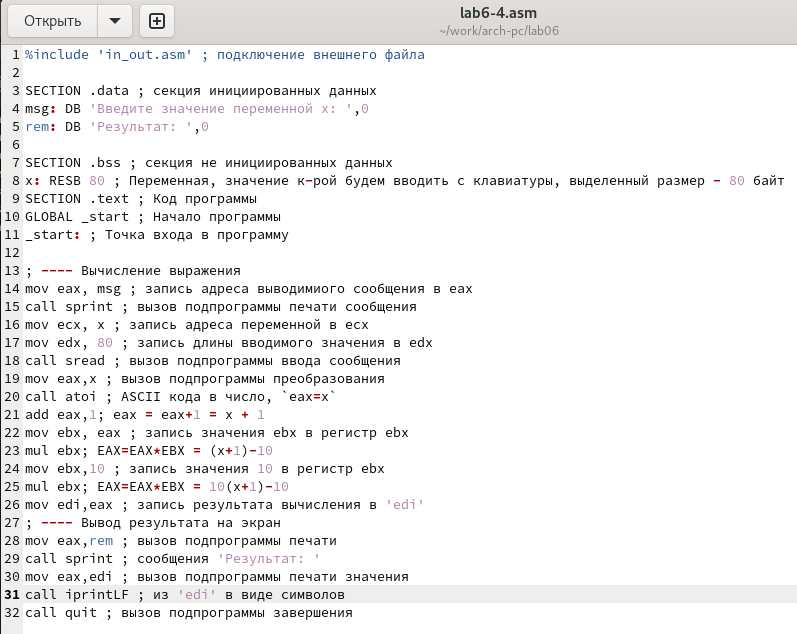


Figure 18: Редактирование файла

Проводим привычные операции и запускаем исполняемый файл, выполняем устную проверку и убеждаемся в правильности работы программы.(рис. [19](#fig:019))

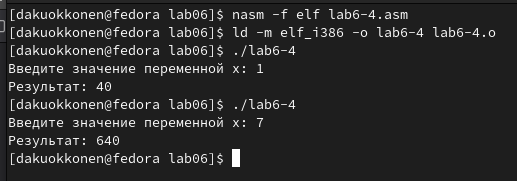


Figure 19: Компиляция, обработка и запуск исполняемого файла

Листинг 4.1 - Программа для вычисления значения выражения из варианта 11.

%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data ; секция инициированных данных  
msg: DB 'Введите значение переменной х: ',0  
rem: DB 'Результат: ',0  
SECTION .bss ; секция не инициированных данных  
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
; ---- Вычисление выражения  
mov eax, msg ; запись адреса выводимиого сообщения в eax  
call sprint ; вызов подпрограммы печати сообщения  
mov ecx, x ; запись адреса переменной в ecx  
mov edx, 80 ; запись длины вводимого значения в edx  
call sread ; вызов подпрограммы ввода сообщения  
mov eax,x ; вызов подпрограммы преобразования  
call atoi ; ASCII кода в число, `eax=x`  
add eax,1; eax = eax+1 = x + 1  
mov ebx, eax ; запись значения ebx в регистр ebx  
mul ebx; EAX=EAX\*EBX = (x+1)-10  
mov ebx,10 ; запись значения 10 в регистр ebx  
mul ebx; EAX=EAX\*EBX = 10(x+1)-10  
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 iprintLF ; из 'edi' в виде символов  
call quit ; вызов подпрограммы завершения

# 5 Выводы

При выполнении лабораторной работы я освоила арифметические инструкции языка ассемблера NASM и приобрел практический опыт работы с ними.

# Список литературы

1. [Архитектура компьютера и ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089086/mod_resource/content/0/Лабораторная%20работа%20№6.%20Арифметические%20операции%20в%20NASM..pdf)