Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Самойлова Софья Дмитриевна

Содержание

# 1 Цель работы

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

# 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Ответы на вопросы
4. 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 №7. Перехожу в созданный каталог с помощью утилиты cd и создаю файл lab6-1.asm (рис. 1).

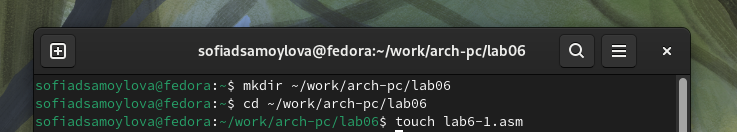


Рис. 1: Создание директории

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. 2).

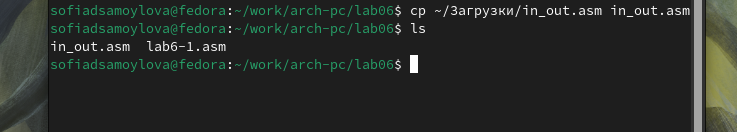


Рис. 2: Создание копии файла

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax, создаю исполняемый файл программы и запускаю его (рис. 3).

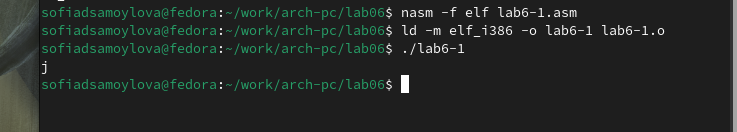


Рис. 3: Работа файла

Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

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



Рис. 4: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 5). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

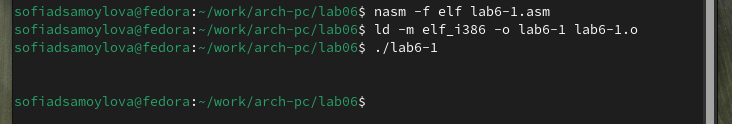


Рис. 5: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touchи ввожу в файл текст другой программы для вывода значения регистра eax (рис. 6).

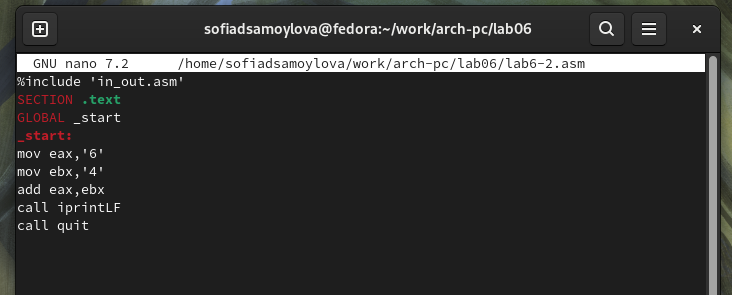


Рис. 6: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 7). Теперь вывод число 106, что является результатом сложения кодов символов “6” и “4”.

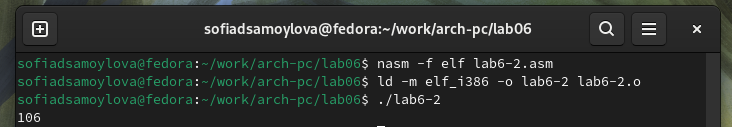


Рис. 7: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4(рис. 8).

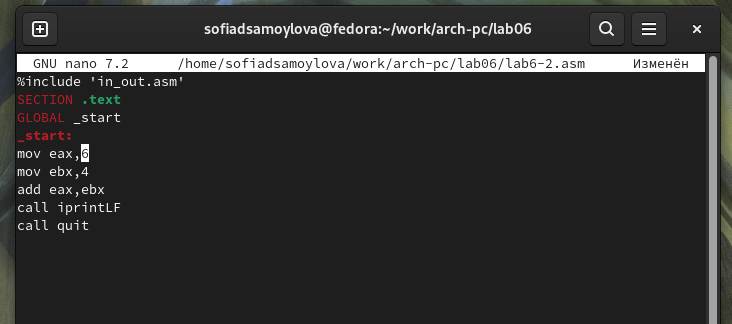


Рис. 8: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 9). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

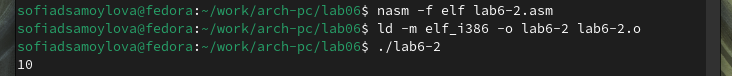


Рис. 9: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint(рис. 10).

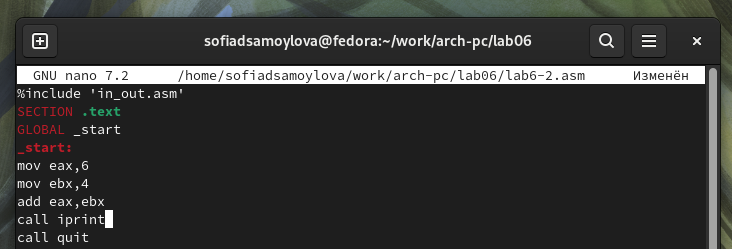


Рис. 10: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 11). Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

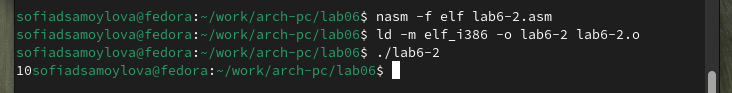


Рис. 11: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch и ввожу в созданный файл текст программы для вычисления значения выражения *f(x) = (5*  2 + 3)/3\* (рис. 12).

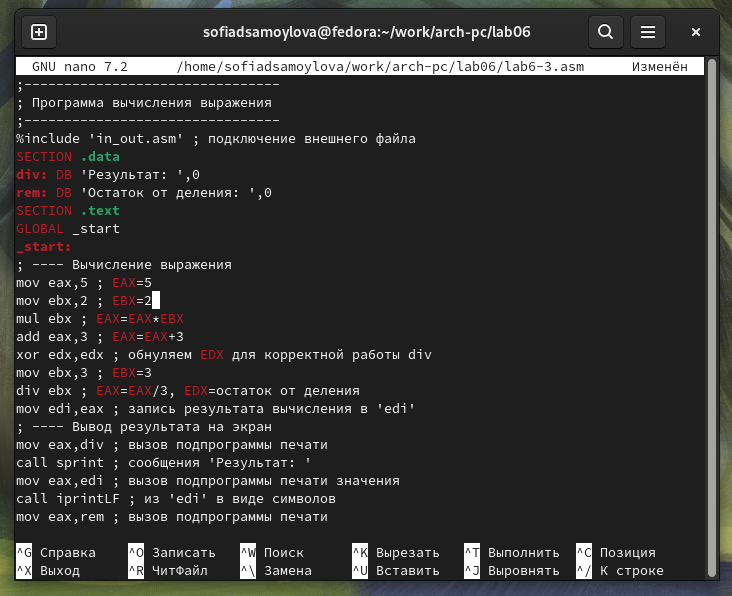


Рис. 12: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 13).

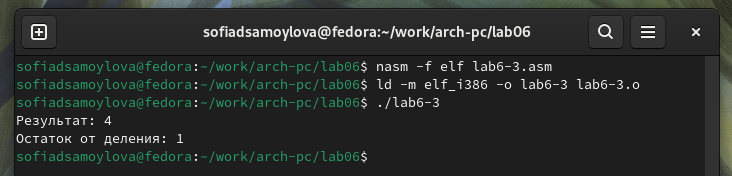


Рис. 13: Запуск файла

Изменяю программу так, чтобы она вычисляла значение выражения *f(x) = (4*  6 + 2)/5\* (рис. 14).

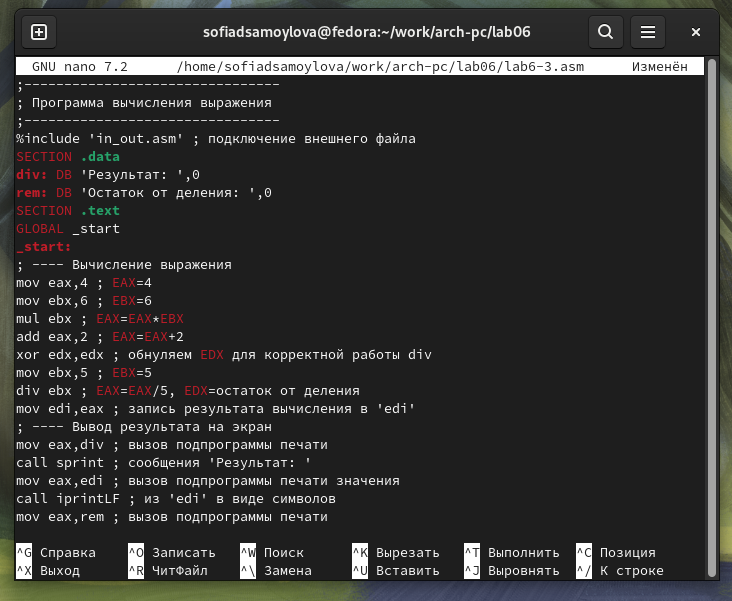


Рис. 14: Изменение программы

Создаю исполняемый файл и проверяю его работу (рис. 15).

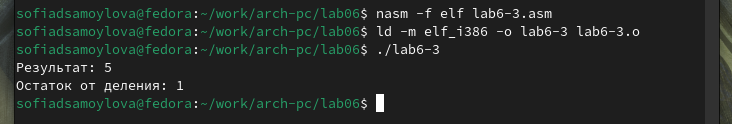


Рис. 15: Изменение программы

Создаю файл variant.asm в каталоге ~/work/arch-pc/lab06, ввожу текст программы вычисления варианта задания по номеру студенческого билета, создаю исполняемый файл и запускаю его (рис. 16).

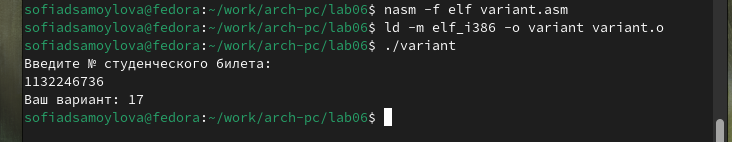


Рис. 16: Запуск файла

## 4.3 Ответы на вопросы

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

mov eax,rem  
call sprint

1. 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

## 4.4 Выполнение заданий для самостоятельной работы

Программа для вычисления выражения варианта №17:

;--------------------------------  
; Программа вычисления выражения 18(x + 1)/6  
;------------------------------  
%include 'in\_out.asm' ; подключение внешнего файла  
  
SECTION .data ; секция инициированных данных  
msg: DB 'Введите значение переменной x: ', 0  
rem: DB 'Результат: ', 0  
  
SECTION .bss ; секция не инициированных данных  
x: RESB 80 ; Переменная, значение которой будем вводить с клавиатуры, выделенный размер - 80 байт  
  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
  
\_start: ; Точка входа в программу  
 ; ---- Ввод значения x  
 mov eax, msg ; запись адреса выводимого 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 ; преобразование ASCII кода в число (eax = x)  
   
 ; ---- Вычисление выражения 18(x + 1) / 6  
 add eax, 1 ; eax = x + 1  
 mov ebx, 18 ; запись значения 18 в ebx  
 mul ebx ; EAX = EAX \* EBX (EAX = 18 \* (x + 1))  
   
 mov ebx, 6 ; запись значения 6 в ebx для деления  
 xor edx, edx ; обнуляем EDX для корректной работы div  
 div ebx ; EAX = EAX / EBX (EAX = (18 \* (x + 1)) / 6)  
  
 ; ---- Вывод результата на экран  
 mov edi, eax ; запись результата вычисления в 'edi'  
   
 mov eax, rem ; вызов подпрограммы печати  
 call sprint ; сообщение 'Результат: '  
   
 mov eax, edi ; вызов подпрограммы печати значения  
 call iprintLF ; из 'edi' в виде символов  
   
 call quit ; вызов подпрограммы завершения

Создаю файл lab6-4.asm в каталоге ~/work/arch-pc/lab06, вношу программу для вычисления выражения 18(x + 1)/6, создаю исполняемый файл и проверяю его работу для значений *х1 = 3* и *х2 = 1* (рис. 17).

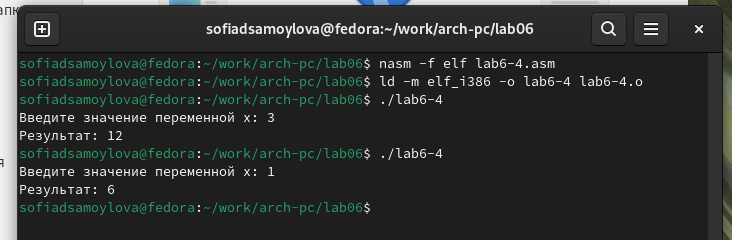


Рис. 17: Работа программы

# 5 Выводы

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