Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Кирьянова Екатерина Андреевна

Содержание

# 1 Цель работы

Освоить арифметические инструкции 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.

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

Создаю новый каталог и файл в нем (рис. 1).

Рис. 1: Создание

Рис. 1: Создание

Ввожу текст программы из листинга 6.1 (рис. 2)

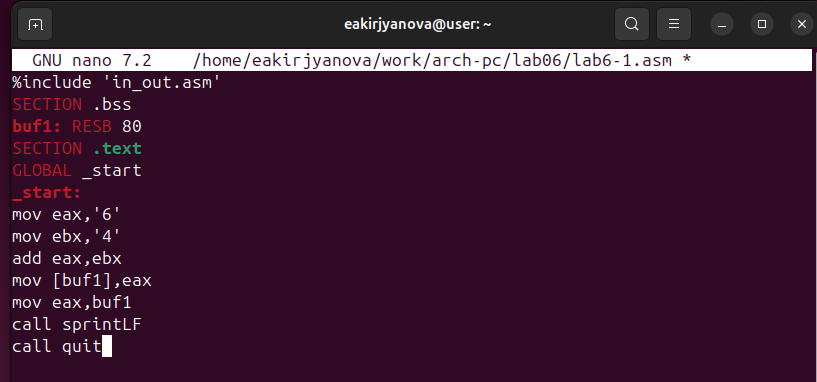


Рис. 2: Программа

Создаю исполняемый файл и запускаю его (рис. 3).

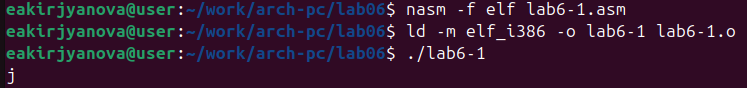


Рис. 3: Запуск

Редактирую текст программы (рис. 4).

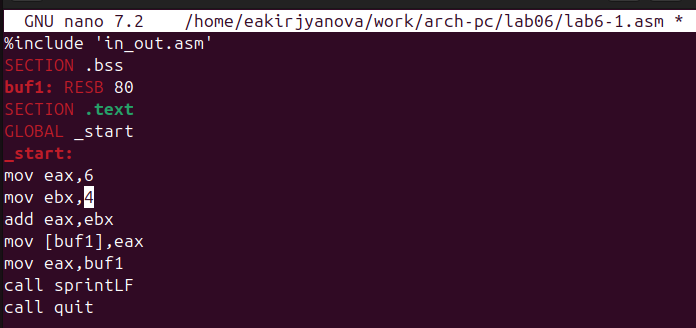


Рис. 4: Редактирование

Создаю обновленный исполняемый файл и запускаю его (рис. 5).

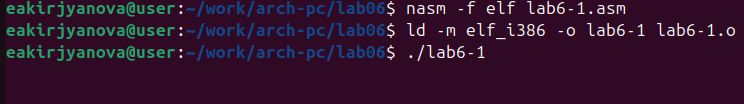


Рис. 5: Запуск

Создаю новый файл (рис. 6).

Рис. 6: Новый файл

Рис. 6: Новый файл

Ввожу текст программы из листинга 6.2 (рис. 7).

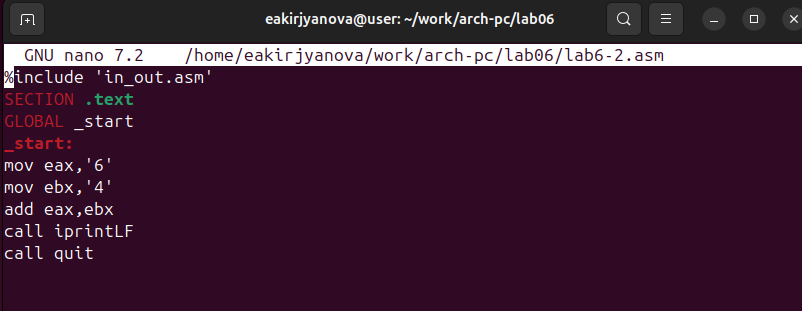


Рис. 7: Вторая программа

Создаю исполняемый файл и запускаю его (рис. 8).

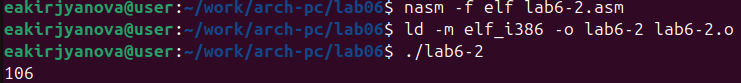


Рис. 8: Запуск программы

Аналогично предыдущему меняю символы на числа (рис. 9).

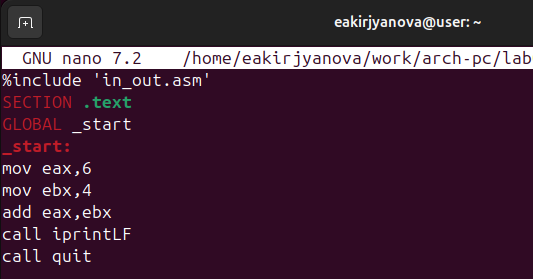


Рис. 9: Редактирование

Создаю исполняемый файл и запускаю его (рис. 10).

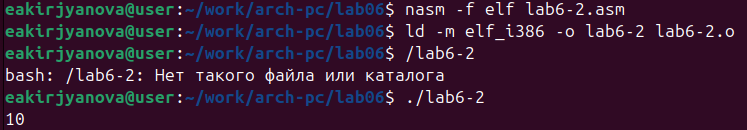


Рис. 10: Запуск

Меняю iprintLF на iprint(рис. 11).

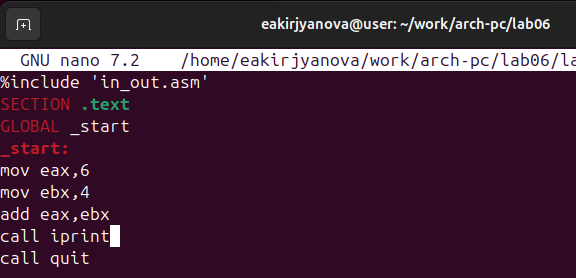


Рис. 11: Редактирование

Создаю исполняемый файл и запускаю его (рис. 12).

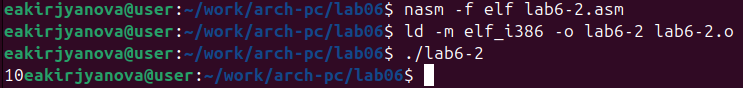


Рис. 12: Запуск

Вывод изменился, так как iprintLF переносит строку, а iprint нет.

## 4.2 Выполнение арифметических операций в NASM

Создаю новый файл (рис. 13).

Рис. 13: Новый файл

Рис. 13: Новый файл

Ввожу текст программы из листинга 6.3 (рис. 14).

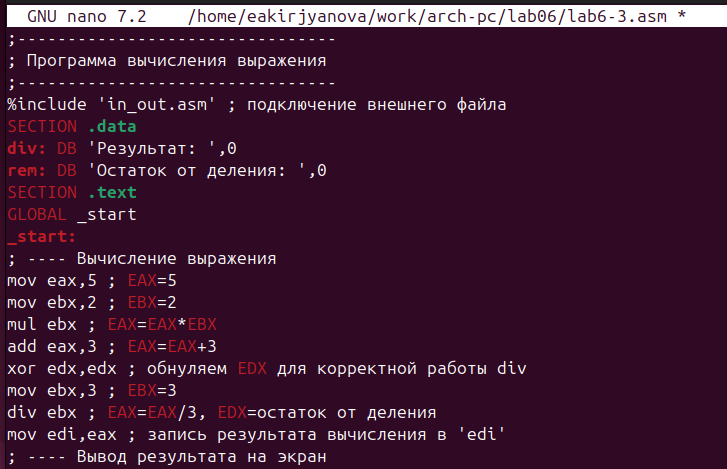


Рис. 14: Треть программа

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

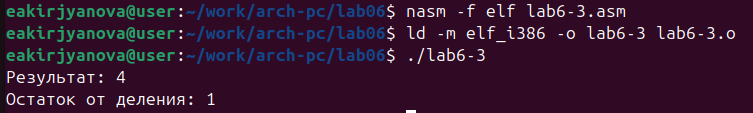


Рис. 15: Запуск

Редактирую (рис. 16).

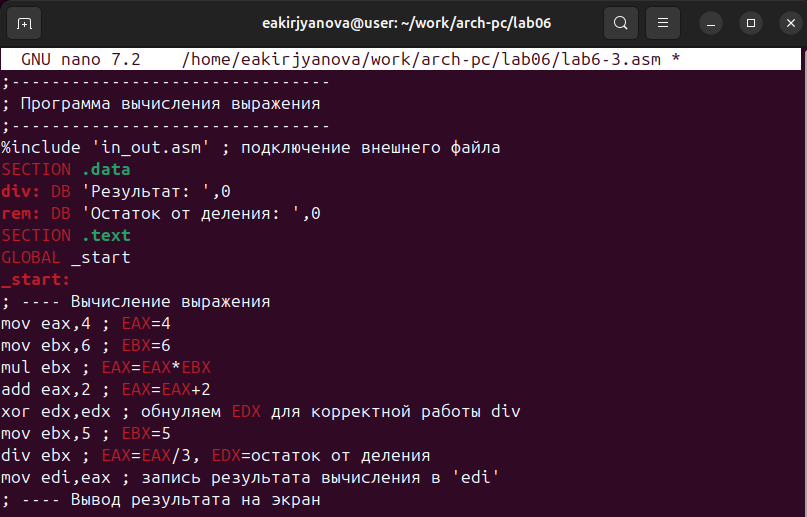


Рис. 16: Редактирование

Создаю исполняемый файл и запускаю (рис. 17).

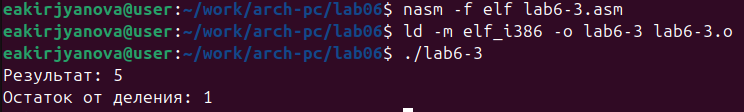


Рис. 17: Запуск

Создаю новый файл (рис. 18).

Рис. 18: Создание файла

Рис. 18: Создание файла

Ввожу текст программы из листинга 6.4 (рис. 19).

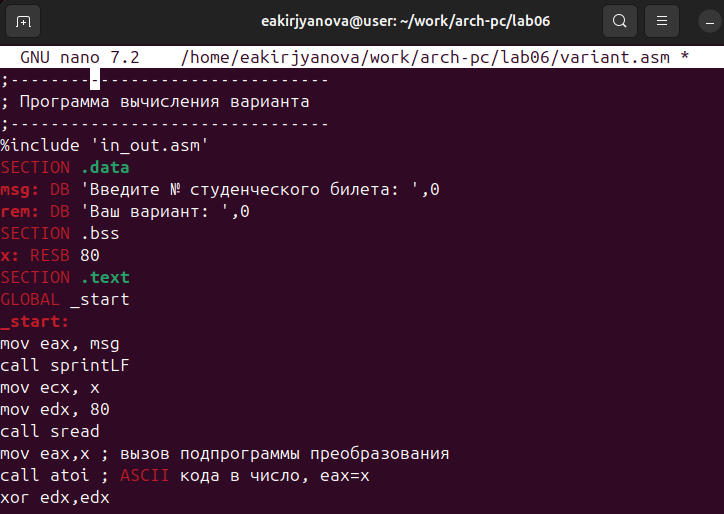


Рис. 19: Четвертая программа

Создаю исполняемый файл и запускаю его (рис. 20).

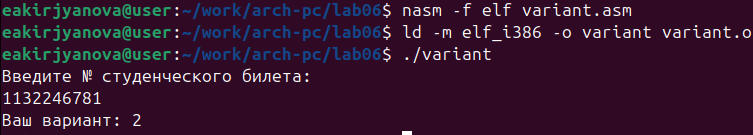


Рис. 20: Запуск

Ответы на вопросы: 1. mov eax,rem ; call sprint 2. 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

1. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
2. 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
3. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
4. inc edx увеличивает значение регистра edx на 1
5. mov eax,edx call iprintLF

## 4.3 Задание для самостоятельной работы

Создаю файл (рис. 21).

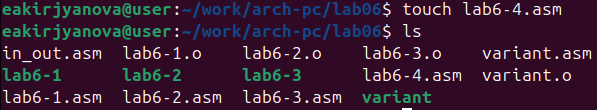


Рис. 21: Создание файла

Ввожу текст программы для вычисления выражения под 2 вариантом (рис. 22).

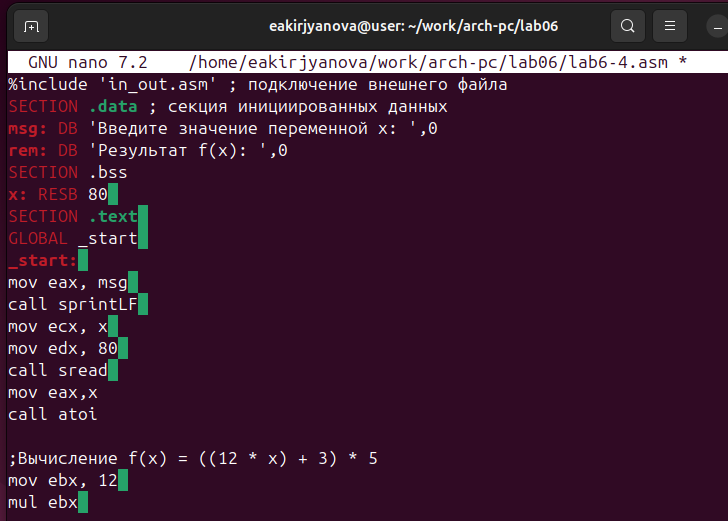


Рис. 22: Программа

Создаю исполняемый файл, запускаю и ввожу значения x1, x2 для проверки (рис. 23).

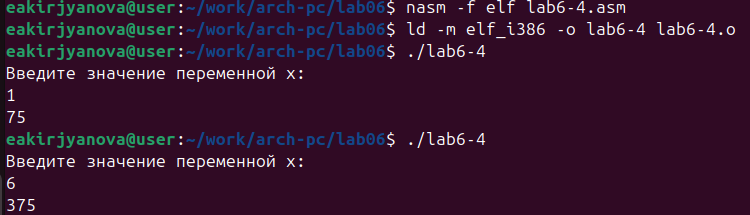


Рис. 23: Выражение

Текст программы:

%include ‘in\_out.asm’ SECTION .data msg: DB ‘Введите значение переменной х:’,0 rem: DB ‘Результат f(x):’,0 SECTION .bss x: RESB 80 SECTION .text GLOBAL \_start \_start: mov eax, msg call sprintLF mov ecx, x mov edx, 80 call sread mov eax,x call atoi

;Вычисление f(x) = ((12 \* x) + 3) \* 5 mov ebx, 12 mul ebx add eax, 3 mov ebx,5 mul ebx mov edi,eax mov eax, edi call iprintLF call quit

# 5 Выводы

В ходе выполнения данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

# 6 Список литературы

1. [Лабораторная работа №6](https://esystem.rudn.ru/pluginfile.php/2089542/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)