Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Гасанова Шакира Чингизовна

Содержание

1 Цель работы ............................................................................................……………………………….............. 3

2 Задание ................................................................................................................…………………………………. 4

3 Теоретическое введение ....................................................................................………………………… 5

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM.……………….......................................................6

4.2 Выполнение арифметических операций в NASM......………….......................................10

4.2.1 Ответы на вопросы по программе...................................................………………………......12

4.3 Выполнение заданий для самостоятельной работы.......................................……….13

5 Выводы ...............................................................................................................………………………………….16

6 Источники ..........................................................................................................………………………………..17

# 1 Цель работы

Целью данной лабораторной работы является освоение арифметческих инструкций 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, для которых необходимо указать, где находятся обрабатываемые данные. Операнд может находиться в регистре или в ячейке памяти. Существуют три основных способа адресации: 1. Регистровая адресация: данные хранятся в регистрах, и команда содержит указание на их имена. Например: mov ax, bx. 2. Непосредственная адресация: значение операнда записывается непосредственно в команде. Например: mov ax, 2. 3. Адресация памяти: команда содержит указание на адрес в памяти, где находятся данные. В этом случае используется символическое обозначение ячейки памяти, над содержимым которой выполняется операция. Ввод данных с клавиатуры и их вывод на экран происходит в виде символов. Для кодировки символов используется таблица ASCII (American Standard Code for Information Interchange – Американский стандартный код для обмена информацией). Согласно этому стандарту, каждый символ кодируется одним байтом. Инструкции NASM не предусматривают непосредственного вывода чисел в их числовом представлении. Поэтому, чтобы вывести число, его необходимо сначала преобразовать в ASCII-коды соответствующих цифр и выводить уже эти коды. В противном случае, при выводе числа без преобразования, оно будет интерпретировано как последовательность ASCII-символов, соответствующих байтам числа. Экран отобразит символы, а не само число. Аналогичная ситуация возникает при вводе данных с клавиатуры. Введенные значения воспринимаются как символы, что делает невозможным выполнение арифметических операций без предварительного преобразования. Для корректной работы необходимо преобразовывать символы ASCII в числа и обратно.

# 4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM Перехожу в созданный каталог с помощью утилиты cd и с помощью команды touch создаю файл lab6-1.asm (рис. 1).

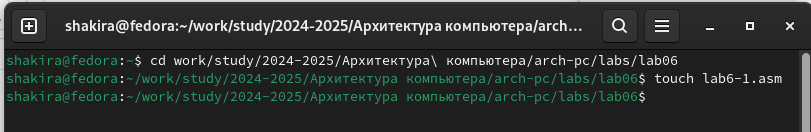


Рис. 1: 1

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

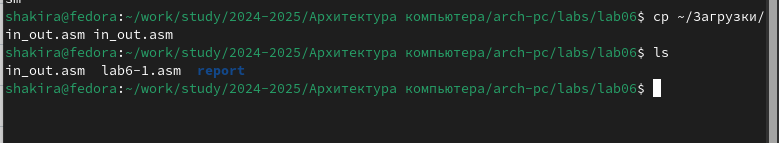


Рис. 2: 2

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. 3).

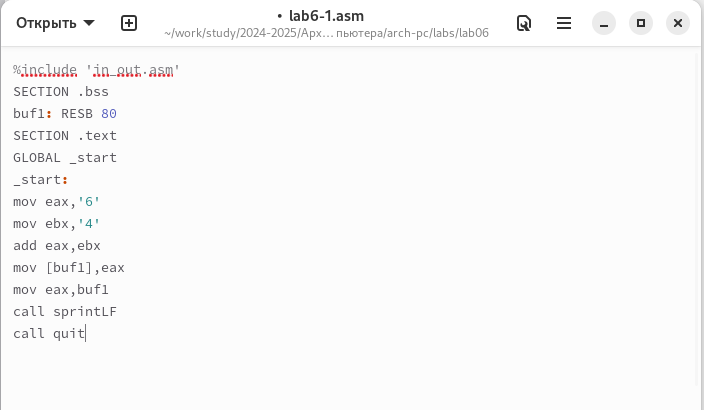


Рис. 3: 3

Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6 (рис. 4).

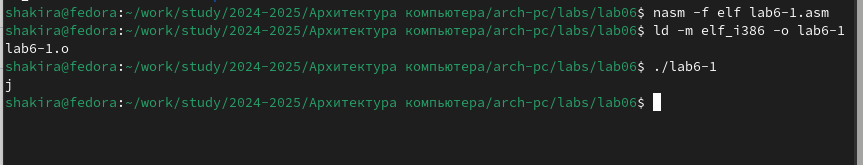


Рис. 4: 4

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 5).



Рис. 5: 5

Создаю новый исполняемый файл программы и запускаю его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран (рис. 6).

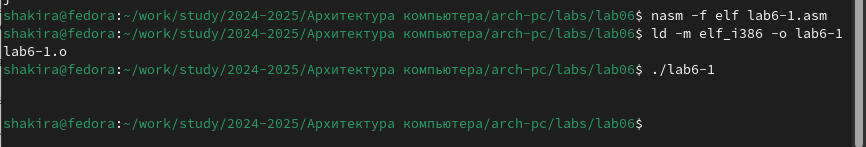


Рис. 6: 6

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 7).

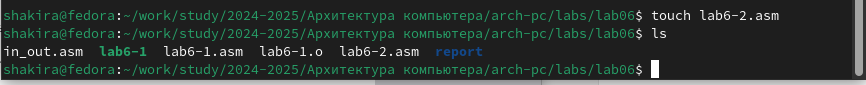


Рис. 7: 7

Ввожу в файл текст другойпрограммы для вывода значения регистра eax (рис. 8).



Рис. 8: 8

Создаю и запускаю исполняемый файл lab6-2. Теперь выводится число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4” (рис. 9).

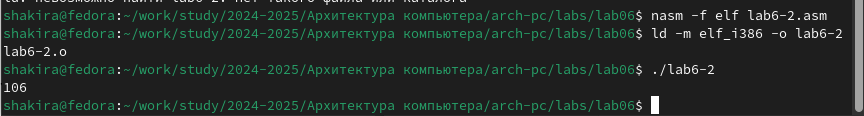


Рис. 9: 9

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 10).



Рис. 10: 10

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. 11).

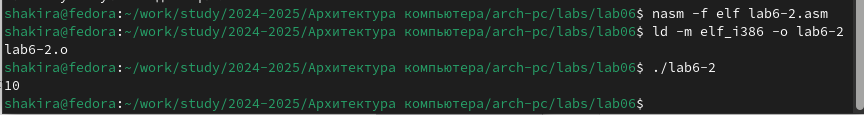


Рис. 11: 11

Заменяю в тексте программы функцию iprintLF на iprint (рис. 12).



Рис. 12: 12

Создаю и запускаю новый исполняемый файл. 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. 13).

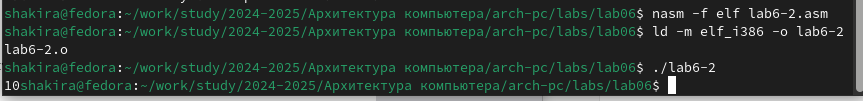


Рис. 13: 13

4.2 Выполнение аримфетических операций в NASM Создаю файл lab7-3.asm с помощью утилиты touch (рис. 14).

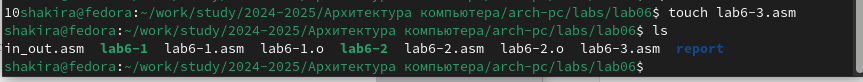


Рис. 14: 14

Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3 (рис. 15).

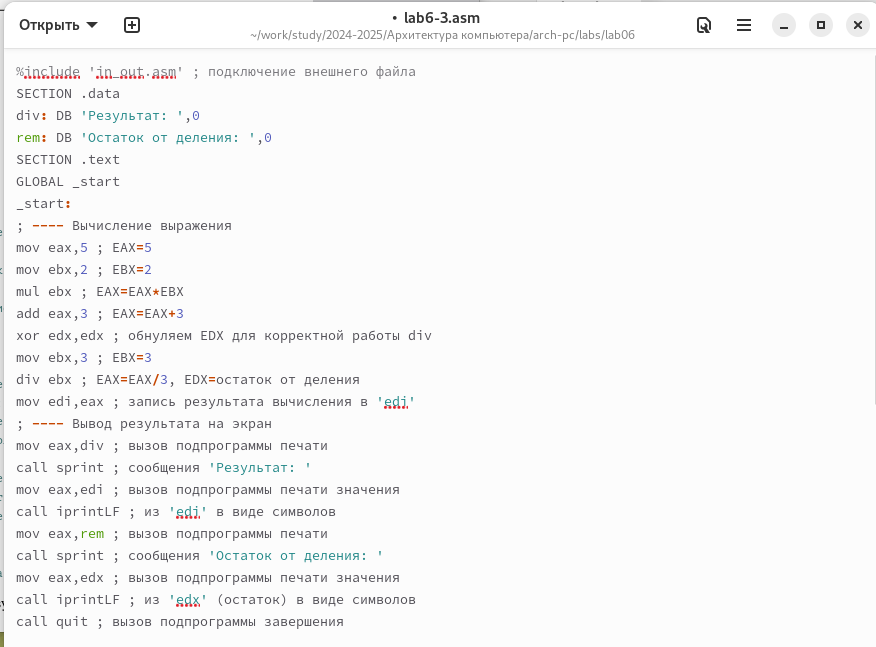


Рис. 15: 15

Создаю исполняемый файл и запускаю его (рис. 16).

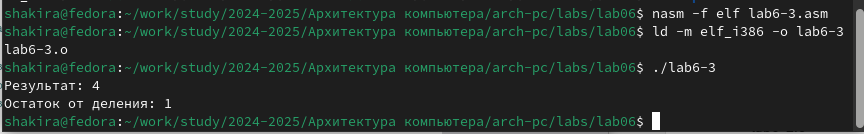


Рис. 16: 16

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. 17).

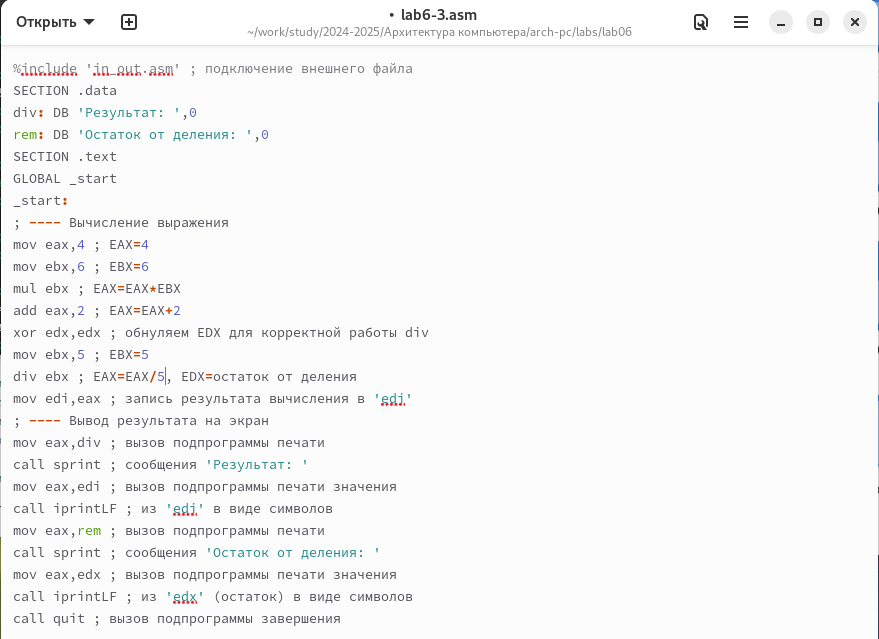


Рис. 17: 17

Создаю и запускаю новый исполняемый файл. Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно (рис. 18).

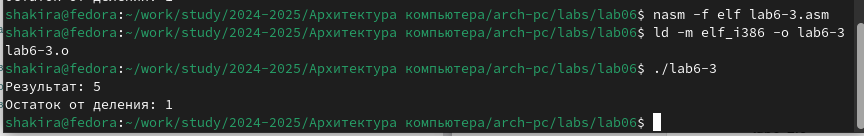


Рис. 18: 18

Создаю файл variant.asm с помощью утилиты touch (рис. 19).

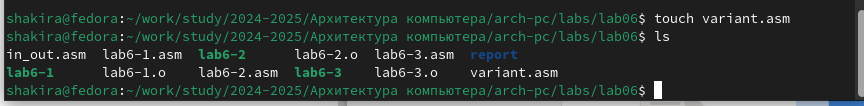


Рис. 19: 19

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 20).

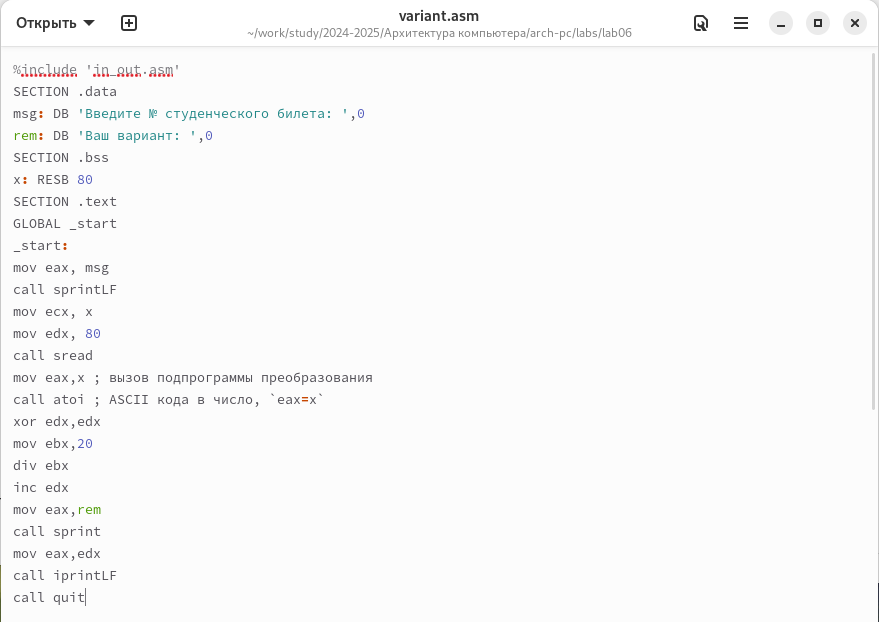


Рис. 20: 20

Создаю и запускаю исполняемый файл (рис. 21). Ввожу номер своего студентечского билета, программа вывела, что мой вариант - 4 (рис. 21).

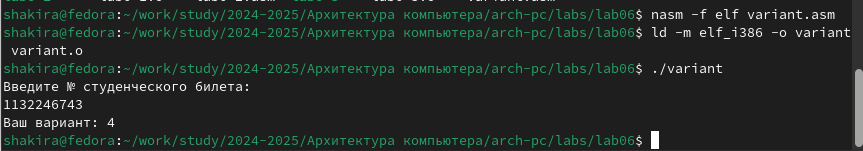


Рис. 21: 21

4.2.1 Ответы на вопоросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода: mov eax,rem call sprint

2. Инструкция mov ecx, x используется для помещения адреса вводимой строки x в регистр ecx. Инструкция mov edx, 80 задаёт длину вводимой строки, записывая значение 80 в регистр edx. Вызов подпрограммы call sread из внешнего файла обе спечивает ввод сообщения с клавиатуры.Инструкция mov ecx, x используется для помещения адреса вводимой строки x в регистр ecx. Инструкция mov edx, 80 задаёт длину вводимой строки, записывая значение 80 в регистр edx. Вызов подпрограммы call sread из внешнего файла обеспечивает ввод сообщения с клавиатуры.

3. Инструкция call atoi вызывает подпрограмму из внешнего файла, которая преобразует символ в формате ASCII в целое число и сохраняет результат в регистре eax.

4. 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

5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx

6. Инструкция inc edx увеличивает значение регистра edx на 1

7. За вывод на экран результатов вычислений отвечают строки: mov eax,edx call iprintLF

4.3 Выполнение заданий для самостоятельной работы Создаю файл lab7-4.asm с помощью утилиты touch (рис. 22).

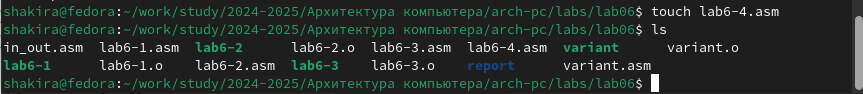


Рис. 22: 22

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения 4/3\*(x − 1) + 5. Это выражение было под вариантом 4 (рис. 23).



Рис. 23: 23

Создаю и запускаю исполняемый файл. При вводе значения 4, вывод - 9 (рис. 24).

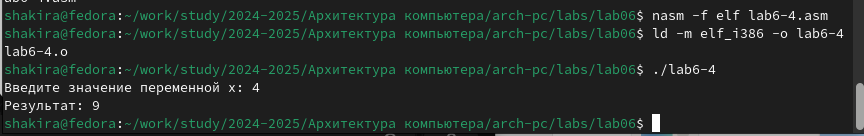


Рис. 24: 24

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе. Программа отработала верно (рис. 25).

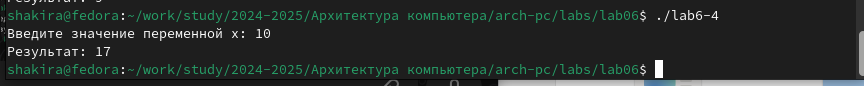


Рис. 25: 25

Код программы:

%include 'in\_out.asm'

SECTION .data

msg: DB 'Введите значение переменной x: ',0

rem: DB 'Результат: ',0

SECTION .bss

x: RESB 80

SECTION .text

GLOBAL \_start

\_start:

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

sub eax,1; eax = eax - 1 = x - 1

mov ebx,4

mul ebx ; eax = eax \* 4 = 4 \* (x - 1)

mov ebx,3

div ebx ; eax = eax / 3 = 4/3 \* (x - 1)

add eax,5; eax = eax + 5 = 4/3 \* (x-1) + 5

mov edi,eax ; запись результата вычисления в 'edi'

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

# 5 Выводы

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

# Список литературы

1. [Архитектура ЭВМ (rudn.ru)](https://esystem.rudn.ru/)
2. Таблица ASCII