Отчет по лабораторной работе №6

Дисциплина: Архитектура компьютера

Баазова Нина Эдгаровна

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Выполнение порядка лабораторной рабоы №6
2. Задание для самостоятельной работы

# 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес опе- ранда 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 ко- манде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символи- ческое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

# 4 Выполнение лабораторной работы

1. Порядок выполнения лабораторной работы:

1.1 Символьные и численные данные в NASM.

1). Создаем каталог для программам лабораторной работы № 6, перейдем в него и создадим файл lab6-1.asm:

mkdir ~/work/arch-pc/lab06 cd ~/work/arch-pc/lab06 touch lab6-1.asm

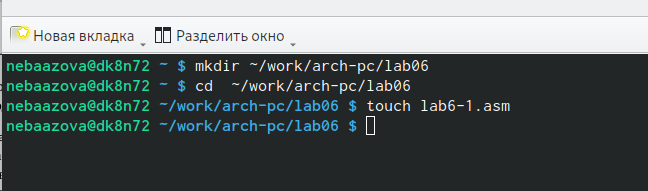


рис 1

2). Вставим в него код из Листинга 6.1. Создадим исполняемый файл и запустим его и видем результат: j.

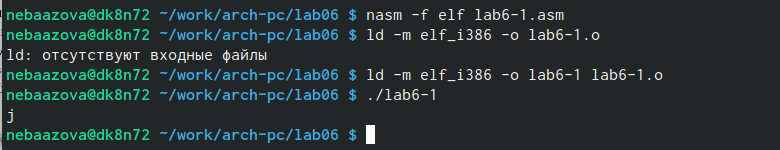


рис 2

3). Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы (Листинг 6.1) следующим образом:

заменим строки mov eax,‘6’ mov ebx,‘4’

на строки mov eax,6 mov ebx,4

Затем также создаем исполняемый файл и запускаем его, результатом ничего не будет.

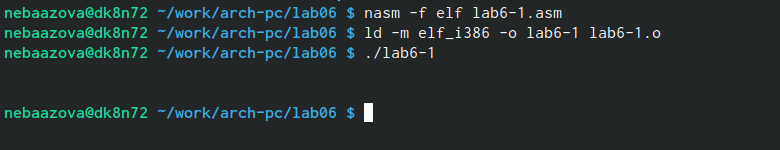


рис 3

4). Создаем файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введим в него текст программы из листинга 6.2. С помощью команды: touch ~/work/arch-pc/lab06/lab6-2.asm Проверяем с помощью команды ls и создаем исполняемый файл и запускаем его. Результатом работы получим: 106.

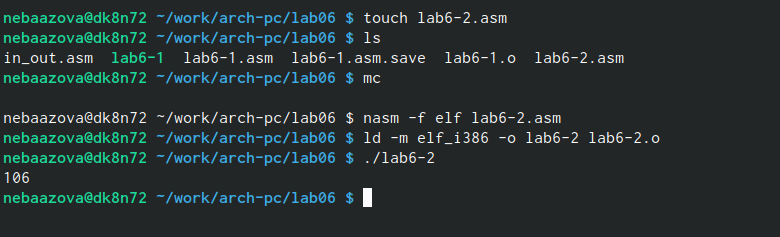


рис 4

5). Аналогично предыдущему примеру изменим символы на числа.

заменим строки mov eax,‘6’ mov ebx,‘4’

на строки mov eax,6 mov ebx,4

Не забудем создать исполняемый файл и запустить его. В результате получим число 10.

Если мы заменем функцию iprintLF на iprint, создадим и запустим отредактированный файл, то увидим, что программа выполнилась без переноса на новую строку. В этом и состоит разница этих функций. Функция iprintLF запрашивает перенос на новою строку, а функция iprint - нет. В результате тоже получим 10, но без переноса следующей строки.

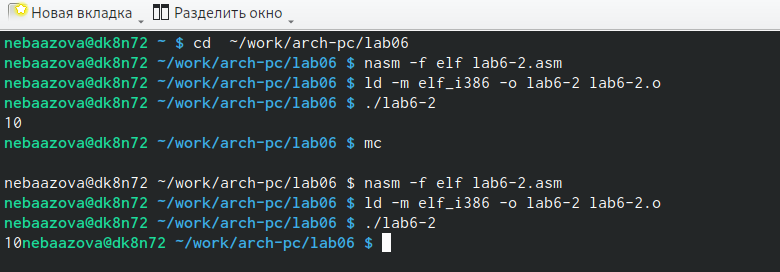


рис 5

1.2 Выполнение арифметических операций в NASM.

1). Создаем файл lab6-3.asm в каталоге ~/work/arch-pc/lab06: touch ~/work/arch-pc/lab06/lab6-3.asm. Копируем туда Листинг 6.3, создаем исполняемый файл и запускаем его. Результат работы программы получится следующим: (рис 6)

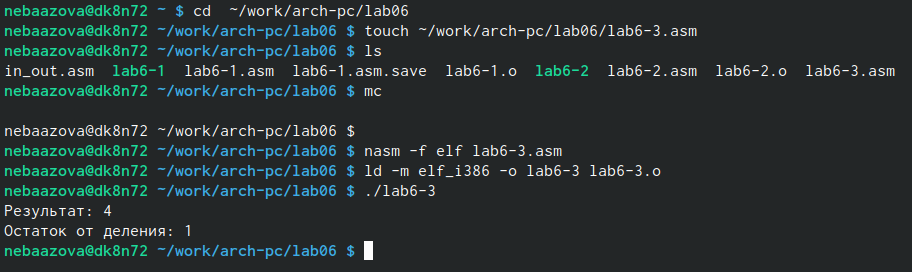


рис 6

2). Изменем текст программы для вычисления выражения 𝑓(𝑥) = (4 ∗ 6 + 2)/5. Создаем исполняемый файл и проверяем его работу.

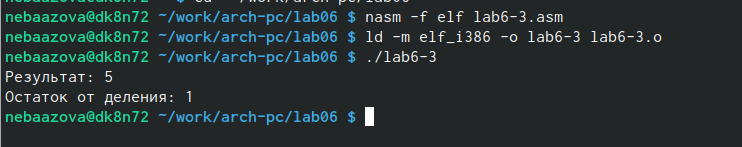


рис 7

3). Создаем файл variant.asm в каталоге ~/work/arch-pc/lab06: touch ~/work/arch-pc/lab06/variant.asm. Внимательно изучаем текст программы из Листинга 6.4 и вводим в файл variant.asm. Создаем исполняемый файл и запускаем его. Проверяем результат работы программы, вычислив номер варианта аналитически. Мне попался вариант 17 (функция 18\*(x+1)/6).

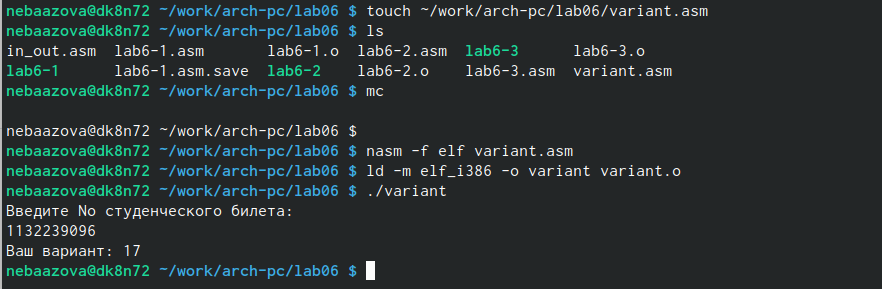


рис 8

1.3 ОТВЕТЫ НА ВОПРОСЫ:

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

2). Следующие инструкции используются для: mov ecx, x - перемещает адрес вводимой строки x в регистр ecx; mov edx, 80 - запись в регистр edx длину вводимой строки; call sread - вызов подпрограммы из внешнего файла, обеспечивающий ввод сообщения с клавиатуры.

3). call atoi - используется для вызова подпрограммы из внешнего файла, которая преобразует ASII код символа в целое число и записывает результат в регист 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” увеличивают значения регистра edx на 1.

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

ВЫВОД: Мы ознакомились с порядком выполнении лабораторной работы №6.

1. Задание для самостоятельной работы:

1). Напишем программу вычисления выражения 𝑦 = 𝑓(𝑥). Программа должна выводить выражение для вычисления, выводить запрос на ввод значения 𝑥, вычислять задан- ное выражение в зависимости от введенного 𝑥, выводить результат вычислений. Вид функции 𝑓(𝑥) выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. У меня это номер 17.

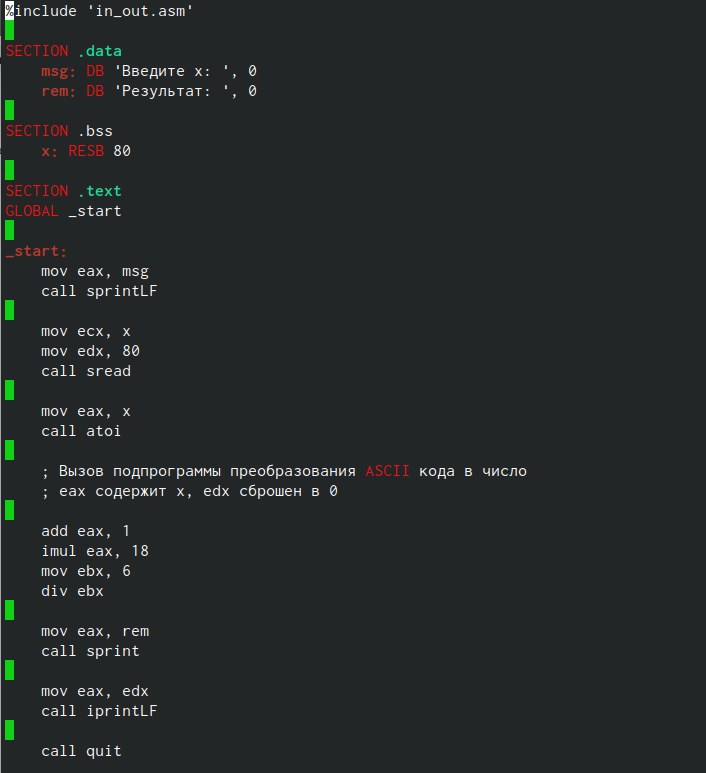


рис 9

2). Создадим исполняемый файл и проверим его работу для значений 𝑥1 и 𝑥2 из 6.3.

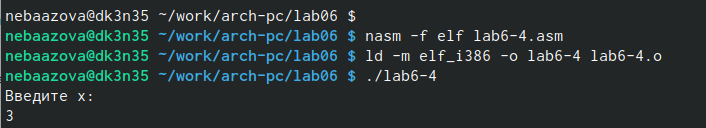


рис 10

ВЫВОД: Мы выполнили задание для самостоятельной работы.

# 5 Вывод лабораторной работы

Мы освоили арифметические инструкции языка ассемблера NASM.