Отчёт по лабораторной работе №6

Архитектура компьютеров и операционные системы.

Брыляков Никита Евгеньевич

Содержание

# 1 Цель работы

Освоение арифметических 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. 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.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 (символы с кодами 0-127) является универсальной (см. Приложение.), а вторая (коды 128-255) 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). По- этому, 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 дан- ные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций.

# 4 Выполнение лабораторной работы

## 4.1 Начало работы

Создаю каталог для работы, перехожу в него и создаю файл lab6-1.asm (рис. [1]).

fig:
Создание каталога и файла внутри

Рис. 1 Создание каталога и файла внутри

Вручную переношу файл in\_out.asm и проверяю его наличие в каталоге (рис. [2]).

fig:
Просмотр каталога

Рис. 2 Просмотр каталога

Открываю файл lab6-1.asm и ввожу нужную программу (рис. [3]).

fig:
Открытие и ввод программы

Рис. 3 Открытие и ввод программы

Создаю исполняемый файл и запускаю его (рис. [4]).

fig:
Создание и запуск

Рис. 4 Создание и запуск

Заменяю некоторые строки в программе (рис. [5]).

fig:
Замена кода в программе

Рис. 5 Замена кода в программе

Создаю новый исполняемый файл и запускаю его. Код 10 соответствует символу переноса строки. Он не выводится на экран(рис. [6]).

fig:
Создание и запуск

Рис. 6 Создание и запуск

Создаю новый файл в каталоге (рис. [7]).

fig:
Создание файла

Рис. 7 Создание файла

Ввожу в него текст программы (рис. [8]).

fig:
Ввод программы

Рис. 8 Ввод программы

Создаю исполняемый файл и запускаю его. В данном случае программа производит сложение кодов символов и выводит число, а не символ, кодом которого является это число (рис. [9]).

fig:
Создание и запуск

Рис. 9 Создание и запуск

Заменяю некоторые строки в программе (рис. [10]).

fig:
Замена кода в программе

Рис. 10 Замена кода в программе

Создаю новый исполняемый файл и запускаю его. Теперь программа складывает сами числа (рис. [11]).

fig:
Создание и запуск

Рис. 11 Создание и запуск

Заменяю функцию iprintLF на iprint (рис. [12]).

fig:
Изменение программы

Рис. 12 Изменение программы

Создаю исполняемый файл и запускаю его. Вывод не изменился. С iprintLF символ перевода строки не отображался, а iprint, в отличие от iprintLF, не производит перевод строки (рис. [13]).

fig:
Создание и проверка

Рис. 13 Создание и проверка

## 4.2 Выполнение арифметических операций в NASM

Создаю новый файл в каталоге (рис. [14]).

fig:
Создание файла

Рис. 14 Создание файла

Ввожу в него текст программы. (рис. [15]).

fig:
Ввод программы

Рис. 15 Ввод программы

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

fig:
Создание и запуск

Рис. 16 Создание и запуск

Изменяю текст программы для вычисления выражения 𝑓(𝑥) = (4 ∗ 6 + 2)/5. (рис. [17]).

fig:
Изменение программы

Рис. 17 Изменение программы

Создаю новый исполняемый файл и запускаю его. Всё верно (рис. [18]).

fig:
Создание и запуск

Рис. 18 Создание и запуск

Создаю файл variant.asm в каталоге (рис. [19]).

fig:
Создание файла

Рис. 19 Создание файла

Ввожу в него текст программы для вычисления варианта. (рис. [20]).

fig:
Ввод программы

Рис. 20 Ввод программы

Создаю исполняемый файл и запускаю его. Мой вариант - 5. (рис. [21]).

fig:
Создание и запуск

Рис. 21Создание и запуск

### 4.2.1 Ответы на вопросы

1. За вывод на экран сообщения ‘Ваш вариант:’ отвечают строки “mov eax,rem” и “call sprint”
2. Инструкция “mov ecx, x” чтобы положить адрес строки, которая вводится, в регистр. “mov edx, 80” в регистр запись длины вводимой строки. “call sread” вывод подпрограммы, которая обеспечивает ввод с клавиатуры.
3. “call atoi” используется для вывода подпрограммы, которая производит преобразование ascii кода символа в целое число и записывает результат в регистр eax.
4. За вычисление варианта отвечают эти строки: “xor edx, edx”, “mov ebx, 20”, “div ebx”, “inc edx”
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 результата вычислений отвечают эти строки: “mov eax, edx”, “call iprintLF”.

## 4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm в каталоге (рис. [22]).

fig:
Создание файла

Рис. 22 Создание файла

Ввожу в него текст программы для вычисления значения выражения моего варианта (9x - 8)/8. (рис. [23]).

fig:
Ввод программы

Рис. 23 Ввод программы

Создаю исполняемый файл и запускаю его. Проверяю его для двух значениев x: 8 и 64. Я также проверил вручную и результаты оказались верны. (рис. [24]).

fig:
Создание и запуск

Рис. 24 Создание и запуск

# 5 Вывод

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

# 6 Список литературы

1. https://esystem.rudn.ru/pluginfile.php/2089086/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