Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Даровских Александра Сергеевна

Содержание

[1 Цель работы 1](#_Toc151165970)

[2 Задание 1](#_Toc151165971)

[3 Теоретическое введение 1](#_Toc151165972)

[4 Выполнение лабораторной работы 2](#_Toc151165973)

[4.1 Символьные и численные данные в NASM 2](#_Toc151165974)

[4.2 Выполнение арифметических операций в NASM 5](#_Toc151165975)

[4.2.1 Ответы на вопросы по программе 8](#_Toc151165976)

[4.3 Выполнение заданий для самостоятельной работы 9](#_Toc151165977)

[5 Выводы 11](#_Toc151165978)

[6 Список литературы 11](#_Toc151165979)

# 1 Цель работы

Цель данной лабораторной работы - 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. 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 команды mkdir создаем директорию, в которой будем создавать файлы с программами для лабораторной работы №6 (рис. 1). Переходим в созданный каталог с помощью команды cd.

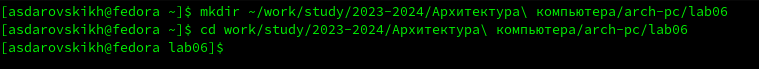


Рис. 1: Создание директории

С помощью команды touch создаем файл lab6-1.asm (рис. 2).

Изображение выглядит как снимок экрана, Шрифт, зеленый

Автоматически созданное описание

Рис. 2: Создание файла

Копируем в текущий каталог файл in\_out.asm с помощью команды cp, т. к. он будет использоваться в других программах (рис. 3).



Рис. 3: Создание копии файла

Открываем созданный файл lab6-1.asm, вставляем в него программу вывода значения регистра eax (рис. 4).

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рис. 4: Редактирование файла

Создаем исполняемый файл программы и запускаем его (рис. 5). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

Изображение выглядит как снимок экрана, Шрифт, зеленый

Автоматически созданное описание

Рис. 5: Запуск исполняемого файла

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

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рис. 6: Редактирование файла

Создаем новый исполняемый файл программы и запускаем его (рис. 7). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

Изображение выглядит как снимок экрана, зеленый, Шрифт

Автоматически созданное описание

Рис. 7: Запуск исполняемого файла

Создаем новый файл lab6-2.asm с помощью команды touch (рис. 8).



Рис. 8: Создание файла

Вводим в файл текст другойпрограммы для вывода значения регистра eax (рис. 9).

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Рис. 9: Редактирование файла

Создаем и запускаем исполняемый файл lab6-2 (рис. 10). Теперь вывод числа 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

Изображение выглядит как снимок экрана, зеленый, Шрифт

Автоматически созданное описание

Рис. 10: Запуск исполняемого файла

Заменяем в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 11).

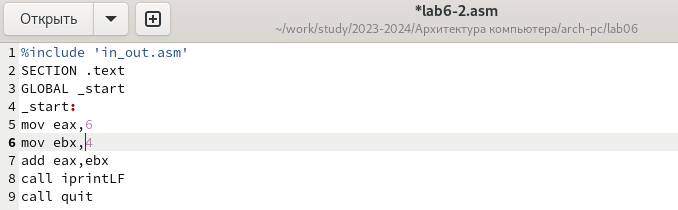


Рис. 11: Редактирование файла

Создаем и запускаем новый исполняемый файл (рис. 12). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

Изображение выглядит как снимок экрана, Шрифт, зеленый

Автоматически созданное описание

Рис. 12: Запуск исполняемого файла

Заменяем в тексте программы функцию iprintLF на iprint (рис. 13).

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Рис. 13: Редактирование файла

Создаем и запускаем новый исполняемый файл (рис. 14). 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.

Изображение выглядит как снимок экрана, зеленый, Шрифт, Графика

Автоматически созданное описание

Рис. 14: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

Создаем файл lab6-3.asm с помощью команды touch (рис. 15).



Рис. 15: Создание файла

Вводим в созданный файл текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3 (рис. 16).

Изображение выглядит как текст, снимок экрана, документ, число

Автоматически созданное описание

Рис. 16: Редактирование файла

Создаем исполняемый файл и запускаем его (рис. 17).

Изображение выглядит как снимок экрана, Шрифт, зеленый, текст

Автоматически созданное описание

Рис. 17: Запуск исполняемого файла

Изменяем программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. 18).

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рис. 18: Изменение программы

Создаем и запускаем новый исполняемый файл (рис. 19). Посчитаем правильность выполнения работы самостоятельно, придем к выводу, что программа верно исполнилась.

Изображение выглядит как снимок экрана, текст, Шрифт, зеленый

Автоматически созданное описание

Рис. 19: Запуск исполняемого файла

Создаем файл variant.asm с помощью команды touch (рис. 20).



Рис. 20: Создание файла

Вводим в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 21).

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

Рис. 21: Редактирование файла

Создаем и запускаем исполняемый файл (рис. 22). Вводим номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 18.

Изображение выглядит как снимок экрана, текст, Шрифт, зеленый

Автоматически созданное описание

Рис. 22: Запуск исполняемого файла

### 4.2.1 Ответы на вопросы по программе

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

mov eax,rem  
call sprint

1. 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
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

## 4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью команды touch (рис. 23).



Рис. 23: Создание файла

Открываем созданный файл для редактирования, вводим в него текст программы для вычисления значения выражения 3(х+10) - 20 (рис. 24). Это выражение было под вариантом 18.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис. 24: Написание программы

Создаем и запускаем исполняемый файл (рис. 25). При вводе значения 1, вывод 13.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 25: Запуск исполняемого файла

Проводим еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 26). Программа отработала верно.

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание

Рис. 26: Запуск исполняемого файла

**Листинг 4.1. Программа для вычисления значения выражения 3(х+10) − 20.**

%include 'in\_out.asm' ; подключение внешнего файла  
SECTION .data ; секция инициированных данных  
msg: DB 'Введите значение переменной х: ',0  
rem: DB 'Результат: ',0  
SECTION .bss ; секция не инициированных данных  
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт  
SECTION .text ; Код программы  
GLOBAL \_start ; Начало программы  
\_start: ; Точка входа в программу  
; ---- Вычисление выражения  
mov eax, msg ; запись адреса выводимиого 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 ; ASCII кода в число, `eax=x`  
add eax,10; eax = eax+10 = x + 10  
mov ebx,3 ; запись значения 3 в регистр ebx  
mul ebx; EAX= EBX\*EAX = 3\*(x+10)  
add eax,-20; eax = eax-20 = 3\*(x+10)-20  
mov edi,eax ; запись результата вычисления в 'edi'  
; ---- Вывод результата на экран  
mov eax,rem ; вызов подпрограммы печати  
call sprint ; сообщения 'Результат: '  
mov eax,edi ; вызов подпрограммы печати значения  
call iprint ; из 'edi' в виде символов  
call quit ; вызов подпрограммы завершения

# 5 Выводы

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

# 6 Список литературы

1. <https://esystem.rudn.ru/pluginfile.php/2089086/mod_resource/content/0/Лабораторная%20работа%20№6.%20Арифметические%20операции%20в%20NASM..pdf>