Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютеров

Даровских Александра Сергеевна

**Содержание**

[1 Цель работы 3](#_Toc151780382)

[2 Задание 4](#_Toc151780383)

[3 Теоретическое введение 5](#_Toc151780384)

[4 Выполнение лабораторной работы 6](#_Toc151780385)

[4.1 Реализация переходов в NASM 6](#_Toc151780386)

[4.2 Изучение структуры файлы листинга 10](#_Toc151780387)

[4.3 Задания для самостоятельной работы 11](#_Toc151780388)

[5 Выводы 16](#_Toc151780389)

[6 Список литературы 17](#_Toc151780390)

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

# 2 Задание

1. Реализация переходов в NASM.
2. Изучение структуры файлы листинга.
3. Задания для самостоятельной работы.

# 3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

• условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.

• безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Безусловный переход выполняется инструкцией jmp. Инструкция cmp является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция cmp является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

# 4 Выполнение лабораторной работы

## 4.1 Реализация переходов в NASM

С помощью команды mkdir создаем директорию, в которой будем создавать файлы с программами для лабораторной работы №7 (рис. 1). Переходим в созданный каталог с помощью команды cd

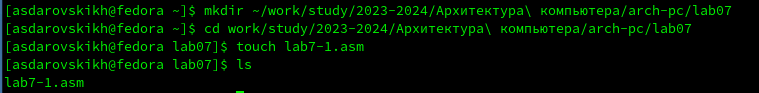


Рис. 1: Создание директории

С помощью команды touch создаем файл lab7-1.asm (рис. 2).

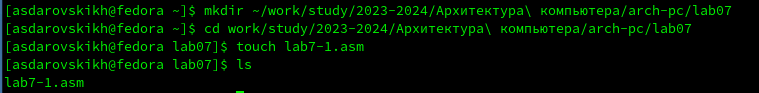


Рис. 2: Создание файла

Открываем созданный файл lab7-1.asm, вставляем программу из листинга 7.1 (рис.3).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис. 3: Ввод текста программы из листинга 7.1

Создаем исполняемый файл и запускаем его. (рис.4).

Изображение выглядит как снимок экрана, текст, Шрифт, зеленый

Автоматически созданное описание

Рис. 4: Запуск программного кода

jmp \_label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки \_label2, пропустив вывод первого сообщения.

Изменяем программу, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу в соответствии с листингом 7.2. (рис.5).

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

Автоматически созданное описание

Рис. 5: Изменение текста программы

Создаем новый исполняемый файл программы и запускаем его. (рис. 6).

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание

Рис. 6: Создание исполняемого файла

Затем изменяем текст программы, добавив в начале программы jmp \_label3, jmp \_label2 в конце метки jmp \_label3, jmp \_label1 добавляем в конце метки jmp \_label2, и добавляем jmp \_end в конце метки jmp \_label1, (рис. 7).

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рис. 7: Изменение текста программы

Создаем новый исполняемый файл программы и запускаем его: (рис. 8).

Изображение выглядит как снимок экрана, текст, Шрифт, зеленый

Автоматически созданное описание

Рис. 8: Вывод программы

Рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. Значения для A и C задаются в программе, значение B вводиться с клавиатуры.

Создаю файл lab7-2.asm в каталоге ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab07. (рис. 9).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 9: Создание файла

Текст программы из листинга 7.3 вводим в lab7-2.asm. (рис. 10).

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

Автоматически созданное описание

Рис. 10: Ввод текста программы из листинга 7.3

Создаем новый исполняемый файл программы и запускаем его. (рис. 11).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 11: Проверка работы файла

Файл работает корректно.

## 4.2 Изучение структуры файлы листинга

Создаем файл листинга для программы из файла lab7-2.asm. (рис. 12).



Рис. 12: Создание файла листинга

Открываем файл листинга lab7-2.lst с помощью текстового редактора и изучаем содержимое. (рис. 13).

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рис. 13: Изучение файла листинга

В представленных трех строчках содержаться следующие данные: (рис. 14).



Рис. 14: Выбранные строки файла

“2” - номер строки кода, “; Функция вычисления длинны сообщения” - комментарий к коду, не имеет адреса и машинного кода.

“3” - номер строки кода, “slen” - название функции, не имеет адреса и машинного кода.

“4” - номер строки кода, “00000000” - адрес строки, “53” - машинный код, “push ebx” - исходный текст программы, инструкция “push” помещает операнд “ebx” в стек.

Открываем файл с программой lab7-2.asm и удаляем выделенный операнд. (рис. 15).



Рис. 15: Удаление выделенного операнда из кода

Выполняем трансляцию с получением файла листинга. (рис. 16).



Рис. 16: Получение файла листинга

Инструкция mov (единственная в коде, содержащая два операнда) не может работать только с одним операндом, что нарушает код.

## 4.3 Задания для самостоятельной работы

1. Мой вариант под номером 18, a,b,c - 83, 73 и 30. (рис. 17).

Изображение выглядит как текст, программное обеспечение, веб-страница, Значок на компьютере

Автоматически созданное описание

Рис. 17: Написание программы

Создаю исполняемый файл и проверяю его работу, подставляя необходимые значение. (рис. 18).

Изображение выглядит как снимок экрана, Шрифт, текст, зеленый

Автоматически созданное описание

Рис. 18: Запуск файла и проверка его работы

Программа работает корректно.

Код программы:

include 'in\_out.asm'

section .data

msg db "Наименьшее число: ",0h

A dd '83'

B dd '73'

C dd '30'

section .bss

min resb 10

section .text

global \_start

\_start:

; ---------- Записываем 'B' в переменную 'min'

mov ecx,[B] ; 'ecx = B'

mov [min],ecx ; 'min = B'

; ---------- Сравниваем 'B' и 'С' (как символы)

cmp ecx,[B] ; Сравниваем 'B' и 'С'

jg check\_B ; если 'B<C', то переход на метку 'check\_A',

mov ecx,[C] ; иначе 'ecx = C'

mov [min],ecx ; 'min = C'

; ---------- Преобразование 'min(B,C)' из символа в число

check\_B:

mov eax,min

call atoi ; Вызов подпрограммы перевода символа в число

mov [min],eax ; запись преобразованного числа в min

; ---------- Сравниваем 'min(B,C)' и 'A' (как числа)

mov ecx,[min]

cmp ecx,[A] ; Сравниваем 'min(B,C)' и 'A'

jl fin ; если 'min(B,C)<A', то переход на 'fin',

mov ecx,[A] ; иначе 'ecx = A'

mov [min],ecx

; ---------- Вывод результата

fin:

mov eax, msg

call sprint ; Вывод сообщения 'Наименьшее число: '

mov eax,[min]

call iprintLF ; Вывод 'min(A,B,C)'

call quit ; Выход

Код программы

1. a^2, a=1; 10+x, a=1 (рис. 19).

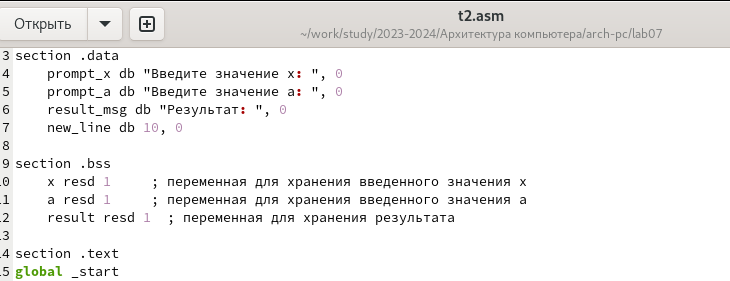


Рис. 19: Написание программы

Создаю исполняемый файл и проверяю его работу для значений х и а соответственно: (1;2), (2;1). (рис. 20).

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Автоматически созданное описание

Рис. 20: Запуск файла и проверка его работы

Программа работает корректно.

Код программы:

%include 'in\_out.asm'

section .data

prompt\_x db "Введите значение x: ", 0

prompt\_a db "Введите значение a: ", 0

result\_msg db "Результат: ", 0

new\_line db 10, 0

section .bss

x resd 1 ; переменная для хранения введенного значения x

a resd 1 ; переменная для хранения введенного значения a

result resd 1 ; переменная для хранения результата

section .text

global \_start

\_start:

; запрос значения x

mov edx, prompt\_x

call sprint

call sread

mov [x], eax ; сохраняем x

; запрос значения a

mov edx, prompt\_a

call sprint

call sread

mov [a], eax ; сохраняем a

; вычисление и вывод результата

mov eax, [a]

cmp eax, 1

je a\_is\_1 ; если a=1, переходим к вычислению 10 + x

imul eax, eax ; умножаем a на само себя

jmp print\_result

a\_is\_1:

add eax, [x] ; вычисляем 10 + x

print\_result:

mov [result], eax ; сохраняем результат

mov edx, result\_msg

call sprint

mov eax, [result]

call iprintLF

; выход

call quit

# 5 Выводы

При выполнении данной лабораторной работы я команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов и ознакомилась с назначением и структурой файла листинга, что поможет мне при выполнении последующих лабораторных работ.

# 6 Список литературы

1. [Лабораторная работа №7. Команды безусловного и условного переходов в Nasm. Программирование ветвлений.](https://esystem.rudn.ru/pluginfile.php/2089087/mod_resource/content/0/Лабораторная%20работа%20№7.%20Команды%20безусловного%20и%20условного%20переходов%20в%20Nasm.%20Программирование%20ветвлений..pdf)