**Отчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютеров**

Чувакина Мария Владимировна

**Содержание**

**1  Цель работы 4**

**2  Задание 5**

**3  Теоретическое введение 6**

**4  Выполнение лабораторной работы 7**

4.1 **Реализация переходов в NASM** ..........................................................7

* 1. **Изучение структуры файлы листинга** . . . . . . . . . . . . . . …………12

4.3 **Задания для самостоятельной работы** . . . . . . . . . . . . . . ….……...13

**5  Выводы 19**

**6  Список литературы 20**

**Список иллюстраций**

4.1 Создание директории…………………………………………………………………7

4.2 Создание файла . . . . . . . . . . . . ……………………………………………………….7

4.2 Ввод текста программы из листинга 7.1 . . . . . . . . . . . . . . . ………….……………..8

4.4 Запуск программного кода ..................... …………………………….………………8

4.5 Изменение текста программы...................................................................................... 9

4.6 Создание исполняемого файла ................... ……………………………………..…..9

4.7 Изменение текста программы.................................................................................... 10

4.8 Выводпрограммы........................................................................................................ 10

4.9 Создание файла ........................... …………………………………………………...11

4.10 Ввод текста программы из листинга 7.3 . . . . . . ……………………... . . . . . . . . 11

4.11 Проверка работы файла ....................... ………………...…………………………12

4.12 Создание файла листинга ....................................................................................... 12

4.13 Изучение файла листинга........................................................................................ 12

4.14 Выбранные строки файла ....................................................................................... 13

4.15 Удалениев ыделенного операнда из кода . . . . ……………………... . . . . . . . . . 13

4.16 Получение файла листинга .................................................................................... 13

4.17 Написание программы ............................................................................................ 14

4.18 Запуск файла и проверка его работы .................................................................... 14

4.19 Написание программы ........................................................................................... 16

4.20 Запуск файла и проверка его работы .................................................................... 17

**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 ко- манды 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 утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №7 (рис. 4.1). Перехожу в созданный каталог с помощью утилиты cd.

Изображение выглядит как текст, программное обеспечение, веб-страница, Мультимедийное программное обеспечение

Автоматически созданное описание Рис. 4.1: Создание директории

С помощью утилиты touch создаю файл lab7-1.asm (рис. 4.2).

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

Автоматически созданное описание

Рис. 4.2: Создание файла

Ввожу в файл lab7-1.asm текст программы из листинга 7.1. (рис. 4.3).

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

Автоматически созданное описание

Рис. 4.3: Ввод текста программы из листинга 7.1

Создаю исполняемый файл и запускаю его. (рис. 4.4).

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание Рис. 4.4: Запуск программного кода

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

Изменю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого изменяю текст программы в соответствии с листингом 7.2. (рис. 4.5).

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

Автоматически созданное описание ю Рис. 4.5: Изменение текста программы

Создаю исполняемый файл и проверяю его работу. (рис. 4.6).

Изображение выглядит как текст, программное обеспечение, веб-страница, Веб-сайт

Автоматически созданное описание Рис. 4.6: Создание исполняемого файла

Затем изменяю текст программы, добавив в начале программы jmp \_label3, jmp \_label2 в конце метки jmp \_label3, jmp \_label1 добавляю в конце метки jmp \_label2, и добавляю jmp \_end в конце метки jmp \_label1, (рис. 4.7).

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

Автоматически созданное описание

Рис. 4.7: Изменение текста программы

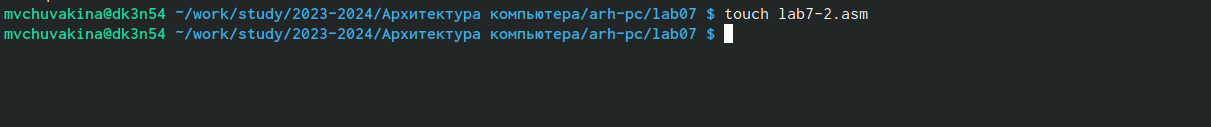
чтобы вывод программы был следующим: (рис. 4.8).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание Рис. 4.8: 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 про- грамме, значение B вводиться с клавиатуры.

Создаю файл lab7-2.asm в каталоге ~/work/arh-pc/lab07. (рис. 4.9).

 Рис. 4.9: Создание файла

Текст программы из листинга 7.3 ввожу в lab7-2.asm. (рис. 4.10).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание Рис. 4.10: Ввод текста программы из листинга 7.3

Создаю исполняемый файл и проверьте его работу. (рис. 4.11).

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание я Рис. 4.11: Проверка работы файла

Файл работает корректно.

**4.2 Изучение структуры файлы листинга**

Создаю файл листинга для программы из файла lab7-2.asm. (рис. 4.12).

 Рис. 4.12: Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора и внимательно изучаю его формат и содержимое. (рис. 4.13).

Изображение выглядит как текст, программное обеспечение, Операционная система, снимок экрана

Автоматически созданное описание Рис. 4.13: Изучение файла листинга

В представленных трех строчках содержаться следующие данные: (рис. 4.14).

 Рис. 4.14: Выбранные строки файла

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

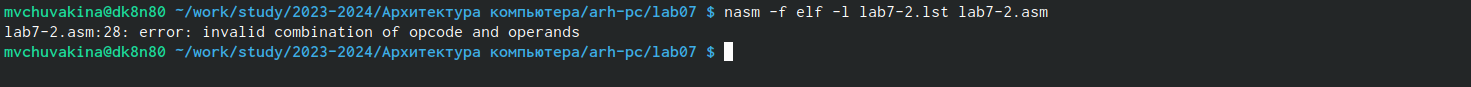
“3” - номер строки кода, “slen” - название функции, не имеет адреса и машин- ного кода.

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

Открываю файл с программой lab7-2.asm и в выбранной мной инструкции с двумя операндами удаляю выделенный операнд. (рис. 4.15).

 Рис. 4.15: Удаление выделенного операнда из кода

Выполняю трансляцию с получением файла листинга. (рис. 4.16).

 Рис. 4.16: Получение файла листинга

На выходе я не получаю ни одного файла из-за ошибки:инструкция mov (единственная в коде содержит два операнда) не может работать, имея только один операнд, из-за чего нарушается работа кода.

**4.3 Задания для самостоятельной работы**

1. Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбираю из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы №6. Мой вариант под номером 16, поэтому мои значения - 44, 74 и 17. (рис. 4.17).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание и и Рис. 4.17: Написание программы

Создаю исполняемый файл и проверяю его работу, подставляя необходимые значение. (рис. 4.18).

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание Рис. 4.18: Запуск файла и проверка его работы

Программа работает корректно. Код программы:

%include 'in\_out.asm'

section .data

msg db "Наименьшее число: ",0h

A dd '44'

B dd '74'

C dd '17'

section .bss

min resb 10

section .text

global \_start

\_start:

; ---------- Записываем 'A' в переменную 'min'

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

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

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

cmp ecx,[C] ; Сравниваем 'A' и 'С'

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

mov ecx,[C] ; иначе 'ecx = C'

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

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

mov ecx,[min]

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

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

mov ecx,[B] ; иначе 'ecx = B'

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

2. Пишу программу, которая для введенных с клавиатуры значений х и а вычисляет значение и выводит результат вычислений заданной для моего варианта функции f(x):

х + 4, если х < 4

a\*x, если х >=4 (рис. 4.1)Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Значок на компьютере

Автоматически созданное описание Рис. 4.19: Написание программы

Создаю исполняемый файл и проверяю его работу для значений х и а соответственно. (рис. 4.20).

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Шрифт

Автоматически созданное описание Рис. 4.20: Запуск файла и проверка его работы

Код программы:

%include 'in\_out.asm'

section .data

vvodx: db "Введите x: ",0

vvoda: db "Введите a: ",0

vivod: db "Результат: ",0

section .bss

x: resb 80

a: resb 80

section .text

global \_start

\_start:

mov eax,vvodx

call sprint

mov ecx,x

mov edx,80

call sread

mov eax,x

call atoi

cmp eax,4

jg \_functionx

mov eax,vvoda

call sprint

mov ecx,a

mov edx,80

call sread

mov eax,a

call atoi

jmp \_functiona

\_functiona:

mov edx,1

mul edx

jmp \_end

\_functionx:

add eax,4

jmp \_end

\_end:

mov ecx,eax

mov eax,vivod

call sprint

mov eax,ecx

call iprintLF

call quit

**5 Выводы**

По итогам данной лабораторной работы я изучила 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. GDB:TheGNUProjectDebugger.—URL:https://www.gnu.org/software/gdb/.
2. GNUBashManual.—2016.—URL:https://www.gnu.org/software/bash/manual/.
3. MidnightCommanderDevelopmentCenter.—2021.—URL:https://midnight-

commander. org/.

1. NASMAssemblyLanguageTutorials.—2021.—URL:https://asmtutor.com/.
2. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly

Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL:

http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.

1. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN

978-1491941591.

1. TheNASMdocumentation.—2021.—URL:https://www.nasm.us/docs.php.
2. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
3. КолдаевВ.Д.,ЛупинС.А.АрхитектураЭВМ.—М.:Форум,2018.
4. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. :

Солон-Пресс, 2017.

1. НовожиловО.П.АрхитектураЭВМисистем.—М.:Юрайт,2016.
2. Расширенныйассемблер:NASM.—2021.—URL:https://www.opennet.ru/docs/RUS/nasm/.
3. РобачевскийА.,НемнюгинС.,СтесикО.ОперационнаясистемаUNIX.—2-е

изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.

1. СтоляровА.ПрограммированиенаязыкеассемблераNASMдляОСUnix.—2-

е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm\_unix.

21

15. ТаненбаумЭ.Архитектуракомпьютера.—6-еизд.—СПб.:Питер,2013.— 874 с. — (Классика Computer Science).

16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,2015. — 1120 с. — (Классика Computer Science).