

Отчет по лабораторной работе №7

дисциплина: Архитектура компьютера

Бондарь Татьяна Владимировна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файлы листинга	12
5	Задания для самостоятельной работы	14
6	Выводы	17

Список иллюстраций

4.1	Переход в каталог и создание файла	8
4.2	Программа с использованием инструкции jmp	9
4.3	Исполнение программы из листинга 7.1	9
4.4	Исправленный текст программы lab7-1.asm	10
4.5	Исполнение программы lab7-1	10
4.6	Программа из листинга 7.3 часть 1	11
4.7	Программа из листинга 7.3 часть 2	11
4.8	Исполнение программы из листинга 7.3	11
4.9	Создания файла листинга программы	12
4.10	Содержимое файла листинга	12
4.11	Удаление операнда	13
4.12	Трансляция	13
5.1	Текст программы lab7-3.asm	14
5.2	Запуск программы	15
5.3	Текст программы lab7-4.asm часть 1	15
5.4	Текст программы lab7-4.asm часть 2	16
5.5	Запуск программы	16

Список таблиц

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

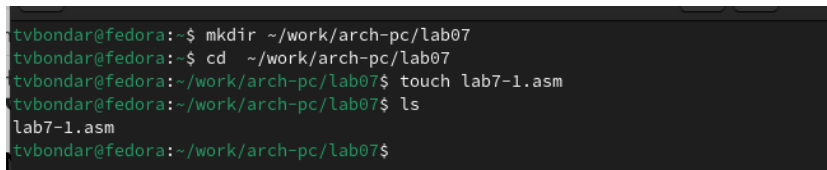
1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.
2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

3 Теоретическое введение

4 *Выполнение лабораторной работы*

4.1 *Реализация переходов в NASM*

1. Создаю каталог для программ лабораторной работы №7, перехожу в него и создаю файл lab7-1.asm.



```
tvbondar@fedora:~$ mkdir ~/work/arch-pc/lab07
tvbondar@fedora:~$ cd ~/work/arch-pc/lab07
tvbondar@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
tvbondar@fedora:~/work/arch-pc/lab07$ ls
lab7-1.asm
tvbondar@fedora:~/work/arch-pc/lab07$
```

Рис. 4.1: Переход в каталог и создание файла

2. Ввожу в файл lab7-1.asm текст программы из листинга 7.1. Запускаю исполняемый файл.


```

; Листинг 7.1. Программа с использованием инструкции jmp
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 4.2: Программа с использованием инструкции jmp

```

tvbondar@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
tvbondar@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
tvbondar@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o
tvbondar@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
tvbondar@fedora:~/work/arch-pc/lab07$

```

Рис. 4.3: Исполнение программы из листинга 7.1

3. Изменим текст программы так, чтобы она сначала выводила “Сообщение №2”, потом “Сообщение №1” и завершала работу. Запустим исправленную программу.

```

; Листинг 7.2. Программа с использованием инструкции jmp
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 4.4: Исправленный текст программы lab7-1.asm

```

tvbondar@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
tvbondar@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
tvbondar@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o
tvbondar@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
tvbondar@fedora:~/work/arch-pc/lab07$

```

Рис. 4.5: Исполнение программы lab7-1

4. Создадим файл lab7-2.asm. Введем в файл текст программы из листинга 7.3. Программа определяет и выводит на экран наибольшую из целочисленных переменных A, B, C. Значения для A, C задаются в программе, значение B вводится с клавиатуры. Запускаю исполняемый файл.

```

; Листинг 7.3. Программа, которая запрашивает и выводит на экран наибольшее из 3 введенных переменных: A, B и C.
%include "io_out.asm"
section .data
max db "Введите B: ",0h
max2 db "Наибольшее число: ",0h
A dd 20
C dd 50
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения "Введите B: "
mov eax,max
call sprint
; ----- Ввод "B".
mov eax,B
mov eax,10
call read
; ----- Преобразование "B" из символа в число
mov eax,B
call atoi ; Вывод подготовленный переводов символа в число
mov [B],eax ; запись преобразованного числа в "B".
; ----- Сравнение "A" в переменную "max".
mov ecx,[A] ; ecx = A.
mov [max],ecx ; max = A.
; ----- Сравнение "A" и "C" (как число)

```

Рис. 4.6: Программа из листинга 7.3 часть 1

```

mov ecx,[C] ; ecx = C.
mov [max],ecx ; max = C.
; ----- Преобразование "max(A,C)" из символа в число
check_B:
mov eax,max
call atoi ; Вывод подготовленный переводов символа в число
mov [max],eax ; запись преобразованного числа в "max"
; ----- Сравнение "max(A,C)" и "B" (как число)
mov ecx,[max]
cmp ecx,[B] ; Сравнение "max(A,C)" и "B".
jg fin ; Если "max(A,C)" > "B", то переход на "fin".
mov ecx,[B] ; иначе "ecx" = B.
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,max2
call sprint ; Вывод сообщения "Наибольшее число: "
mov eax,[max]
call printf ; Вывод "max(A,B,C)"
call quit ; Выход

```

Рис. 4.7: Программа из листинга 7.3 часть 2

```

tvbondar@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
tvbondar@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
tvbondar@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
tvbondar@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
tvbondar@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.o
tvbondar@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
tvbondar@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 0
Наибольшее число: 50
tvbondar@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 50
Наибольшее число: 50
tvbondar@fedora:~/work/arch-pc/lab07$

```

Рис. 4.8: Исполнение программы из листинга 7.3

4.2 Изучение структуры файлы листинга

5. Создаю файл листинга для программы из файла lab7-2.asm. Открываю файл листинга в любом текстовом редакторе.

```
tvbondar@fedora: ~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
tvbondar@fedora: ~/work/arch-pc/lab07$ ls
in_out.asm lab7-1.asm lab7-2 lab7-2.lst
lab7-1 lab7-1.o lab7-2.asm lab7-2.o
tvbondar@fedora: ~/work/arch-pc/lab07$
```

Рис. 4.9: Создания файла листинга программы

```
1 ; Листинг 7.3. Программа, которая определяет и выводит на экран наибольшее из 3 чисел: A, B и C.
2 %include "in_out.asm"
3
4 <> ; ----- also -----
5 <> ; Вычисляем максимум двух чисел
6
7 <> ;
8 00000000 53 <> push eax
9 00000001 85C3 <> pop eax, eax
10
11 <> ;
12 <> ;
13 00000003 803800 <> mov byte [eax], 0
14 00000006 7403 <> jz finished
15 0000000A 40 <> inc eax
16 0000000D E8E8 <> jmp max1char
17
18 <> ;
19 <> ;
20 <> finished:
21 <> sub eax, eax
22 <> pop eax
<> ;
<> ; ----- print -----
<> ; Выводим результат сообщения
<> ; Выводим данные: max, <message>
<> print:
```

Рис. 4.10: Содержимое файла листинга

6. Объясняю три строчки из файла листинга: 23 00000106 E891FFFFFF call atoi - Вызов подпрограммы перевода символа в число; 23 - номер строки, 00000106 - адрес, E891FFFFFF - машинный код; 41 0000014B 7F0C jg fin - переход на label 'fin', если 'max(A,C)>B'; 41 - номер строки, 0000014B - адрес, 7F0C - машинный код; 50 0000016D E869FFFFFF call quit - Выход из программы; 50 - номер строки; 0000016D - адрес; E869FFFFFF - машинный код.
7. Открываю файл с программой lab7-2.asm и в одной из инструкций с двумя операндами удаляю один операнд. Транслирую файл с текстом программы с получением файла листинга. Я не получаю выходных файлов, программа выдает ошибку, так как в данной операции должны присутствовать два операнда, а не один.

```

; ----- Сравниваем 'A' и 'C' (как символы)
cmp ех|          ; Сравниваем 'A' и 'C'
jg check_B       ; если 'A>C', то переход на метку 'check_B',

```

Рис. 4.11: Удаление операнда

```

tvbondar@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:29: error: invalid combination of opcode and operands
tvbondar@fedora:~/work/arch-pc/lab07$

```

Рис. 4.12: Трансляция

5 Задания для самостоятельной работы

1. Создаю файл lab7-3.asm и ввожу в него текст программы для нахождения наименьшей из трех целочисленных переменных a, b, c. Мой вариант 12. Программа работает корректно.

```
; Программа, которая определяет и выводит на экран наименьшее из 3 целочисленных переменных: A, B и C.
#include 'in_out.asm'
section .data
    msg1 db "Наименьшее число: ",0h
    A dd '99'
    B dd '29'
    C dd '26'
section .bss
    min resb 10
section .text
    global _start
_start:
    ; ----- Преобразование 'B' из символа в число
    mov eax, B
    call atoi           ; Вывод преобразованной переменной символа в число
    mov [B],eax         ; Записываем преобразованное число в 'B'
    ; ----- Записываем 'A' в переменную 'min'
    mov eax, [A]
    mov [min],eax       ; 'min' = A
    ; ----- Сравниваем 'A' и 'C' (как символы)
    cmp eax, [C]        ; Сравниваем 'A' и 'C'
    jl check_B          ; Если 'A' < 'C', то переходим на метку 'check_B'
    mov eax, [C]         ; Если 'A' > 'C', то устанавливаем 'min' = C
    mov [min],eax       ; 'min' = C
    ; ----- Преобразование 'min(A,C)' из символа в число
    check_B:
    mov eax, min
    call atoi           ; Вывод преобразованной переменной символа в число
    mov [min],eax       ; Записываем преобразованное число в 'min'
    ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
    mov eax, [min]
    cmp eax, [B]        ; Сравниваем 'min(A,C)' и 'B'
    jl fin             ; Если 'min(A,C)' < 'B', то переходим на 'fin'
    mov eax, [B]        ; Иначе устанавливаем 'min' = B
    mov [min],eax
    ; ----- Вывод результата
    fin:
    mov eax, msg1
    call sprint         ; Вывод преобразованного наименьшего числа: '
    mov eax, [min]
    call printf         ; Вывод 'min(A,B,C)'
    call quit           ; Выход
```

Рис. 5.1: Текст программы lab7-3.asm

```

tvbondar@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1.asm lab7-2 lab7-2.lst lab7-3.lst
lab7-1 lab7-1.o lab7-2.asm lab7-3.asm lab7-3.o
tvbondar@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
tvbondar@fedora:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 26
tvbondar@fedora:~/work/arch-pc/lab07$

```

Рис. 5.2: Запуск программы

2. Создаю файл lab7-4.asm и ввожу в него текст программы, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции и выводит результат вычислений. Мой вариант - 12.

```

#include 'in_out.asm'
section .data
msg1 db 'Введите x: ', 0h
msg2 db 'Введите a: ', 0h
msg3 db 'Результат: ', 0h
section .bss
x resb 11
a resb 11
res resb 12
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите x: '
mov eax, msg1
call sprint

; ----- Ввод 'x'
mov ecx, x
mov edx, 10
call sread

; ----- Преобразование 'x' из символа в число
mov eax, x
call atoi
mov [x], eax

; ----- Вывод сообщения 'Введите a: '
mov eax, msg2
call sprint

; ----- Ввод 'a'
mov ecx, a
mov edx, 10
call sread

; ----- Преобразование 'a' из символа в число
mov eax, a
call atoi
mov [a], eax

```

Рис. 5.3: Текст программы lab7-4.asm часть 1

```

; ----- Сравниваем 'x' и '5'
mov eax, [x]      ; Загружаем значение 'x' в EAX
cmp eax, 5        ; Сравниваем EAX с 5
jle do_first      ; если 'x <= 5', то переход на метку 'do_first'
jge do_second     ; иначе переход на метку 'do_second'

do_first:
mov eax, [x]      ; Загружаем значение 'x' в EAX
mov ebx, [a]      ; Загружаем значение 'a' в EBX
mul ebx          ; Умножаем EAX на EBX (x * a)
mov ecx, eax      ; Переносим результат в ECX
jmp fin

do_second:
mov eax, [x]      ; Загружаем значение 'x' в EAX
sub eax, 5        ; Вычитаем 5 из EAX
mov ecx, eax      ; Переносим результат в ECX
jmp fin

; ----- Вывод результата
fin:
mov eax, msg3
call sprint      ; Вывод сообщения 'Result: '
mov eax, ecx     ; Переносим результат из ECX в EAX
call iprintfLF  ; Вывод 'x!'
call quit       ; Выход

```

Рис. 5.4: Текст программы lab7-4.asm часть 2

```

tvbondar@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
tvbondar@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
tvbondar@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 7
Результат: 9
tvbondar@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
tvbondar@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
tvbondar@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 7
Результат: 21
tvbondar@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 6
Введите a: 4
Результат: 1
tvbondar@fedora:~/work/arch-pc/lab07$

```

Рис. 5.5: Запуск программы

6 Выводы

В результате выполнения лабораторной работы я изучила команды условного и безусловного переходов, а так же приобрела навыки написания программ с использованием переходов. Познакомилась с назначением и структурой файла листинга.