

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Кузьмина Мария Константиновна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация переходов в NASM	6
3.2	Изучение структуры файла листинга	11
3.3	Выполнение заданий для самостоятельной работы.	14
4	Выводы	21

Список иллюстраций

3.1	снимок экрана	6
3.2	снимок экрана	7
3.3	снимок экрана	7
3.4	снимок экрана	8
3.5	снимок экрана	10
3.6	снимок экрана	10
3.7	снимок экрана	10
3.8	снимок экрана	11
3.9	снимок экрана	11
3.10	снимок экрана	12
3.11	снимок экрана	14
3.12	снимок экрана	15
3.13	снимок экрана	17
3.14	снимок экрана	18
3.15	снимок экрана	20

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 файла листинга
3. Выполнение заданий для самостоятельной работы.

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

Создаем директорию с помощью `mkdir`, переходим в нее и создаем файл `lab7-1.asm` (рис. 3.1):



```
mkkuzmina@VirtualBox:~$ mkdir ~/work/arch-pc/lab07
mkkuzmina@VirtualBox:~$ cd ~/work/arch-pc/lab07
mkkuzmina@VirtualBox:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 3.1: снимок экрана

Вводим в файл `lab7-1.asm` текст программы из листинга 7.1 (рис. 3.2):

```

%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. 3.2: снимок экрана

Создаем исполняемый файл и запускаем его. Результат работы данной программы будет следующим (рис. 3.3):

```

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

```

Рис. 3.3: снимок экрана

Изменяем программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Для этого в текст программы

после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). (рис. 3.4):

```
SECTION .text
GLOBAL _start

_start:
jmp _label3

_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'
jmp _label2

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.4: снимок экрана

```
%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 _label3

_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'
jmp _label2

_end:
call quit ; вызов подпрограммы завершения

```

(рис. 3.5):

```

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

```

Рис. 3.5: снимок экрана

Создаем файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и вводим в lab7-2.asm текст из листинга 7.3. (рис. 3.6):

```

mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ touch lab7-2.asm

```

Рис. 3.6: снимок экрана

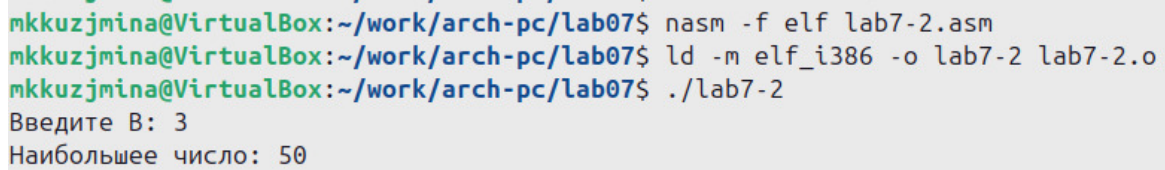
```

%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 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,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'

```

Рис. 3.7: снимок экрана

Вводим значение В и проверяем правильность выполнения (рис. 3.8):

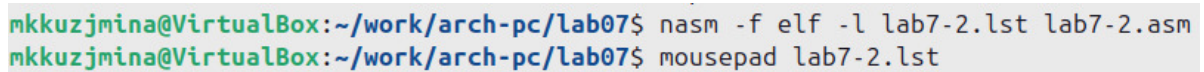


```
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 3
Наибольшее число: 50
```

Рис. 3.8: снимок экрана

3.2 Изучение структуры файла листинга

Создаем файл листинга для программы из файла lab7-2.asm, открываем файл листинга lab7-2.lst с помощью текстового редактора mousepad и в инструкции с двумя операндами удаляем один операнд. (рис. 3.9):



```
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ mousepad lab7-2.lst
```

Рис. 3.9: снимок экрана

```

mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',

```

Рис. 3.10: снимок экрана

```

#include 'in_out.asm'

section .data
msg1 db 'Введите B: ',0h
msg2 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,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',

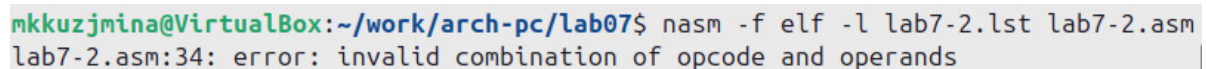
```

```

mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Выполняем трансляцию с получением файла листинга, получаем ошибку. (Подробно объяснить содержимое трёх строк файла листинга по выбору:) `mov ecx, [A]` - загружает значение переменной A в `ecx`. `mov [max], ecx` - копирует значение из регистра `ecx` в переменную `max`. `cmp ecx, [C]` - сравнивает значение в регистре `ecx` с числом, хранящимся в переменной C. Если мы удалим операнд, это вызовет ошибку и выходной файл не создастся, а в листинге появится текст ошибки (рис. 3.11):



```

mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands

```

Рис. 3.11: снимок экрана

3.3 Выполнение заданий для самостоятельной работы.

Создаем исполняемый файл `lab7-3.asm`, в котором пишем программу нахождения наименьшей из 3 целочисленных переменных `a`, `b` и `c` (значения из 15 варианта) (рис. 3.12), и проверяем его работу, введя значение B (рис. 3.13):

```

%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db 'Наименьшее число: ',0h
A dd '32'
C dd '54'
section .bss
min resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'

```

Рис. 3.12: снимок экрана

```

%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db 'Наименьшее число: ',0h
A dd '32'
C dd '54'
section .bss
min resb 10
B resb 10
section .text
global _start

```

```

_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
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'
jb fin ; если 'min(A,C)<B', то переход на 'fin',

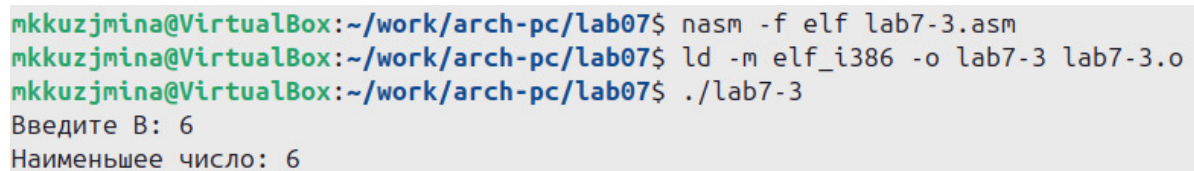
```



```

mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```



```

mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 6
Наименьшее число: 6

```

Рис. 3.13: снимок экрана

Создаем файл lab7-4.asm, пишем программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выберем из таблицы вариантов заданий (15ый). (рис. 3.14):

```

#include 'in_out.asm'
section .data
msg_x db 'Введите значение переменной x: ',0h
msg_a db 'Введите значение переменной a: ',0h
res db 'Результат: ',0h

section .bss
x resb 80
a resb 80

section .text
global _start
_start:
; ----- Ввод значения x
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax ; edi = x
; ----- Ввод значения a

```

Рис. 3.14: снимок экрана

```

#include 'in_out.asm'
section .data
msg_x db 'Введите значение переменной x: ',0h
msg_a db 'Введите значение переменной a: ',0h
res db 'Результат: ',0h

section .bss
x resb 80
a resb 80

section .text

```

```

global _start
_start:
; ----- Ввод значения x
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax ; edi = x
; ----- Ввод значения a
mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax ; esi = a
; ----- Сравнение x и a
cmp edi, esi
jl less_than_a ; если x < a, переход на less_than_a
; ----- x >= a, вычисляем x + 10
mov eax, edi
add eax, 10
jmp print_result
less_than_a:
; ----- x < a, вычисляем a + 10

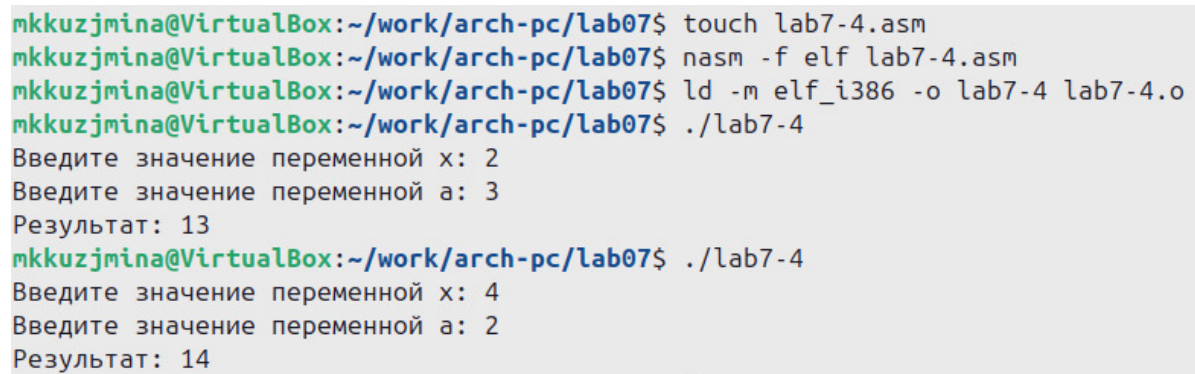
```

```

mov eax, esi
add eax, 10
; ----- Вывод результата
print_result:
mov edi, eax
mov eax, res
call sprint ; Вывод сообщения 'Результат: '
mov eax, edi
call iprintLF ; Вывод результата
call quit ; Выход

```

Создаем исполняемый файл и проверяем его работу для значений x и a (рис. 3.15):



```

mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ touch lab7-4.asm
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 2
Введите значение переменной a: 3
Результат: 13
mkkuzjmina@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 4
Введите значение переменной a: 2
Результат: 14

```

Рис. 3.15: снимок экрана

4 Выводы

Были изучены команды условного и безусловного переходов. Приобретены навыки написания программ с использованием переходов, ознакомились с значением и структурой файла листинга.