

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Абакумов Тимофей Александрович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация переходов в NASM	7
3.2	Изучение структуры файлы листинга	15
3.3	Выполнение заданий для самостоятельной работы	16
4	Выводы	24

Список иллюстраций

3.1	Создание каталога и файла	7
3.2	Код программы	8
3.3	Работа программы	9
3.4	Код программы	10
3.5	Работа программы	11
3.6	Работа программы	11
3.7	Код программы	12
3.8	Создание файла	13
3.9	Работа программы	15
3.10	Создание файла	15
3.11	Ошибка в листинге	16
3.12	Код программы	17
3.13	Работа программы	19
3.14	Создание файла	20
3.15	Код программы	21
3.16	Создание исполняемого файла	23
3.17	Работа программы	23

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

Порядок выполнения лабораторной работы

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

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

1. Для начала создадим каталог для программ лабораторной работы № 7, перейдём в него и создадим файл lab7-1.asm (рис. 3.1).

```
taabakumov@dk3n55 ~ $ mkdir ~/work/arch-pc/lab07
taabakumov@dk3n55 ~ $ cd ~/work/arch-pc/lab07
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-1.asm
taabakumov@dk3n55 ~/work/arch-pc/lab07 $
```

Рис. 3.1: Создание каталога и файла

2. Рассмотрим пример программы с использованием инструкции jmp. Введём в файл lab7-1.asm текст программы из листинга 7.1 (рис. 3.2).

```

lab7-1.asm      [-M--] 41 L:[ 1+21 22/ 22] *(707 / 707b) <EOF>
#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 sprintLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
78 Демидова А. В.
Архитектура ЭВМ
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 3.2: Код программы

Код программы из пункта 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

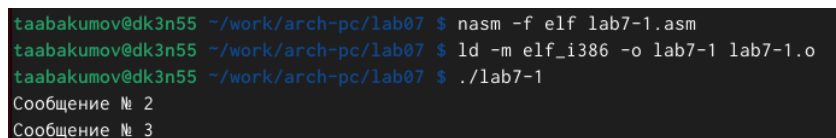
call sprintLF ; 'Сообщение № 1'

_label2:

mov eax, msg2 ; Вывод на экран строки


```
call sprintLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

3. Создадим исполняемый файл и запустим его. Результат работы данной программы будет следующим (рис. 3.3).



```
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 3.3: Работа программы

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

4. Изменим программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Изменим текст программы в соответствии с листингом 7.2 (рис. 3.4).

```

lab7-1.asm      [-M--]  8 L:[ 3+10 13/ 22] *(367 / 670b)
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. 3.4: Код программы

Код программы из пункта 4:

%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

```

5. Создадим исполняемый файл и проверим его работу (рис. 3.5).

```

taabakumov@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1

```

Рис. 3.5: Работа программы

6. Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим: `user@dk4n31:~$./lab7-1` Сообщение № 3 Сообщение № 2 Сообщение № 1 `user@dk4n31:~$` (рис. 3.6).

```

taabakumov@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 3.6: Работа программы

(рис. 3.7).

```

lab7-1.asm      [----] 11 L: [ 2+13 15/ 23] *(388 / 682b) 0103 0x067 [*][X]
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. 3.7: Код программы

Код программы из пункта 6:

%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

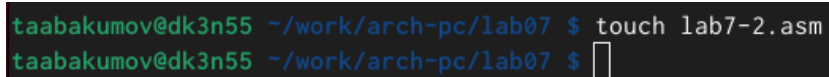
call sprintf ; 'Сообщение № 2'

```

jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

7. Далее создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучим текст программы из листинга 7.3 и введём в lab7-2.asm (рис. 3.8).



```

taabakumov@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-2.asm
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ 

```

Рис. 3.8: Создание файла

Код программы из пункта 7:

```

%include 'in_out.asm'
section .data
msg1 db 'Введите В:',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 В:'

```

```

mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- Записываем 'А' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'А' и 'С' (как символы)
cmp ecx,[C] ; Сравниваем 'А' и 'С'
jg check_B ; если 'А>С', 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,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в max
; ----- Сравниваем 'max(A,C)' и 'В' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'В'
jg fin ; если 'max(A,C)>В', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = В'
mov [max],ecx

```

; ——— Вывод результата

fin:

mov eax, msg2

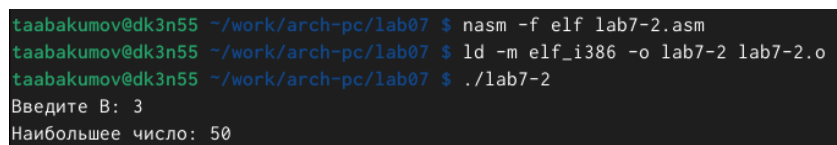
call sprint ; Вывод сообщения 'Наибольшее число:'

mov eax,[max]

call iprintLF ; Вывод 'max(A,B,C)'

call quit ; Выход

8. Создадим исполняемый файл и проверим его работу для разных значений (рис. 3.9).



```
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 3
Наибольшее число: 50
```

Рис. 3.9: Работа программы

Обратим внимание, в данном примере переменные A и C сравниваются как символы, а переменная B и максимум из A и C как числа (для этого используется функция atoi преобразования символа в число).

3.2 Изучение структуры файлы листинга

9. Создадим файл листинга для программы из файла lab7-2.asm, а затем откроем файл листинга lab7-2.lst с помощью текстового редактора mcedit:(рис. 3.10).



```
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst
```

Рис. 3.10: Создание файла

10. Откроем файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалим один операнд, после чего выполним трансляцию с получением файла листинг (рис. 3.11).

```

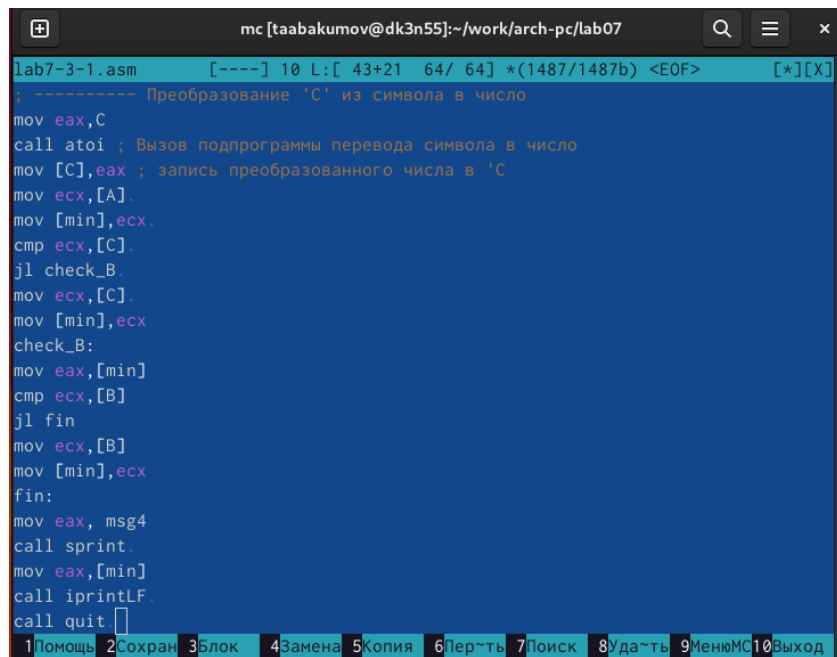
13                                     ; ----- Вывод сообщения 'Введите B:
14 000000E8 B8[00000000]             mov eax,msg1
15 000000ED E81DFFFFFF               call sprint
16                                     ; ----- Ввод 'B'
17                                     mov e
17 *****                          error: symbol `e' not defined
18 000000F2 BA0A000000               mov edx,10
19 000000F7 E847FFFFFF               call sread
20                                     ; ----- Преобразование 'B' из симво
21 000000FC B8[0A000000]             mov eax,B
22 00000101 E896FFFFFF               call atoi ; Вызов подпрограммы перевода
23 00000106 A3[0A000000]             mov [B],eax ; запись преобразованного чи
24                                     ; ----- Записываем 'A' в переменную
25 0000010B 8B0D[35000000]           mov ecx,[A] ; 'ecx = A'
26 00000111 890D[00000000]           mov [max],ecx ; 'max = A'

```

Рис. 3.11: Ошибка в листинге

3.3 Выполнение заданий для самостоятельной работы

11. 1 Задание: Написать 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 №6. Создать исполняемый файл и проверить его работу.
12. Для начала создадим файл lab7-3-1.asm. Мой вариант из прошлой лабораторной работы - 9, соответственно программа должна выводить минимальное число среди чисел: 24,98,15 (рис. 3.12).



```
lab7-3-1.asm [----] 10 L: [ 43+21 64/ 64] *(1487/1487b) <EOF> [*][X]
; ----- Преобразование 'C' из символа в число
mov eax,C
call atoi ; Вызов подпрограммы перевода символа в число
mov [C],eax ; запись преобразованного числа в 'C'
mov ecx,[A].
mov [min],ecx.
cmp ecx,[C].
jl check_B.
mov ecx,[C].
mov [min],ecx
check_B:
mov eax,[min]
cmp ecx,[B]
jl fin
mov ecx,[B]
mov [min],ecx
fin:
mov eax, msg4
call sprint.
mov eax,[min]
call iprintLF.
call quit

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 3.12: Код программы

Код программы из пункта 12:

include 'in_out.asm'

section .data

msg1 db 'Введите A:',0h

msg2 db 'Введите B:',0h

msg3 db 'Введите C:',0h

msg4 db "Наименьшее число:",0h

section .bss

min resb 10

A resb 10

B resb 10

C resb 10

section .text

global _start

_start:

; ----- Вывод сообщения 'Введите A:'

```

mov eax,msg1
call sprint
; ----- Ввод 'А'
mov ecx,A
mov edx,10
call sread
; ----- Преобразование 'А' из символа в число
mov eax,A
call atoi ; Вызов подпрограммы перевода символа в число
mov [A],eax ; запись преобразованного числа в 'В'
; ----- Вывод сообщения 'Введите В:'
mov eax,msg2
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
mov eax,B
call atoi
mov [B],eax
; ----- Вывод сообщения 'Введите С:'
mov eax,msg3
call sprint
; ----- Ввод 'С'
mov ecx,C
mov edx,10
call sread
; ----- Преобразование 'С' из символа в число
mov eax,C

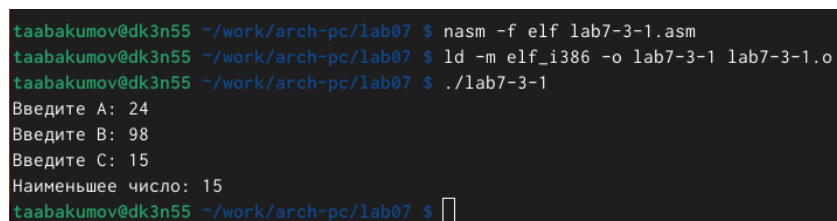
```

```

call atoi ; Вызов подпрограммы перевода символа в число
mov [C],eax ; запись преобразованного числа в 'C'
mov ecx,[A]
mov [min],ecx
cmp ecx,[C]
jl check_B
mov ecx,[C]
mov [min],ecx
check_B:
mov eax,[min]
cmp ecx,[B]
jl fin
mov ecx,[B]
mov [min],ecx
fin:
mov eax, msg4
call sprint
mov eax,[min]
call iprintLF
call quit

```

13. Создадим исполняемый файл и проверим его работу (рис. 3.13).



```

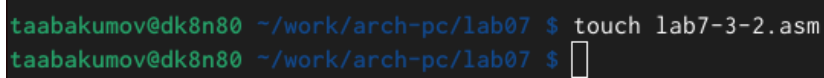
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3-1.asm
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3-1 lab7-3-1.o
taabakumov@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-3-1
Введите A: 24
Введите B: 98
Введите C: 15
Наименьшее число: 15
taabakumov@dk3n55 ~/work/arch-pc/lab07 $

```

Рис. 3.13: Работа программы

14. 2 Задание: Написать 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. После этого необходимо создать исполняемый файл и проверить его работу для значений x и a из 7.6.

Для начала создадим файл lab7-3-2.asm (рис. 3.14).



```
taabakumov@dk8n80 ~/work/arch-pc/lab07 $ touch lab7-3-2.asm
taabakumov@dk8n80 ~/work/arch-pc/lab07 $
```

Рис. 3.14: Создание файла

15. Мой вариант из прошлой лабораторной работы - 9, соответственно при вводе числа 5, должно выводиться число 7, и точно также при вводе числа 6, должно выводиться число 4 (рис. 3.15).

```

mov ecx,[X]
mov [F],ecx

cmp ecx,[A]
jl check_or
mov ecx,[A]
mov [F],ecx
jmp fin

check_or:
mov eax,[A]
mov ecx,[X]
add eax,ecx
mov [F],eax

fin:
mov eax,otv
call sprint
mov eax,[F]
call iprintLF
call quit

```

Рис. 3.15: Код программы

Код программы из пункта 15:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
prim1 DB 'a+x,x<=a',0
```

```
prim2 DB 'a, x>a',0
```

```
X1 DB 'Введите значение x:',0
```

```
A1 DB 'Введите значение a:',0
```

```
otv DB 'Ответ:',0
```

```
SECTION .bss
```

```
X RESB 20
```

```
A RESB 20
```

```

F RESB 20
SECTION .text
GLOBAL _start
_start:
mov eax,prim1
call sprintLF
mov eax,prim2
call sprintLF
mov eax,X1
call sprint
mov ecx,X
mov edx,10
call sread
mov eax,X
call atoi
mov [X],eax
mov eax,A1
call sprint
mov ecx,A
mov edx,10
call sread
mov eax,A
call atoi
mov [A],eax
mov ecx,[X]
mov [F],ecx
cmp ecx,[A]
jl check_or
mov ecx,[A]

```

```

mov [F],ecx
jmp fin
check_or:
mov eax,[A]
mov ecx, [X]
add eax,ecx
mov [F],eax
fin:
mov eax,otv
call sprint
mov eax,[F]
call iprintLF
call quit

```

16. После написания программы создадим исполняемый файл (рис. 3.16).

```

taabakumov@dk8n80 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3-2.asm
taabakumov@dk8n80 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3-2 lab7-3-2.o

```

Рис. 3.16: Создание исполняемого файла

17. Проверим работу программы (рис. 3.17).

```

taabakumov@dk8n80 ~/work/arch-pc/lab07 $ ./lab7-3-2
a+x ,x<=a
a, x>a
Введите значение x:5
Введите значение a:7
Ответ: 12
taabakumov@dk8n80 ~/work/arch-pc/lab07 $ ./lab7-3-2
a+x ,x<=a
a, x>a
Введите значение x:6
Введите значение a:4
Ответ: 4
taabakumov@dk8n80 ~/work/arch-pc/lab07 $ 

```

Рис. 3.17: Работа программы

4 Выводы

Были изучены основные принципы работы с условным и безусловным переходом в assembler и изучены основы чтения файлов листинга.