

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Морошан Матвей Корнелиович

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	9
3.3	Выполнение заданий для самостоятельной работы	11
4	Выводы	13

Список иллюстраций

3.1	Создание каталога, переход в него и создание файла	6
3.2	Копирование файла	6
3.3	Текст программы листинга 7.1	7
3.4	Создание и запуск файла	7
3.5	Текст программы листинга 7.2	7
3.6	Создание исполняемого файла и запуск его	8
3.7	Изменение текста программы	8
3.8	Создание файла и запуск	8
3.9	Создание файла	8
3.10	Текст программы листинга 7.3	9
3.11	Проверка программы	9
3.12	Создание файла листинга	10
3.13	Объяснение первой строки	10
3.14	Объяснение второй строки	10
3.15	Объяснение третьей строки	10
3.16	Удаление одного из операндов	10
3.17	Трансляция с получением файла листинга	10
3.18	2 Операнда	10
3.19	Создание файла	11
3.20	Текст программы задания №1	11
3.21	Создание файла и запуск	11
3.22	Создание файла	12
3.23	Текст программы задания №2	12
3.24	Создание файла и запуск	12

1 Цель работы

Целью данной лабораторной работы является 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

Создаю каталог для программ лабораторной работы №7, перехожу в него и создаю файл lab7-1.asm (рис. 3.1)

```
mkmoroshan@dk2n21 ~ $ mkdir ~/work/arch-pc/lab07
mkmoroshan@dk2n21 ~ $ cd ~/work/arch-pc/lab07
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 3.1: Создание каталога, переход в него и создание файла

Копирую файл in_out.asm из загрузок в соответствующую папку для дальнейшей работы (рис. 3.2)

```
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ cp ~/Загрузки/in_out.asm in_out.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ls
in_out.asm  lab7-1.asm
```

Рис. 3.2: Копирование файла

Ввожу в lab7-1.asm текст программы из листинга 7.1 (рис. 3.3)

```

lab7-1.asm  [~M--] 41 L:[ 1+19 20/ 20] *(649 / 649b) <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 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. 3.3: Текст программы листинга 7.1

Создаю исполняемый файл и запускаю его (рис. 3.4)

```

mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3

```

Рис. 3.4: Создание и запуск файла

Ввожу текст программы листинга 7.2 (рис. 3.5)

```

lab7-1.asm  [~M--] 41 L:[ 1+21 22/ 22] *(670 / 670b) <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 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. 3.5: Текст программы листинга 7.2

Создаю исполняемый файл и запускаю его (рис. 3.6)

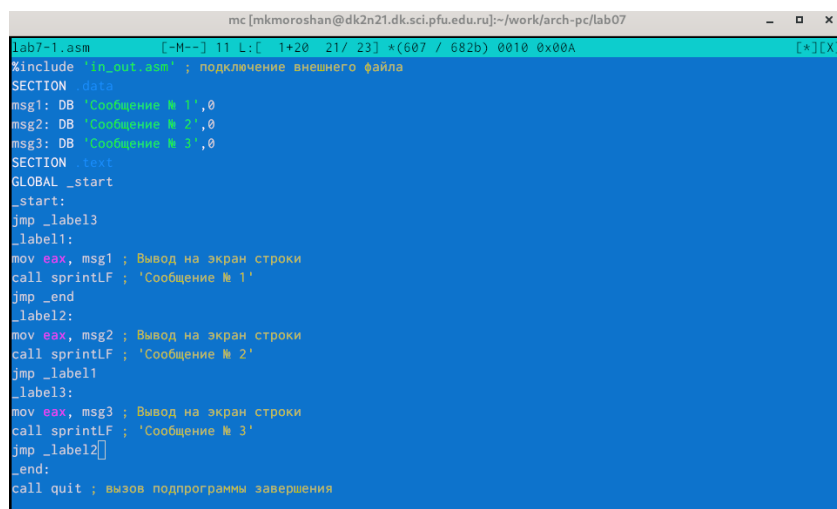
```

mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1

```

Рис. 3.6: Создание исполняемого файла и запуск его

Изменяю текст программы так, чтоб вывод был в нужной последовательности (рис. 3.7)



```

lab7-1.asm  [~M--] 11 L: [ 1+20 21/ 23] *(607 / 682b) 0010 0x00A  [*][X]
%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. 3.7: Изменение текста программы

Создаю исполняемый файл и запускаю его (рис. 3.8)

```

mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 3.8: Создание файла и запуск

Создаю файл lab7-2.asm (рис. 3.9)

```

mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ touch lab7-2.asm

```

Рис. 3.9: Создание файла

Ввожу текст программы из листинга 7.3 в файл (рис. 3.10)


```

lab7-2.asm [~M--] 17 L: [ 1+30 31/ 49] *(978 /1743b) 0109 0x06D [*][X]
#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,max
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

```

Рис. 3.10: Текст программы листинга 7.3

Создаю исполняемый файл и проверяю его работу для разных значений B (рис. 3.11)

```

mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 48
Наибольшее число: 50
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 52
Наибольшее число: 52

```

Рис. 3.11: Проверка программы

3.2 Изучение структуры файла листинга

Создаю файл листинга для программы из файла lab7-2.asm и открываю его (рис. 3.12)

```
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 3.12: Создание файла листинга

Объяснение строки: В регистр `eax` мы вносим значение 4 (рис. 3.13)

```
34 00000022 B804000000 <1> mov eax, 4
```

Рис. 3.13: Объяснение первой строки

Объяснение строки: В регистр `eax` мы вносим значение 3 (рис. 3.14)

```
66 0000004A B803000000 <1> mov eax, 3
```

Рис. 3.14: Объяснение второй строки

Объяснение строки: Вызов подпрограммы перевода символа в число (рис. 3.15)

```
35 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
```

Рис. 3.15: Объяснение третьей строки

Удаляю один из операндов (рис. 3.16)

```
14 000000E8 B8[00000000] mov eax,
```

Рис. 3.16: Удаление одного из операндов

Выполняю трансляцию с получением файла листинга, но ничего не выводит (рис. 3.17)

```
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $
```

Рис. 3.17: Трансляция с получением файла листинга

После трансляции захожу обратно и вижу, что операнд вернулся на место (рис. 3.18)

```
14 000000E8 B8[00000000] mov eax,msg1
```

Рис. 3.18: 2 Операнда

3.3 Выполнение заданий для самостоятельной работы

Создаю файл для задания №1 (рис. 3.19)

```
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ touch lab7-3.asm
```

Рис. 3.19: Создание файла

Записываю текст программы для нахождения наименьшего числа (рис. 3.20)

```
> mc [mkmoroshan@dk2n21.dk.sci.pfu.edu.ru:~/work/arch-pc/lab07]
lab7-3.asm [-M--] 17 L: [ 1+40 41/ 41] *(1487/1487b) <EOF> [*][X]
%include "in_out.asm"
section .data
msg2 db "Наименьшее число: ",0h
A dd '8'
C dd '88'
B dd '68'
section .bss
min resb 10
section .text
global _start
_start:
; ----- 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'
jb 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 ; Перевод 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
mov eax,[min]
call iprintfLF ; Вывод 'min(A,B,C)'
call quit ; Выход
```

Рис. 3.20: Текст программы задания №1

Создаю исполняемый файл и проверяю правильность работы (рис. 3.21)

```
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 8
```

Рис. 3.21: Создание файла и запуск

Создаю файл для задания №2 (рис. 3.22)

```
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ touch lab7-4.asm
```

Рис. 3.22: Создание файла

Записываю текст программы для вычисления выражения $f(x)$ (рис. 3.23)

```
> mc [mkmoroshan@dk2n21.dk.sci.pfu.edu.ru:~/work/arch-pc/lab07]
lab7-4.asm [-----] 8 L: [ 1+ 8 9/ 47] *(216 / 635b) 0048 0x030 [*)(X]
#include "in_out.asm"
section .data
msg1 db "Введите число X: ",0h
msg2 db "Введите число A: ",0h
msg3 db "Значение функции f(x) = ",0h
section .bss
X resb 10
A resb 10
F resb 10
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,X
mov edx,10
call sread
mov eax,X
call atoi
mov [X],eax
mov eax,msg2
call sprint
mov ecx,A
mov edx,10
call sread
mov eax,A
call atoi
mov [A],eax
mov eax,[A]
cmp eax,0
jne com1
mov eax,[X]
add eax,eax
add eax,1
mov[F],eax
jmp fin
com1:
mov eax,[X]
add eax,eax
add eax,[A]
mov[F],eax
fin:
```

Рис. 3.23: Текст программы задания №2

Создаю исполняемый файл, запускаю и убеждаюсь в правильности программы (рис. 3.24)

```
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-4
Введите число X: 3
Введите число A: 0
Значение функции f(x) = 7
mkmoroshan@dk2n21 ~/work/arch-pc/lab07 $ ./lab7-4
Введите число X: 3
Введите число A: 2
Значение функции f(x) = 8
```

Рис. 3.24: Создание файла и запуск

4 Выводы

При выполнении данной лабораторной работы я изучил команды условного и безусловного переходов, приобрёл навыки написания программ с использованием переходов, познакомился с назначением и структурой файла листинга