

Отчет по лабораторной работе №7

Дисциплина: Архитектура компьютера

Быкова Алина Александровна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задание для самостоятельной работы	16
4	Выводы	20

Список иллюстраций

2.1	Создание каталога и файла lab7-1.asm	6
2.2	Ввела программу из листинга	7
2.3	Запустила исполняемый файл	8
2.4	Изменение программы	8
2.5	Запустила исполняемый файл	9
2.6	Изменение программы	10
2.7	Запустила исполняемый файл	11
2.8	Создала файл lab7-2.asm	11
2.9	Ввела программу из листинга	12
2.10	Создала исполняемый файл	13
2.11	Проверила для 10	13
2.12	Проверила для 20	13
2.13	Проверила для 50	13
2.14	Открыла файл листинга lab7-2.lst	14
2.15	Появление ошибки в файле листинга	15
3.1	Изменение программы	17
3.2	Создала исполняемый файл	18
3.3	Изменение программы	19

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 Выполнение лабораторной работы

Создала каталог для программ лабораторной работы № 7, перешла в него и создала файл lab7-1.asm.

```
aabihkova@dk3n61 ~ $ mkdir ~/work/arch-pc/lab07  
aabihkova@dk3n61 ~ $ cd ~/work/arch-pc/lab07  
aabihkova@dk3n61 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 2.1: Создание каталога и файла lab7-1.asm

Ввела в файл lab7-1.asm текст программы из листинга.

```
GNU nano 6.4
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 2.2: Ввела программу из листинга

Создала исполняемый файл и запустила его. Результат работы данной программы: Сообщение № 2 Сообщение № 3

```

aabihekova@dk3n61 ~ $ cd ~/work/arch-pc/lab07
aabihekova@dk3n61 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aabihekova@dk3n61 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aabihekova@dk3n61 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3

```

Рис. 2.3: Запустила исполняемый файл

Изменила программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу.

```

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

```

Рис. 2.4: Изменение программы

Создала исполняемый файл и запустила его. Результат работы данной программы: Сообщение № 2 Сообщение № 1


```
aabihkova@dk3n61 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aabihkova@dk3n61 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aabihkova@dk3n61 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1
aabihkova@dk3n61 ~/work/arch-pc/lab07 $
```

Рис. 2.5: Запустила исполняемый файл

Изменила текст программы в соответствии с листингом, изменив инструкции jmp.

```

GNU nano 0.4 /afs/.../home/.../...
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 2.6: Изменение программы

Создала исполняемый файл и проверила его работу. Результат работы данной программы: Сообщение № 3 Сообщение № 2 Сообщение № 1

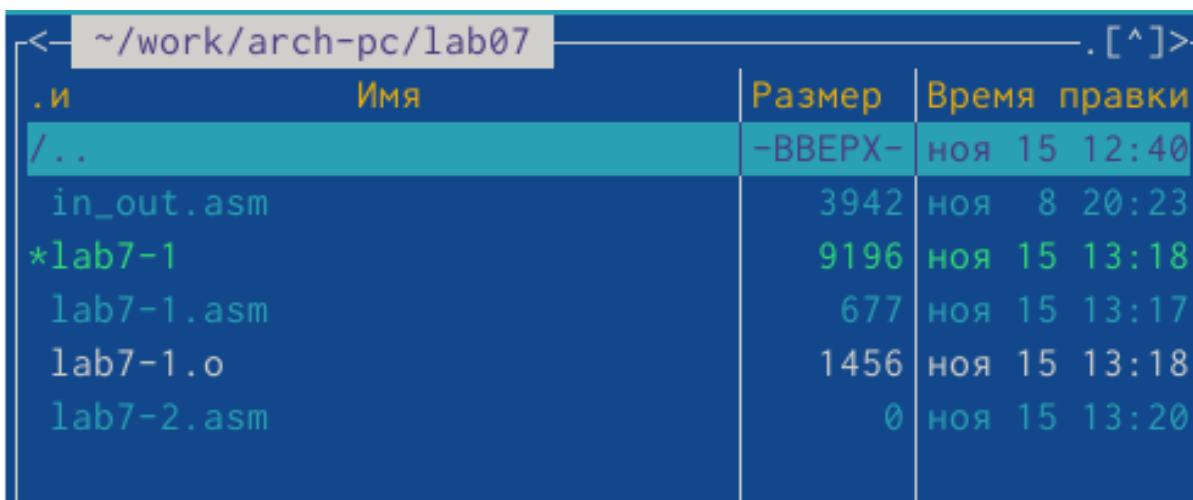
```

aabihkova@dk3n61 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
aabihkova@dk3n61 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
aabihkova@dk3n61 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
aabihkova@dk3n61 ~/work/arch-pc/lab07 $ █

```

Рис. 2.7: Запустила исполняемый файл

Создала файл lab7-2.asm в каталоге ~/work/arch-pc/lab07.



.и	Имя	Размер	Время правки
/..		-ВВЕРХ-	ноя 15 12:40
	in_out.asm	3942	ноя 8 20:23
*lab7-1		9196	ноя 15 13:18
	lab7-1.asm	677	ноя 15 13:17
	lab7-1.o	1456	ноя 15 13:18
	lab7-2.asm	0	ноя 15 13:20

Рис. 2.8: Создала файл lab7-2.asm

Ввела в файл lab7-2.asm текст программы из листинга.

```

%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)

```

Рис. 2.9: Ввела программу из листинга

Создала исполняемый файл и проверила его работу для разных значений В.

```
aabihkova@dk3n61 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
aabihkova@dk3n61 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aabihkova@dk3n61 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 
```

Рис. 2.10: Создала исполняемый файл

```
aabihkova@dk8n54 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
aabihkova@dk8n54 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aabihkova@dk8n54 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 10
Наибольшее число: 50
```

Рис. 2.11: Проверила для 10

```
aabihkova@dk8n54 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
aabihkova@dk8n54 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aabihkova@dk8n54 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 20
Наибольшее число: 50
```

Рис. 2.12: Проверила для 20

```
aabihkova@dk8n54 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
aabihkova@dk8n54 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
aabihkova@dk8n54 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 50
Наибольшее число: 50
```

Рис. 2.13: Проверила для 50

Создала файл листинга для программы из файла lab7-2.asm и открыла его с помощью текстового редактора mcedit.

```
lab7-2.lst [-----] 8 L: 12 0 1/225] <W 7/44585) 0012 0x020
1 %include "in_out.asm"
2 <<> ;----- silen -----
3 <<> ; функция вычисления длины сообщения
4 00000000 53 <<> silen:
5 00000001 85C3 <<> push ebx
6 <<> mov ebx, eax
7 <<> nextchar:
8 00000003 803800 <<> cmp byte [eax], 0
9 00000006 7403 <<> jz finished
10 00000008 40 <<> inc eax
11 00000009 EBF8 <<> jmp nextchar
12 <<>
13 <<> finished:
14 0000000B 29D8 <<> sub eax, ebx
15 0000000D 5B <<> pop ebx
16 0000000E C3 <<> ret
17 <<>
18 <<>
19 <<> ;----- sprint -----
20 <<> ; функция печати сообщения
21 <<> ; входные данные: mov eax, <message>
22 <<> sprint:
23 0000000F 52 <<> push edx
24 00000010 51 <<> push ecx
25 00000011 53 <<> push ebx
26 00000012 50 <<> push eax
27 00000013 E8E8FFFFFF <<> call silen
28 <<>
29 00000018 85C2 <<> mov edx, eax
30 0000001A 58 <<> pop eax
31 <<>
32 0000001B 85C1 <<> mov ecx, eax
33 0000001D BB01000000 <<> mov ebx, 1
34 00000022 BB04000000 <<> mov eax, 4
35 00000027 CD80 <<> int 80h
36 <<>
37 00000029 5B <<> pop ebx
38 0000002A 59 <<> pop ecx
```

Рис. 2.14: Открыла файл листинга lab7-2.lst

В строке 8 содержится адрес “00000003”, машинный код “803800” и содержимое строки кода “cmp byte [eax], 0”. В строке 10 содержится адрес “00000008”, машинный код “40” и содержимое строки кода “inc eax”. В строке 24 содержится адрес “00000010”, машинный код “51” и содержимое строки кода “push ecx”.

Открыла файл с программой lab7-2.asm и в инструкции удалила один операнд. Выполнила трансляцию. Если в коде появляется ошибка, то ее описание появится в файле листинга.

```

169 000000E5 CD80          <1>      int      80h
170 000000E7 C3          <1>      ret
2
3 00000000 D092D0B2D0B5D0B4D0-    section .data
3 00000009 B8D182D0B520423A20-    msg1 db 'Введите B: ',0h
3 00000012 00.....
4 00000013 D09DD0B0D0B8D0B1D0-    msg2 db "Наибольшее число: ",0h
4 0000001C BED0BBD18CD188D0B5-
4 00000025 D0B520D187D0B8D181-
4 0000002E D0BBD0BE3A2000.....
5 00000035 32300000          A dd '20'
6 00000039 35300000          C dd '50'
7
8 00000000 <res Ah>        section .bss
9 0000000A <res Ah>        max resb 10
10
11                                B resb 10
12                                section .text
13                                global _start
14                                _start:
15                                ; ----- Вывод сообщения 'Введите B: '
16                                mov eax,msg1
17                                error: invalid combination of opcode and operands
18                                call sprint
19                                ; ----- Ввод 'B'
20                                mov ecx,B
21                                mov edx,10
22                                call sread
23                                ; ----- Преобразование 'B' из символа в число
24                                mov eax,B
25                                call atoi ; Вызов подпрограммы перевода символа в число
26                                mov [B],eax ; запись преобразованного числа в 'B'
27                                ; ----- Записываем 'A' в переменную 'max'
28                                mov ecx,[A] ; 'ecx = A'
29                                mov [max],ecx ; 'max = A'
30                                ; ----- Преобразование 'max(A,C)' из символа в число

```

1 Помощь 2 Сохранить 3 Блок 4 Замена 5 Копия 6 Перезаписать 7 Поиск 8 Удалить 9 Меню MC 10 Выход

Рис. 2.15: Появление ошибки в файле листинга

3 Задание для самостоятельной работы

Вариант 17

Задание №1

Создала файл lab7-3.asm в каталоге ~/work/arch-pc/lab07 и написала программу нахождения наименьшей из 3 целочисленных переменных a,b и c. a=68 b=26 c=12


```

GNU nano 6.4 /afs/.dk.sci.pfu.с
%include 'in_out.asm'
section .data
msg2 db "Наименьшее число: ",0h
A dd '68'
B dd '26'
C dd '12'
section .bss
min resb 10
section .text
global _start
_start:
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'
jl 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, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '

```

Рис. 3.1: Изменение программы

Создала исполняемый файл и запустила его. Результат: программа вывела наименьшее число 12.

```
aabihkova@dk8n64 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
aabihkova@dk8n64 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
aabihkova@dk8n64 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 12
```

Рис. 3.2: Создала исполняемый файл

Задание №2

Создала файл lab7-4.asm в каталоге ~/work/arch-pc/lab07 и написала программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений.

```

#include 'in_out.asm'
section .data
input1 db "Введите x: ",0h
input2 db "Введите a: ",0h
section .bss
max resb 10
x resb 10
a resb 10
section .text
global _start
_start:
mov eax,input1
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
call atoi ;
mov [x],eax ;
mov eax,input2
call sprint
mov ecx,a
mov edx,10
call sread
mov eax,a
call atoi ;
mov [a],eax ;
mov ebx, 8
cmp [a], ebx
jge check_a
mov eax, [a]
mov ebx, 8
add eax, ebx
call iprintLF

```

Рис. 3.3: Изменение программы

4 Выводы

Я изучила команды условного и безусловного переходов и научилась писать программы с использованием этих переходов.