

Отчёт по лабораторной работе №8

Компьютерные науки и технология программирования

Сячинова Ксения Ивановна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задание для самостоятельной работы	15

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Текст программы	7
2.3	Компиляция и выполнение файла	7
2.4	Изменение программы	9
2.5	Результат	10
2.6	Изменение программы	11
2.7	Результат	12
2.8	Создание файла	12
2.9	Текст программы	12
2.10	Текст программы	13
2.11	Результат программы	13
2.12	Создание файла листинга	13
2.13	Строка 1	14
2.14	Строка 2	14
2.15	Строка 3	14
2.16	Удаление операнда	14
3.1	Создание файла	15
3.2	Текст программы	16
3.3	Текст программы	17
3.4	Выполнение программы	17
3.5	Система	17
3.6	Создание файла	17
3.7	Текст программы	18
3.8	Текст программы	19
3.9	Результат программы	19

Список таблиц

1 Цель работы

Изучить команды условного и безусловного перехода. Приобрести навыки написания программ с использованием переходов, а так же знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

1. Создаём каталог для программ лабораторной работы №8, переходим в него и создаём файл 'lab8-1.asm'.(рис. 2.1)

```
kisyachinova1@dk6n58 ~/work/arch-pc $ mkdir lab08
kisyachinova1@dk6n58 ~/work/arch-pc $ cd lab08
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 2.1: Создание каталога и файла

2. Рассмотрим пример программы, открываем файл и вводим текст программы.(рис. 2.2), (рис. 2.3)

```

lab8-1.asm      [----] 27 L:
#include 'in_out.asm'

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
    call sprintLF
    ....
_label2:
    mov eax, msg2
    call sprintLF
    ....
_label3:
    mov eax, msg3
    call sprintLF
    ....
_end:
    call quit

```

Рис. 2.2: Текст программы

```

kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение № 2
Сообщение № 3

```

Рис. 2.3: Компиляция и выполнение файла

Команда 'jmp' позволяет начать использование инструкции с отмеченной метки, в нашем случае с '_label2'.

Изменим программу так, чтобы она выводила сначала 'Сообщение №2', потом 'Сообщение №1' и завершала работу.(рис. 2.4), (рис. 2.5)


```

lab8-1.asm      [----] 15 L
#include 'in_out.asm'

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
    call sprintLF
    jmp _end

....
_label2:
    mov eax, msg2
    call sprintLF
    jmp _label1

....
_label3:
    mov eax, msg3
    call sprintLF

....
_end:
    call quit

```

Рис. 2.4: Изменение программы

```
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение № 2
Сообщение № 1
```

Рис. 2.5: Результат

Затем, изменим программу так, чтобы сообщения выводились в обратном порядке. Программы работает корректно. (рис. 2.6), (рис. 2.7)

```

lab8-1.asm      [----] 0
%include 'in_out.asm'

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
    call sprintLF
    jmp _end
....
_label2:
    mov eax, msg2
    call sprintLF
    jmp _label1
....
_label3:
    mov eax, msg3
    call sprintLF
    jmp _label2
....
_end:
    call quit

```

Рис. 2.6: Изменение программы

```

kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 2.7: Результат

3. Создадим файл 'lab8-2.asm' и вводим текст программы.(рис. 2.8), (рис. 2.9), (рис. 2.10),(рис. 2.11)

```

kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ touch lab8-2.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ mcedit lab8-2.asm

```

Рис. 2.8: Создание файла

```

lab8-2.asm      [----]  8 L:[  1+ 0  1
#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:
mov eax, msg1
call sprint

mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi
mov [B], eax

```

Рис. 2.9: Текст программы

```

mov ecx,[A]
mov [max],ecx
[
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx

check_B:
mov eax,max
call atoi
mov [max], eax

mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx

fin:
mov eax, msg2
call sprint
mov eax, [max]
call iprintLF
call quit

```

Рис. 2.10: Текст программы

```

kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ ./lab8-2
Введите B: 10
Наибольшее число: 50

```

Рис. 2.11: Результат программы

4. Для получения файл листинга указывает ключ '-l'. Создадим файл листинга для программы из файла 'lab8-2.asm'. и откроем его с помощью текстового редактора.(рис. 2.12)

```

kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ nasm -f elf -l lab8-2.lst lab8-2.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab08 $ mcedit lab8-2.lst

```

Рис. 2.12: Создание файла листинга

Объясним содержимое трёх строк файла.

- 1) 24 - номер строки файла листинга, 00000106 - смещение машинного кода от начала текущего сегмента, E891FFFFFF - машинный код, в который ассемблируется данная инструкция в виде шестнадцатиричной последовательности, `call atoi` - исходная строка программы.(рис. 2.13)

```
24 00000106 E891FFFFFF      call atoi
```

Рис. 2.13: Строка 1

- 2) 47 - номер строка файла листинга, 00000159 - смещение машинного кода от начала текущего сегмента, 'B8[13000000]' - машинный код, в который ассемблируется данная инструкция в виде шестнадцатиричной последовательности, 'mov eax,msg1' - исходная строка программы. (рис. 2.14)

```
47 00000159 B8[13000000]    mov eax, msg2
```

Рис. 2.14: Строка 2

- 3) 20 - номер строка файла листинга, 000000F7 - смещение машинного кода от начала текущего сегмента, 'BA0A000000' - машинный код, в который ассемблируется данная инструкция в виде шестнадцатиричной последовательности, 'mov edx,10' - исходная строка программы. (рис. 2.15)

```
20 000000F7 BA0A000000     mov edx,10
```

Рис. 2.15: Строка 3

Удалим один операнд в программе, выполним трансляцию с получением файла. В файле листинга нам также выдаёт ошибку. (рис. 2.16)

```
47      mov eax,  
47      ***** error: invalid combination of opcode and operands  
48 00000159 E8D1FFFFFF     call msg1
```

Рис. 2.16: Удаление операнда

3 Задание для самостоятельной работы

1. Напишем программу для нахождения наименьшей из 3 целочисленных переменных a, b и c. В соответствии с лабораторной вариантом №7 мой вариант 11. Числа A=21, B=28, C=34 (рис. 3.1), (рис. 3.2), (рис. 3.3), (рис. 3.4),

A screenshot of a terminal window with a black background and green text. It shows two commands being executed in a directory ~/work/arch-pc/lab08. The first command is 'touch lab8-1.1.asm' and the second is 'mcedit lab8-1.1.asm'. Both commands are preceded by the prompt 'kisyachinova1@dk6n52 \$'.

```
kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ touch lab8-1.1.asm
kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ mcedit lab8-1.1.asm
```

Рис. 3.1: Создание файла

```

lab8-1.1.asm      [----]  9 L:[  1+ 0  1/
#include 'in_out.asm'
section .data
msg1 db 'Введите В: ',0h
msg2 db 'Наименьшее число: ',0h
A dd '21'
C dd '34'

section .bss
min<--->resb 10
B<----->resb 10

section .text

global _start
_start:
mov eax,msg1
call sprint

mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi
mov [B],eax

```

Рис. 3.2: Текст программы


```

mov ecx,[A]
mov [min],ecx

cmp ecx,[C]
jb check_B
mov ecx,[C]
mov [min],ecx

check_B:
mov eax,min
call atoi
mov [min],eax

mov ecx,[min]
cmp ecx,[B]
jb fin
mov ecx,[B]
mov [min],ecx

fin:
mov eax, msg2
call sprint
mov eax,[min]
call iprintLF
call quit

```

Рис. 3.3: Текст программы

```

kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.1.asm
kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1.1 lab8-1.1.o
kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ ./lab8-1.1
Введите B: 28
Наименьшее число: 21

```

Рис. 3.4: Выполнение программы

2. Напишем программу, которая для введённых значений 'x' и 'a' вычисляет значение заданной функции. Согласно лабораторной №7, мой вариант 11. Тогда имеем систему рис. 3.5. Проверим её с помощью значений (0;3) и (1;2)

$$11 \quad \begin{cases} 4a, & x = 0 \\ 4a + x, & x \neq 0 \end{cases} \quad (0;3) \quad (1;2)$$

Рис. 3.5: Система

```

kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ touch lab8-1.2.asm
kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ mcedit lab8-1.2.asm

```

Рис. 3.6: Создание файла

```

lab8-1.2.asm      [----]  0 L:[ 1+10 11/ 6
#include 'in_out.asm'
section .data
msg1: db 'Введите x: ',0h
msg2: db 'Введите a: ',0h
msg3: db 'Функция равна: ',0h

section .bss
x:<---->resb 10
a:<---->resb 10
answer:>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

```

Рис. 3.7: Текст программы

```

....
    mov eax,a
    call atoi
    mov [a],eax
....
    mov eax, [x]
....
    cmp eax, 0
....
    je Func1
    jne Func2
....
Func1:
    mov eax, 4
    mov ebx, [a]
    mul ebx
    mov [answer], eax
    jmp Final
....
Func2:
    mov eax, 4
    mov ebx, [a]
    mul ebx
    add eax, [x]
    mov [answer], eax
    jmp Final
Final:
    mov eax,msg3
    call sprint
    mov eax, [answer]
    call iprintLF
    call quit

```

Рис. 3.8: Текст программы

Программа выдаёт верный результат. При (0;3) имеем значение выражение равное 12, в при (1;2) равное 9.(рис. 3.9),

```

kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.2.asm
kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1.2 lab8-1.2.o
kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ ./lab8-1.2
Введите x: 0
Введите a: 3
Функция равна: 12
kisyachinova1@dk6n52 ~/work/arch-pc/lab08 $ ./lab8-1.2
Введите x: 1
Введите a: 2
Функция равна: 9

```

Рис. 3.9: Результат программы

#Вывод

В ходе выполнения данной лабораторной работы я изучила команды условного и безусловного перехода. Преобрела навыки по написанию программ с использованием переходов. Так же познакомилась с назначением и структурой файла листинга