

Отчёт по лабораторной работе №7

Архитектура компьютера

Морозова Мария Вячеславовна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выполнение самостоятельной работы	11
6	Выводы	12
7	Листинги	13

Список иллюстраций

4.1	Создание файла	8
4.2	Создание файла, запуск	8
4.3	Создание файла, запуск	8
4.4	Результат работы программы	9
4.5	Создание файла, запуск	9
4.6	Файл листинга	10
4.7	Ошибка	10
5.1	Результат работы программы	11
5.2	Компоновка файла, запуск программы.	11

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

Написать программу для нахождения наименьшего из трёх чисел, программу для вычисления значения функции $f(x)$.

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: • условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

4 Выполнение лабораторной работы

Создала каталог для программ лабораторной работы № 7, перешла в него и создала файл lab7-1.asm: (рис. 4.1).

```
mvmorozova@dk8n81 ~ $ mkdir ~/work/arch-pc/lab07
mvmorozova@dk8n81 ~ $ cd ~/work/arch-pc/lab07
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ touch lab7-1.asm
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $
```

Рис. 4.1: Создание файла

Создала исполняемый файл и запустила его с кодом листинга 7.1. (рис. 4.2).

```
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $
```

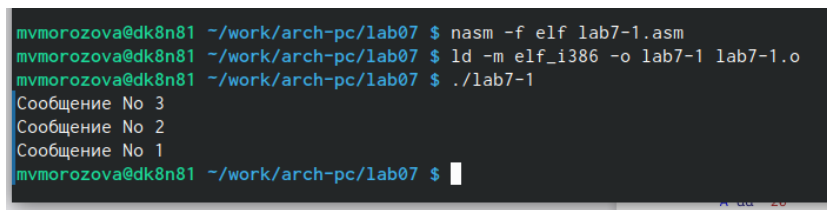
Рис. 4.2: Создание файла, запуск

Создала исполняемый файл и запустила его с кодом листинга 7.2. (рис. 4.3).

```
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $
```

Рис. 4.3: Создание файла, запуск

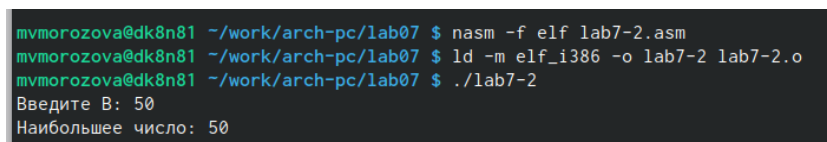
Создала исполняемый файл и запустила его с изменениями инструкции `jmp`. (рис. 4.4).



```
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $
```

Рис. 4.4: Результат работы программы

Создала файл `lab7-2.asm` в каталоге `~/work/arch-pc/lab07`. Запустила с использованием программы из листинга 7.3. (рис. 4.5).



```
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
mvmorozova@dk8n81 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 50
Наибольшее число: 50
```

Рис. 4.5: Создание файла, запуск

Создала файл листинга для программы из файла `lab7-2.asm`, открыла файл листинга `lab7-2.lst` с помощью текстового редактора. Рассмотрим несколько строк: строке 9 содержится номер сторки [9], адресс [00000003], машинный код [803800] и содержимое строки кода [`cmp byte [eax], 0`] в строке 11 содержится номер сторки [11], адресс [00000008], машинный код [40] и содержимое строки кода [`inc eax`] в строке 24 содержится номер сторки [24], адресс [0000000F], машинный код [52] и содержимое строки кода [`push edx`] (рис. 4.6).

```

lab7-2.lst      [----]  0 L:[ 1+ 0  1/225] *(0  /14458b) 0032 0x020
1               %include 'in_out.asm'
1               <1> ;----- slen -----
2               <1> ; Функция вычисления длины сообщения
3               <1> slen:
4 00000000 53    <1> push    ebx
5 00000001 89C3  <1> mov     ebx, eax
6               <1>
7               <1> nextchar:
8 00000003 803800 <1> cmp     byte [eax], 0
9 00000006 7403   <1> jz      finished
10 00000008 40    <1> inc     eax
11 00000009 EBF8  <1> jmp     nextchar
12               <1>
13               <1> finished:
14 0000000B 29D8  <1> sub     eax, ebx
15 0000000D 5B    <1> pop     ebx
16 0000000E C3    <1> ret
17               <1>
1 Помощь 2 Сохранить 3 Блок 4 Замена 5 Копия 6 Переместить 7 Поиск

```

Рис. 4.6: Файл листинга

Удалила один операнд, чтобы посмотреть какая будет получена ошибка. (рис. 4.7).

```

14 000000E8 B3[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16               ; ----- Ввод 'B'
17               mov ecx
17               ***** error: invalid combination of opcode and operands
18 000000F2 BA0A000000 mov edx,10
19 000000F7 E847FFFFFF call sread
20               ; ----- Преобразование 'B' из символа в число

```

Рис. 4.7: Ошибка

5 Выполнение самостоятельной работы

Запустила программу для вычисления наименьшего из трёх чисел. (рис. 5.1).

```
mvmorozova@dk5n59 ~/work/arch-pc/lab07 $ mc
mvmorozova@dk5n59 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 15
mvmorozova@dk5n59 ~/work/arch-pc/lab07 $
```

Рис. 5.1: Результат работы программы

Создала исполняемый файл и запустила программу для нахождения $f(x)$, проверила работу для двух пар x, a . (рис. 5.2).

```
mvmorozova@dk4n62 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
mvmorozova@dk4n62 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
mvmorozova@dk4n62 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 1
Введите a: 1
6
mvmorozova@dk4n62 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 2
Введите a: 1
3
mvmorozova@dk4n62 ~/work/arch-pc/lab07 $ mc
mvmorozova@dk4n62 ~/work/arch-pc/lab07 $
```

Рис. 5.2: Компоновка файла, запуск программы.

6 Выводы

Изучили команды условного и безусловного переходов. Приобрели навыки написания программ с использованием переходов. Познакомились с назначением и структурой файла листинга.

7 Листинги

lab7-3

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

```
section .data
```

```
    msg1 db "Наименьшее число:"
```

```
    a dd 45
```

```
    b dd 67
```

```
    c dd 15
```

```
section .bss
```

```
    min resb 10
```

```
section .text
```

```
global _start
```

```
_start:
```

```
    mov eax, msg1
```

```
    call sprint
```

```
    mov ecx, [a]
```

```
    mov [min], ecx ; 'min = A'
```

```

; ----- Сравниваем 'A' и 'C' (как числа)
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:

```

; ----- 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, [min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

lab7-4

```

#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, [x]
cmp [a], ebx
je check
```

```
mov eax, [a]
mov ebx, [x]
add eax, ebx
call iprintLF
call quit
```

```
check:
mov eax, [a]
mov ebx, 6
mul ebx
call iprintLF
call quit
```