

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Максимова Дарья Валерьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файлы листинга	12
4.3	Задание для самостоятельной работы	15
4.4	задание№1	15
4.5	задание№2	17
5	Выводы	18

Список иллюстраций

4.1	листинг	8
4.2	Результат	8
4.3	редактирование текста	9
4.4	редактирование	9
4.5	вывод	10
4.6	текст программы	11
4.7	проверяю программу	12
4.8	создание файла листинга	12
4.9	файл листинга	12
4.10	три строчки программы	13
4.11	исходное	14
4.12	после редактирования	14
4.13	итог	14
4.14	мой файл для заданий	15
4.15	текст программы	16
4.16	вывод программы	17
4.17	текст программы	17
4.18	Проверка программы	17

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

1.Реализация переходов в NASM 1.Изучение структуры файлы листинга 1.Задание для самостоятельной работы 1.задание№1 1.задание№2

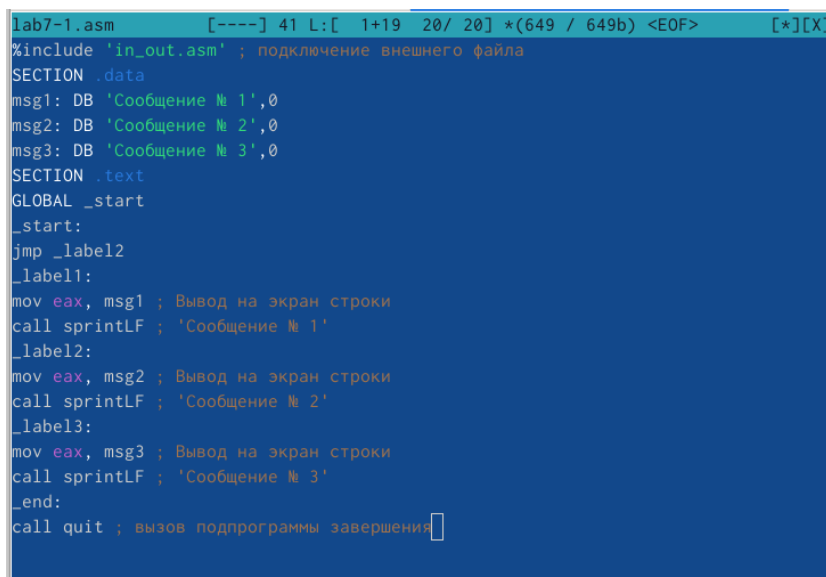
3 Теоретическое введение

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

4.1 Реализация переходов в NASM

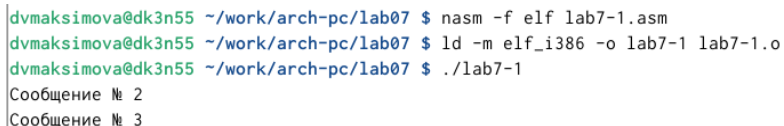
Создаю каталог для выполнения лабораторной работы, перехожу в него и там создаю файл lab7-1.asm. Затем ввожу в этот файл текст с листинга 7.1 (рис. 4.1).



```
lab7-1.asm [----] 41 L: [ 1+19 20/ 20] *(649 / 649b) <EOF> [*][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 _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. 4.1: листинг

Создаю исполняемый файл и запускаю его. Результат программы получился вот таким (рис. 4.2).



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

Рис. 4.2: Результат

В соответствии с листингом 7.2 я редактирую текст программы и теперь программа выводит на экран сначала строчку “Сообщение №2”, а затем строчку “Сообщение №1” и завершает работу: (рис. 4.3).

```

dvmaksimova@dk3n55 ~/work/arch-pc/lab07 $ mc

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

```

Рис. 4.3: редактирование текста

Затем я самостоятельно редактирую тест программы: (рис. 4.4).

```

lab7-1.asm      [----] 41 L: [ 1+22 23/ 23] *(682 / 682b) <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 _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. 4.4: редактирование

После редактирования программы вывод на экран получается таким (рис. 4.5).

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

Рис. 4.5: ВЫВОД

Для следующего задания я создаю файл lab7-2.asm и ввожу текст из листинга 7.3(рис. 4.6).

```

lab7-2.asm      [-M--] 17 L:[ 11+38  49/ 49] *(1743/1743b
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
; ----- 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

```

Рис. 4.6: текст программы

И проверяю программу, вводя разные переменные B, которые у меня запрашивает программа(рис. 4.7).

```

dvmaksimova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
dvmaksimova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
dvmaksimova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 8777
Наибольшее число: 8777
dvmaksimova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 1
Наибольшее число: 50
dvmaksimova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 51
Наибольшее число: 51
dvmaksimova@dk3n55 ~/work/arch-pc/lab07 $ 

```

Рис. 4.7: проверяю программу

4.2 Изучение структуры файлы листинга

Для того, чтобы получить файл листинга я указываю ключ `-l` и задаю имя файла листинга в командной строке. Пользуясь этим, создаю файл листинга моего файла `lab7-2.asm`(рис. 4.8).

```

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

```

Рис. 4.8: создание файла листинга

Открываю файл листинга, который выглядит следующим образом:(рис. 4.9).

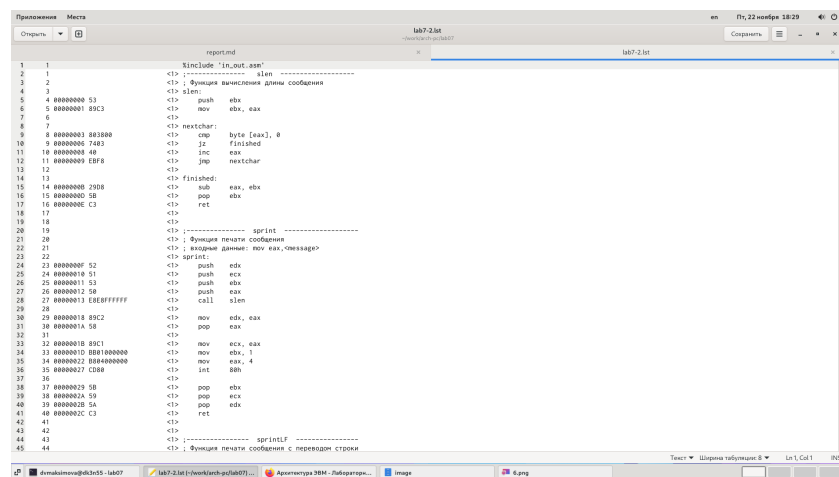


Рис. 4.9: файл листинга

Комментируя данные строчки, могу утверждать, что первая строчка отвечает за перемещение символа `B` в переменную `eax`, второй строчкой мы вызываем

попрограмму atoi, которая в свою очередь переводит символ в число, и третья строка выполняет перемещение eax в переменную "B". Таким образом, введя в программу три эти строки, мы преобразовали символ в число, которое теперь находится в "B".(рис. 4.10).

```
;  
-----  
mov  eax,B  
call atoi ;  
mov  [B],eax
```

Рис. 4.10: три строки программы

Я открываю файл lab7-2.asm и редактирую так, что в любой инструкции удаляю один из двух операндов (рис. 4.11).

```

,
mov ecx,B
mov edx
call sread
. ----- |

```

Рис. 4.11: исходное

редактирую(рис. 4.12).

```

,
mov ecx,B
mov edx,10
call sread
. ----- |

```

Рис. 4.12: после редактирования

Вот что получаю, когда хочу сделать файл листинга(рис. 4.13).

```

dvmaksimova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:18: error: invalid combination of opcode and operands

```

Рис. 4.13: итог

4.3 Задание для самостоятельной работы

При выполнении лабораторной работы №6 у меня был второй вариант, поэтому при выполнении заданий я буду использовать значения переменных соответственные второму варианту.

4.4 задание№1

В том же каталоге, где я выполняла свою лабораторную работу, я создаю файл для выполнения задания с именем test.asm(рис. 4.14).

```
dvmaksimova@dk3n55 ~/work/arch-pc/lab07 $ touch nomer1.asm
```

Рис. 4.14: мой файл для заданий

Затем пишу программу, которая найдет наименьшую целочисленную переменную из a,b и c(рис. 4.1).

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'наименьшее значение: ',0h
4 A dd 82
5 B dd 61
6 C dd 59
7 section .bss
8 min resb 10
9 section .text
10 global _start
11 _start:
12
13 mov eax,[A]
14 mov [min],eax
15 mov ebx,[B]
16 mov ecx,[C]
17
18 cmp eax,ebx;сравниваю а и б
19 jl aaa
20 mov [min],ebx
21
22 aaa:
23 mov ebx,[min]
24 cmp ebx,ecx
25 jl fff
26 mov [min],ecx
27
28 fff:
29 mov eax,msg1
30 call sprint
31 mov eax,[min]
32 call iprintLF
33 call quit

```

Рис. 4.15: текст программы

Создаю исполняемый файл и запускаю. Всё работает правильно, поэтому перехожу к выполнению второго задания (рис. 4.16).

```
dvmaksimova@dk8n63 ~/work/arch-pc/lab07 $ touch test.asm
```

Рис. 4.16: вывод программы

4.5 задание№2

Создаю файл test2.asm, в который самостоятельно записываю программу для выполнения задания(рис. 4.17).

```
dvmaksimova@dk8n63 ~/work/arch-pc/lab07 $ nasm -f elf test.asm
dvmaksimova@dk8n63 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o test test.o
dvmaksimova@dk8n63 ~/work/arch-pc/lab07 $ ./test
наименьшее значение: 59
```

Рис. 4.17: текст программы

Создаю исполняемый файл, запускаю его, ввожу переменные и проверяю работу для введенных мной значений из таблицы 7.6(рис. 4.18).

```
dvmaksimova@dk8n63 ~/work/arch-pc/lab07 $ nasm -f elf testt.asm
dvmaksimova@dk8n63 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o testt testt.o
dvmaksimova@dk8n63 ~/work/arch-pc/lab07 $ ./testt
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 6
dvmaksimova@dk8n63 ~/work/arch-pc/lab07 $ ./testt
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
```

Рис. 4.18: Проверка программы

5 Выводы

Я изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. и Знакомство с назначением и структурой файла листинга.