

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

-04-24

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

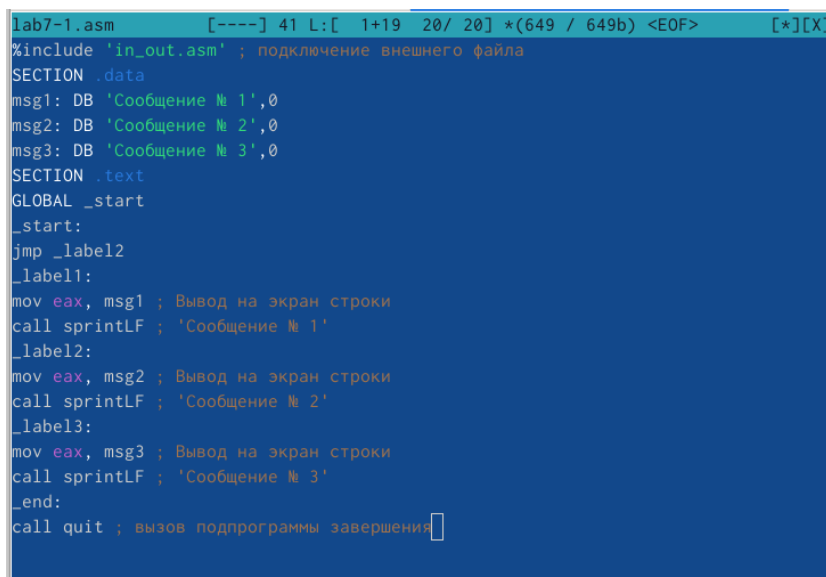
3 Теоретическое введение

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

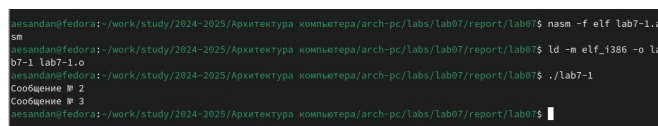
Создаю каталог для выполнения лабораторной работы, перехожу в него и там создаю файл lab7-1.asm. Затем ввожу в этот файл текст с листинга 7.1 (рис. 4.1).



```
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).



```
mesandam@fedora1:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/labs/lab07/report/lab07$ nasm -f elf lab7-1.asm
mesandam@fedora1:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/labs/lab07/report/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mesandam@fedora1:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/labs/lab07/report/lab07$ ./lab7-1
Сообщение № 1
Сообщение № 2
Сообщение № 3
mesandam@fedora1:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/labs/lab07/report/lab07$
```

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).

```
aesandangfedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ nasm -f elf lab7-1.asm
aesandangfedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aesandangfedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
aesandangfedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/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).

```
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ nasm -f elf lab7-1.asm
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$
```

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).

```

Введите В: 100
Наибольшее число: 100
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ nasm -f elf lab7-2.asm
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ ./lab7-2
Введите В: 49
Наибольшее число: 50
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/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).

```

aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/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).

```

1 1          ;include 'in_out.asm'
2 1          ;<> ;----- silen -----
3 2          ;<> ; Функция вычисления длины сообщения
4 3          ;<> silen:
5 4 00000000 51          ;<> push ebx
6 5 00000000 89C3        ;<> mov ebx, eax
7 6          ;<>
8 7          ;<> nextchar:
9 8 00000000 000000      ;<> cmp byte [eax], 0
10 9 00000000 7403        ;<> jz finished
11 10 00000000 40          ;<> inc eax
12 11 00000000 EBF8        ;<> jmp nextchar
13 12          ;<>
14 13          ;<> finished:
15 14 00000000 2008        ;<> sub eax, ebx
16 15 00000000 5B          ;<> pop ebx
17 16 00000000 C3          ;<> ret
18 17          ;<>
19 18          ;<>
20 19          ;<> ;----- print -----
21 20          ;<> ; Функция печати сообщения
22 21          ;<> print:
23 22          ;<> ;<>
24 23 00000000 52          ;<> push edx
25 24 00000000 51          ;<> push ecx
26 25 00000000 53          ;<> push ebx
27 26 00000000 50          ;<> push eax
28 27 00000000 EB07        ;<> call silen
29 28          ;<>
30 29 00000000 89C2        ;<> mov edx, eax
31 30 00000000 5B          ;<> pop ebx
32 31          ;<>
33 32 00000000 89C1        ;<> mov ecx, eax
34 33 00000000 00000000    ;<> mov ebx, 1
35 34 00000000 00000000    ;<> mov eax, 4
36 35 00000000 C906        ;<> int 0x06
37 36          ;<>
38 37 00000000 5B          ;<> pop ebx
39 38 00000000 5A          ;<> pop ecx
40 39 00000000 5A          ;<> pop edx
41 40 00000000 C3          ;<> ret
42 41          ;<>
43 42          ;<>
44 43          ;<> ;----- printf -----
45 44          ;<> ; Функция печати сообщения с переводом строки.

```

Рис. 4.9: файл листинга

Комментируя данные строки, могу утверждать, что первая строка отвечает за перемещение символа В в переменную `eax`, второй строкой 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

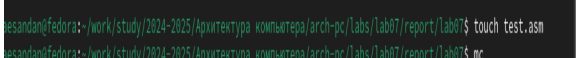
Выдает ошибку,когда хочу сделать файл листинга.

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).



```
asandam@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ touch test.asm
asandam@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ nc
```

Рис. 4.14: мой файл для заданий

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

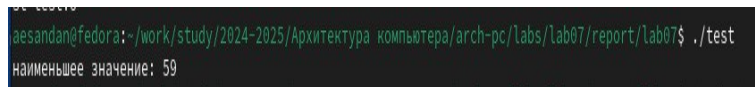
```

#include 'in_out.asm'
SECTION .data
msg1: DB 'наименьшее значение: ', 0h
A dd 82
B dd 61
C dd 59
SECTION .bss
min resb 10
SECTION .text.
GLOBAL _start.
_start:
mov eax, [A]
mov [min],eax
mov ebx, [B]
mov ecx, [C]
cmp eax,ebx; сравниваю а и б
jl aaa
mov [min],ebx
aaa:
mov ebx, [min]
cmp ebx,ecx
jl fff.
mov [min],ecx
fff:
mov eax,msg1.
call sprint.
mov eax, [min].
call iprintLF
call quit

```

Рис. 4.15: текст программы

Создаю исполняемый файл и запускаю. Всё работает правильно. (рис. 4.16).

A terminal window with a dark background. The prompt is 'aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07\$'. The command './test' has been entered. The output is 'наименьшее значение: 59'.

```
aesandan@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab07/report/lab07$ ./test
наименьшее значение: 59
```

Рис. 4.16: вывод 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.