

Отчет по выполнению лабораторной работы №7

Дисциплина: архитектура компьютеров

Намруев Максим Саналович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Изучение структуры файла листинга	13
5	Задание для самостоятельной работы	16
6	Выводы	23
	Список литературы	24

Список иллюстраций

4.1	Создание файла	8
4.2	Ввод текста из листинга 7.1	9
4.3	Запуск файла	9
4.4	Изменение файла	10
4.5	Запуск измененного файла	10
4.6	Редактирование файла	11
4.7	Запуск программы	11
4.8	Ввод текста из листинга 7.3	12
4.9	Запуск программы	13
4.10	Создание файла листинга	13
4.11	Открывание файла листинга	14
4.12	Удаление операнда	15
4.13	Попытка создание файла	15
5.1	Создание файла для самостоятельной работы	16
5.2	Написание программы	17
5.3	Проверка работы программы	17
5.4	Создание файла	19
5.5	Написание программы	20
5.6	Проверка программы	20

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
2. Изучение структуры файлы листинга

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. Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

4 Выполнение лабораторной работы

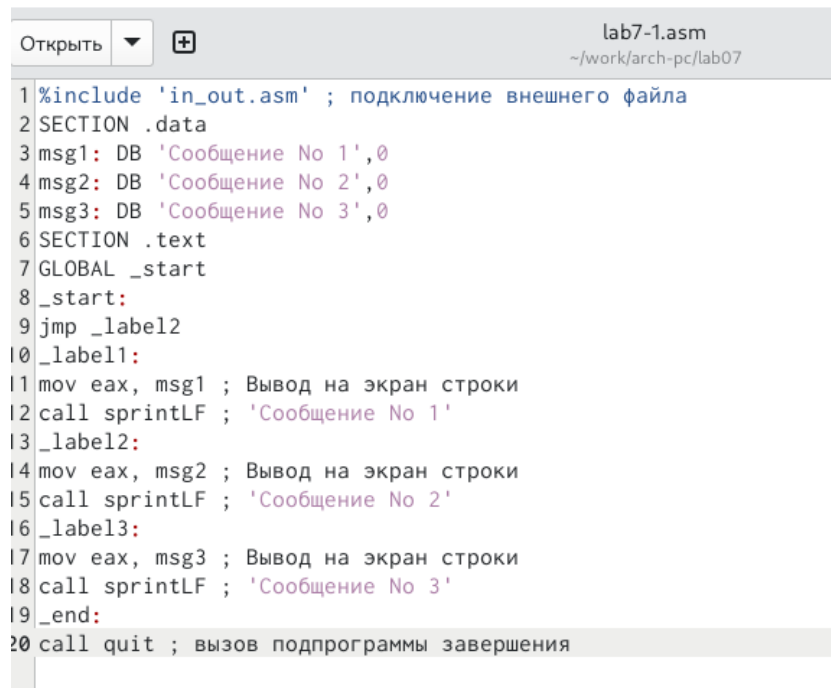
##Реализация переходов в NASM

Открываю терминал и создаю каталог для программ лабораторной работы, перехожу в него и создаю файл lab7-1.asm (рис. [4.1]).

```
msnamruev@dk8n60 ~ $ mkdir ~/work/arch-pc/lab07
msnamruev@dk8n60 ~ $ cd ~/work/arch-pc/lab07
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ touch lab7-1.asm
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ ls
lab7-1.asm
msnamruev@dk8n60 ~/work/arch-pc/lab07 $
```

Рис. 4.1: Создание файла

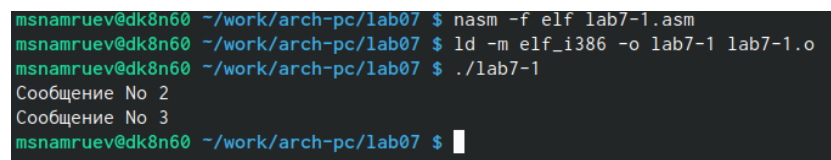
Ввожу в файл lab7-1.asm текст программы из листинга 7.1.(рис. [4.2])



```
Открыть ▼ + lab7-1.asm
~/work/arch-pc/lab07
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение No 1',0
4 msg2: DB 'Сообщение No 2',0
5 msg3: DB 'Сообщение No 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintLF ; 'Сообщение No 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintLF ; 'Сообщение No 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintLF ; 'Сообщение No 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Ввод текста из листинга 7.1

Создаю исполняемый файл и запускаю его.(рис. [4.3])



```
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
msnamruev@dk8n60 ~/work/arch-pc/lab07 $
```

Рис. 4.3: Запуск файла

Изменяю текст программы в соответствии с листингом 7.2.(рис. [4.4])

```

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

```

Рис. 4.4: Изменение файла

Создаю исполняемый файл и запускаю его.(рис. [4.5])

```

msnamruev@dk8n60 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1

```

Рис. 4.5: Запуск измененного файла

Далее изменяю файл так, чтобы он выводил “Сообщения” в обратном порядке.(рис. [4.6])

```

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

```

Рис. 4.6: Редактирование файла

Создаю исполняемый файл и запускаю его.(рис. [4.7])

```

msnamruev@dk8n60 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
msnamruev@dk8n60 ~/work/arch-pc/lab07 $

```

Рис. 4.7: Запуск программы

Убеждаюсь, что всё работает верно.

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и вставляю в него текст программы из листинга 7.3.(рис. [4.8])

```

открыть  lab/-2.asm
~/work/arch-pc/lab07

#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)
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

```

Рис. 4.8: Ввод текста из листинга 7.3

Создаю исполняемый файл и запускаю его.(рис. [4.9])

```
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 3
Наибольшее число: 50
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 60
Наибольшее число: 60
msnamruev@dk8n60 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 50
Наибольшее число: 50
msnamruev@dk8n60 ~/work/arch-pc/lab07 $
```

Рис. 4.9: Запуск программы

4.1 Изучение структуры файла листинга

Создаю файл листинга для программы из файла lab7-2.asm.(рис. [4.10])

```
msnamruev@dk6n66 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 4.10: Создание файла листинга

Далее открываю файл листинга lab7-2.lst с помощью текстового редактора gedit.(рис. [4.11])

report.md		lab7-2.lst
1	1	%include 'in_out.asm'
2	1	<1> ;----- slen -----
3	2	<1> ; Функция вычисления длины сообщения
4	3	<1> slen:
5	4 00000000 53	<1> push ebx
6	5 00000001 89C3	<1> mov ebx, eax
7	6	<1>
8	7	<1> nextchar:
9	8 00000003 803800	<1> cmp byte [eax], 0
10	9 00000006 7403	<1> jz finished
11	10 00000008 40	<1> inc eax
12	11 00000009 EBF8	<1> jmp nextchar
13	12	<1>
14	13	<1> finished:
15	14 0000000B 29D8	<1> sub eax, ebx
16	15 0000000D 5B	<1> pop ebx
17	16 0000000E C3	<1> ret
18	17	<1>
19	18	<1>
20	19	<1> ;----- sprint
21	20	<1> ; Функция печати сообщения
22	21	<1> ; входные данные: mov eax, <message>
23	22	<1> sprint:
24	23 0000000F 52	<1> push edx
25	24 00000010 51	<1> push ecx
26	25 00000011 53	<1> push ebx
27	26 00000012 50	<1> push eax
28	27 00000013 E8E8FFFFFF	<1> call slen
29	28	<1>
30	29 00000018 89C2	<1> mov edx, eax
31	30 0000001A 58	<1> pop eax
32	31	<1>
33	32 0000001B 89C1	<1> mov ecx, eax
34	33 0000001D BB01000000	<1> mov ebx, 1
35	34 00000022 B804000000	<1> mov eax, 4
36	35 00000027 CD80	<1> int 80h

Рис. 4.11: Открывание файла листинга

Опишу строчку номер 16:

Здесь “15”-это номер строчки в коде программы “0000000D”- это адрес “5B”- это машинный код “ret” - исходный кол программы

Опишу строчку номер 36:

Здесь “35”-это номер строчки в коде программы “00000027”- это адрес “CD80”- это машинный код “int” - исходный кол программы

Опишу строчку номер 24:

Здесь “23”-это номер строчки в коде программы “0000000F”- это адрес “52”- это машинный код “push” - исходный кол программы

Теперь открываю файл с программой lab7-2.asm и удаляю один операнд в случайном месте.(рис. [4.12])

```

jg спешк_в ; если А/С , то переход на метку спешк
mov ecx ; иначе 'ecx = C'

```

Рис. 4.12: Удаление операнда

Пытаюсь создать файл листинга, но он не создается из-за ошибки.(рис. [4.13])

```

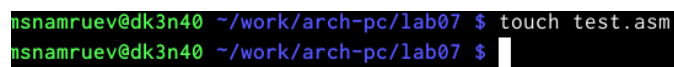
msnamruev@dk6n66 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:30: error: invalid combination of opcode and operands

```

Рис. 4.13: Попытка создание файла

5 Задание для самостоятельной работы

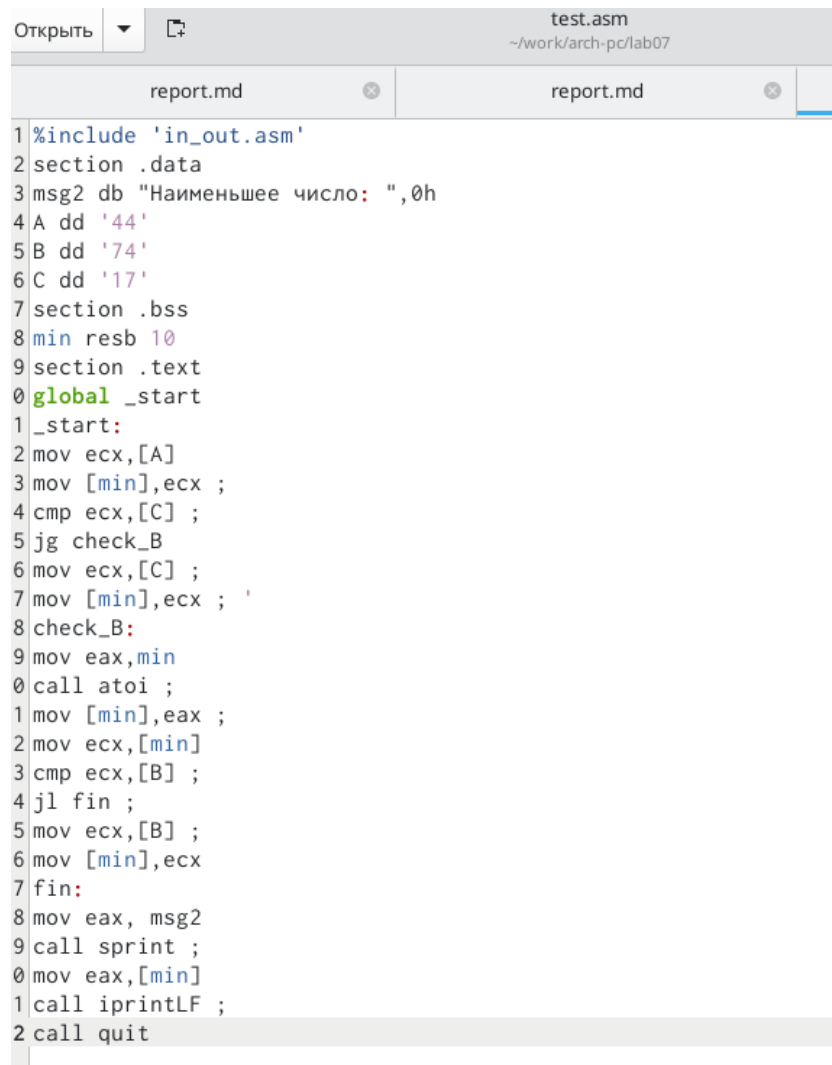
1. Создаю файл test.asm.(рис. [5.1])



```
nsnamruev@dk3n40 ~/work/arch-pc/lab07 $ touch test.asm
nsnamruev@dk3n40 ~/work/arch-pc/lab07 $
```

Рис. 5.1: Создание файла для самостоятельной работы

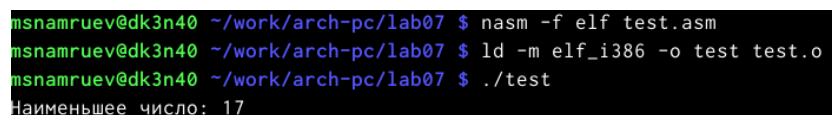
Далее открываю его и пишу программу для нахождения наименьшей из переменных a,b,c в соответствии с моим вариантом (вариант 16).(рис. [5.2])



```
1 %include 'in_out.asm'
2 section .data
3 msg2 db "Наименьшее число: ",0h
4 A dd '44'
5 B dd '74'
6 C dd '17'
7 section .bss
8 min resb 10
9 section .text
0 global _start
1 _start:
2 mov ecx,[A]
3 mov [min],ecx ;
4 cmp ecx,[C] ;
5 jg check_B
6 mov ecx,[C] ;
7 mov [min],ecx ;
8 check_B:
9 mov eax,min
0 call atoi ;
1 mov [min],eax ;
2 mov ecx,[min]
3 cmp ecx,[B] ;
4 jl fin ;
5 mov ecx,[B] ;
6 mov [min],ecx
7 fin:
8 mov eax, msg2
9 call sprint ;
0 mov eax,[min]
1 call iprintLF ;
2 call quit
```

Рис. 5.2: Написание программы

Создаю исполняемый файл и запускаю его.(рис. [5.3])



```
msnamruev@dk3n40 ~/work/arch-pc/lab07 $ nasm -f elf test.asm
msnamruev@dk3n40 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o test test.o
msnamruev@dk3n40 ~/work/arch-pc/lab07 $ ./test
Наименьшее число: 17
```

Рис. 5.3: Проверка работы программы

Программа работает верно

Код программы: %include 'in_out.asm'

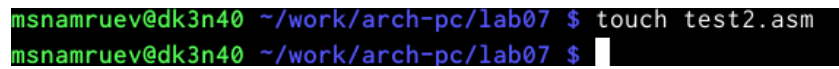
```

section .data
msg2 db "Наименьшее число:",0h
A dd '44'
B dd '74'
C dd '17'
section .bss
min resb 10
section .text
global _start
_start:
mov ecx,[A]
mov [min],ecx ;
cmp ecx,[C] ;
jg check_B
mov ecx,[C] ;
mov [min],ecx ;
check_B:
mov eax,min
call atoi ;
mov [min],eax ;
mov ecx,[min]
cmp ecx,[B] ;
jl fin ;
mov ecx,[B] ;
mov [min],ecx
fin:
mov eax, msg2
call sprint ;
mov eax,[min]

```

```
call iprintLF ;  
call quit
```

2. Создаю файл test2.asm.(рис. [5.4])



```
msnamruev@dk3n40 ~/work/arch-pc/lab07 $ touch test2.asm  
msnamruev@dk3n40 ~/work/arch-pc/lab07 $
```

Рис. 5.4: Создание файла

Начинаю написание программы, которая для введенных с клавиатуры значение x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений.(рис. [5.5])

```

#include 'in_out.asm'
section .data
msg1 db 'Введите X: ',0h
msg2 db 'Введите A: ',0h
msg3 db "Результат: ",0h
section .bss
x resb 10
a resb 10
f resb 10
section .text
global _start
_start:

mov eax,msg1
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,msg2
call sprint
mov ecx,a
mov edx,10
call sread
mov eax,x
call atoi
mov [x],eax
mov eax,a
call atoi
mov [a],eax
mov edi,[x]
mov ecx,edi
add ecx,4
mov [f],ecx
cmp edi,4
jl fin ;
mov ebx,[a]
mov edi,[x]

```

Рис. 5.5: Написание программы

Создаю исполняемый файл и запускаю его.(рис. [5.6])

```

msnamruev@dk3n40 ~/work/arch-pc/lab07 $ nasm -f elf test2.asm
msnamruev@dk3n40 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o test2 test2.o
msnamruev@dk3n40 ~/work/arch-pc/lab07 $ ./test2
Введите X: 1
Введите A: 1
Результат: 5
msnamruev@dk3n40 ~/work/arch-pc/lab07 $ ./test2
Введите X: 7
Введите A: 1
Результат: 7
msnamruev@dk3n40 ~/work/arch-pc/lab07 $

```

Рис. 5.6: Проверка программы

Программа работает верно

Код программы:

```
%include 'in_out.asm'
section .data
msg1 db 'Введите X:',0h
msg2 db 'Введите A:',0h
msg3 db "Результат:",0h
section .bss
x resb 10
a resb 10
f resb 10
section .text
global _start
_start:
    mov eax,msg1
    call sprint
    mov ecx,x
    mov edx,10
    call sread
    mov eax,msg2
    call sprint
    mov ecx,a
    mov edx,10
    call sread
    mov eax,x
    call atoi
    mov [x],eax
    mov eax,a
    call atoi
```

```
mov [a],eax
mov edi,[x]
mov ecx,edi
add ecx,4
mov [f],ecx
cmp edi,4
jl fin ;
mov ebx,[a]
mov edi,[x]
mul edi ;
mov [f],eax ;
fin:
mov eax, msg3
call sprint
mov eax,[f]
call iprintLF ;
call quit
```

6 Выводы

После выполнения данной лабораторной работы я изучил команды условного и безусловного переходов, приобрел навыки написания программ с использованием переходов и познакомился с назначением и структурой файла листинга.

Список литературы