

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Логинов Георгий Евгеньевич

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 файла листинга	13
4.3	Выполнение заданий для самостоятельной работы (вар. 20) . . .	15
5	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	Создание каталога и файла в ней	8
4.2	Редактирование файла	9
4.3	Исполнение программы	9
4.4	Редактирование файла	10
4.5	Исполнение программы	10
4.6	Редактирование файла	11
4.7	Исполнение программы	11
4.8	Создание файла	11
4.9	Редактирование файла	12
4.10	Исполнение программы для разных значений В	12
4.11	Название рисунка	13
4.12	Редактирование файла	14
4.13	Открытие файла листинга	14
4.14	Создание и редактирование файла	15
4.15	Исполнение программы	17
4.16	Создание и редактирование файла	18
4.17	Исполнение 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 файла листинга
3. Задание для самостоятельной работы

3 Теоретическое введение

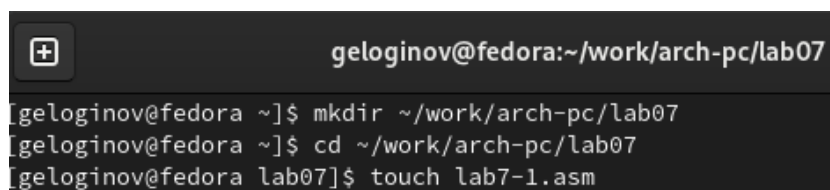
Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- 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 программ для лабораторной работе №7, перехожу в него и создаю файл lab7-1.asm (рис. 4.1).

A terminal window with a dark background. The title bar shows a plus icon and the text 'geloginov@fedora:~/work/arch-pc/lab07'. The terminal contains three lines of text: the first line shows the user at the prompt creating the directory with 'mkdir ~/work/arch-pc/lab07'; the second line shows the user changing to that directory with 'cd ~/work/arch-pc/lab07'; the third line shows the user creating an empty file with 'touch lab7-1.asm'.

```
geloginov@fedora ~]$ mkdir ~/work/arch-pc/lab07
geloginov@fedora ~]$ cd ~/work/arch-pc/lab07
geloginov@fedora lab07]$ touch lab7-1.asm
```

Рис. 4.1: Создание каталога и файла в ней

Ввожу в файл lab7-1.asm текст программы с использованием функции jmp (рис. 4.2).

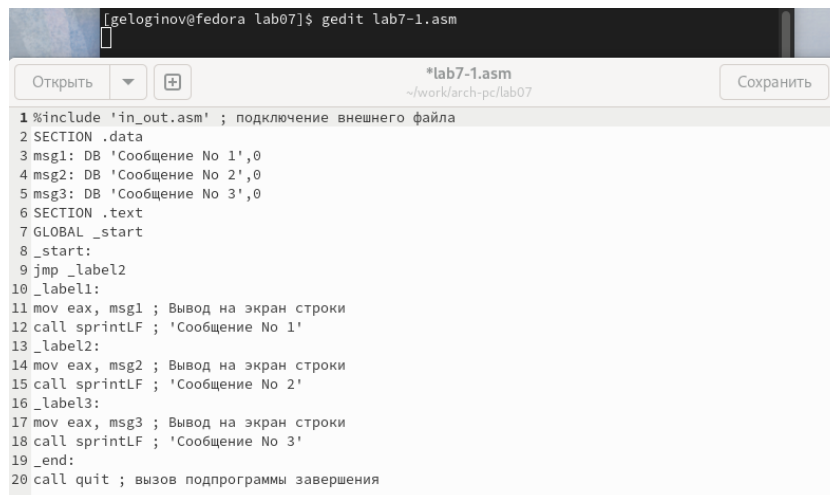


Рис. 4.2: Редактирование файла

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

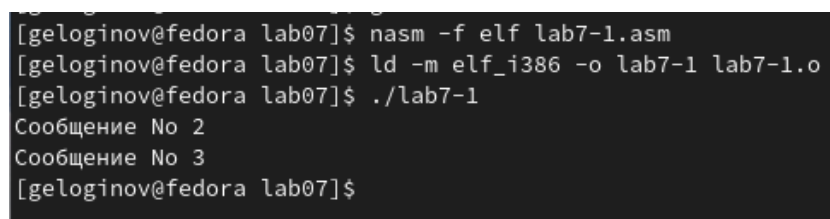


Рис. 4.3: Исполнение программы

Изменяю программу таким образом, чтобы она выводила сначала 'Сообщение No 2', потом 'Сообщение No 1' и завершала работу (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
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение No 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение No 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение No 3'
21 _end:
22 call quit ; вызов подпрограммы завершения

```

Рис. 4.4: Редактирование файла

Создаю исполняемый файл и проверяю его работу (рис. 4.5).

```

[geloginov@fedora lab07]$ nasm -f elf lab7-1.asm
[geloginov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[geloginov@fedora lab07]$ ./lab7-1
Сообщение No 2
Сообщение No 1
[geloginov@fedora lab07]$

```

Рис. 4.5: Исполнение программы

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение No 3’, потом ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу (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
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение No 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение No 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение No 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения

```

Рис. 4.6: Редактирование файла

Создаю исполняемый файл и проверяю корректность работы программы (рис. 4.7). Программа отработала корректно.

```

[geloginov@fedora lab07]$ nasm -f elf lab7-1.asm
[geloginov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[geloginov@fedora lab07]$ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
[geloginov@fedora lab07]$

```

Рис. 4.7: Исполнение программы

Создаю файл lab7-2.asm (рис. 4.8).

```

[geloginov@fedora lab07]$ touch lab7-2.asm
[geloginov@fedora lab07]$

```

Рис. 4.8: Создание файла

Ввожу в созданный файл текст программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С (рис. 4.9).

```

Открыть  + ~/work/arch-pc/lab07
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprint ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call iprintLF ; Вывод 'max(A,B,C)'
49 call quit ; Выход

```

Рис. 4.9: Редактирование файла

Создаю исполняемый файл и проверяю его работу для разных значений B (рис. 4.10). Программа работала корректно.

```

[geloginov@fedora lab07]$ nasm -f elf lab7-2.asm
[geloginov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[geloginov@fedora lab07]$ ./lab7-2
Введите B: 30
Наибольшее число: 50
[geloginov@fedora lab07]$ ./lab7-2
Введите B: 60
Наибольшее число: 60
[geloginov@fedora lab07]$ ./lab7-2
Введите B: 10
Наибольшее число: 50
[geloginov@fedora lab07]$

```

Рис. 4.10: Исполнение программы для разных значений B

4.2 Изучение структуры файла листинга

Создание файла листинга и его просмотр в текстовом редакторе gedit (рис. 4.11).

```
[geloginov@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[geloginov@fedora lab07]$ gedit 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 B801000000 <1> mov ebx, 1
35 34 00000022 B804000000 <1> mov eax, 4
36 35 00000027 CD80 <1> int 80h
37 36 <1>
38 37 00000029 5B <1> pop ebx
39 38 0000002A 59 <1> pop ecx
```

Рис. 4.11: Название рисунка

1. В строке 5 содержится собственно номер строки[5], адрес[00000001], машинный код[89C3] и содержимое строки кода[mov ebx, eax].
2. В строке 11 содержится собственно номер строки[11], адрес[00000009], машинный код[EBF8] и содержимое строки кода[jmp nextchar].
3. В строке 14 содержится собственно номер строки[14], адрес[0000000B], машинный код[29D8] и содержимое строки кода[sub eax, ebx].

Открываю файл lab7-2.asm и удаляю в инструкции mov вторгй операнд (рис. 4.12).

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,

```

Рис. 4.12: Редактирование файла

Открытие файла листинга после трансляции (рис. 4.13). Если в коде появляется ошибка, то её описание появится в файле листинга.

```

166 165 <1> ; Функция завершения программы
167 166 <1> quit:
168 167 000000DB B800000000 <1> mov ebx, 0
169 168 000000E0 B801000000 <1> mov eax, 1
170 169 000000E5 CD80 <1> int 80h
171 170 000000E7 C3 <1> ret
172 2 section .data
173 3 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
174 3 00000009 B8D182D0B520423A20-
175 3 00000012 00
176 4 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
177 4 0000001C BED0BBD18CD188D0B5-
178 4 00000025 D0B520D187D0B8D181-
179 4 0000002E D0BBD0BE3A2000
180 5 00000035 32300000 A dd '20'
181 6 00000039 35300000 C dd '50'
182 7 section .bss
183 8 00000000 <res Ah> max resb 10
184 9 0000000A <res Ah> B resb 10
185 10 section .text
186 11 global _start
187 12 _start:
188 13 ; ----- Вывод сообщения 'Введите B: '
189 14 mov eax
190 14 ***** error: invalid combination of opcode and operands

```

Рис. 4.13: Открытие файла листинга

4.3 Выполнение заданий для самостоятельной работы

(вар. 20)

Создаю файл lab7-3.asm, пишу в нём программу для нахождения наименьшей из трёх целочисленных переменных a, b и c (рис. 4.14).

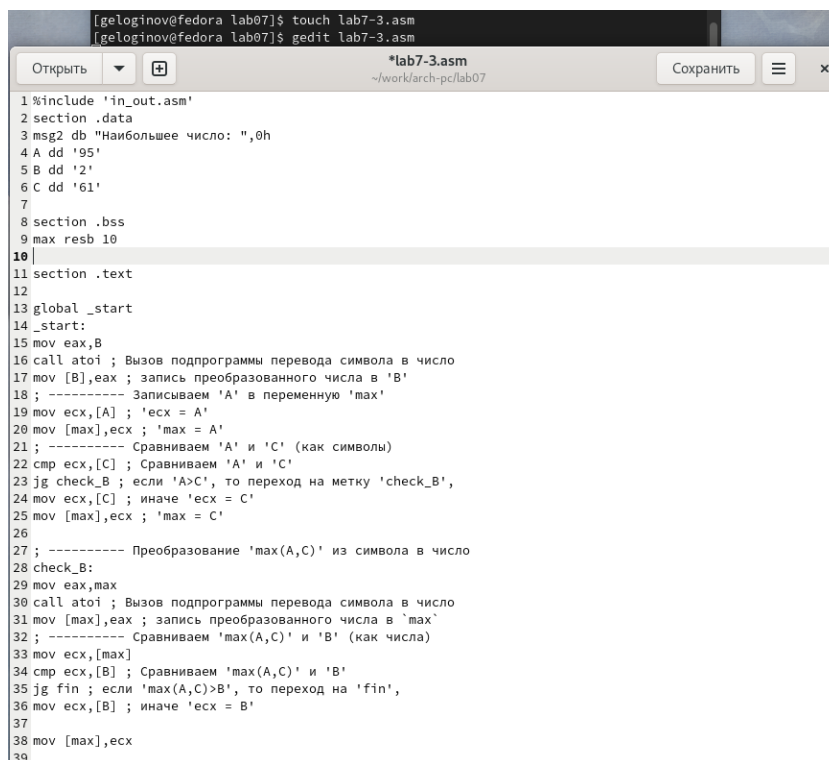


Рис. 4.14: Создание и редактирование файла

Текст программы в файле lab7-3.asm:

```
%include 'in_out.asm'

section .data
msg2 db "Наибольшее число: ",0h
A dd '95'
B dd '2'
C dd '61'
```

```

section .bss
max resb 10

section .text

global _start
_start:
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

```

Создаю исполняемый файл и проверяю его работу (рис. 4.15). Программа отработала корректно.

```

[geloginov@fedora lab07]$ nasm -f elf lab7-3.asm
[geloginov@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[geloginov@fedora lab07]$ ./lab7-3
Наибольшее число: 95
[geloginov@fedora lab07]$

```

Рис. 4.15: Исполнение программы

Создаю файл lab7-4.asm, пишу в нём программу, которая для введённых с клавиатуры значений x и a вычисляет значение функции $f(x)$, которая равна $x-a$ при $x \geq a$, и 5, когда $x < a$ и выводит результат вычислений. (рис. 4.16).

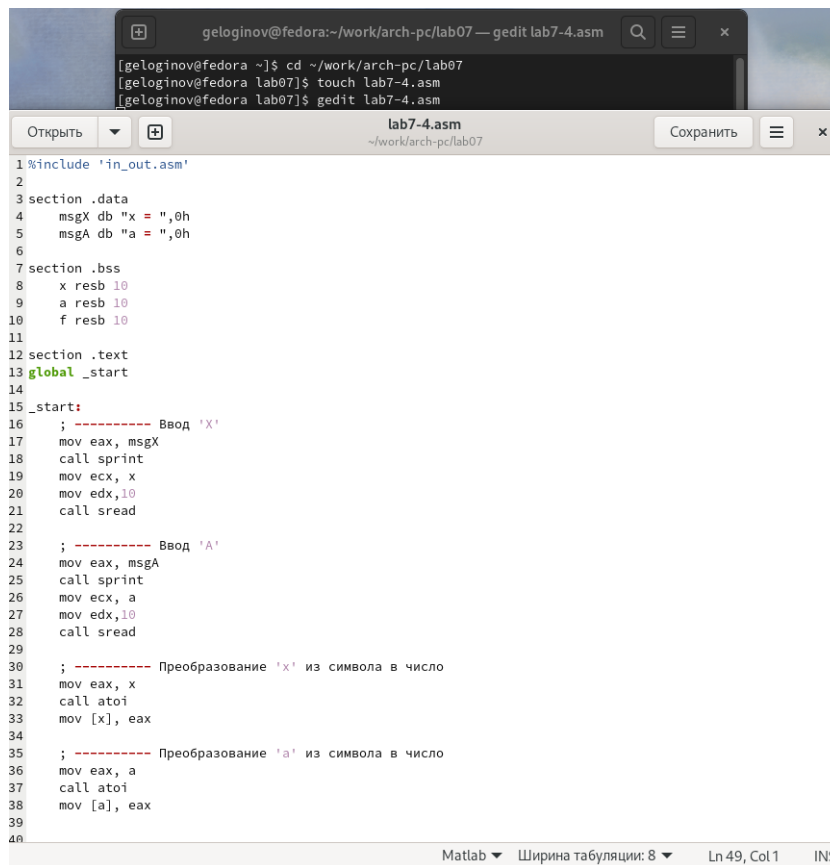


Рис. 4.16: Создание и редактирование файла

Текст программы в файле lab7-4.asm:

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

```
section .data
```

```
msgX db "x = ",0h
```

```
msgA db "a = ",0h
```

```
section .bss
```

```
x resb 10
```

```
a resb 10
```

```
f resb 10
```

```

section .text
global _start

_start:
    ; ----- Ввод 'X'
    mov eax, msgX
    call sprint
    mov ecx, x
    mov edx, 10
    call sread

    ; ----- Ввод 'A'
    mov eax, msgA
    call sprint
    mov ecx, a
    mov edx, 10
    call sread

    ; ----- Преобразование 'x' из символа в число
    mov eax, x
    call atoi
    mov [x], eax

    ; ----- Преобразование 'a' из символа в число
    mov eax, a
    call atoi
    mov [a], eax

```

```
mov ecx, [x]
cmp ecx, [a]
```

```
jge func
```

```
mov eax, 5
jmp fin
```

func:

```
mov eax, [x]
mov ecx, [a]
sub eax, ecx
```

fin:

```
call iprintLF
call quit
```

Создаю исполняемый файл и проверяю его работу для пар x и a (1,2) и (2,1) (рис. 4.17). Программа отработала верно.

```
[geloginov@fedora lab07]$ nasm -f elf lab7-4.asm
[geloginov@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[geloginov@fedora lab07]$ ./lab7-4
x = 1
a = 2
5
[geloginov@fedora lab07]$ ./lab7-4
x = 2
a = 1
1
[geloginov@fedora lab07]$
```

Рис. 4.17: Исполнение программы

5 Выводы

В ходе выполнения лабораторной работы я освоил принципы условного и безусловного перехода в NASM.

Список литературы

1. Лабораторная работа №6