

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Мирзоян Ян Игоревич

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	12
5	Выводы	14
	Список литературы	15

Список иллюстраций

4.1	Создаю каталог для программ для лабораторной работе №7, перехожу в него и создаю файл lab7-1.asm	8
4.2	Ввожу в файл lab7-1.asm текст программы с использованием функции jmp	8
4.3	Создаю исполняемый файл и запускаю его	9
4.4	Изменяю программу таким образом, чтобы она выводила сначала 'Сообщение No 2', потом 'Сообщение No 1' и завершала работу . .	9
4.5	Создаю исполняемый файл и проверяю его работу	9
4.6	Изменяю программу таким образом, чтобы она выводила сначала 'Сообщение No 3', потом 'Сообщение No 2', потом 'Сообщение No 1' и завершала работу	10
4.7	Создаю исполняемый файл и проверяю корректность работы программы	10
4.8	Создаю файл lab7-2.asm	10
4.9	Ввожу в созданный файл текст программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C	11
4.10	Создаю исполняемый файл и проверяю его работу для разных значений B	11
4.11	Создание файла листинга и его просмотр в текстовом редакторе gedit	12
4.12	Открываю файл lab7-2.asm и удаляю в инструкции mov второй операнд	12
4.13	Открытие файла листинга после трансляции, если в коде есть ошибка, то ее описание появится в файле листинга	13

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 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.

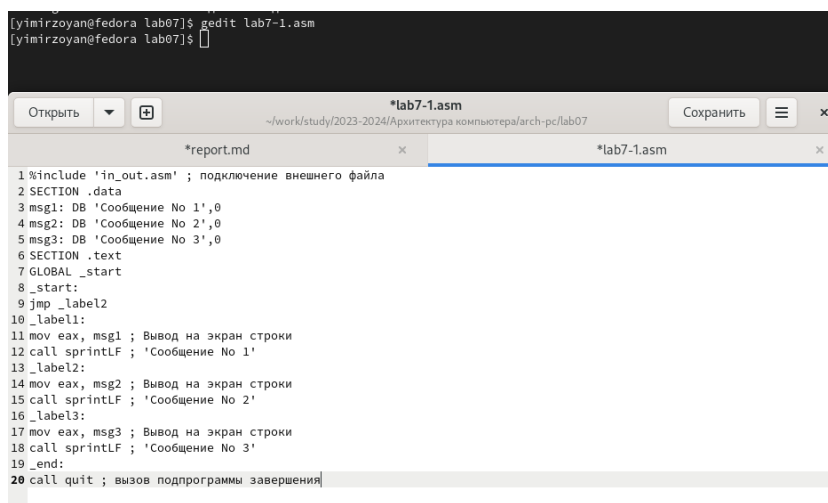
4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

```
[yimirzoyan@fedora arch-pc]$ mkdir lab07  
[yimirzoyan@fedora arch-pc]$ cd lab07  
[yimirzoyan@fedora lab07]$ touch lab7-1.asm
```

Рис. 4.1: Создаю каталог для программ для лабораторной работе №7, перехожу в него и создаю файл lab7-1.asm

```
[yimirzoyan@fedora lab07]$ gedit lab7-1.asm  
[yimirzoyan@fedora 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 sprintfLF ; 'Сообщение No 1'  
13 _label2:  
14 mov eax, msg2 ; Вывод на экран строки  
15 call sprintfLF ; 'Сообщение No 2'  
16 _label3:  
17 mov eax, msg3 ; Вывод на экран строки  
18 call sprintfLF ; 'Сообщение No 3'  
19 _end:  
20 call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Ввожу в файл lab7-1.asm текст программы с использованием функции jmp


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

Рис. 4.3: Создаю исполняемый файл и запускаю его



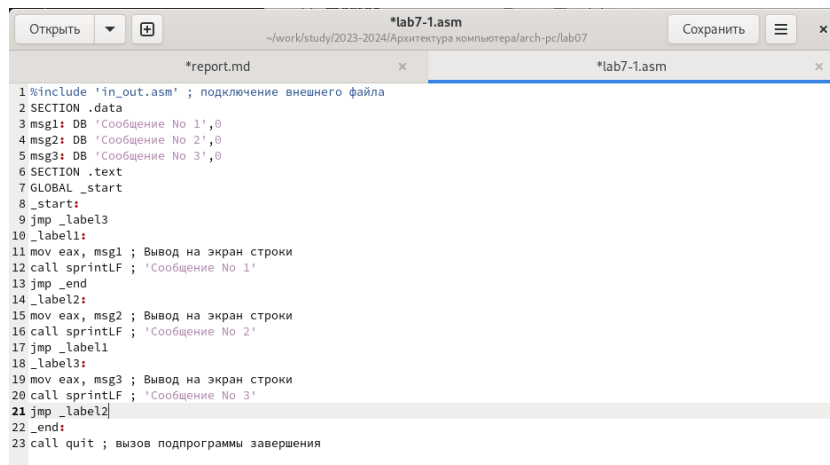
```
lab7-1.asm
~/.work/study/2023-2024/Архитектура компьютера/arch-pc/lab07
Сохранить

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: Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу

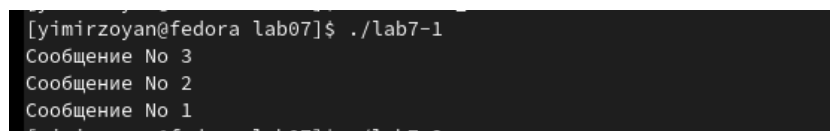
```
[yimirzoyan@fedora lab07]$ nasm -f elf lab7-1.asm
[yimirzoyan@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[yimirzoyan@fedora lab07]$ ./lab7-1
Сообщение No 2
Сообщение No 1
[yimirzoyan@fedora lab07]$
```

Рис. 4.5: Создаю исполняемый файл и проверяю его работу



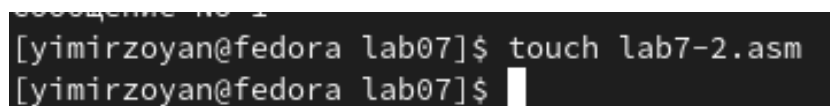
```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение No 1',0
4 msg2: DB 'Сообщение No 2',0
5 msg3: DB 'Сообщение No 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call printf ; 'Сообщение No 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call printf ; 'Сообщение No 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call printf ; 'Сообщение No 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 4.6: Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение No 3’, потом ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу



```
[yimirzoyan@fedora lab07]$ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
```

Рис. 4.7: Создаю исполняемый файл и проверяю корректность работы программы



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

Рис. 4.8: Создаю файл lab7-2.asm

```

*lab7-2.asm
~\work\study\2023-2024\Архитектура компьютера\arch-pc\lab07
Сохранить

*report.md
*lab7-2.asm

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: Ввожу в созданный файл текст программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C

```

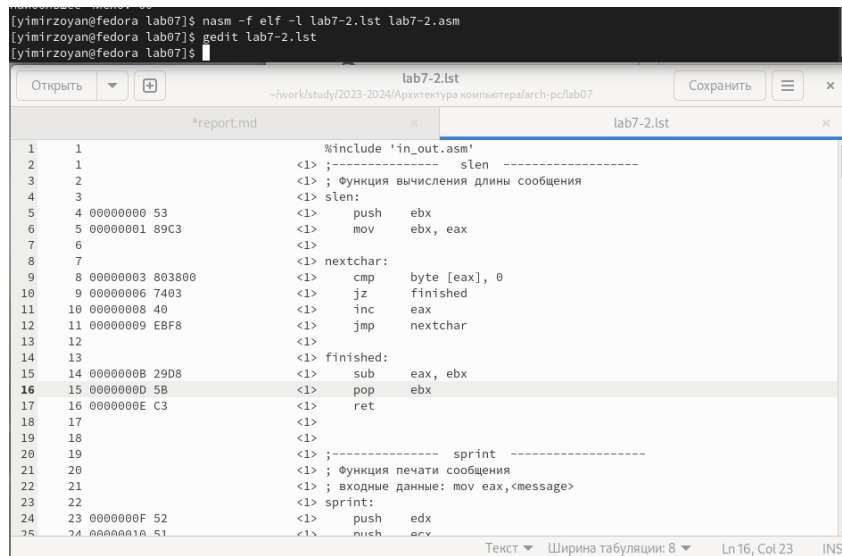
[yimirzoyan@fedora lab07]$ nasm -f elf lab7-2.asm
[yimirzoyan@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[yimirzoyan@fedora lab07]$ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
[yimirzoyan@fedora lab07]$ ./lab7-2
Введите B: 10
Наибольшее число: 50
[yimirzoyan@fedora lab07]$ ./lab7-2
Введите B: 30
Наибольшее число: 50
[yimirzoyan@fedora lab07]$ ./lab7-2
Введите B: 60
Наибольшее число: 60
[yimirzoyan@fedora lab07]$

```

Рис. 4.10: Создаю исполняемый файл и проверяю его работу для разных значений B

4.2 Изучение структуры файла листинга

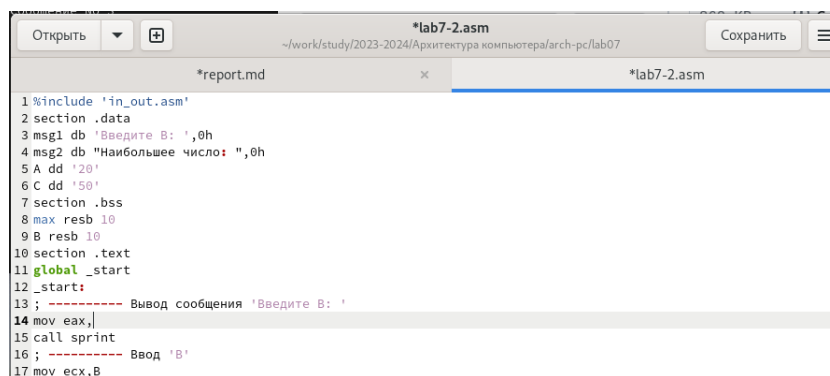
```
[yimirzoyan@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[yimirzoyan@fedora lab07]$ gedit lab7-2.lst
[yimirzoyan@fedora lab07]$
```



```
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
```

Рис. 4.11: Создание файла листинга и его просмотр в текстовом редакторе gedit

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].



```
*lab7-2.asm
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,|
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
```

Рис. 4.12: Открываю файл lab7-2.asm и удаляю в инструкции mov вторгй операнд

```

167 166 <1> quit:
168 167 000000D8 B800000000 <1> mov ebx, 0
169 168 000000E0 B801000000 <1> mov eax, 1
170 169 000000E5 CD00 <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 D09D0B00D0B0D0B1D0- msg2 db "Наибольшее число: ",0h
177 4 0000001C BE0B8B018CD188D0B5-
178 4 00000025 D0B5200187D0B8D181-
179 4 0000002E D0BBD0B0E3A2000
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 000000E8 B8[00000000] mov eax,msg1
190 15 000000ED E81DFFFFFF call sprint
191 16 ; ----- Ввод 'B'
192 17 000000F2 B9[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10
194 19 000000FC E842FFFFFF call sread
195 20 ; ----- Преобразование 'B' из символа в число
196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
198 23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 880D[35000000] mov ecx,[A] ; 'есх = A'
201 26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
204 29 00000122 7F0C jg check_B ; если 'A'>'C', то переход на метку 'check_B',
205 30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'есх = C'
206 31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'

```

Рис. 4.13: Открытие файла листинга после трансляции, если в коде есть ошибка, то ее описание появится в файле листинга

5 Выводы

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

Список литературы

1. [Лабораторная работа №6](https://esystem.rudn.ru/pluginfile.php/2089545/mod_resource/content/1)