

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Гаврилейко Алина Александровна

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
4.3	Задания для самостоятельной работы	15
5	Выводы	22
	Список литературы	23

Список иллюстраций

4.1	Создание каталога и файла для программы	8
4.2	Сохранение программы	8
4.3	Запуск программы	9
4.4	Изменение программы	9
4.5	Запуск измененной программы	9
4.6	Изменение программы	10
4.7	Проверка изменений	10
4.8	Сохранение новой программы	11
4.9	Проверка программы из листинга	12
4.10	Проверка файла листинга	12
4.11	Удаление операнда из программы	14
4.12	Просмотр ошибки в файле листинга	15
4.13	Первая программа самостоятельной работы	16
4.14	Проверка работы первой программы	18
4.15	Вторая программа самостоятельной работы	19
4.16	Проверка работы второй программы	21

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 файлов листинга
3. Самостоятельное написание программ по материалам лабораторной 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 лабораторной работы №7 (рис. -fig. 4.1).

```
alina@gavrileykoalina:~$ mkdir ~/work/arch-pc/lab07
alina@gavrileykoalina:~$ cd ~/work/arch-pc/lab07
alina@gavrileykoalina:~/work/arch-pc/lab07$ touch lab7-1.asm
alina@gavrileykoalina:~/work/arch-pc/lab07$
```

Рис. 4.1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. -fig. 4.2).

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 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 № 1'
13 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintf ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintf ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Сохранение программы

При запуске программы я убедилась в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. -fig. 4.3).


```

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

```

Рис. 4.3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. -fig. 4.4).

```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 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 № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintfLF ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintfLF ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения

```

Рис. 4.4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. -fig. 4.5).

```

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

```

Рис. 4.5: Запуск измененной программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. -fig. 4.6).

```
1 %include 'in_out.asm' ; подключение внешнего файла
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1',0
5 msg2: DB 'Сообщение № 2',0
6 msg3: DB 'Сообщение № 3',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label3
13
14 _label1:
15 mov eax, msg1 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 1'
17 jmp _end
18
19 _label2:
20 mov eax, msg2 ; Вывод на экран строки
21 call sprintf ; 'Сообщение № 2'
22 jmp _label1
23
24 _label3:
25 mov eax, msg3 ; Вывод на экран строки
26 call sprintf ; 'Сообщение № 3'
27 jmp _label2
28 |
29 _end:
30 call quit ; вызов подпрограммы завершения
```

Рис. 4.6: Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. -fig. 4.7).

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

Рис. 4.7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. -fig. 4.8).

```
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,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
```

Рис. 4.8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяя работу программы с разными входными данными (рис. -fig. 4.9).

```

alina@gavrileykoalina:~/work/arch-pc/lab07$ touch lab7-2.asm
alina@gavrileykoalina:~/work/arch-pc/lab07$ mousepad lab7-2.asm
alina@gavrileykoalina:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
alina@gavrileykoalina:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
alina@gavrileykoalina:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
alina@gavrileykoalina:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
alina@gavrileykoalina:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
alina@gavrileykoalina:~/work/arch-pc/lab07$

```

Рис. 4.9: Проверка программы из листинга

4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mousepad (рис. -fig. 4.10).

```

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
37 | 36 | <1>
38 | 37 | 00000029 5B <1> pop ebx
39 | 38 | 0000002A 59 <1> pop ecx
40 | 39 | 0000002B 5A <1> pop edx
41 | 40 | 0000002C C3 <1> ret
42 | 41 | <1>
43 | 42 | <1>
44 | 43 | <1> ;----- sprintf -----
45 | 44 | <1> ; Функция печати сообщения с переводом строки

```

Рис. 4.10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпа-

дать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. -fig. 4.11).

```

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,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|
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

```

Рис. 4.11: Удаление операнда из программы

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. -fig. 4.12).

```

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 mov eax
197 21 ***** error: invalid combination of opcode and operands
198 22 00000101 E896FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
199 23 00000106 A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 0000010B 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
202 26 00000111 890D[00000000] mov [max],ecx ; 'max = A'
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 00000117 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
205 29 0000011D 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
206 30 0000011F 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
207 31 00000125 890D[00000000] mov [max],ecx ; 'max = C'
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 0000012B B8[00000000] mov eax,max
211 35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
212 36 00000135 A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 0000013A 8B0D[00000000] mov ecx,[max]
215 39 00000140 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
216 40 00000146 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
217 41 00000148 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
218 42 0000014E 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
221 45 00000154 B8[13000000] mov eax, msg2
222 46 00000159 E8B1FFFFFF call sprint ; Вывод сообщения 'Наибольшее число: '
223 47 0000015E A1[00000000] mov eax,[max]
224 48 00000163 E81EFFFFFF call iprintLF ; Вывод 'max(A,B,C)'
225 49 00000168 E86EFFFFFF call quit ; Выход
226

```

Рис. 4.12: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Возвращаю операнд к функции в программе и изменяю ее так, чтобы она вы-
водила переменную с наименьшим значением (рис. -fig. 4.13).

```

1
2 %include 'in_out.asm'
3
4 SECTION .data
5 msg1 db 'Введите B: ', 0h
6 msg2 db 'Наименьшее число: ', 0h
7 A dd '41'
8 C dd '35'
9
10 SECTION .bss
11 min resb 10
12 B resb 10
13
14 SECTION .text
15 GLOBAL _start
16 _start:
17
18 mov eax, msg1
19 call sprint
20
21 mov ecx, B
22 mov edx, 10
23 call sread
24
25 mov eax, B
26 call atoi
27 mov [B], eax
28
29 mov ecx, [A]
30 mov [min], ecx
31
32 cmp ecx, [C]
33 jg check_B
34 mov ecx, [C]
35 mov [min], ecx
36
37 check_B:
38 mov eax, min
39 call atoi
40 mov [min], eax
41
42 mov ecx, [min]
43 cmp ecx, [B]
44 jb fin
45 mov ecx, [B]

```

Рис. 4.13: Первая программа самостоятельной работы

Код первой программы:

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

```
SECTION .data
```

```
msg1 db 'Введите B: ', 0h
```

```
msg2 db 'Наименьшее число: ', 0h
```

```
A dd '41'
```

```
C dd '35'
```



```
SECTION .bss
```

```
min resb 10
```

```
B resb 10
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg1
```

```
call sprint
```

```
mov ecx, B
```

```
mov edx, 10
```

```
call sread
```

```
mov eax, B
```

```
call atoi
```

```
mov [B], eax
```

```
mov ecx, [A]
```

```
mov [min], ecx
```

```
cmp ecx, [C]
```

```
jg check_B
```

```
mov ecx, [C]
```

```
mov [min], ecx
```

```
check_B:
```

```

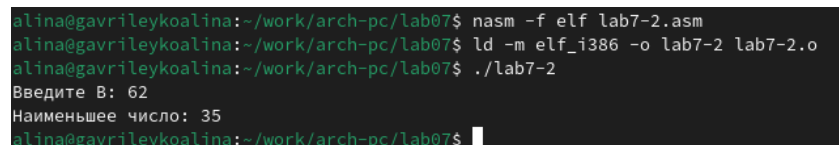
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit

```

Проверяю корректность написания первой программы (рис. -fig. 4.14).



```

alina@gavrileykoalina:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
alina@gavrileykoalina:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
alina@gavrileykoalina:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 62
Наименьшее число: 35
alina@gavrileykoalina:~/work/arch-pc/lab07$

```

Рис. 4.14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных а и х (рис. -fig. 4.15).

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg_x: DB 'Введите значение переменной x: ', 0
4 msg_a: DB 'Введите значение переменной a: ', 0
5 res: DB 'Результат: ', 0
6 SECTION .bss
7 x: RESB 80
8 a: RESB 80
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg_x
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax, x
18 call atoi
19 mov edi, eax
20
21 mov eax, msg_a
22 call sprint
23 mov ecx, a
24 mov edx, 80
25 call sread
26 mov eax, a
27 call atoi
28 mov esi, eax
29
30 cmp edi, esi
31 jle add_values
32 mov eax, esi
33 jmp print_result
34
35 add_values:
36 mov eax, edi
37 add eax, esi
38
39 print_result:
40 mov edi, eax
41 mov eax, res
42 call sprint
43 mov eax, edi
44 call iprintLF
45 call quit

```

Рис. 4.15: Вторая программа самостоятельной работы

Код второй программы:

```

#include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0

SECTION .bss
x: RESB 80
a: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

```

```

cmp edi, esi
jle add_values
mov eax, esi
jmp print_result

```

```

add_values:
mov eax, edi
add eax, esi

```

```

print_result:
mov edi, eax
mov eax, res
call sprint
mov eax, edi
call iprintLF
call quit

```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. -fig. 4.16).

```

alina@gavrileykoalina:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
alina@gavrileykoalina:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
alina@gavrileykoalina:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 3
Введите значение переменной a: 0
Результат: 0
alina@gavrileykoalina:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 1
Введите значение переменной a: 2
Результат: 3
alina@gavrileykoalina:~/work/arch-pc/lab07$

```

Рис. 4.16: Проверка работы второй программы

5 Выводы

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файлов листинга.

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.