

Лабораторная работа №8

Щетинин Даниил Николаевич

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
4	Задание для самостоятельной работы	8
5	Выводы	16

Список иллюстраций

4.1	работа lab8-1.asm	9
4.2	работа нового lab8-1	9
4.3	код lab8-2	9
4.4	проверка lab8-2	10
4.5	Наибольшая переменная с разными В	10
4.6	Файл листинга	11
4.7	Ошибка в файле листинга	12
4.8	код lab8-3.asm	12
4.9	код lab8-3.asm	13
4.10	проверка lab8-3.asm	13
4.11	код lab8-4.asm	14
4.12	код lab8-4.asm	15
4.13	проверка lab8-4.asm	15

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

3 Выполнение лабораторной работы

Шаг 1

Создадим каталог для лабораторной работы 8, а также файл lab8-1.asm:

Введём в него текст из Листинга 8.1, для того чтобы понять принцип работы инструкции `jmp`

Создадим исполняемый файл и запустим его:

(рис. 4.1)

Как мы видим, сообщение 1 нету в терминале, хотя оно присутствует в коде файла. Это потому, что использование инструкции `jmp _label2` меняет порядок исполнения инструкций, и позволяет пропустить инструкцию.

Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу, введём текст из Листинга 8.2, и проверим работу файла:

(рис. 4.2)

Шаг 2

Изменим код, чтобы сообщения выводились в порядке убывания, при этом только добавляя инструкцию `jmp`:

(рис. 4.3)

(рис. 4.4)

Шаг 3 Создадим файл lab8-2.asm, и введем в него текст из Листинга 8.3, для того чтобы найти меньшее из 3 чисел, два из которых заданы программой.

Проверим его работу для различных В:

(рис. 4.5)

Шаг 4

Создадим файл листинга для программы из предыдущего файла, для этого используем ключ -l в команде nasm, и откроем его, чтобы ознакомиться с содержанием. Рассмотрим 3 строчки для примера структуры листинга:

(рис. 4.6)

```
38 0000013F 8B0D[00000000]      mov esx,[max]
39 00000145 3B0D[0A000000]      cmp esx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 0000014B 7F0C                jg fin ; если 'max(A,C)>B', то переход на 'fi
```

38, 39, 40 отвечают за номер строки;

00000... - адрес, он отвечает за то, чтобы инструкции выполнялись по порядку;

8B0D, 3B0D, 7F0C - машинный код, инструкция на машинном языке, они отвечают за перемещение переменной max в esx, сравнение esx и B, и переход к 'fin' если esx больше B соответственно;

Справа находится исходный текст нашей программы.

Допустим ошибку в нашем коде, удалим один операнд в операции, требующей два, создадим файл листинга, lab8-2.lst и посмотрим как он изменился:

(рис. 4.7)

Как мы видим, в файле листинга около строчки кода находится предупреждение об ошибке.

4 Задание для самостоятельной работы

Шаг 1

Создадим файл lab8-3.asm для создания программы для нахождения наименьшей из 3 переменных a,b,c, за основу взяв код lab8-2.asm

Как в листинге 8.2, мы поочерёдно записываем с клавиатуры 3 переменные, переводим их в числа (необязательно)

Далее можно просто оставить код Листинга 8.3, заменив `jb` (переход если больше) на `jl` (переход если меньше) (Также См комментарии в коде)

(рис. 4.8)

(рис. 4.9)

Проверяем:

(рис. 4.10)

Создадим файл lab8-4.asm для создания программы нахождения ответа на систему уравнений с использованием 2 переменных a,x за основу снова взяв код lab8-2.asm

Как и указано в комментариях, мы записываем введённые значения a и x в переменные A и X поочерёдно, преобразуем их в числа для работы с операциями сложения и умножения, и сравниваем X и 4.

Если $X < 4$ то мы переходим к `Xmin`, и решаем уравнение $x + 4$, выводим результат, в противном случае умножаем a на x и переходим в конец, где мы выводим сообщение 'Ответ:' на экран

Код (рис. 4.11)

(рис. 4.12)

Проверим

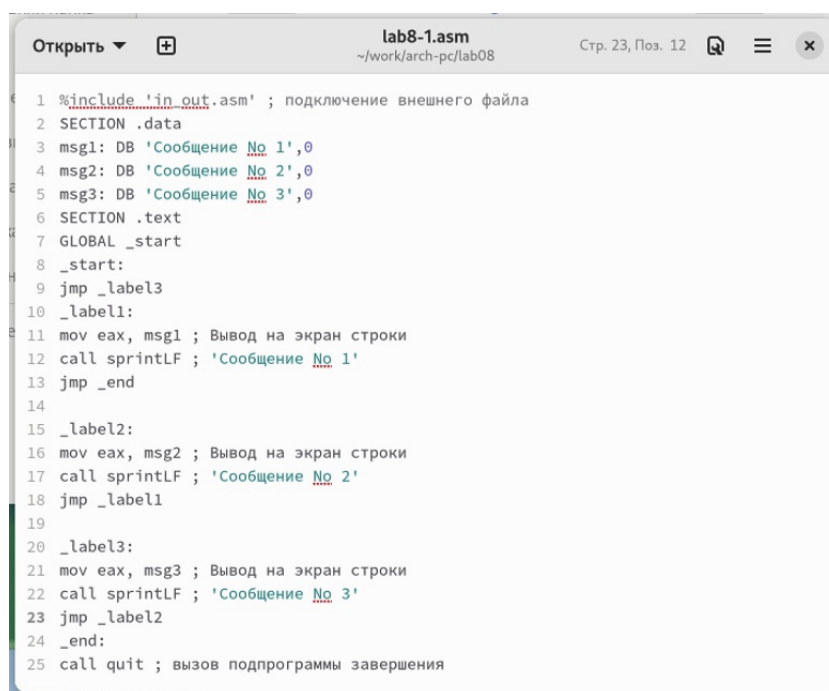
(рис. 4.13)

```
[dnthetinin@fedora lab08]$ nasm -f elf lab8-1.asm
[dnthetinin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[dnthetinin@fedora lab08]$ ./lab8-1
Сообщение No 2
Сообщение No 3
```

Рис. 4.1: работа lab8-1.asm

```
[dnthetinin@fedora lab08]$ nasm -f elf lab8-1.asm
[dnthetinin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[dnthetinin@fedora lab08]$ ./lab8-1
Сообщение No 2
Сообщение No 1
```

Рис. 4.2: работа нового lab8-1



```
lab8-1.asm
~/work/arch-pc/lab08
Стр. 23, Поз. 12

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
15 _label2:
16 mov eax, msg2 ; Вывод на экран строки
17 call sprintf ; 'Сообщение No 2'
18 jmp _label1
19
20 _label3:
21 mov eax, msg3 ; Вывод на экран строки
22 call sprintf ; 'Сообщение No 3'
23 jmp _label2
24 _end:
25 call quit ; вызов подпрограммы завершения
```

Рис. 4.3: код lab8-2

```
[dnthetinin@fedora lab08]$ nasm -f elf lab8-1.asm
[dnthetinin@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[dnthetinin@fedora lab08]$ ./lab8-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
```

Рис. 4.4: проверка lab8-2

```
[dnthetinin@fedora lab08]$ ./lab8-2
Введите B: 10
Наибольшее число: 50
[dnthetinin@fedora lab08]$ ./lab8-2
Введите B: 22
Наибольшее число: 50
[dnthetinin@fedora lab08]$ ./lab8-2
Введите B: 52
Наибольшее число: 52
[dnthetinin@fedora lab08]$ ./lab8-2
Введите B: 56.1
Наибольшее число: 56
[dnthetinin@fedora lab08]$
```

Рис. 4.5: Наибольшая переменная с разными B

Открыть

+

lab8-2.lst

~/work/arch-pc/lab08

Стр. 1, Поз. 1

🔍

☰

✕

lab8-1.asm

lab8-2.asm

lab8-2.lst

✕

188

13

189

14

000000F8 B8[00000000]

190

15

000000ED E81DFFFFFF

191

16

192

17

000000F2 B9[0A000000]

193

18

000000F7 BA0A000000

194

19

000000FC E842FFFFFF

195

20

196

21

00000101 B8[0A000000]

197

22

00000106 E891FFFFFF

198

23

0000010B A3[0A000000]

199

24

200

25

00000110 8B0D[35000000]

201

26

00000116 890D[00000000]

202

27

203

28

0000011C 3B0D[39000000]

204

29

00000122 7F0C

205

30

00000124 8B0D[39000000]

206

31

0000012A 890D[00000000]

207

32

208

33

209

34

00000130 B8[00000000]

210

35

00000135 E862FFFFFF

211

36

0000013A A3[00000000]

212

37

213

38

0000013F 8B0D[00000000]

214

39

00000145 3B0D[0A000000]

215

40

0000014B 7F0C

216

41

0000014D 8B0D[0A000000]

217

42

00000153 890D[00000000]

218

43

219

44

220

45

00000159 B8[13000000]

221

46

0000015E E8ACFFFFFF

222

47

00000163 A1[00000000]

223

48

00000168 E819FFFFFF

224

49

0000016D E869FFFFFF

;

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'

jb

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'

jb

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: Файл листинга

```

212 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
213 38 0000013F 8B0D[00000000] mov ecx,[max]
214 39 00000145 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
215 40 0000014B 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
216 41 0000014D 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
217 42 00000153 890D[00000000] mov [max],ecx
218 43 ; ----- Вывод результата
219 44 fin:
220 45 mov eax, |
221 45 ***** error: invalid combination of opcode and operands
222 46 00000159 E8B1FEFFFF call sprint ; Вывод сообщения 'Наибольшее число: '
223 47 0000015F A1[00000000] mov eax,[max]
224 48 00000163 E81EFFFF call iprintf ; Вывод 'max(A,B,C)'
225 49 00000168 E86EFFFF call quit ; Выход

```

Рис. 4.7: Ошибка в файле листинга

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наименьшее число: ",0h
5 msg3 db 'Введите A: ',0h
6 msg4 db 'Введите C: ',0h
7 section .bss
8 max resb 10
9 A resb 10
10 B resb 10
11 C resb 10
12 section .text
13 global _start
14 _start:
15 ; ----- Вывод сообщения 'Введите A: '
16 mov eax,msg3
17 call sprint
18 ; ----- Ввод 'A'
19 mov ecx,A
20 mov edx,10
21 call sread
22 ; ----- Преобразование 'A' из символа в число
23 mov eax,A
24 call atoi ; Вызов подпрограммы перевода символа в число
25 mov [A],eax ; запись преобразованного числа в 'B'
26 ; ----- Вывод сообщения 'Введите B: '
27 mov eax,msg1
28 call sprint
29 ; ----- Ввод 'B'
30 mov ecx,B
31 mov edx,10
32 call sread
33 ; ----- Преобразование 'B' из символа в число
34 mov eax,B
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [B],eax ; запись преобразованного числа в 'B'
37 ; ----- Вывод сообщения 'Введите C: '
38 mov eax,msg4

```

Рис. 4.8: код lab8-3.asm

```

39 call sprint
40 ; ----- Ввод 'C'
41 mov ecx,C
42 mov edx,10
43 call sread
44 ; ----- Преобразование 'C' из символа в число
45 mov eax,C
46 call atoi ; Вызов подпрограммы перевода символа в число
47 mov [C],eax ; запись преобразованного числа в 'B'
48 ; ----- Записываем 'A' в переменную 'max'
49 mov ecx,[A] ; 'ecx = A'
50 mov [max],ecx ; 'max = A'
51 ; ----- Сравниваем 'A' и 'C' (как символы)
52 cmp ecx,[C] ; Сравниваем 'A' и 'C'
53 jl check_B ; если 'A<C', то переход на метку 'check_B',
54 mov ecx,[C] ; иначе 'ecx = C'
55 mov [max],ecx ; 'max = C'
56 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
57 check_B:
58 mov ecx,[max]
59 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
60 jl fin ; если 'max(A,C)<B', то переход на 'fin',
61 mov ecx,[B] ; иначе 'ecx = B'
62 mov [max],ecx
63 ; ----- Вывод результата
64 fin:
65 mov eax,msg2
66 call sprint ; Вывод сообщения 'Наибольшее число: '
67 mov eax,[max]
68 call iprintLF ; Вывод 'max(A,B,C)'
69 call quit ; Выход

```

Рис. 4.9: код lab8-3.asm

```

[dnthetinin@fedora lab08]$ ./lab8-3
Введите A: 44
Введите B: 74
Введите C: 17
Наименьшее число: 17
[dnthetinin@fedora lab08]$

```

Рис. 4.10: проверка lab8-3.asm

```
report.md lab8-4.asm x
1 ;x + 4 x < 4
2 ;a*x x >= 4
3 %include 'in_out.asm'
4 section .data
5 msg1 db 'Введите x: ',0h
6 msg2 db 'Введите a: ',0h
7 msg3 db 'Ответ: ',0h
8 section .bss
9 max resb 10
10 A resb 10
11 X resb 10
12 section .text
13 global _start
14 _start:
15 ; ----- Вывод сообщения 'Введите A: '
16 mov eax,msg2
17 call sprint
18 ; ----- Ввод 'A'
19 mov ecx,A
20 mov edx,10
21 call sread
22 ; ----- Преобразование 'A' из символа в число
23 mov eax,A
24 call atoi ; Вызов подпрограммы перевода символа в число
25 mov [A],eax ; запись преобразованного числа в 'A'
26 ; ----- Вывод сообщения 'Введите X: '
27 mov eax,msg1
28 call sprint
29 ; ----- Ввод 'X'
30 mov ecx,X
31 mov edx,10
32 call sread
33 ; ----- Преобразование 'x' из символа в число
34 mov eax,X
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [X],eax ; запись преобразованного числа в 'X'
```

Рис. 4.11: код lab8-4.asm

```

37 ; ----- Записываем 'X' в переменную 'max'
38 mov ecx,[X] ; 'ecx = X'
39 mov [max],ecx ; 'max = X'
40 ; ----- Сравниваем 'X' и '4'
41 mov edx, 4
42 cmp ecx,edx ; Сравниваем 'X' и '4'
43 jl Xmin ; если 'X<4', то переход на метку 'Xmin',
44 mov edx,[A]
45 mul edx
46 mov [max],eax
47 jmp fin ; завершение
48 ; ----- FX
49 Xmin:
50 add ecx,4 ; X + 4
51 mov [max],ecx
52 ; ----- Вывод результата
53 fin:
54 mov eax, msg3
55 call sprint ; Вывод сообщения
56 mov eax,[max]
57 call iprintf ; Вывод
58 call quit ; Выход

```

Рис. 4.12: код lab8-4.asm

```

[dnthetinin@fedora lab08]$ ./lab8-4
Введите a: 1
Введите x: 1
Ответ: 5
[dnthetinin@fedora lab08]$ ./lab8-4
Введите a: 1
Введите x: 7
Ответ: 7
[dnthetinin@fedora lab08]$

```

Рис. 4.13: проверка lab8-4.asm

5 Выводы

Я смог написать базовую программу вычисления функции на языке ассемблера
NASM