

Отчёт по лабораторной работе 7

Команды безусловного и условного переходов в Nasm.
Программирование ветвлений

Дурдыев Безирген

Содержание

1	Цель работы	5
2	Теоретическое введение	6
2.1	Команды перехода	6
2.2	Листинг	7
3	Выполнение лабораторной работы	8
3.1	Реализация переходов в NASM	8
3.2	Изучение структуры файлы листинга	15
3.3	Задание для самостоятельной работы	18
4	Выводы	23

Список иллюстраций

3.1	Программа в файле lab7-1.asm	9
3.2	Запуск программы lab7-1.asm	10
3.3	Программа в файле lab7-1.asm	11
3.4	Запуск программы lab7-1.asm	11
3.5	Программа в файле lab7-1.asm	12
3.6	Запуск программы lab7-1.asm	13
3.7	Программа в файле lab7-2.asm	14
3.8	Запуск программы lab7-2.asm	15
3.9	Файл листинга lab7-2	16
3.10	Ошибка трансляции lab7-2	17
3.11	Файл листинга с ошибкой lab7-2	18
3.12	Программа в файле task-1.asm	19
3.13	Запуск программы task-1.asm	20
3.14	Программа в файле task-2.asm	21
3.15	Запуск программы task-2.asm	22

Список таблиц

1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Теоретическое введение

2.1 Команды перехода

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.

Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление.

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания.

2.2 Листинг

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.

Итак, структура листинга:

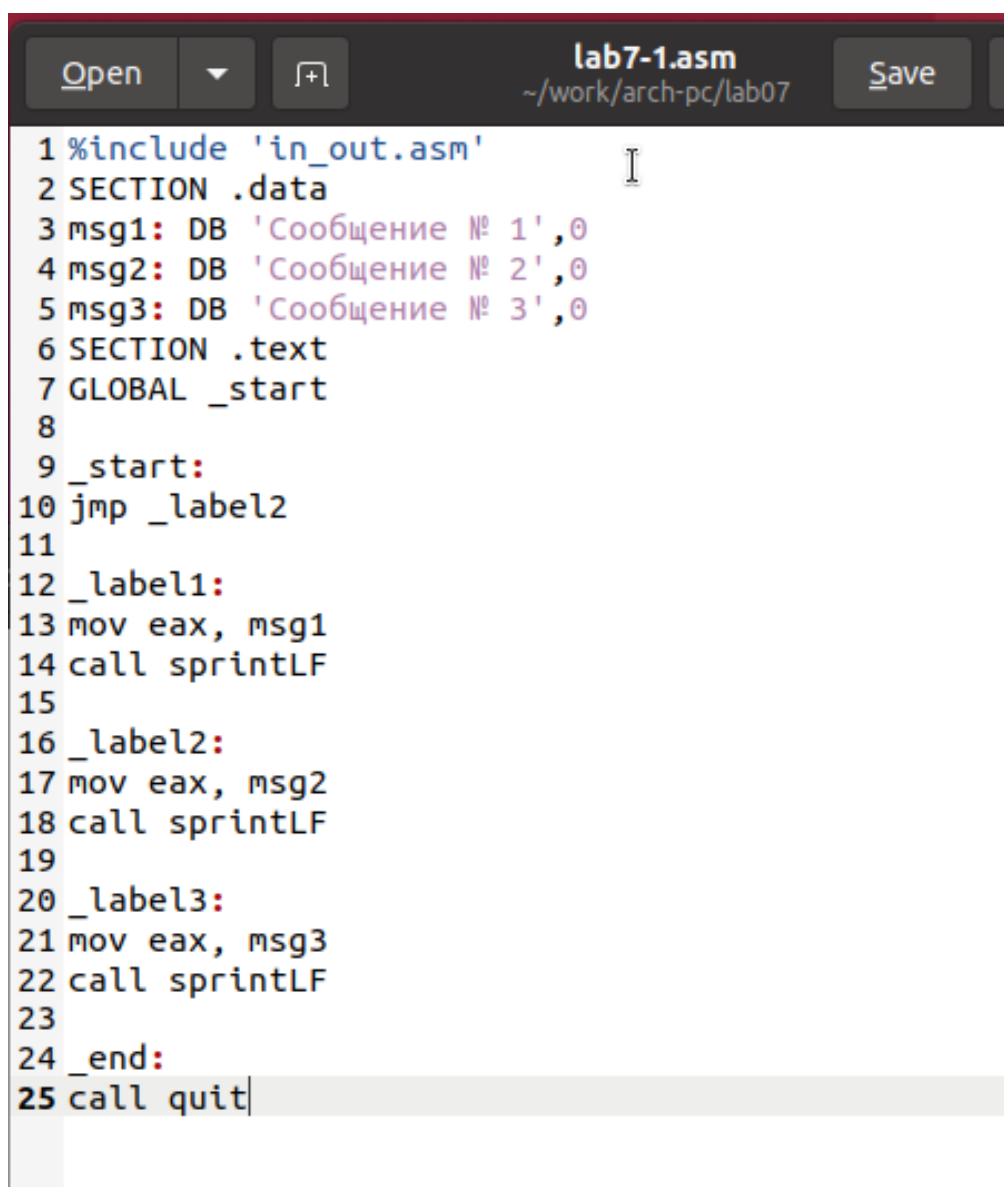
- номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы);
- адрес — это смещение машинного кода от начала текущего сегмента;
- машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция `int 80h` начинается по смещению `00000020` в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция `int 80h` ассемблируется в `CD80` (в шестнадцатеричном представлении); `CD80` — это инструкция на машинном языке, вызывающая прерывание ядра)
- исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается)

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

Создал каталог для программам лабораторной работы № 7 и файл lab7-1.asm

Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл lab7-1.asm текст программы из листинга 7.1.



```
1 %include 'in_out.asm'
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
9 _start:
10 jmp _label2
11
12 _label1:
13 mov eax, msg1
14 call sprintfLF
15
16 _label2:
17 mov eax, msg2
18 call sprintfLF
19
20 _label3:
21 mov eax, msg3
22 call sprintfLF
23
24 _end:
25 call quit
```

Рис. 3.1: Программа в файле lab7-1.asm

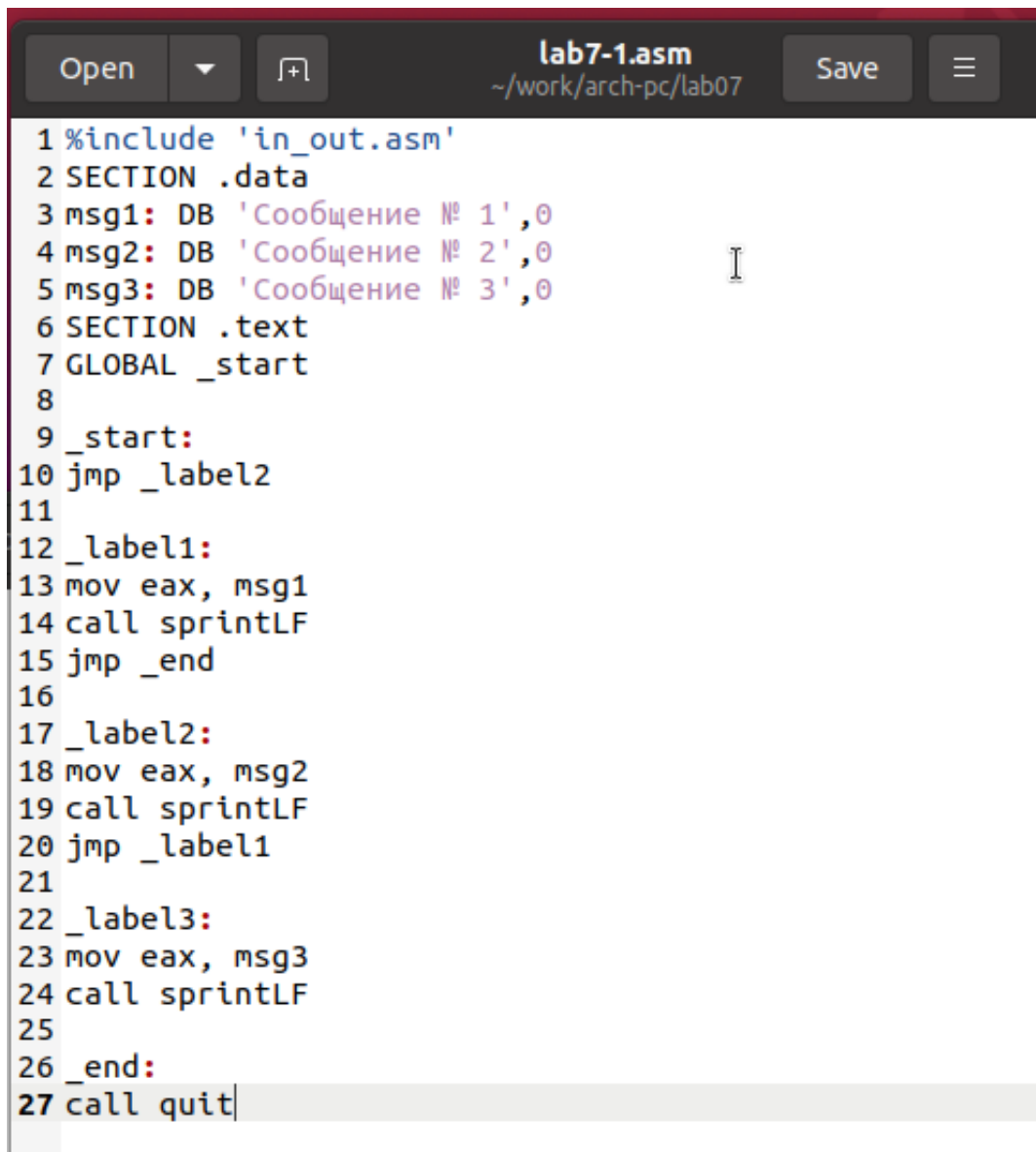
Создал исполняемый файл и запустил его.

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

Рис. 3.2: Запуск программы lab7-1.asm

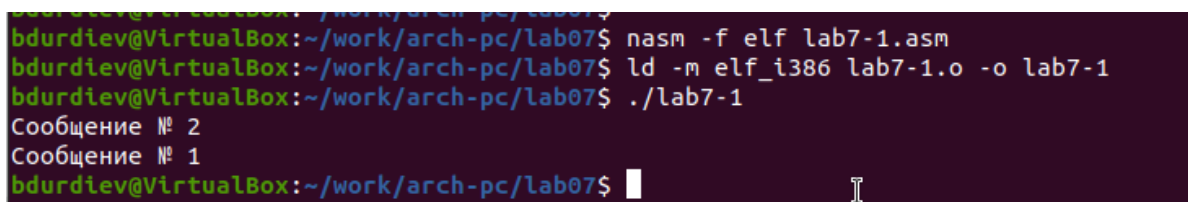
Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`).

Изменил текст программы в соответствии с листингом 7.2.



```
1 %include 'in_out.asm'
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
9 _start:
10 jmp _label2
11
12 _label1:
13 mov eax, msg1
14 call sprintf
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintf
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintf
25
26 _end:
27 call quit
```

Рис. 3.3: Программа в файле lab7-1.asm



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

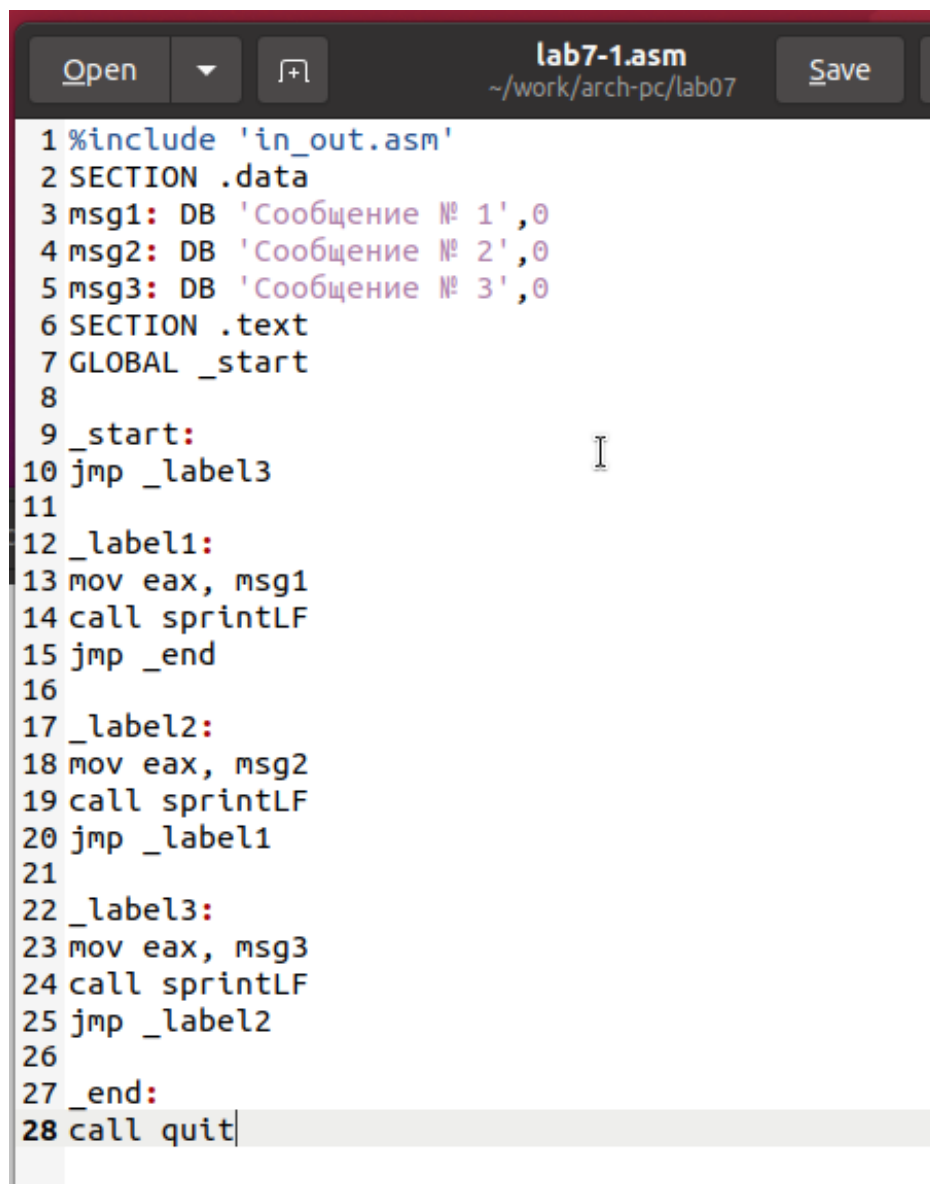
Рис. 3.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
1 %include 'in_out.asm'
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
9 _start:
10 jmp _label3
11
12 _label1:
13 mov eax, msg1
14 call sprintLF
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintLF
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintLF
25 jmp _label2
26
27 _end:
28 call quit
```

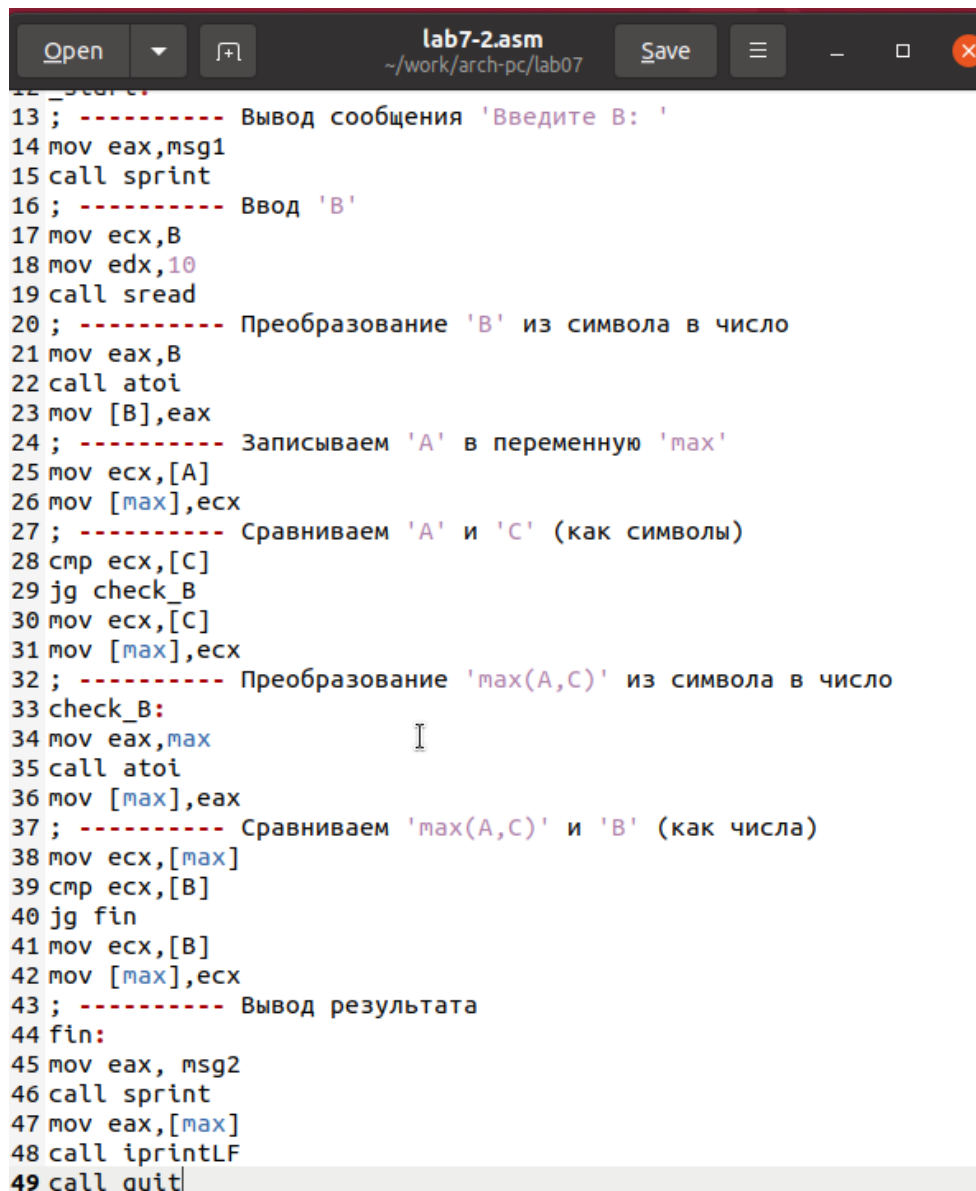
Рис. 3.5: Программа в файле lab7-1.asm

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

Рис. 3.6: Запуск программы lab7-1.asm

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений В.



```
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
23 mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A]
26 mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C]
29 jg check_B
30 mov ecx,[C]
31 mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi
36 mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39 cmp ecx,[B]
40 jg fin
41 mov ecx,[B]
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprint
47 mov eax,[max]
48 call iprintLF
49 call quit
```

Рис. 3.7: Программа в файле lab7-2.asm

```
bduddiev@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
bduddiev@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
bduddiev@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
bduddiev@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 40
Наибольшее число: 50
bduddiev@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 70
Наибольшее число: 70
bduddiev@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 3.8: Запуск программы lab7-2.asm

3.2 Изучение структуры файлы листинга

Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке.

Создал файл листинга для программы из файла lab7-2.asm

```
lab7-2.lst
~/.work/arch-pc/lab07
lab7-2.asm
lab7-2.lst

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
198 23 0000010B A3[0A000000] mov [B],eax
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A]
201 26 00000116 890D[00000000] mov [max],ecx
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C]
204 29 00000122 7F0C jg check_B
205 30 00000124 8B0D[39000000] mov ecx,[C]
206 31 0000012A 890D[00000000] mov [max],ecx
207 32 ; ----- Преобразование 'max(A,C)' из символа в число
208 33 check_B:
209 34 00000130 B8[00000000] mov eax,max
210 35 00000135 E862FFFFFF call atoi
211 36 0000013A A3[00000000] mov [max],eax
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]
215 40 0000014B 7F0C jg fin
216 41 0000014D 8B0D[0A000000] mov ecx,[B]
217 42 00000153 890D[00000000] mov [max],ecx
218 43 ; ----- Вывод результата
219 44 fin:
220 45 00000159 B8[13000000] mov eax, msg2
221 46 0000015E E8ACFFFFFF call sprint
222 47 00000163 A1[00000000] mov eax,[max]
223 48 00000168 E819FFFFFF call iprintLF
224 49 0000016D E869FFFFFF call quit
```

Рис. 3.9: Файл листинга lab7-2

Внимательно ознакомился с его форматом и содержимым. Подробно объясню содержимое трёх строк файла листинга по выбору.

строка 203

- 28 - номер строки в подпрограмме
- 0000011C - адрес
- 3B0D[39000000] - машинный код
- `cmp ecx,[C]` - код программы - сравнивает регистр `ecx` и переменную `C`

строка 204

- 29 - номер строки в подпрограмме
- 00000122 - адрес

- 7F0C - машинный код
- `jb check_B` - код программы - если `>`, то переход к метке `check_B`

строка 205

- 30 - номер строки в подпрограмме
- 00000124 - адрес
- `8B0D[39000000]` - машинный код
- `mov esx,[C]` - код программы - перекладывает в регистр `esx` значение переменной `C`

Открыл файл с программой `lab7-2.asm` и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга.

```

bduirdiev@VirtualBox:~/work/arch-pc/lab07$
bduirdiev@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
bduirdiev@VirtualBox:~/work/arch-pc/lab07$
bduirdiev@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:36: error: invalid combination of opcode and operands
bduirdiev@VirtualBox:~/work/arch-pc/lab07$
bduirdiev@VirtualBox:~/work/arch-pc/lab07$
bduirdiev@VirtualBox:~/work/arch-pc/lab07$

```

Рис. 3.10: Ошибка трансляции `lab7-2`

```

191 16 ; ----- Ввод 'В'
192 17 000000F2 B9[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10
194 19 000000FC E842FFFFFF call spread
195 20 ; ----- Преобразование 'В' из символа в число
196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 E891FFFFFF call atoi
198 23 0000010B A3[0A000000] mov [B],eax
199 24 ; ----- Записываем 'А' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A]
201 26 00000116 890D[00000000] mov [max],ecx
202 27 ; ----- Сравниваем 'А' и 'С' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C]
204 29 00000122 7F0C jg check_B
205 30 00000124 8B0D[39000000] mov ecx,[C]
206 31 0000012A 890D[00000000] mov [max],ecx
207 32 ; ----- Преобразование 'max(A,C)' из символа в число
208 33 check_B:
209 34 00000130 B8[00000000] mov eax,max
210 35 00000135 E862FFFFFF call atoi
211 36 mov [max],
212 36 ***** error: invalid combination of opcode and operands
213 37 ; ----- Сравниваем 'max(A,C)' и 'В' (как числа)
214 38 0000013A 8B0D[00000000] mov ecx,[max]
215 39 00000140 3B0D[0A000000] cmp ecx,[B]
216 40 00000146 7F0C jg fin
217 41 00000148 8B0D[0A000000] mov 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
223 47 0000015E A1[00000000] mov eax,[max]
224 48 00000163 E81EFFFFFF call iprintLF
225 49 00000168 E86EFFFFFF call quit

```

Рис. 3.11: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаваться из-за ошибки. Но получился листинг, где выделено место ошибки.

3.3 Задание для самостоятельной работы

Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу

для варианта 11 - 21,28,34

```
34    mov eax,B
35    call atoi
36    mov [B],eax
37
38    mov eax,msgC
39    call sprint
40    mov ecx,C
41    mov edx,80
42    call sread
43    mov eax,C
44    call atoi
45    mov [C],eax
46 ; _____algorithm_____
47
48    mov ecx,[A] ;ecx = A
49    mov [min],ecx ;min = A
50
51    cmp ecx, [B] ; A&B
52    jl check_C ; if a<b: goto check_C
53    mov ecx, [B]
54    mov [min], ecx ;else min = B
55
56 check_C:
57    cmp ecx, [C]
58    jl finish
59    mov ecx,[C]
60    mov [min],ecx
61
62 finish:
63    mov eax,answer
64    call sprint
65
66    mov eax, [min]
67    call iprintLF
68
69    call quit
70
71
```

Рис. 3.12: Программа в файле task-1.asm

```
bdurdiev@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf task-1.asm
bdurdiev@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 task-1.o -o task-1
bdurdiev@VirtualBox:~/work/arch-pc/lab07$ ./task-1
Input A: 21
Input B: 28
Input C: 34
Smallest: 21
bdurdiev@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 3.13: Запуск программы task-1.asm

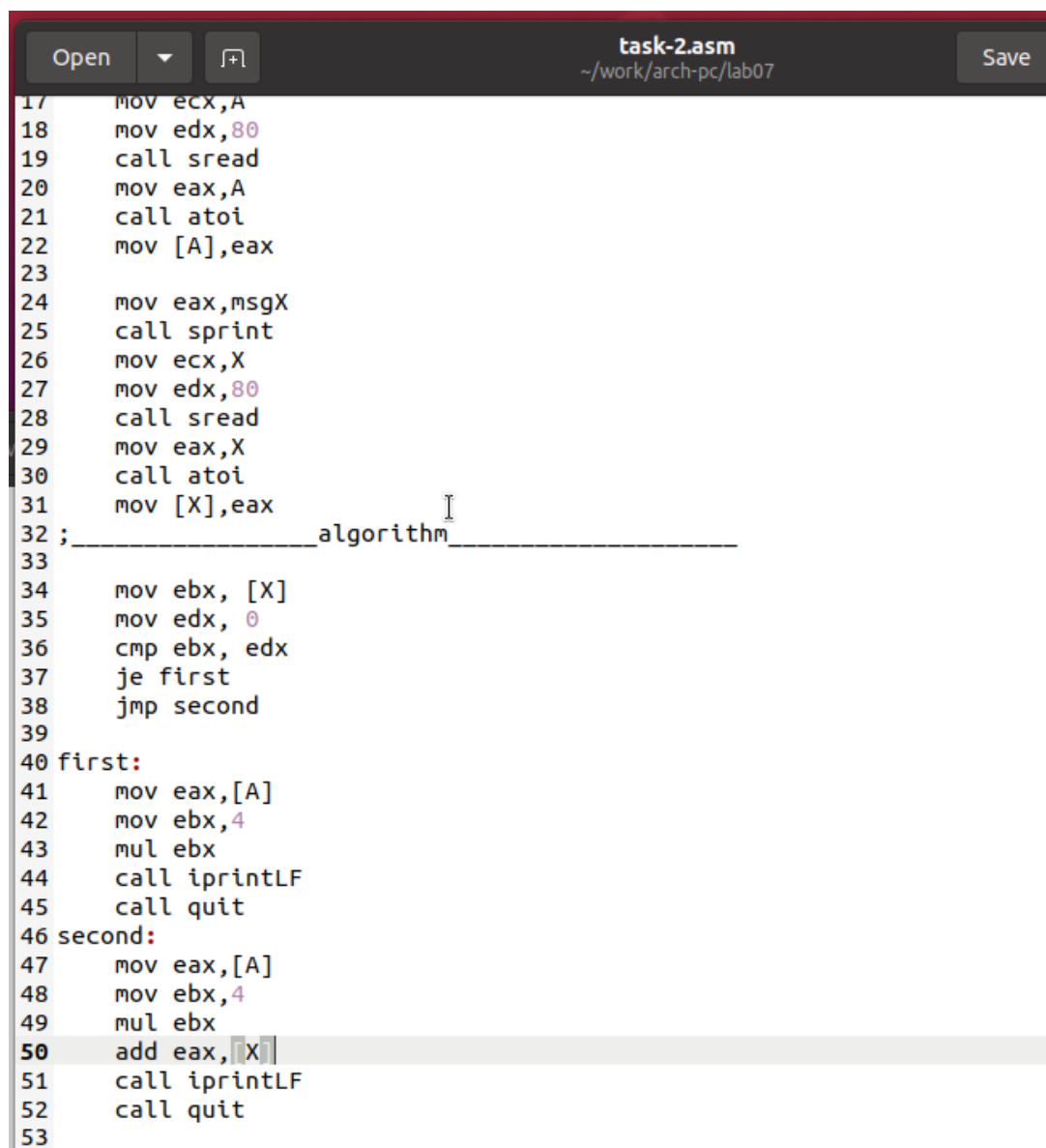
Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений X и a из 7.6.

для варианта 11

$$\begin{cases} 4a, x = 0 \\ 4a + x, x \neq 0 \end{cases}$$

Если подставить $x = 0, a = 3$ получается 12.

Если подставить $x = 1, a = 2$ получается 9.



```
17    mov ecx,A
18    mov edx,80
19    call sread
20    mov eax,A
21    call atoi
22    mov [A],eax
23
24    mov eax,msgX
25    call sprintf
26    mov ecx,X
27    mov edx,80
28    call sread
29    mov eax,X
30    call atoi
31    mov [X],eax
32 ; _____algorithm_____
33
34    mov ebx, [X]
35    mov edx, 0
36    cmp ebx, edx
37    je first
38    jmp second
39
40 first:
41    mov eax,[A]
42    mov ebx,4
43    mul ebx
44    call iprintLF
45    call quit
46 second:
47    mov eax,[A]
48    mov ebx,4
49    mul ebx
50    add eax,[X]
51    call iprintLF
52    call quit
53
```

Рис. 3.14: Программа в файле task-2.asm

```
bduardiev@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf task-2.asm
bduardiev@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 task-2.o -o task-2
bduardiev@VirtualBox:~/work/arch-pc/lab07$ ./task-2
Input A: 3
Input X: 0
12
bduardiev@VirtualBox:~/work/arch-pc/lab07$ ./task-2
Input A: 2
Input X: 1
9
bduardiev@VirtualBox:~/work/arch-pc/lab07$
```

Рис. 3.15: Запуск программы task-2.asm

4 Выводы

Изучили команды условного и безусловного переходов, познакомились с фактом листинга.