

Отчёта по лабораторной работе 8

Команды безусловного и условного переходов в Nasm.

Программирование ветвлений

Агоссоу Вигнон Тримегистре Разиел НФИбд-05-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	22

Список иллюстраций

2.1	Файл lab8-1.asm:	7
2.2	Программа lab8-1.asm:	8
2.3	Файл lab8-1.asm:	9
2.4	Программа lab8-1.asm:	10
2.5	Файл lab8-1.asm	11
2.6	Программа lab8-1.asm	12
2.7	Файл lab8-2.asm	13
2.8	Программа lab8-2.asm	14
2.9	Файл листинга lab8-2	15
2.10	ошибка трансляции lab8-2	16
2.11	файл листинга с ошибкой lab8-2	17
2.12	Файл lab8-3.asm	18
2.13	Программа lab8-3.asm	19
2.14	Файл lab8-4.asm	20
2.15	Программа lab8-4.asm	21

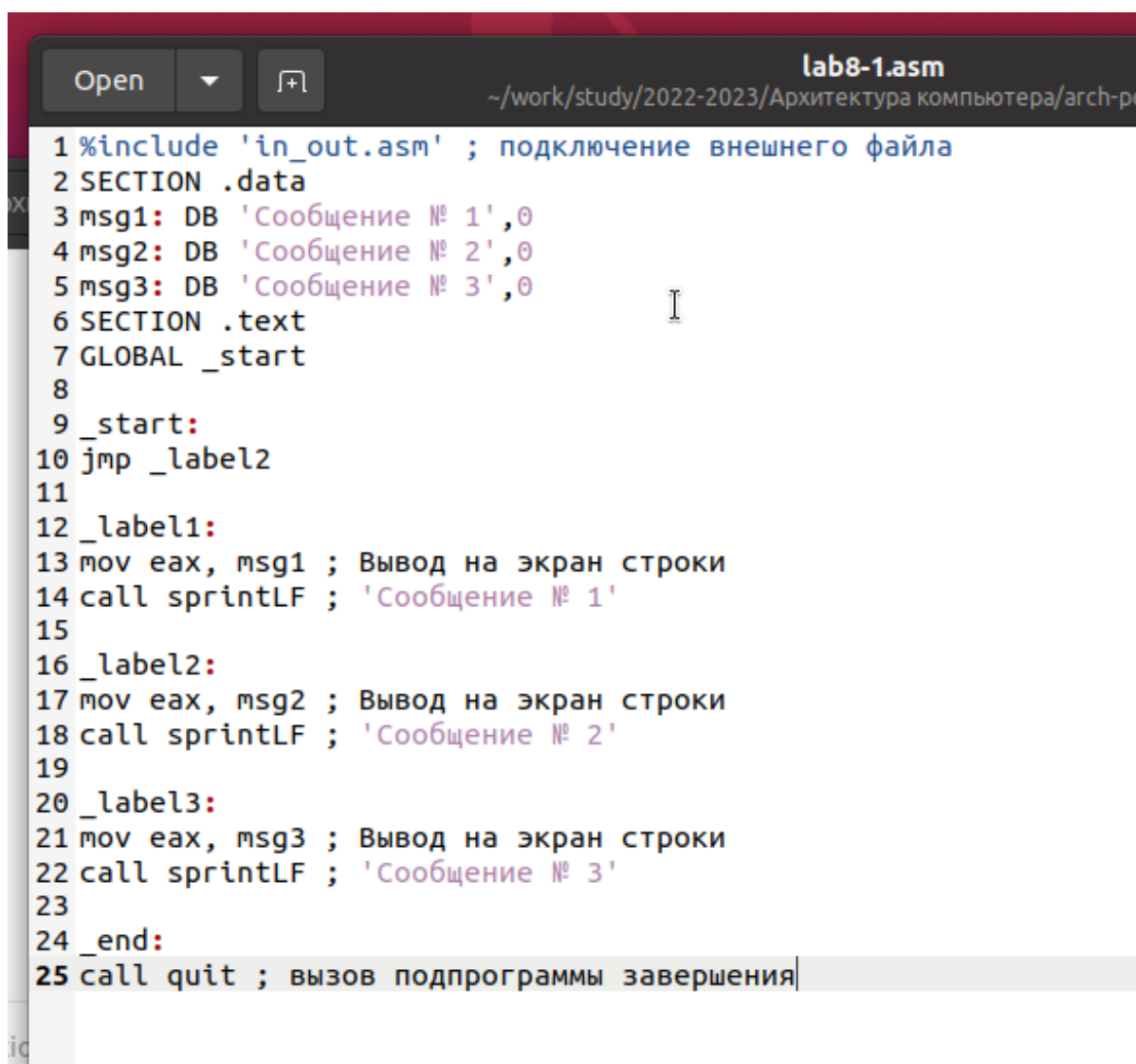
Список таблиц

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 Выполнение лабораторной работы

1. Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm
2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введите в файл lab8-1.asm текст программы из листинга 8.1. (рис. [2.1])

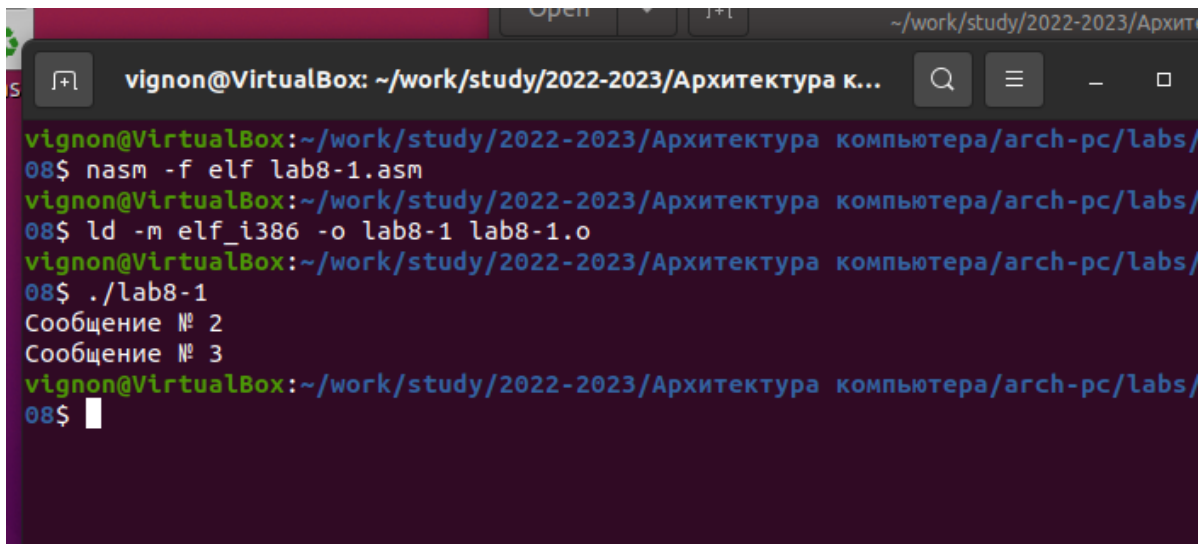


```
lab8-1.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc

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
9 _start:
10 jmp _label2
11
12 _label1:
13 mov eax, msg1 ; Вывод на экран строки
14 call sprintf ; 'Сообщение № 1'
15
16 _label2:
17 mov eax, msg2 ; Вывод на экран строки
18 call sprintf ; 'Сообщение № 2'
19
20 _label3:
21 mov eax, msg3 ; Вывод на экран строки
22 call sprintf ; 'Сообщение № 3'
23
24 _end:
25 call quit ; вызов подпрограммы завершения
```

Рис. 2.1: Файл lab8-1.asm:

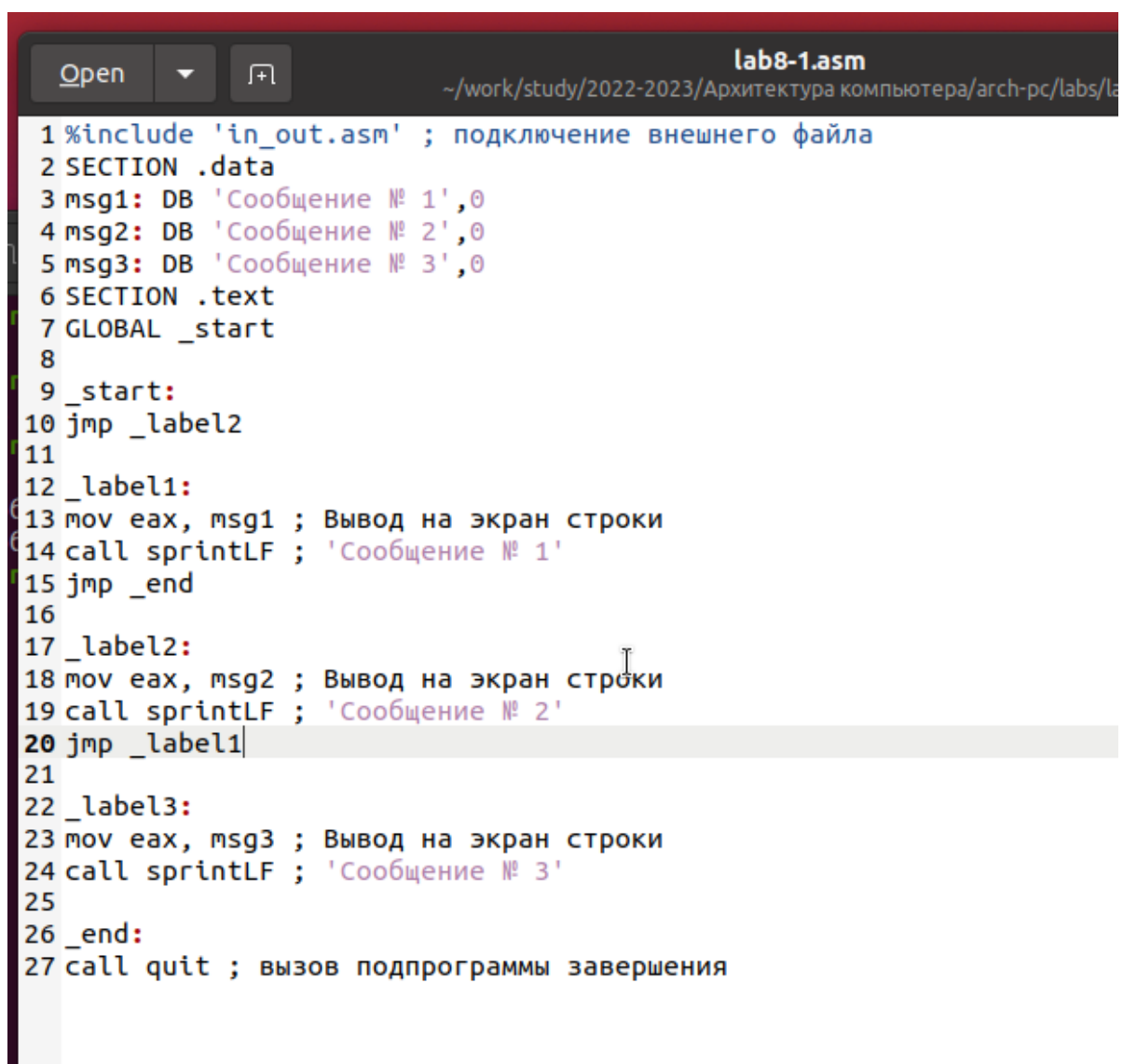
Создайте исполняемый файл и запустите его. (рис. [2.2])



```
vignon@VirtualBox: ~/work/study/2022-2023/Архитектура к...
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/
08$ nasm -f elf lab8-1.asm
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/
08$ ld -m elf_i386 -o lab8-1 lab8-1.o
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/
08$ ./lab8-1
Сообщение № 2
Сообщение № 3
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/
08$
```

Рис. 2.2: Программа lab8-1.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`). Измените текст программы в соответствии с листингом 8.2. (рис. [2.3], [2.4])



```
lab8-1.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab8-1.asm

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
9 _start:
10 jmp _label2
11
12 _label1:
13 mov eax, msg1 ; Вывод на экран строки
14 call sprintLF ; 'Сообщение № 1'
15 jmp _end
16
17 _label2:
18 mov eax, msg2 ; Вывод на экран строки
19 call sprintLF ; 'Сообщение № 2'
20 jmp _label1
21
22 _label3:
23 mov eax, msg3 ; Вывод на экран строки
24 call sprintLF ; 'Сообщение № 3'
25
26 _end:
27 call quit ; вызов подпрограммы завершения
```

Рис. 2.3: Файл lab8-1.asm:

```
08$  
vignon@VirtualBox:~/work/study/2022-2023/Архитектура ко  
08$ nasm -f elf lab8-1.asm  
vignon@VirtualBox:~/work/study/2022-2023/Архитектура ко  
08$ ld -m elf_i386 -o lab8-1 lab8-1.o  
vignon@VirtualBox:~/work/study/2022-2023/Архитектура ко  
08$ ./lab8-1  
Сообщение № 2  
Сообщение № 1  
vignon@VirtualBox:~/work/study/2022-2023/Архитектура ко  
08$ █
```

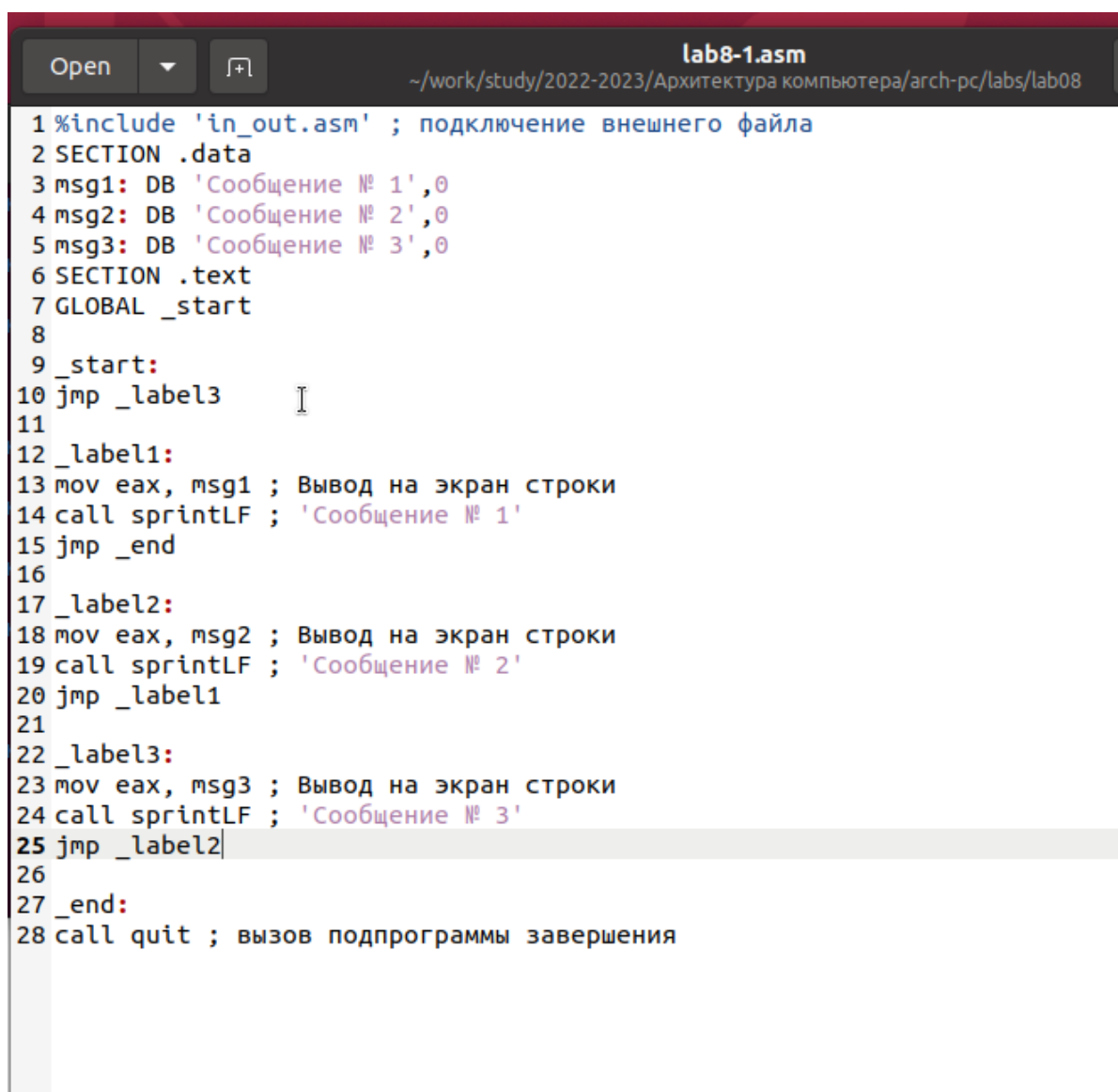
Рис. 2.4: Программа lab8-1.asm:

Измените текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим (рис. [2.5], [2.6]):

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
lab8-1.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

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
9 _start:
10 jmp _label3
11
12 _label1:
13 mov eax, msg1 ; Вывод на экран строки
14 call sprintLF ; 'Сообщение № 1'
15 jmp _end
16
17 _label2:
18 mov eax, msg2 ; Вывод на экран строки
19 call sprintLF ; 'Сообщение № 2'
20 jmp _label1
21
22 _label3:
23 mov eax, msg3 ; Вывод на экран строки
24 call sprintLF ; 'Сообщение № 3'
25 jmp _label2
26
27 _end:
28 call quit ; вызов подпрограммы завершения
```

Рис. 2.5: Файл lab8-1.asm

```
Сообщение № 1
vignon@VirtualBox:~/work/study/2022-2023/Архитектура
08$ nasm -f elf lab8-1.asm
vignon@VirtualBox:~/work/study/2022-2023/Архитектура
08$ ld -m elf_i386 -o lab8-1 lab8-1.o
vignon@VirtualBox:~/work/study/2022-2023/Архитектура
08$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
vignon@VirtualBox:~/work/study/2022-2023/Архитектура
08$
```

Рис. 2.6: Программа lab8-1.asm

3. 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. Создайте исполняемый файл и проверьте его работу для разных значений В. (рис. [2.7], [2.8])

```
lab8-2.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08
Save

lab8-1.asm
lab8-2.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

Saving file "/home/vignon/work/study/2022-2023/Архитектура ...  Matlab Tab Width: 8 Ln 34, Col 12 INS
```

Рис. 2.7: Файл lab8-2.asm

```
vignon@VirtualBox:~/work/study/2022-2023/Архитекту
08$ nasm -f elf lab8-2.asm
vignon@VirtualBox:~/work/study/2022-2023/Архитекту
08$ ld -m elf_i386 -o lab8-2 lab8-2.o
vignon@VirtualBox:~/work/study/2022-2023/Архитекту
08$ ./lab8-2
Введите B: 100
Наибольшее число: 100
vignon@VirtualBox:~/work/study/2022-2023/Архитекту
08$ ./lab8-2
Введите B: 40
Наибольшее число: 50
vignon@VirtualBox:~/work/study/2022-2023/Архитекту
08$ █
```

Рис. 2.8: Программа lab8-2.asm

4. Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Создайте файл листинга для программы из файла lab8-2.asm (рис. [2.9])

```

lab8-2.lst
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08
Save

lab8-1.asm  x  lab8-2.asm  x  lab8-2.lst  x
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
14    13
15    14 0000000B 29D8     <1>    finished:
16    15 0000000D 5B       <1>    sub     eax, ebx
17    16 0000000E C3       <1>    pop     ebx
18    17
19    18
20    19
21    20
22    21
23    22
24    23 0000000F 52       <1>    ;----- sprint -----
25    24 00000010 51       <1>    ; Функция печати сообщения
26    25 00000011 53       <1>    ; входные данные: mov eax,<message>
27    26 00000012 50       <1>    sprint:
28    27 00000013 E8E8FFFF <1>    push    edx
29    28
30    29 00000018 89C2     <1>    push    ecx
31    30 0000001A 58       <1>    push    ebx
32    31
33    32 0000001B 89C1     <1>    push    eax
34    33 0000001D B801000000 <1>    call    strlen
35    34 00000022 B804000000 <1>
36    35 00000027 CD80     <1>    mov     edx, eax
37    36
38    37 00000029 5B       <1>    pop     eax
39    38 0000002A 59       <1>    pop     ebx
40    39 0000002B 5A       <1>    pop     ecx
41    40 0000002C C3       <1>    pop     edx
42    41
Loading file "/home/vignon/work/study/2022-2023/Архитект... Plain Text Tab Width: 8 Ln 1, Col 1 INS

```

Рис. 2.9: Файл листинга lab8-2

Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору.

строка 51

- 51 - номер строки
- 00000033 - адрес
- B80A000000 - машинный код
- mov eax, 0Ah - код программы

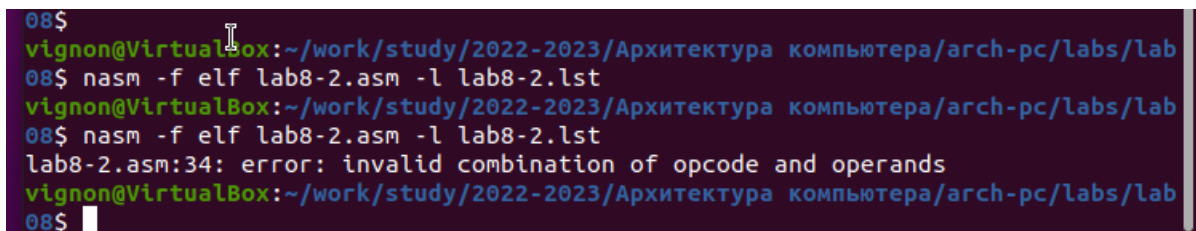
строка 52

- 52 - номер строки
- 00000038 - адрес
- 50 - машинный код
- push eax- код программы

строка 53

- 53 - номер строки
- 00000039 - адрес
- 89E0 - машинный код
- mov eax, esp - код программы

Откройте файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалите один операнд. Выполните трансляцию с получением файла листинга (рис. [2.10],[2.11])



```

08$ 
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab
08$ nasm -f elf lab8-2.asm -l lab8-2.lst
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab
08$ nasm -f elf lab8-2.asm -l lab8-2.lst
lab8-2.asm:34: error: invalid combination of opcode and operands
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab
08$ 
  
```

Рис. 2.10: ошибка трансляции lab8-2


```

lab8-2.lst
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08
Save

lab8-1.asm  x  lab8-2.asm  x  lab8-2.lst  x

число
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
'check_B',
205 30 00000124 8B0D[39000000]
206 31 0000012A 890D[00000000]
207 32
число
208 33
209 34
210 34 *****
211 35 00000130 E867FFFFFF
число
212 36 00000135 A3[00000000]
213 37
214 38 0000013A 8B0D[00000000]
215 39 00000140 3B0D[0A000000]
216 40 00000146 7F0C
217 41 00000148 8B0D[0A000000]
218 42 0000014E 890D[00000000]
219 43
220 44
221 45 00000154 B8[13000000]
222 46 00000159 E8B1FFFFFF
223 47 0000015E A1[00000000]
224 48 00000163 E81EFFFFFF
225 49 00000168 E86EFFFFFF
226 50

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

mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в

check_B:
mov eax
error: invalid combination of opcode and operands
call atoi ; Вызов подпрограммы перевода символа в

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

```

Plain Text Tab Width: 8 Ln 1, Col 1 INS

Рис. 2.11: файл листинга с ошибкой lab8-2

5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу (рис. [2.12],[2.13])

для варианта 11 - 21, 28, 34

```
lab8-3.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08
Open Save
lab8-1.asm lab8-2.asm lab8-2.lst lab8-3.asm
1 %include 'in_out.asm'
2 SECTION .data
3     msgA:    DB 'Input A: ',0
4     msgB:    DB 'Input B: ',0
5     msgC:    DB 'Input C: ',0
6     answer:  DB 'Smallest: ',0
7
8 SECTION .bss
9     A:  RESB 80
10    B:  RESB 80
11    C:  RESB 80
12    result:  RESB 80
13    min: RESB 80
14
15 SECTION .text
16     GLOBAL _start
17
18 _start:
19     mov eax,msgA
20     call sprint
21     mov ecx,A
22     mov edx,80
23     call sread
24     mov eax,A
25     call atoi
26     mov [A],eax
27
28     mov eax, msgB
29     call sprint
30     mov ecx,B
31     mov edx,80
32     call sread
33     mov eax,B
34     call atoi
```

Matlab Tab Width: 8 Ln 1, Co

Рис. 2.12: Файл lab8-3.asm

```

08$
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab
08$ nasm -f elf lab8-3.asm
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab
08$ ld -m elf_i386 -o lab8-3 lab8-3.o
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab
08$ ./lab8-3
Input A: 21
Input B: 28
Input C: 34
Smallest: 21
vignon@VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab
08$

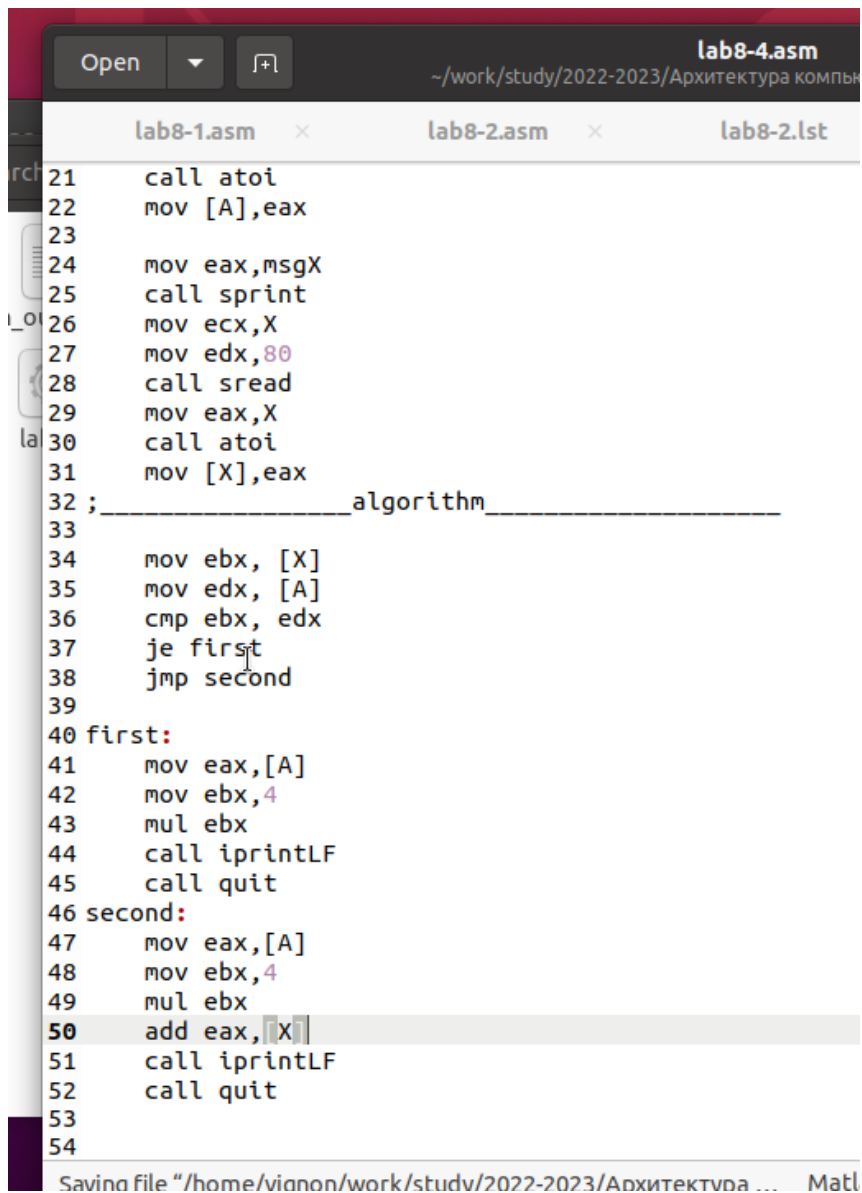
```

Рис. 2.13: Программа lab8-3.asm

6. 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 8.6 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 8.6. (рис. [2.14],[2.15])

для варианта 11

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



```
lab8-4.asm
~/work/study/2022-2023/Архитектура компьютерных систем

lab8-1.asm x lab8-2.asm x lab8-2.lst

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, [A]
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
54

Saving file "/home/vianon/work/study/2022-2023/Архитектура ... Matl
```

Рис. 2.14: Файл lab8-4.asm

```
08$  
vignon@VirtualBox:~/work/study/2022-2023/Apx  
08$ nasm -f elf lab8-4.asm  
vignon@VirtualBox:~/work/study/2022-2023/Apx  
08$ ld -m elf_i386 -o lab8-4 lab8-4.o  
vignon@VirtualBox:~/work/study/2022-2023/Apx  
08$ ./lab8-4  
Input A: 3  
Input X: 0  
12  
vignon@VirtualBox:~/work/study/2022-2023/Apx  
08$ ./lab8-4  
Input A: 2  
Input X: 1  
9  
vignon@VirtualBox:~/work/study/2022-2023/Apx  
08$
```

Рис. 2.15: Программа lab8-4.asm

3 Выводы

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