

Отчёт по лабораторной работе 7

дисциплина: Архитектура компьютера

Шангина В. А НКАбд-05-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Условные переходы	11
2.3	Изучение структуры файла листинга	13
2.4	Самостоятельное задание	16
3	Выводы	21

Список иллюстраций

2.1	Создан каталог	6
2.2	Программа lab7-1.asm	7
2.3	Запуск программы lab7-1.asm	7
2.4	Программа lab7-1.asm	8
2.5	Запуск программы lab7-1.asm	9
2.6	Программа lab7-1.asm	10
2.7	Запуск программы lab7-1.asm	10
2.8	Программа lab7-2.asm	12
2.9	Запуск программы lab7-2.asm	13
2.10	Файл листинга lab7-2	14
2.11	Ошибка трансляции lab7-2	15
2.12	Файл листинга с ошибкой lab7-2	16
2.13	Программа task1.asm	17
2.14	Запуск программы task1.asm	17
2.15	Программа task2.asm	19
2.16	Запуск программы task2.asm	20

Список таблиц

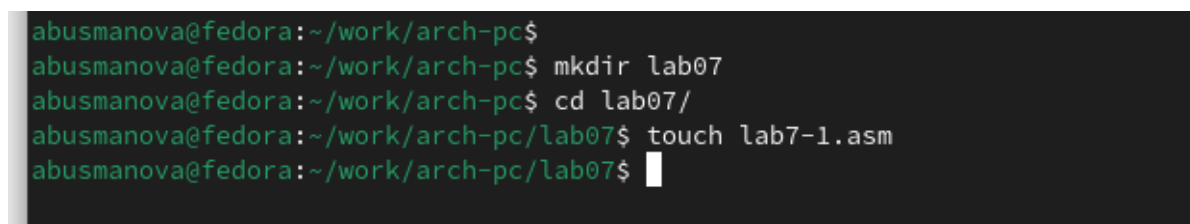
1 Цель работы

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

2.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7 и файл lab7-1.asm.
(рис. 2.1)

A terminal window with a dark background and green text. The prompt is 'abusmanova@fedora:~/work/arch-pc\$'. The first command is 'mkdir lab07'. The second command is 'cd lab07/'. The third command is 'touch lab7-1.asm'. The prompt is now 'abusmanova@fedora:~/work/arch-pc/lab07\$' followed by a cursor.

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

Рис. 2.1: Создан каталог

Инструкция `jmp` в NASM используется для реализации безусловных переходов. Пример программы, демонстрирующей эту инструкцию, приведен в файле lab7-1.asm. (рис. 2.2)

```

%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

_label1:
mov eax, msg1
call sprintLF

_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit

```

Рис. 2.2: Программа lab7-1.asm

Создаю исполняемый файл и запускаю его. (рис. 2.3)

```

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

```

Рис. 2.3: Запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы как вперед, так и назад. Для изменения последовательности вывода программы добавляю метки `_label1` и `_end`. Таким образом, вывод программы изменится: сначала отобразится сообщение № 2, затем сообщение № 1, и программа завершит работу.

Обновляю текст программы согласно листингу 7.2. (рис. 2.4, рис. 2.5)

```
%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label2

_label1:
mov eax, msg1
call sprintf
jmp _end

_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf

_end:
call quit
```

Рис. 2.4: Программа lab7-1.asm


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

Рис. 2.5: Запуск программы lab7-1.asm

Дорабатываю текст программы для вывода следующих сообщений:

Сообщение № 3

Сообщение № 2

Сообщение № 1

Результат показан на рисунках (рис. 2.6, рис. 2.7).

```

%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label3

_label1:
mov eax, msg1
call sprintf
jmp _end

_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf
jmp _label2

_end:
call quit

```

Рис. 2.6: Программа lab7-1.asm

```

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

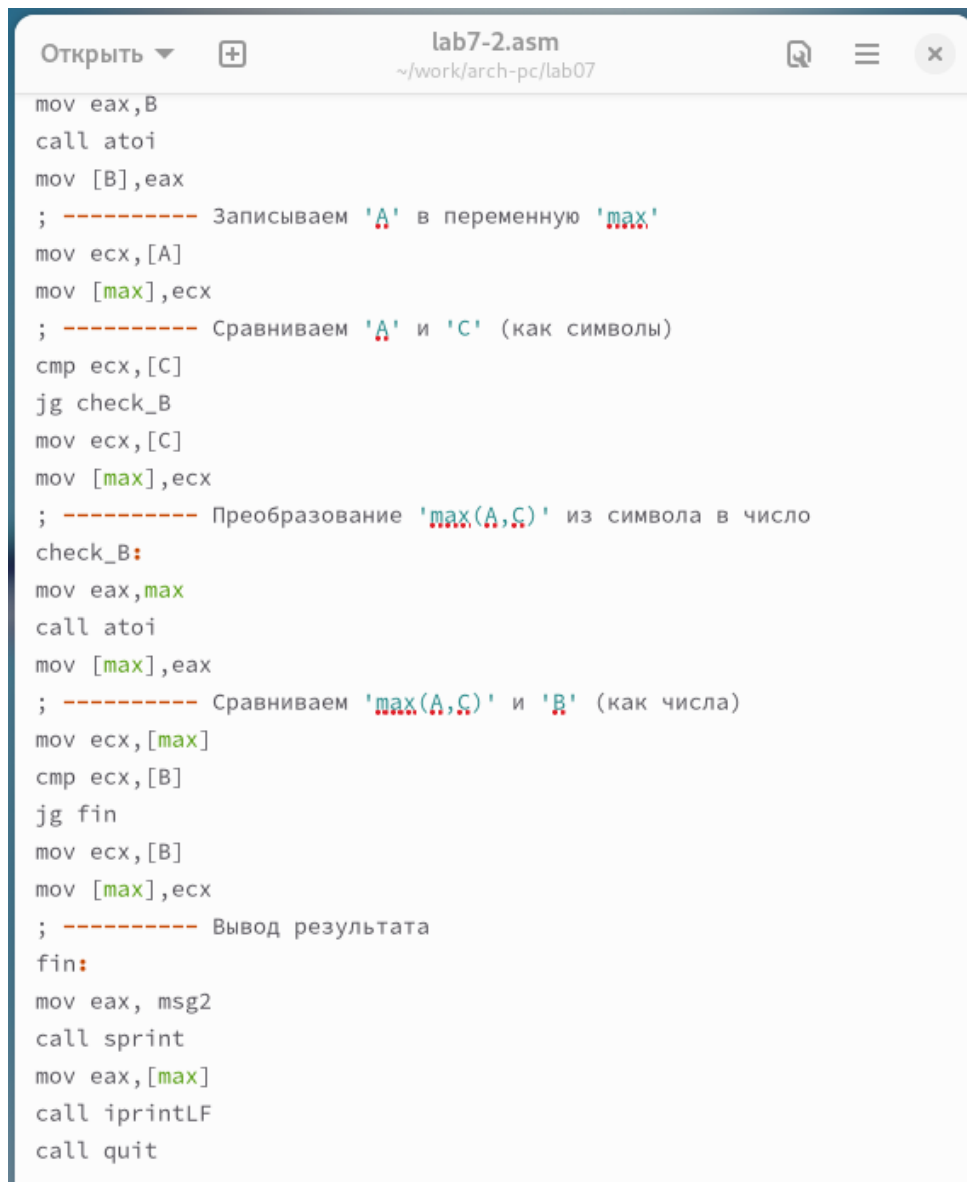
```

Рис. 2.7: Запуск программы lab7-1.asm

Использование инструкции `jmp` обеспечивает переходы независимо от условий. Однако для реализации условных переходов требуется использование дополнительных инструкций.

2.2 Условные переходы

Для демонстрации условных переходов создаю программу, определяющую максимальное значение среди трех переменных: А, В и С. Значения А и С задаются в программе, а В вводится с клавиатуры. Результаты работы программы представлены на рисунках (рис. 2.8, рис. 2.9).



```
Открыть ▾ + lab7-2.asm ~/.work/arch-pc/lab07
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint
mov eax,[max]
call iprintLF
call quit
```

Рис. 2.8: Программа lab7-2.asm

```
abusmanova@fedora:~/work/arch-pc/lab07$  
abusmanova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm  
abusmanova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2  
abusmanova@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 30  
Наибольшее число: 50  
abusmanova@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 50  
Наибольшее число: 50  
abusmanova@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 70  
Наибольшее число: 70  
abusmanova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.9: Запуск программы lab7-2.asm

2.3 Изучение структуры файла листинга

Для получения файла листинга указываю ключ `-l` при ассемблировании. Результат ассемблирования программы lab7-2.asm представлен на рисунке (рис. 2.10).

```

19 00000017 8B7E11C7      mov     esi,esi
20                                ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000]    mov     eax,B
22 00000106 F891FFFFFF      call   atoi
23 0000010B A3[0A000000]    mov     [B],eax
24                                ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000]    mov     ecx,[A]
26 00000116 890D[00000000]    mov     [max],ecx
27                                ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]    cmp     ecx,[C]
29 00000122 7F0C            jg     check_B
30 00000124 8B0D[39000000]    mov     ecx,[C]
31 0000012A 890D[00000000]    mov     [max],ecx
32                                ; ----- Преобразование 'max(A,C)' из символа в число
33                                check_B:
34 00000130 B8[00000000]    mov     eax,max
35 00000135 F862FFFFFF      call   atoi
36 0000013A A3[00000000]    mov     [max],eax
37                                ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000]    mov     ecx,[max]
39 00000145 3B0D[0A000000]    cmp     ecx,[B]
40 0000014B 7F0C            jg     fin
41 0000014D 8B0D[0A000000]    mov     ecx,[B]
42 00000153 890D[00000000]    mov     [max],ecx
43                                ; ----- Вывод результата
44                                fin:
45 00000159 B8[13000000]    mov     eax,msg2
46 0000015E F8ACFFFFFF      call   sprintf
47 00000163 A1[00000000]    mov     eax,[max]
48 00000168 F819FFFFFF      call   iprintLF

```

Рис. 2.10: Файл листинга lab7-2

Анализируя структуру листинга, можно увидеть соответствие строк кода и их машинного представления. Например:

- **Строка 203:**

- Номер строки: 28
- Адрес: 0000011C
- Машинный код: 3B0D[39000000]
- Команда: `cmp ecx,[C]`

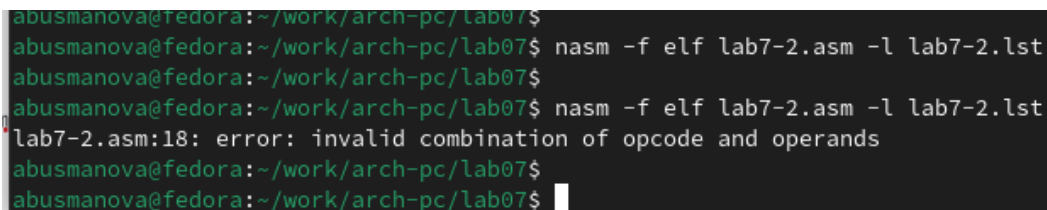
- **Строка 204:**

- Номер строки: 29
- Адрес: 00000122
- Машинный код: 7F0C
- Команда: jg check_B

- **Строка 205:**

- Номер строки: 30
- Адрес: 00000124
- Машинный код: 8B0D[39000000]
- Команда: mov ecx,[C]

Далее изменяю инструкцию с двумя операндами, удаляя один, и повторяю трансляцию. Возникает ошибка, результат которой отображен на рисунках (рис. 2.11, рис. 2.12).



```
abusmanova@fedora:~/work/arch-pc/lab07$
abusmanova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
abusmanova@fedora:~/work/arch-pc/lab07$
abusmanova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:18: error: invalid combination of opcode and operands
abusmanova@fedora:~/work/arch-pc/lab07$
abusmanova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```
Открыть ▾ + lab7-2.lst ~/work/arch-pc/lab07
3 00000009 B8D182D0B520423A20-
3 00000012 00
4 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
4 0000001C BFD0BBBD18CD188D0B5-
4 00000025 D0B520D187D0B8D181-
4 0000002E D0BBD0BE3A2000
5 00000035 32300000 A dd '20'
6 00000039 35300000 C dd '50'
7 section .bss
8 00000000 <res. Ah> max resb 10
9 0000000A <res. Ah> B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 mov edx,
18 ***** error: invalid combination of opcode and operands
19 000000F7 E847FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в число
21 000000FC B8[0A000000] mov eax,B
22 00000101 E896FFFFFF call atoi
23 00000106 A3[0A000000] mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D[35000000] mov ecx,[A]
26 00000111 890D[00000000] mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
```

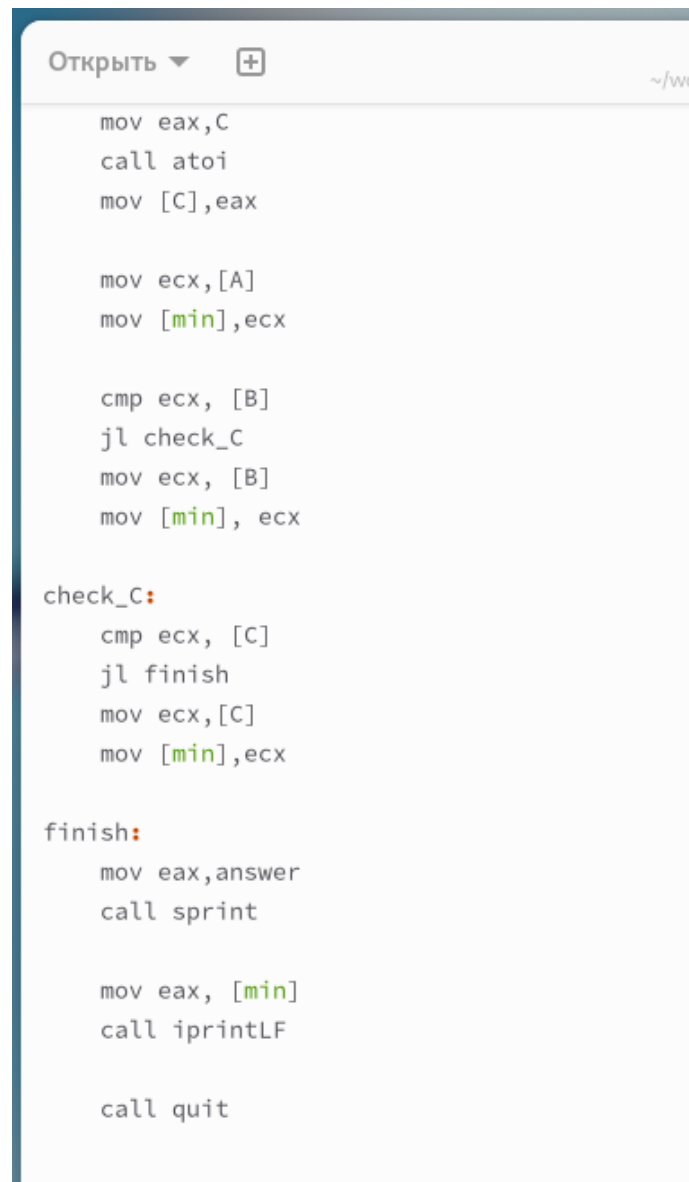
Рис. 2.12: Файл листинга с ошибкой lab7-2

2.4 Самостоятельное задание

1. Напишите программу, которая находит наименьшее значение из трех переменных a, b и c для следующих значений:

Вариант 4: 8,88,68.

Результат работы программы показан на рисунках (рис. 2.13, рис. 2.14).



```
Открыть ▾ + ~/wo
mov eax,C
call atoi
mov [C],eax

mov ecx,[A]
mov [min],ecx

cmp ecx,[B]
jl check_C
mov ecx,[B]
mov [min],ecx

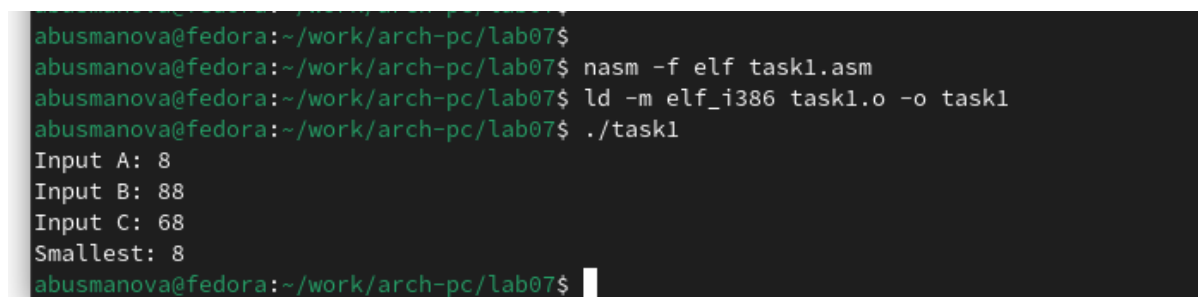
check_C:
cmp ecx,[C]
jl finish
mov ecx,[C]
mov [min],ecx

finish:
mov eax,answer
call sprint

mov eax,[min]
call iprintLF

call quit
```

Рис. 2.13: Программа task1.asm



```
abusmanova@fedora:~/work/arch-pc/lab07$
abusmanova@fedora:~/work/arch-pc/lab07$ nasm -f elf task1.asm
abusmanova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 task1.o -o task1
abusmanova@fedora:~/work/arch-pc/lab07$ ./task1
Input A: 8
Input B: 88
Input C: 68
Smallest: 8
abusmanova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.14: Запуск программы task1.asm

2. Напишите программу для вычисления функции $f(x)$ для введенных значений x и a :


Вариант 4:

$$f(x) = \begin{cases} 2x + a, & \text{если } a \neq 0 \\ 2x + 1, & \text{если } a = 0 \end{cases}$$

При $x = 3, a = 0$ результат: 7.

При $x = 3, a = 2$ результат: 8.

Результаты программы представлены на рисунках (рис. 2.15, рис. 2.16).



```
Открыть ▾ + task2.asm
~\work\arch-pc\lab07

mov ecx,X
mov edx,80
call sread
mov eax,X
call atoi
mov [X],eax

mov ebx, [A]
mov edx, 0
cmp ebx, edx
jne first
jmp second

first:
mov eax,[X]
mov ebx,2
mul ebx
add eax,[A]
call iprintLF
call quit

second:
mov eax,[X]
mov ebx,2
mul ebx
add eax,1
call iprintLF
call quit
```

Рис. 2.15: Программа task2.asm

```
abusmanova@fedora:~/work/arch-pc/lab07$  
abusmanova@fedora:~/work/arch-pc/lab07$  
abusmanova@fedora:~/work/arch-pc/lab07$ nasm -f elf task2.asm  
abusmanova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 task2.o -o task2  
abusmanova@fedora:~/work/arch-pc/lab07$ ./task2  
Input A: 0  
Input X: 3  
7  
abusmanova@fedora:~/work/arch-pc/lab07$ ./task2  
Input A: 2  
Input X: 3  
8  
abusmanova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы task2.asm

3 Выводы

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