

# Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Шабакова Карина Баировна

## Содержание

### 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

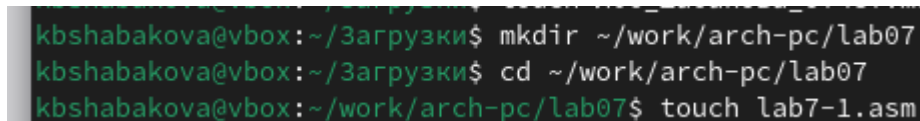
### 3 Теоретическое введение

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. 1).



```
kbshabakova@vbox:~/Загрузки$ mkdir ~/work/arch-pc/lab07
kbshabakova@vbox:~/Загрузки$ cd ~/work/arch-pc/lab07
kbshabakova@vbox:~/work/arch-pc/lab07$ touch lab7-1.asm
```

*Рис. 1: Создание каталога и файла для программы*

Копирую код из листинга в файл будущей программы. (рис. 2).

Рис. 2:

```
*~/work/arch-pc/lab07/lab7-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

### Сохранение программы

При запуске программы я убедилась в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. 3).

```
bash: ./lab7-1: нет такого файла или каталога
kbshabakova@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
kbshabakova@vbox:~/work/arch-pc/lab07$
```

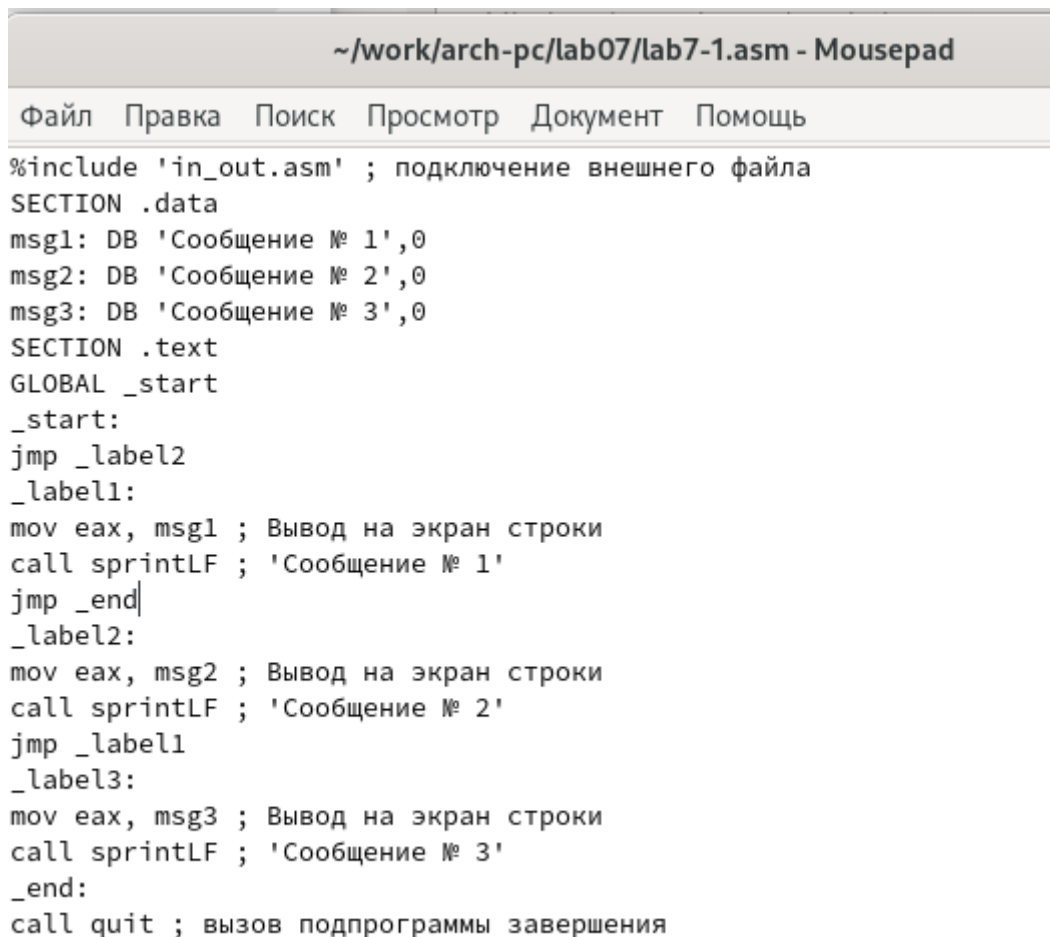
Рис. 3:

программы

Запуск

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4).

Рис. 4:

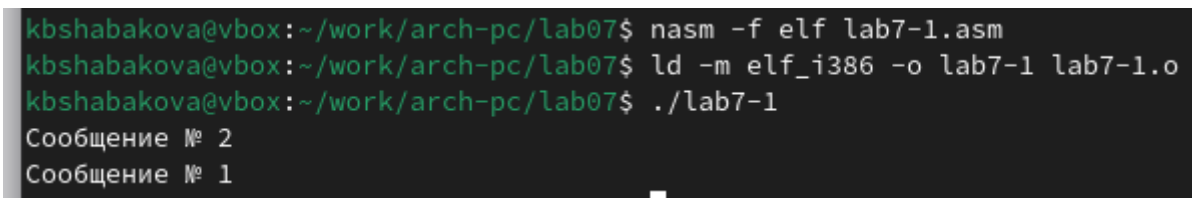


```
~/work/arch-pc/lab07/lab7-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

### Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 5).

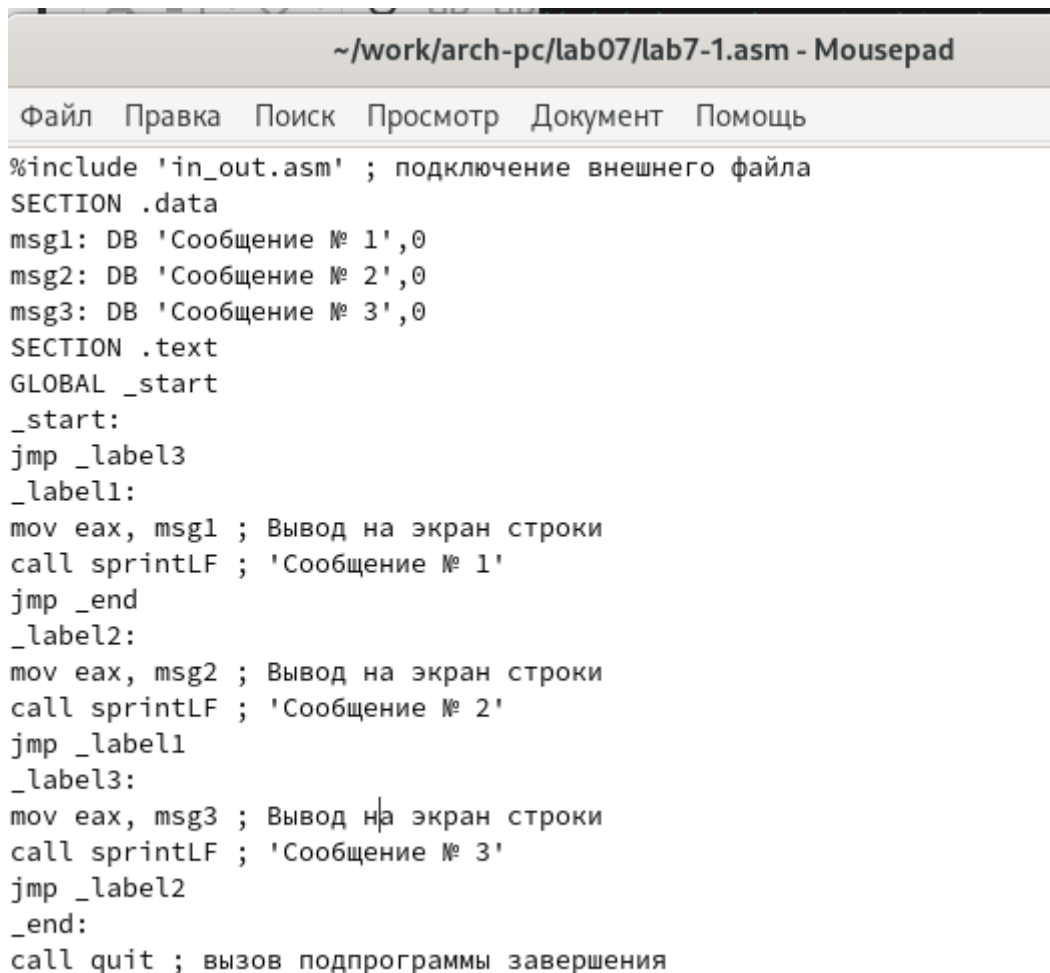


```
kbshabakova@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
kbshabakova@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
kbshabakova@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 5: Запуск измененной программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 6).

Рис. 6:

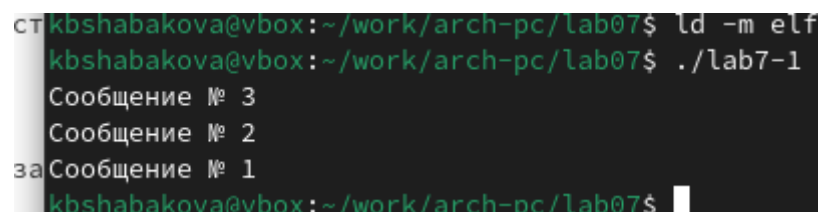


```
~/work/arch-pc/lab07/lab7-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

### Изменение программы

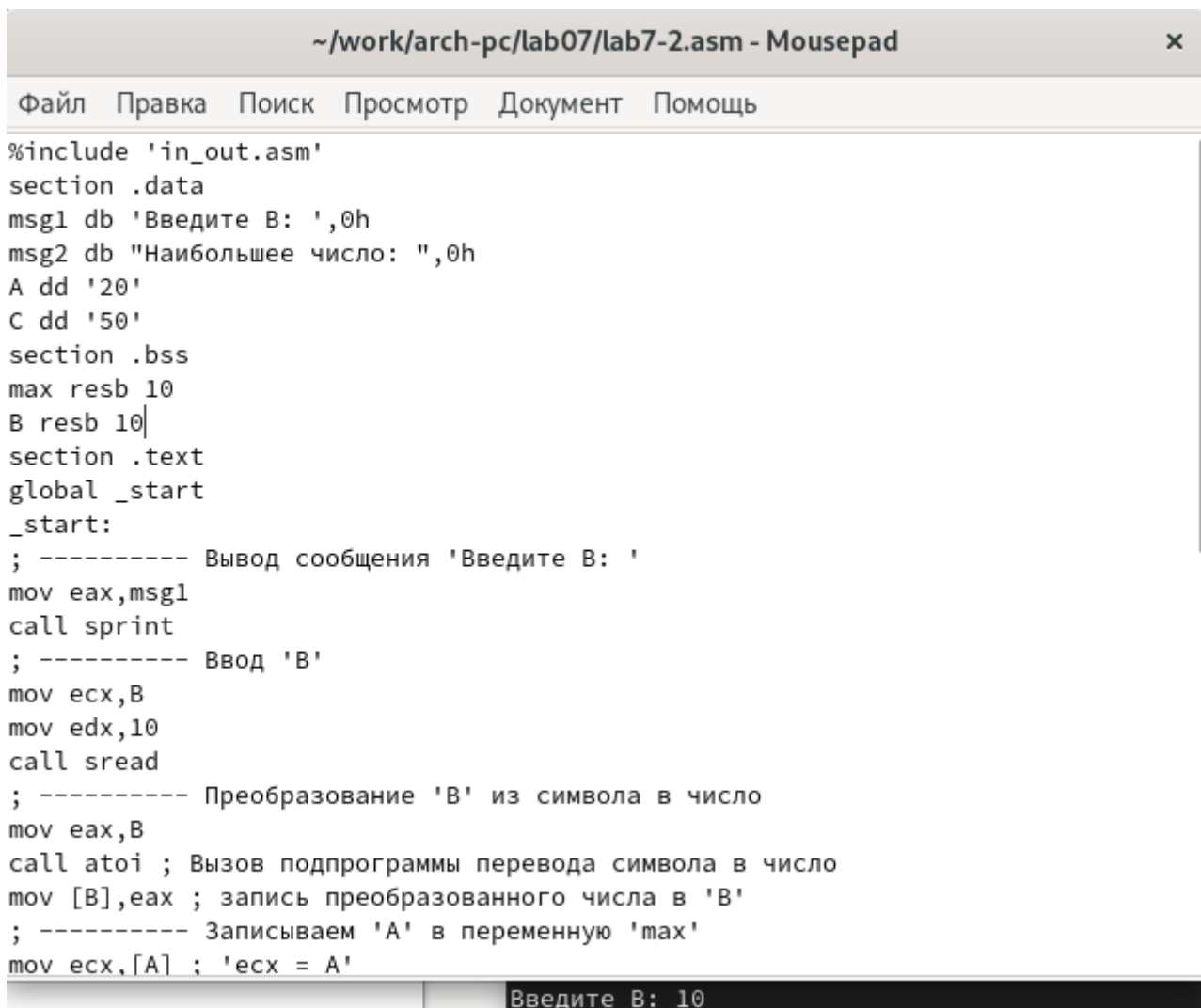
Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 7).



```
ст kbshabakova@vbox:~/work/arch-pc/lab07$ ld -m elf
kbshabakova@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
за Сообщением № 1
kbshabakova@vbox:~/work/arch-pc/lab07$
```

Рис. 7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 8).



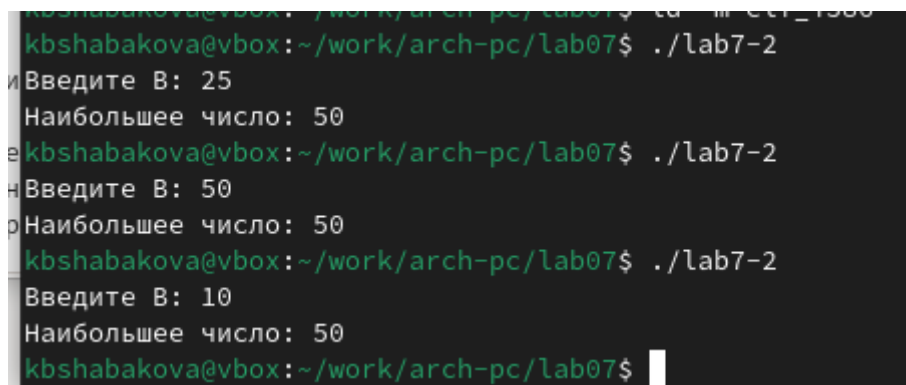
```
~/work/arch-pc/lab07/lab7-2.asm - Mousepad x
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm'
section .data
msg1 db 'Введите В: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите В: '
mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- Записываем 'А' в переменную 'max'
mov ecx,[A] ; 'ecx = A'

Введите В: 10
```

Рис. 8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 9).



```
kbshabakova@vbox: ~/work/arch-pc/lab07$ cd ~
kbshabakova@vbox: ~/work/arch-pc/lab07$ ./lab7-2
Введите В: 25
Наибольшее число: 50
kbshabakova@vbox: ~/work/arch-pc/lab07$ ./lab7-2
Введите В: 50
Наибольшее число: 50
kbshabakova@vbox: ~/work/arch-pc/lab07$ ./lab7-2
Введите В: 10
Наибольшее число: 50
kbshabakova@vbox: ~/work/arch-pc/lab07$
```

Рис. 9:  
программы

Проверка  
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. 10).

```
~/work/arch-pc/lab07/lab7-2.lst - Mousepad x
Файл Правка Поиск Просмотр Документ Помощь

1          %include 'in_out.asm'
1          <1> ;----- slen -----
2          <1> ; Функция вычисления длины сообщения
3          <1> slen:
4 00000000 53          <1>      push    ebx
5 00000001 89C3        <1>      mov     ebx, eax
6          <1>
7          <1> nextchar:
8 00000003 803800      <1>      cmp     byte [eax], 0
9 00000006 7403        <1>      jz      finished
10 00000008 40         <1>      inc     eax
11 00000009 EBF8       <1>      jmp     nextchar
12          <1>
13          <1> finished:
14 0000000B 29D8       <1>      sub     eax, ebx
15 0000000D 5B         <1>      pop     ebx
16 0000000E C3         <1>      ret
17          <1>
18          <1>
19          <1> ;----- sprint -----
20          <1> ; Функция печати сообщения
21          <1> ; входные данные: mov eax,<message>
22          <1> sprint:
23 0000000F 52          <1>      push    edx
24 00000010 51          <1>      push    ecx
25 00000011 53          <1>      push    ebx
26 00000012 50          <1>      push    eax
27 00000013 E8E8FFFFFF <1>      call    slen
28          <1>
29 00000018 89C2       <1>      mov     edx, eax
30 0000001A 58         <1>      pop     eax
31          <1>
32 0000001B 89C1       <1>      mov     ecx, eax
33 0000001D BB01000000    <1>      mov     ebx, 1
34 00000022 B804000000    <1>      mov     eax, 4
35 00000027 CD80       <1>      int     80h
36          <1>
37 00000029 5B         <1>      pop     ebx
```

Рис. 10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать 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. 11).

```
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'
jg 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,
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'
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
```

Рис. 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. 12).

```
~/work/arch-pc/lab07/lab7-2.lst - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
lab7-2.lst  x  lab7-2.asm  x
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,
34 ***** error: invalid combination of opcode and operands
35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
36 00000135 A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013A 8B0D[00000000] mov ecx,[max]
39 00000140 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 00000146 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 00000148 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
42 0000014E 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000154 B8[13000000] mov eax, msg2
46 00000159 E8B1FFFFFF call sprint ; Вывод сообщения 'Наибольшее число: '
47 0000015E A1[00000000] mov eax,[max]
48 00000163 E81EFFFFFF call iprintLF ; Вывод 'max(A,B,C)'
49 00000168 E86EFFFFFF call quit ; Выход
```

Рис. 12: Просмотр ошибки в файле листинга

### 4.3 Задания для самостоятельной работы

Искренне не понимаю, какой вариант я должна была получить во время 7 лабораторной работы, поэтому буду использовать свой вариант - девятый - из предыдущей лабораторной работы. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 13).



```
~/work/arch-pc/lab07/lab7-2.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
lab7-2.lst  x  lab7-2.asm  x
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
```

Рис. 13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

SECTION .data
db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

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. 14).

```

kbshabakova@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
kbshabakova@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
kbshabakova@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 98
Наименьшее число: 15

```

Рис. 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. 15).

```
~/work/arch-pc/lab07/lab7-3.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
lab7-2.lst  x  lab7-2.asm  x  lab7-3.asm  x

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
```

Рис. 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

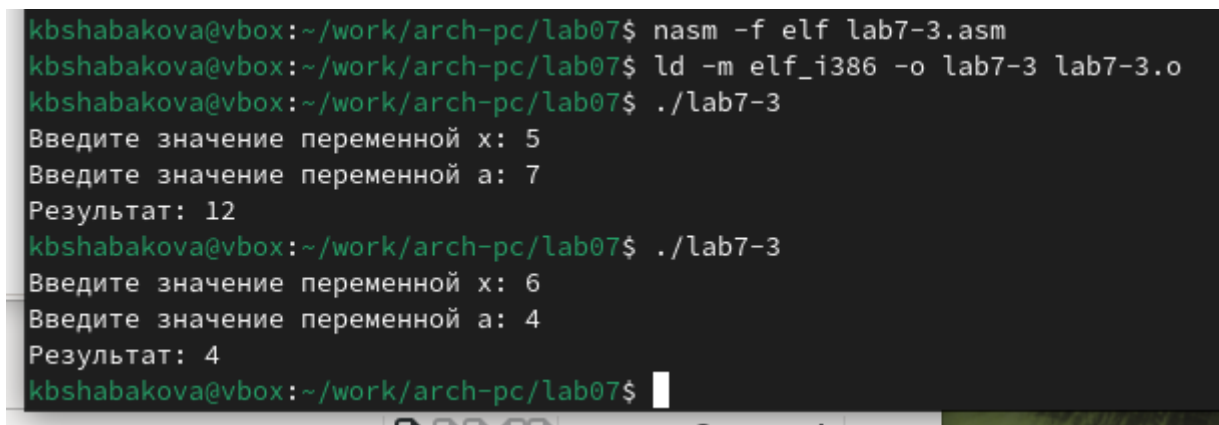
cmp edi, esi
jle add_values
mov eax, esi
jmp print_result

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. 16).



```

kbshabakova@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
kbshabakova@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
kbshabakova@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
kbshabakova@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
kbshabakova@vbox:~/work/arch-pc/lab07$

```

Рис. 16: Проверка работы второй программы

## 5 Выводы

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился 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.