Лабораторная работа №7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений.

Карачевцева Елизавета Васильевна

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 Выполнение лабораторной работы

1) Я создала каталог lab7 и внутри создал файл lab7-1.asm

```
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ mkdir lab07
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd lab07
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch lab7-1.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ls
lab7-1.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ pwd
/home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
```

Рис. 2.1: Создание файла lab7-1.asm

2) Я ввела в файл текст программы и запустила его.

```
/home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-1.asm Изменён
%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
```

Рис. 2.2: Текст в файле lab7-1.asm

3) Я создала исполняемый файл и запустила его. Результат соответствовал нужному.

```
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1
.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o
lab7-1 lab7-1.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 2.3: Запуск программы lab7-1

4)Я изменила текст программы чтобы выводился нужный ответ и создала исполняемый файл.

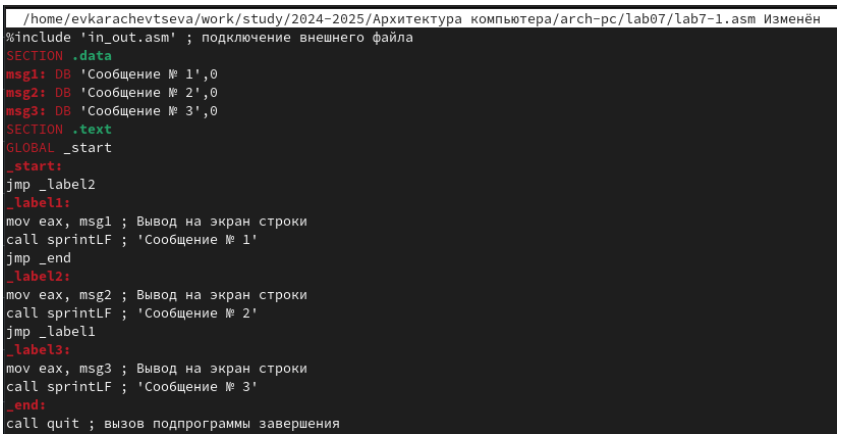

Рис. 2.4: Изменение текста

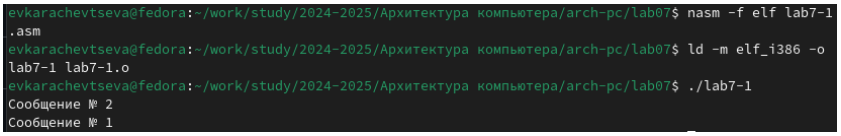

Рис. 2.5: Проверка работы программы

5)Я изменила текст программы чтобы сначала выводило сообщение 3,затем 2, затем 1.

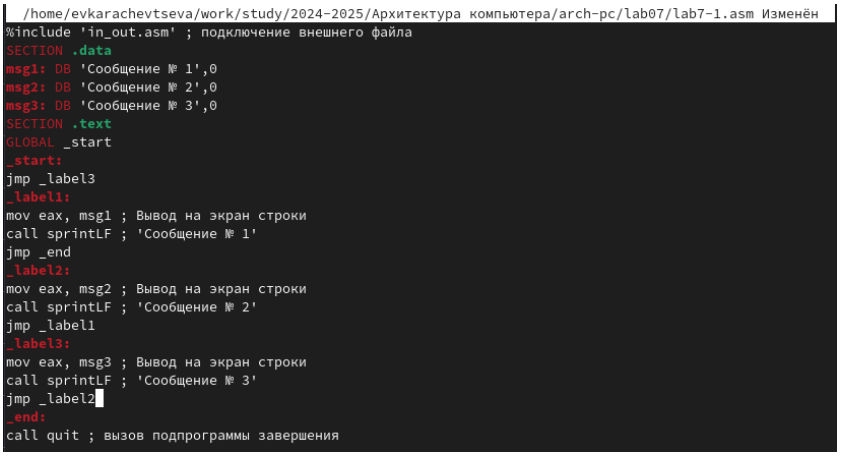

Рис. 2.6: Изменение текста

6) Запустила программу и проверила ее работу.

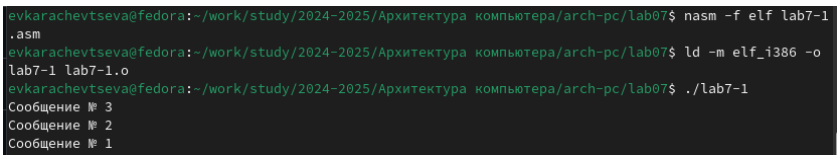

Рис. 2.7: Запуск программы

7) Я создала файл lab7-2.asm и написала текст программы.

```
 GNU nano 7.2
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',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 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'
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,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'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ---------- Вывод результата
fin:
```

Рис. 2.8: Текст программы для сравнения чисел

8) Я ввела два разных числа чтобы проверить как работает программа.

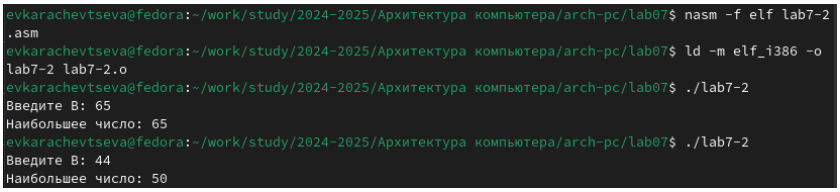

Рис. 2.9: Программа для сравнения чисел

9) Я создала файл листинга lab7-2.lst и открыла его.

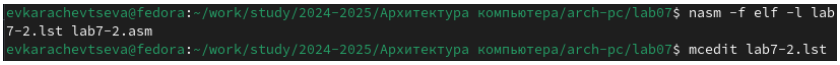

Рис. 2.10: Файл листинга lab7-2.lst

10) Проанализировав файл, я поняла как он работает и какие значения выводит.

11) Эта строка находится на 21 месте, ее адрес “00000101”, Машинный код - B8 [0A000000], а mov eax,B - исходный текст программы, означающий что в регистр eax мы вносим значения переменной B.

Рис. 2.11: Объяснения первой строки

2) Эта строка находится на 35 месте, ее адрес “00000135”, Машинный код - E862FFFFFF, a call atoi - исходный текст программы, означающий что символ лежащий в строке выше переводится в число.

Рис. 2.12: Объяснения второй строки

3) Эта строка находится на 47 месте, ее адрес “00000163”, Машинный код - A1[00000000], a mov eax,[max] - исходный текст программы, означающий что число хранившееся в переменной max записывается в регистр eax.

Рис. 2.13: Объяснения третьей строки

11) В строке mov eax,max я убрала max и попробовал создать файл. Выдало ошибку, так как для программы нужно два операнда.

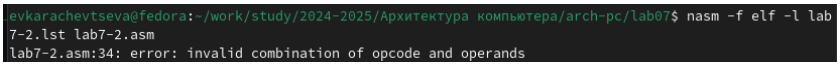

Рис. 2.14: Создание файла без одного операнда

12) В файле листинга показывает где именно ошибка и с чем она связана.

Рис. 2.15: Файл листинга без одного операнда

3 Самостоятельная работа.

1) Я написала программу для нахождения меньшего из трех чисел. Для большего удобства я сделала ввод чисел с клавиатуры.У меня первый вариант поэтому числа были :81,22,72. Программа вывела меньшее из этих чисел.

```
%include 'in_out.asm'

SECTION .data
A1 DB 'Введите число A: ',0h
B1 DB 'Введите число B: ',0h
C1 DB 'Введите число C: ',0h
otv DB 'Наименьшее число: ',0h
SECTION .bss
min RESB 20
A RESB 20
B RESB 20
C RESB 20

SECTION .text
GLOBAL _start
_start:

mov eax,A1
call sprint

mov ecx,A
mov edx,20
call sread

mov eax, A
call atoi
mov [A],eax

xor eax,eax

mov eax,B1
call sprint

mov ecx,B
mov edx,20
call sread

mov eax,B
call atoi
mov [B],eax

xor eax,eax

mov ecx, [A]
```

Рис. 3.1: Текст программы

```
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf 2.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o
2 2.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./2
Введите число A: 81
Введите число B: 22
Введите число C: 72
Наименьшее число: 22
```

Рис. 3.2: Результат работы программы

2) Я написала программу, чтобы она вычисляла выражение при введенных X и A. Для большего удобства, выражение которое будет вычисляться я выводжу вначале работы программы. Так как у меня 14 вариант, то программа написана для 14 варианта.

```
 GNU nano 7.2                                  /home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/3.asm
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data
prim1 DB '3a+1, x<a' ,0
prim2 DB '3x+1, x=>a',0
X1 DB 'Введите значение X:',0
A1 DB 'Введите значение a:',0
otv DB 'Ответ: ',0

SECTION .bss
X RESB 20
A RESB 20
F RESB 20
SECTION .text
GLOBAL _start
_start:

mov eax,prim1
call sprintLF
mov eax,prim2
call sprintLF

mov eax,X1
call sprint

mov ecx,X
mov edx,10
call sread

mov eax,X
call atoi
mov [X],eax

mov eax,A1
call sprint

mov ecx,A
mov edx,10
call sread

mov eax,A
call atoi
mov [A],eax

mov ecx,[X]
mov [F],ecx

cmp ecx,[A]
```

Рис. 3.3: Текст программы

```
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf 3.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o 3 3.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./3
3a+1, x<a
3x+1, x=>a
Введите значение X:2
Введите значение a:3
Ответ: 10
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./3
3a+1, x<a
3x+1, x=>a
Введите значение X:4
Введите значение a:2
```

Рис. 3.4: Проверка работы программы

4 Вывод

Я изучила команды условного и безусловного перехода. Приобрела навыки написания программ с переходами.