Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Хрисанова Ксения Олеговна

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. Выполнение заданий для самостоятельной 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).

Рис. 1: Создание каталога и файла для программы

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

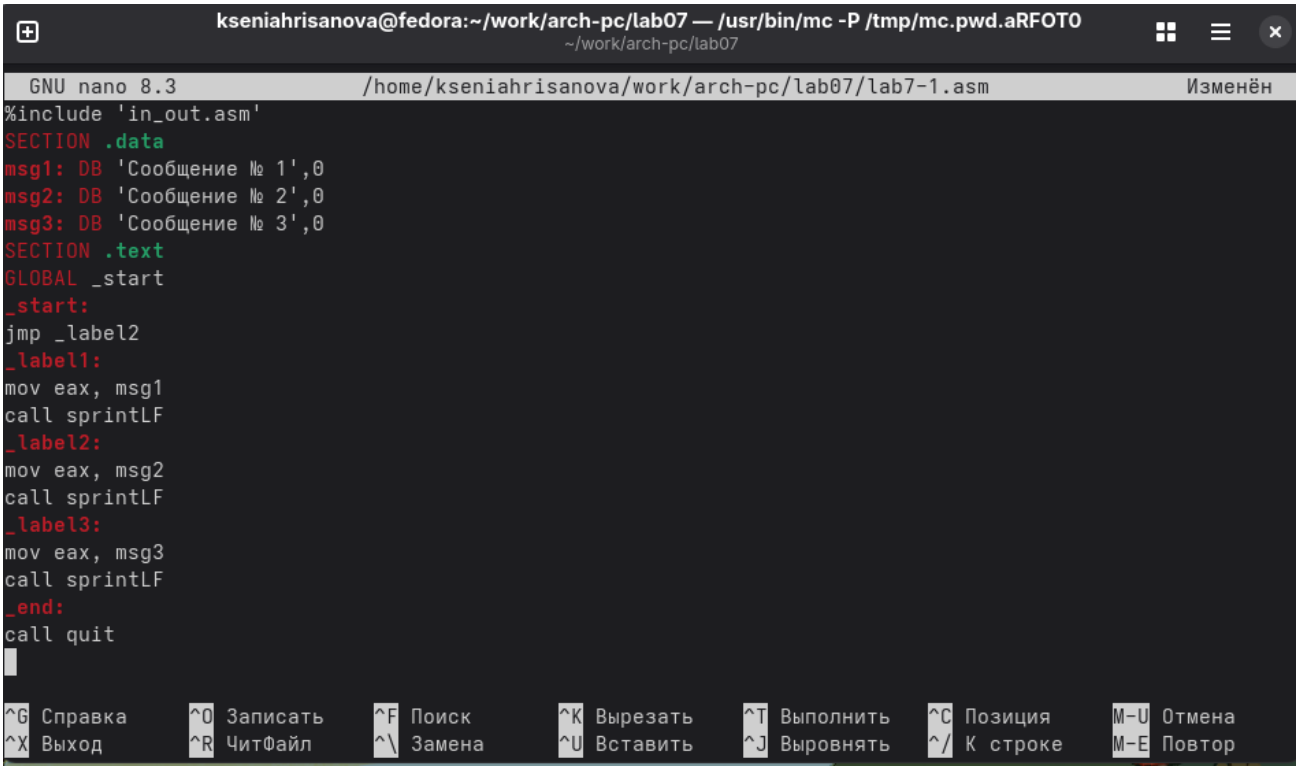

Рис. 2: Сохранение программы

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

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

Рис. 3: Запуск программы

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

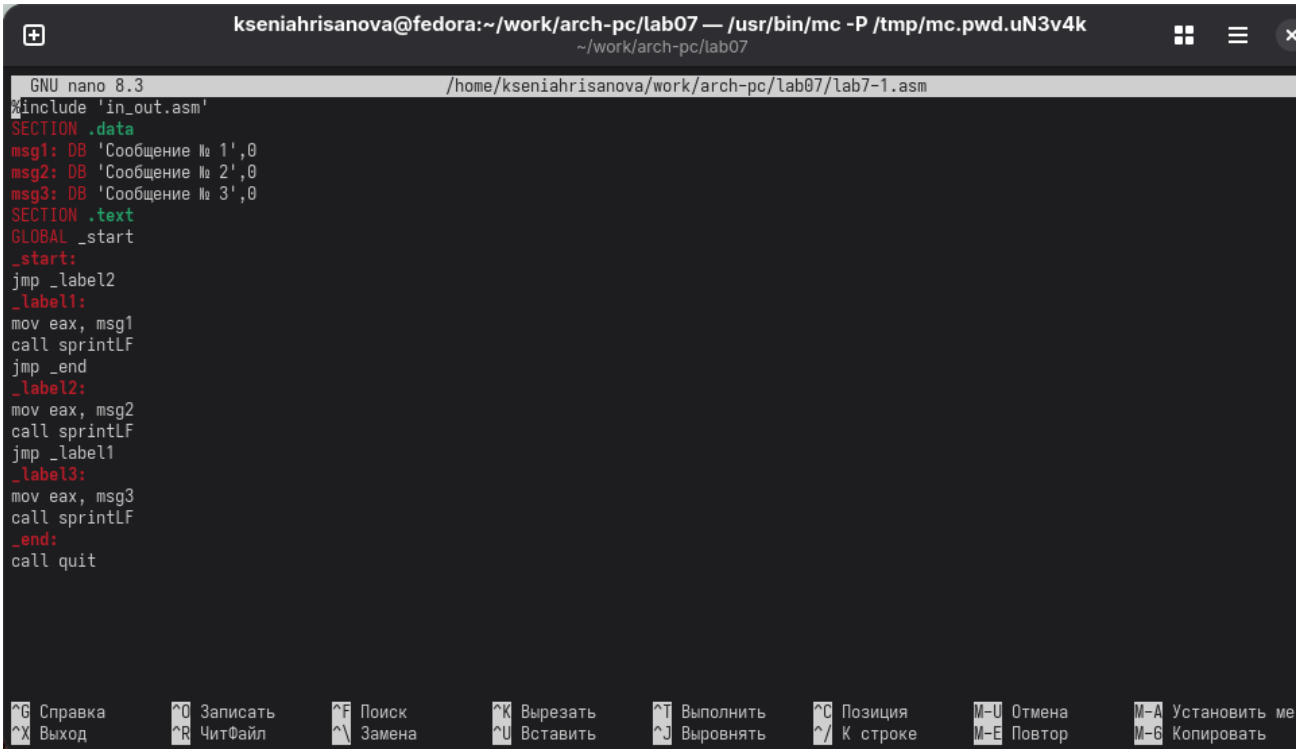

Рис. 4: Изменение программы

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

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

Рис. 5: Запуск измененной программы

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

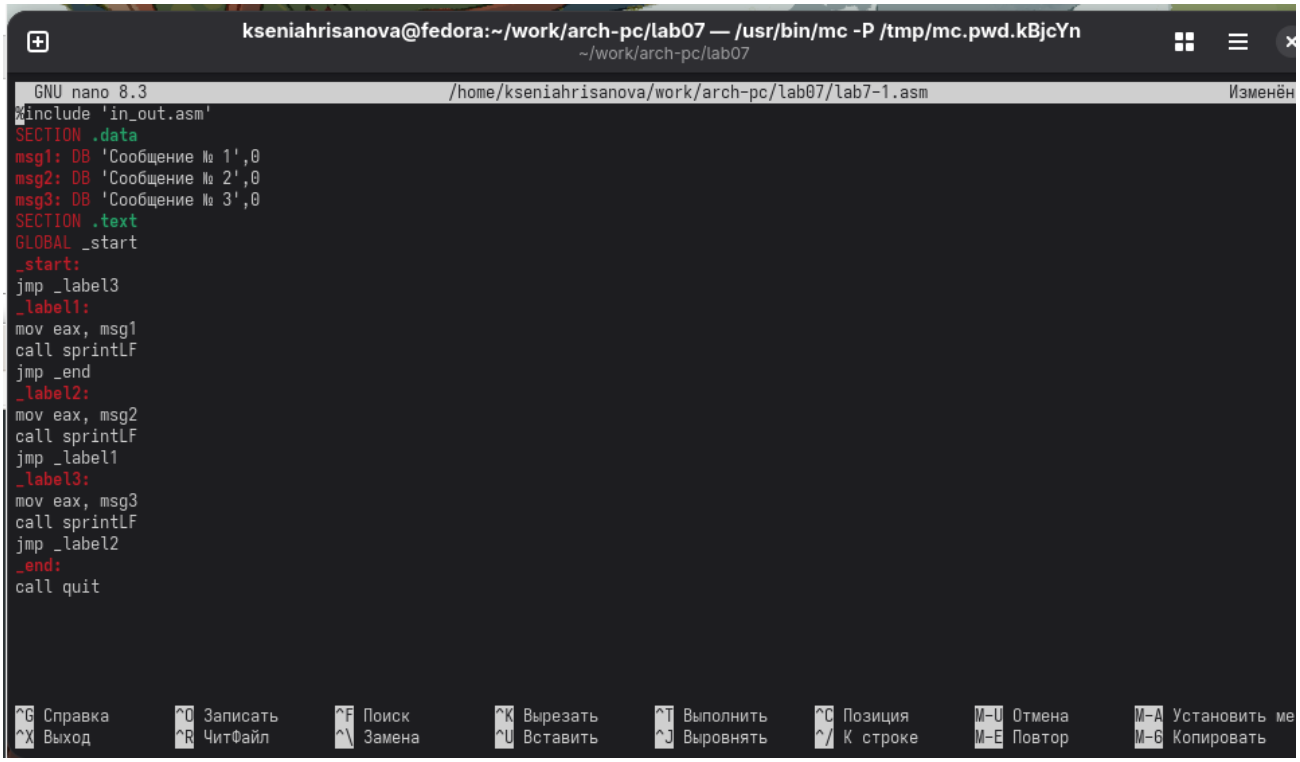

Рис. 6: Изменение программы

Программа работает корректно (рис. 7).

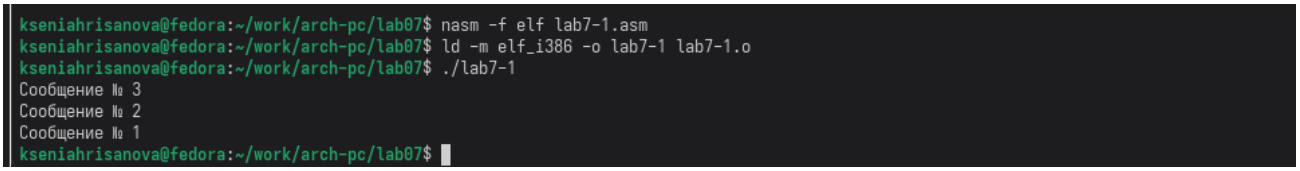

Рис. 7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из листинга 7.3 (рис. 8).

Рис. 8: Сохранение новой программы

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

```
kseniahrisanova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 70
Наибольшее число: 70
kseniahrisanova@fedora:~/work/arch-pc/lab07$
```

рис. 9: Проверка программы 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).

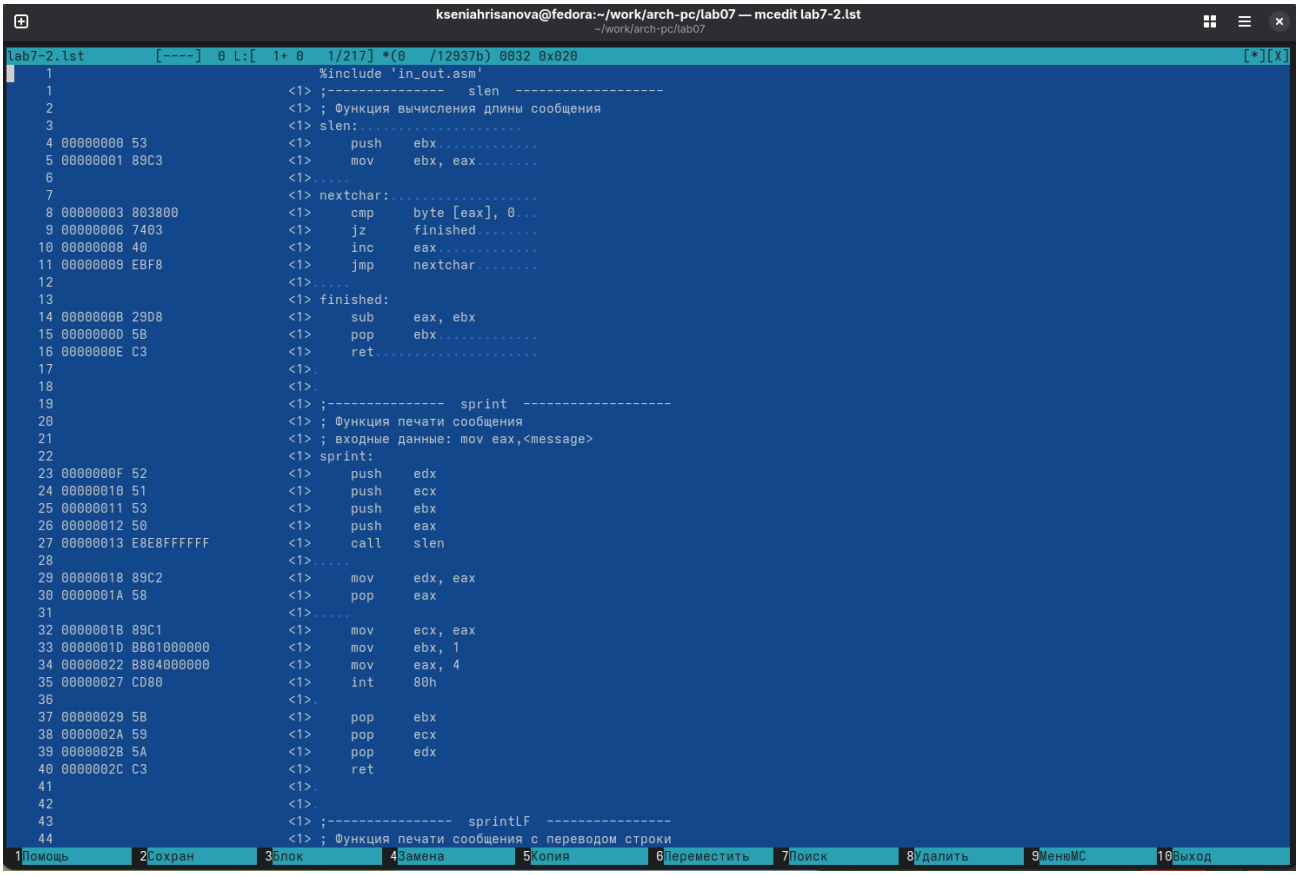

Рис. 10: Проверка файла листинга

Файл листинга NASM представляет собой расширенный вариант исходного текста программы, в котором для каждой строки указаны номер строки, смещение машинного кода, байты сгенерированных инструкций и исходная команда. Листинг позволяет анализировать соответствие команд ассемблера и результата их трансляции.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 11).

Рис. 11: Удаление операнда из программы

Создаётся только файл листинга lab7-2.lst, объектный файл не формируется из-за ошибки. В листинге у строки с испорченной инструкцией появляется сообщение об ошибке ассемблера (текст error: . . . после исходной строки).

(рис. 12).

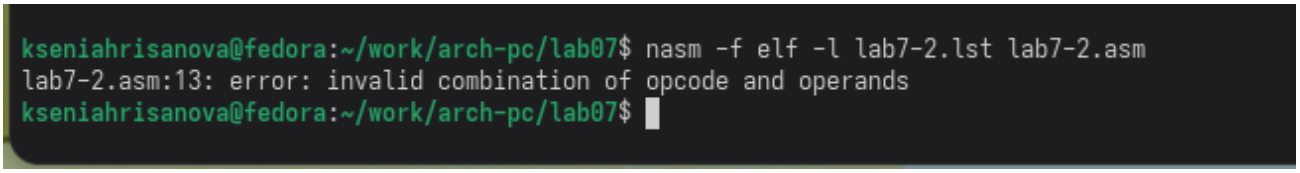

Рис. 12: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 13).

Рис. 13: программа самостоятельной работы

Текст первой программы:
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '17'
C dd '45'
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
```

Проверяю работу первой программы (рис. 14).

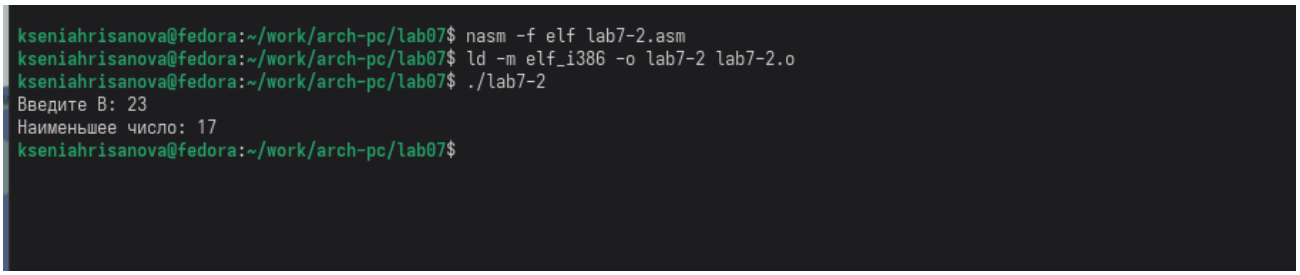

Рис. 14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатурых переменных a и x (рис. 15).

Рис. 15: Вторая программа самостоятельной работы

Текст второй прграммы:

```
%include 'in_out.asm'
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
msg_f: DB 'Результат f(x): ', 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
  jl less_than_a
  mov eax, 8
  jmp print_answer
 less_than_a:
  mov eax, esi
  shl eax, 1
  sub eax, edi
print_answer:
  mov ebx, eax
 mov eax, msg_f
  call sprint
```

Транслирую и компоную файл, запускаю и проверяю работу программмы для значений a и x, которые даны в задании (рис. 16).

```
kseniahrisanova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 1
Введите значение переменной a: 2
Результат f(x): 3
kseniahrisanova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 2
Введите значение переменной a: 1
Результат f(x): 8
kseniahrisanova@fedora:~/work/arch-pc/lab07$
```

Рис. 16: Проверка работы второй программы

5 Выводы

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, приобрела навыки написания программ с использованием переходов и познакомилась с назначением и структурой файлов листинга.

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7