РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: Архитектура компьютера

Студент:Безлепкина Татьяна Игоревна

Группа:НКАбд-01-25

МОСКВА

2025 г.

Оглавление

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Порядок выполнения лабораторной работы

2.1. Реализация переходов в NASM

Создам каталог для программам лабораторной работы № 7, перейду в него и создам файл lab7-1.asm:

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab06$ mkdir ~/work/arch-pc/lab07
tibezlepkina1@localhost-live:~/work/arch-pc/lab06$ cd ~/work/arch-pc/lab07
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рисунок 2.1.1 - создание каталога,переход в него,создание файла

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nano lab7-1.asm
```

Рисунок 2.1.2 - переход в текстовый редактор nano

```
 GNU nano 8.3                                    lab7-1.asm
%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'
```

Рисунок 2.1.3 - ввод программы в текстовый редактор nano

Попытаемся скомпилировать файл,что в итоге не выйдет,так как возникает ошибка.Ошибка означает, что компилятор не нашел файл in_out.asm в текущей директории.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
lab7-1.asm:1: error: unable to open include file `in_out.asm': No such file or directory
```

Рисунок 2.1.4 - попытка компиляции файла

Чтобы решить проблему, найдем файл в системе.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ find / -name "in_out.asm" 2>/dev/null
/home/tibezlepkina1/work/arch-pc/lab06/in_out.asm
/home/tibezlepkina1/work/arch-pc/lab05/in_out.asm
/run/overlayfs/home/tibezlepkina1/work/arch-pc/lab06/in_out.asm
/run/overlayfs/home/tibezlepkina1/work/arch-pc/lab05/in_out.asm
```

Рисунок 2.1.5 - поиск файла «in_out.asm» в системе

Скопируем файл «in_out.asm» в текущую директорию.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ cp /home/tibezlepkina1/work/arch-pc/lab06/in_out.asm ./
```

Рисунок 2.1.6 - копирование файла «in_out.asm»

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рисунок 2.1.7 - компиляция,линковка,запуск.

Инструкция jmp (прыжок) позволяет изменить стандартный порядок выполнения программы. С ее помощью можно "перепрыгнуть" через одни инструкции и начать выполнение с других.

В данной задаче нужно изменить программу так, чтобы:

Сначала выводилось "Сообщение № 2", потом "Сообщение № 1", затем программа завершалась.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nano lab7-1.asm
```

Рисунок 2.1.8 - переход в текстовый редактор nano

```
 GNU nano 8.3                                          lab7-1.asm
%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
```

Рисунок 2.1.9 - изменение текста программы в редакторе nano

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рисунок 2.1.10 - компиляция,линковка,запуск

Чтобы получить вывод в порядке "Сообщение № 3", "Сообщение № 2", "Сообщение № 1", нужно изменить инструкции перехода:

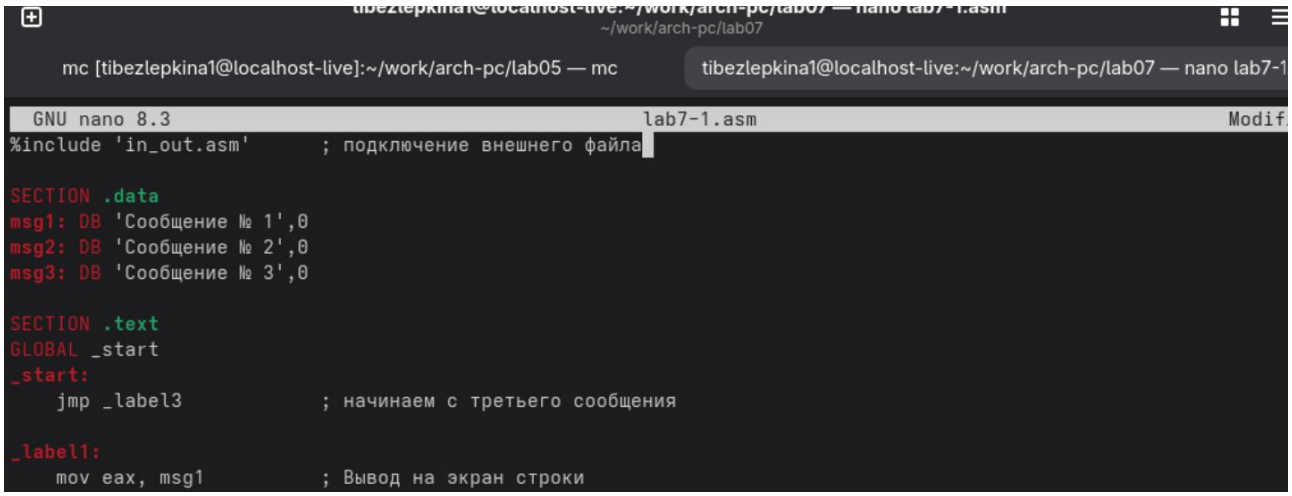

Рисунок 2.1.11 - изменение текста программы.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рисунок 2.1.12 - компиляция,линковка,запуск.

Инструкция jmp выполняет безусловный переход, который происходит всегда. Однако в программировании часто требуется реализовать условные переходы - когда переход выполняется только при выполнении определенного условия.В качестве примера рассмотрим программу, которая находит и выводит наибольшее число из трех целочисленных переменных A, B и C:

Значения переменных A и C задаются непосредственно в коде программы

Значение переменной B вводится пользователем с клавиатуры

Программа сравнивает все три значения и определяет максимальное

Таким образом, вместо простых безусловных переходов здесь потребуются переходы, зависящие от результатов сравнения чисел.

Создам файл lab7-2.asm в каталоге ~/work/arch-pc/lab07.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ touch lab7-2.asm
```

Рисунок 2.1.13 - создание файла lab7-2.asm

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nano lab7-2.asm
```

Рисунок 2.1.14 - переход в текстовый редактор nano

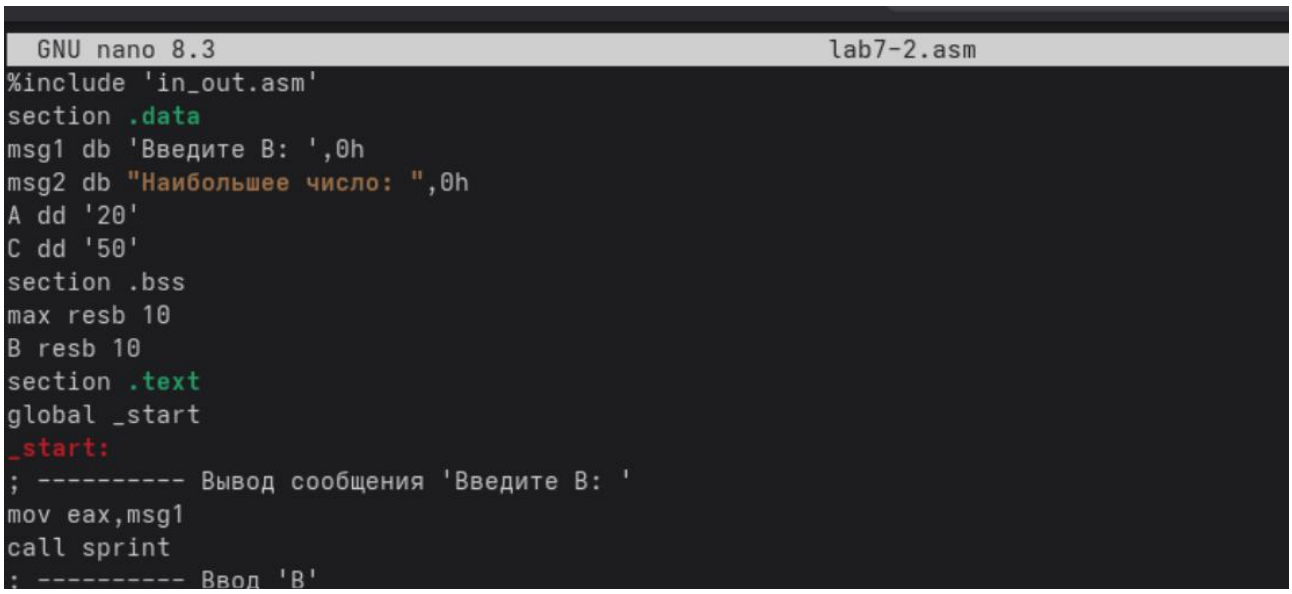

Рисунок 2.1.15 - изменение текста программы

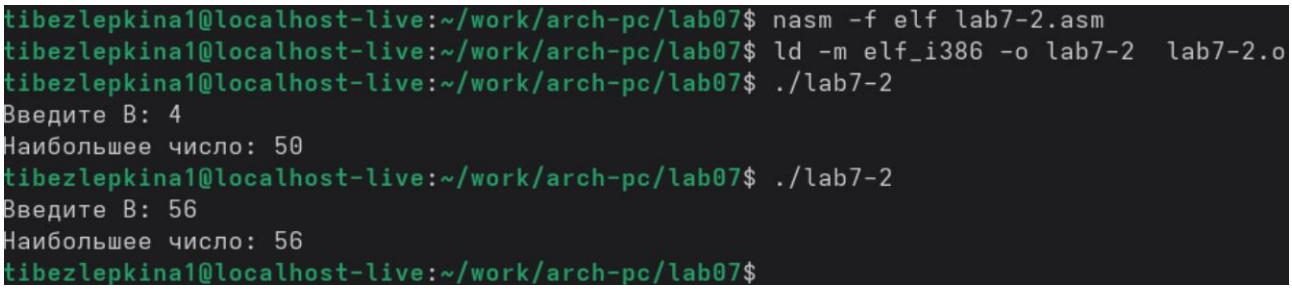

Рисунок 2.1.16 - компиляция,линковка,запуск

2.2 Изучение структуры файлы листинга

Создам файл листинга для программы из файла lab7-2.asm.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рисунок 2.2.1 - создание файла листинга

Открою файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit:

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рисунок 2.2.2 - открытие файла листинга

Подробно объясню содержимое трёх строк файла листинга по выбору.

Возьму строки:14,15,19:

14 000000E8 B8[00000000] mov eax,msg1

Строка 14: Загрузка адреса сообщения:

mov eax,msg1 - загружает адрес строки "Введите B: " в регистр EAX,

B8 - машинный код инструкции MOV EAX,

[00000000] - адрес строки msg1 в памяти,

После этой команды EAX указывает на начало строки для вывода.

15 000000ED E81DFFFFFF call sprint

Строка 15: Вызов функции вывода:

call sprint - вызов функции, которая выводит строку на экран,

E81DFFFFFF - машинный код для перехода к функции sprint,

Функция берет строку из EAX и выводит её,

После выполнения возвращается обратно к следующей инструкции.

19 000000FC E842FFFFFF call sread

Строка 19: Вызов функции ввода:

call sread - вызов функции для чтения ввода с клавиатуры,

E842FFFFFF - машинный код для перехода к функции sread,

Функция читает данные в буфер B (указанный в ECX),

Пользователь вводит значение переменной B.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nano lab7-2.asm
```

Рисунок 2.2.3 - переход в текстовый редактор nano

Внесу ошибку,удалю один операнд:

Изменю строку mov eax, msg1 на mov eax

Выполню трансляцию с получением файла листинга.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
```

Рисунок 2.2.4 - выполнение трансляции с получением листинга.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ls -la lab7-2.*
-rw-r--r--. 1 tibezlepkina1 tibezlepkina1  1737 Nov 23 10:48 lab7-2.asm
-rw-r--r--. 1 tibezlepkina1 tibezlepkina1 14542 Nov 23 10:49 lab7-2.lst
```

Рисунок 2.2.5 - проверка создания файлов

Создаваемые файлы:

Файл листинга lab7-2.lst - создается.

(Создается до проверки синтаксических ошибок, содержит весь исходный код с адресами и машинными кодами, но машинный код для ошибочной строки будет отсутствовать).

Объектный файл lab7-2.o - не создается(не создается из-за ошибки компиляции, трансляция прерывается на этапе синтаксического анализа).

Исполняемый файл - не создается(не создается, так как нет объектного файла для линковки)

Просмотрим листинг:

```
                                %include 'in_out.asm'
    1                       <1> ;---------------   slen  ----
    2                       <1> ; Функция вычисления длины со
    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
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nano lab7-2.lst
```

Рисунок 2.2.6 - просмотр листинга

В листинге добавляется:

Строки до ошибки имеют полную информацию (адреса, машинные коды), строка с ошибкой показывается, но машинный код не генерируется,строки после ошибки также показываются, но без машинных кодов,объектный код не создается из-за ошибки.

3 Задания для самостоятельной работы

1) Напишу программу нахождения наименьшей из 3 целочисленных переменных a,b и . Значения переменных выберу из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создам исполняемый файл и проверю его работу.

Напишу программу для 20 варианта:

Создам файл lab7-3.asm и перейду в тестовый редактор nano.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ touch lab7-3.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nano lab7-3.asm
```

Рисунок 3.1 - создание файла и переход в текстовый редактор

```
 GNU nano 8.3                                  lab7-3.asm
%include 'in_out.asm'

SECTION .data
msg db "Наименьшее число: ",0h
a dd 95
b dd 2
c dd 61

SECTION .text
GLOBAL _start
_start:

; ---------- Записываем 'a' в переменную 'min'
mov ecx,[a]      ; 'ecx = a'
mov [min],ecx    ; 'min = a'
```

Рисунок 3.2 -написание текста программы

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 2
```

Рисунок 3.3 - компиляция,линковка,запуск

2) Напишу программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выберу из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создам исполняемый файл и проверю его работу для значений x и a из 7.6.

Напишу программу для 20 варианта:

Для начала создам файл lab7-4.asm и перейду в текстовый редактор.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ touch lab7-4.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nano lab7-4.asm
```

Рисунок 3.4 - создание файла и переход в редактор

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab05 ×  tibezlepkina1@localhost-live:~
  GNU nano 8.3                                  lab7-4.asm
%include 'in_out.asm'

SECTION .data
msg_x db "Введите x: ",0h
msg_a db "Введите a: ",0h
msg_res db "Результат f(x): ",0h

SECTION .bss
x resb 10
a resb 10
res resb 10

SECTION .text
```

Рисунок 3.5 - написание текста программ

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 2
Результат f(x): 5
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 1
Результат f(x): 1
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$
```

Рисунок 3.6 - компиляция,линковка,запуск

Загрузим данные отчета на github.

Вывод:

В ходе лабораторной работы №7 были изучены команды условных и безусловных переходов в ассемблере NASM. На практике освоены инструкции jmp для безусловных переходов и jg, jl, je для условных переходов на основе анализа флагов процессора. Разработаны программы с ветвлениями, включая поиск наибольшего и наименьшего из трех чисел, а также вычисление значений кусочной функции. Изучена структура файла листинга, создаваемого транслятором NASM, и проведен анализ машинного кода. Полученные навыки позволяют эффективно реализовывать сложную логику ветвлений в низкоуровневом программировании и использовать листинги для отладки программ.

5 Список литературы.

1. https://esystem.rudn.ru/mod/resource/view.php?id=1030495
2. https://esystem.rudn.ru/mod/page/view.php?id=1030492
3. https://esystem.rudn.ru/mod/resource/view.php?id=1030496
4. https://esystem.rudn.ru/mod/resource/view.php?id=1030548