Отчет по лабораторной работе №7

Иванова Анастасия Сергеевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить команды условного и безусловного перехода. Приобрести навыки написания программ с использованием переходов. Познакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

1. Создадим каталог для программам лабораторной работы № 7, перейдем в него и создадим файл lab7-1.asm рис. 2.1:

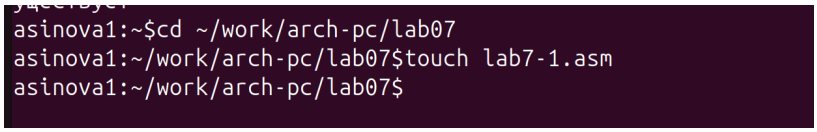

Рисунок 2.1: Создание каталога и файла

2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введем в файл lab7-1.asm текст программы 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
jmp _end

label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рисунок 2.2: Ввод программы

Создайте исполняемый файл и запустите его рис. 2.3:

```
asinova1:~/work/arch-pc/lab07$nasm -f elf lab7-1.asm
asinova1:~/work/arch-pc/lab07$ld -m elf_i386 -o lab7-1 lab7-1.o
asinova1:~/work/arch-pc/lab07$./lab7-1
Сообщение № 2
Сообщение № 3
```

Рисунок 2.3: Проверка работы программы

Таким образом, использование инструкции jmp _label2 меняет порядок испол-

нения инструкций и позволяет выполнить инструкции начиная с метки _label2, пропустив вывод первого сообщения.

Инструкция jmp позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала „Сообщение № 2“, потом „Сообщение № 1“ и завершала работу. Изменим текст программы рис. 2.4:

```
%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
jmp _end

label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рисунок 2.4: Изменение программы

Создадим исполняемый файл и проверим его работу рис. 2.5:

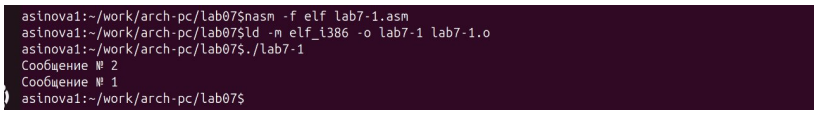

Рисунок 2.5: Проверка работы программы

Изменим текст программы, добавив jmp, чтобы вывод программы был следующим: user@dk4n31:~$ ./lab7-1 Сообщение № 3 Сообщение № 2 Сообщение № 1 user@dk4n31:~$

рис. 2.6:

```
%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
```

Рисунок 2.6: Изменение программы

рис. 2.7:

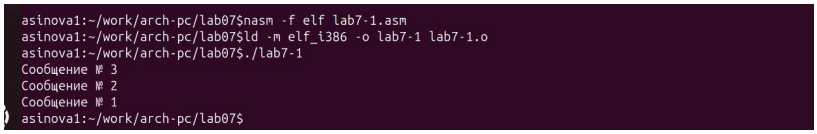

Рисунок 2.7: Изменение программы

рис. 2.8:

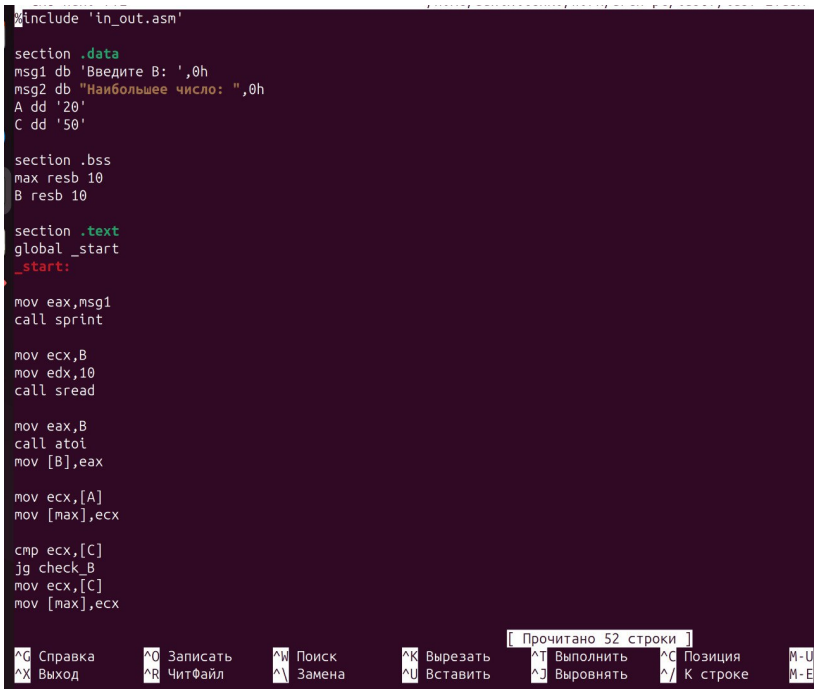

Рисунок 2.8: Проверка работы программы

3. Использование инструкции jmp приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход дол- жен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. Значения для A и C задаются в программе, значение B вводиться с клавиатуры. Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучим текст программы и введем в lab7-2.asm рис. 2.11

```
/home/aantkitenko/work/arch-pc/lab07/lab7-2.lst   [----]  0 L:[  1+ 0   1/228] *(0    /13388b) 0032 0x020
   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>.
```

Рисунок 2.9: Ввод программы

Создадим исполняемый файл и проверим его работу для разных значений B рис. 2.10:

```
asinova1:~/work/arch-pc/lab07$nasm -f elf lab7-2.asm
asinova1:~/work/arch-pc/lab07$ld -m elf_i386 -o lab7-2 lab7-2.o
asinova1:~/work/arch-pc/lab07$.\lab7-2
.lab7-2: команда не найдена
asinova1:~/work/arch-pc/lab07$./lab7-2
Введите B: 25
Наибольшее число: 50
asinova1:~/work/arch-pc/lab07$./lab7-2
Введите B: 60
Наибольшее число: 60
asinova1:~/work/arch-pc/lab07$./lab7-2
Введите B: 10
Наибольшее число: 50
asinova1:~/work/arch-pc/lab07$
```

Рисунок 2.10: Проверка работы программы

Данную программу можно упростить и сравнивать все 3 переменные как символы (т.е. не использовать функцию atoi). Однако если переменные преобразовать из символов числа, над ними можно корректно проводить арифметические операции.

4. Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Откроем файл с программой lab7-2.asm и в инструкции

с двумя операндами удалим один операнд. Выполним трансляцию с получением файла листинга рис. 2.11:

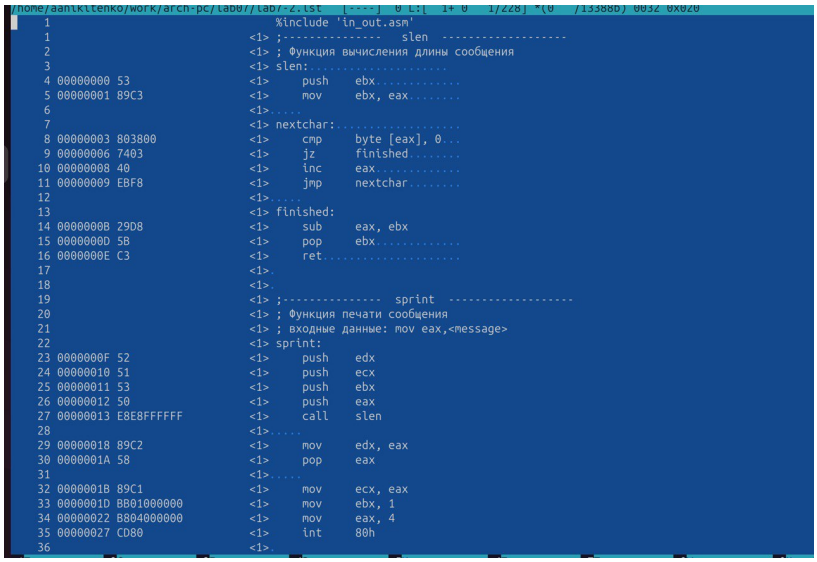

Рисунок 2.11: Просмотр содержимого файла

Объяснение строк: Первое значение в файле листинга - номер строки, и он может вовсе не совпа- дать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредствен- но идет сам машинный код, а заключает строку исходный текст прогарммы с комментариями.

3 Задание для самостоятельной работы

1. Напишем программу нахождения наименьшей из 3 целочисленных переменных a,b и . Значения переменных выбрем из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создадим исполняемый файл и проверим его работу:

рис. 3.1:

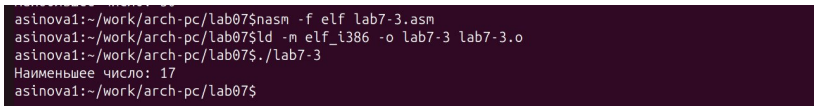

Рисунок 3.1: Выполнение задания

рис. 3.2:

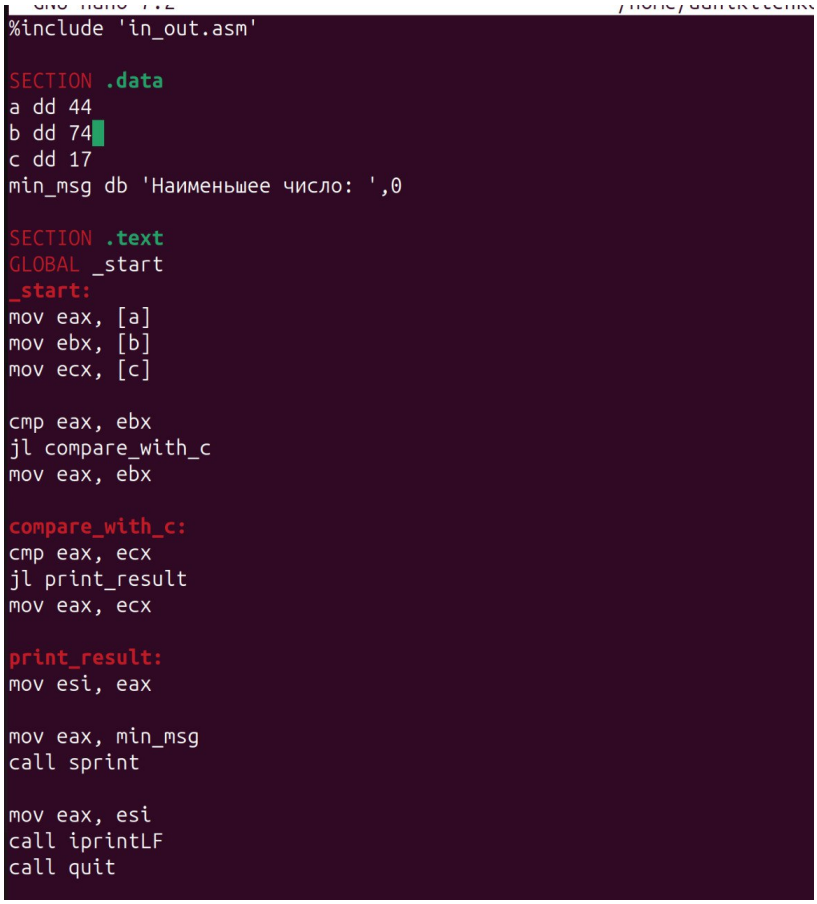

Рисунок 3.2: Выполнение задания

2. Напишем программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбрем из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создадим исполняемый файл и проверим его работу для значений x и a из 7.6:

рис. 3.3:

Рисунок 3.3: Выполнение задания

рис. 3.4:

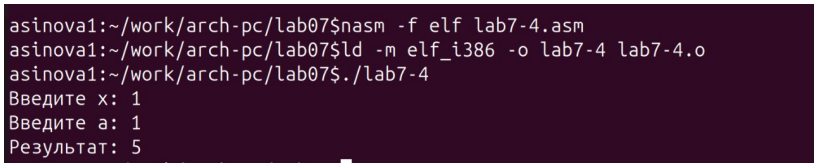

Рисунок 3.4: Выполнение задания

рис. 3.5:

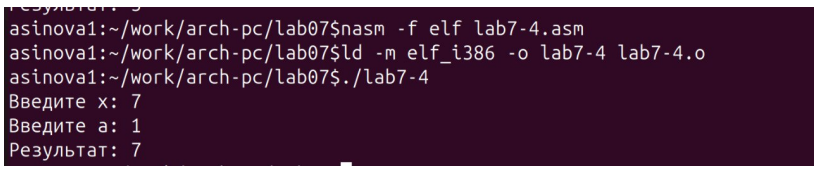

Рисунок 3.5: Выполнение задания

рис. 3.6:

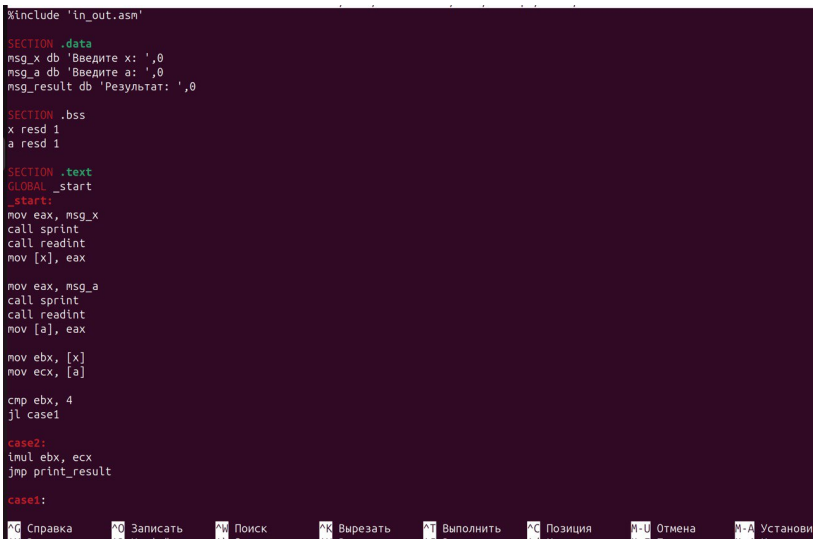

Рисунок 3.6: Выполнение задания

4 Вывод

Мы изучили команды условного и безусловного перехода. Приобрели навыки написания программ с использованием переходов. Познакомились с назначением и структурой файла листинга.

Список литературы