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: Томилова Валентина

Группа: НКАбд-06-25

МОСКВА

2025 г.

Оглавление

Список таблиц 1 Цель работы 4
2 Задания 6
3 Теоретическое введение 7
4 Выполнение работы 8
Реализация переходов в NASM 8
Изучение структуры файлы листинга 13
Объяснение строк листинга 13
Задание для самостоятельной работы 15
5 Выводы 19
Список литературы 20

Список изображений

Рисунок 0.1 (создание каталога) 7
Рисунок 0.2 (переход в каталог) 7
Рисунок 0.3 (создаем файл) 7
Рисунок 0.4 (текст программы) 7
Рисунок 0.5 (создание исполняемого файла) 8
Рисунок 0.6 (запуск исполняемого файла) 8
Рисунок 0.7 (результат работы файла) 8
Рисунок 0.8 (измененный текст программы) 8
Рисунок 0.9 (создание и запуск исполняемого файла) 9
Рисунок 0.10 (измененный текст программы) 9
Рисунок 0.11 (создание исполняемого файла) 9
Рисунок 0.12(запуск исполняемого файла) 10
Рисунок 0.13(создание файла) 10
Рисунок 0.14 (текст программы) 10
Рисунок 0.15 (текст программы) 11
Рисунок 0.16 (создание исполняемого файла и проверка работы) 11
Рисунок 0.17(создание исполняемого файла и проверка работы) 11
Рисунок 0.18(создание исполняемого файла и проверка работы) 11
Рисунок 0.19 (создание файла листинга) 12
Рисунок 0.20 (открытие листинга) 12
Рисунок 0.21 (открытие листинга) 12
Рисунок 0.22 (текст программы с удалением операнды) 13
Рисунок 0.23 (создание файла листинга) 13
Рисунок 0.24 (открытие листинга) 13
Рисунок 0.25 (текст программы) 14
Рисунок 0.26 (текст программы) 14
Рисунок 0.27 (текст программы) 14
Рисунок 0.28 (создание исполняемого файла и проверка его работы) 15
Рисунок 0.29 (текст программы) 15
Рисунок 0.30 (текст программы) 16
Рисунок 0.31 (создание исполняемого файла) 16
Рисунок 0.32 (создание исполняемого файла) 16
Рисунок 0.33 (запуск программы) 16
Рисунок 0.34 (проверка результатов) 16
Рисунок 0.35 (проверка результатов) 16

Список таблиц

1 Цель работы

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

2 Задания

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 работы

Реализация переходов в NASM

Создадим каталог для программам лабораторной работы № 7, перейдем в него и создадим файл lab7-1.asm (рисунок 0.1-0.3)

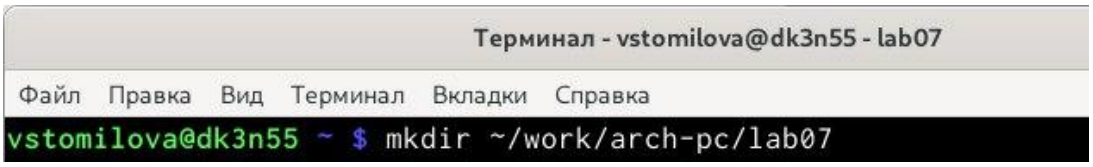

Рисунок 0.1 (создание каталога)

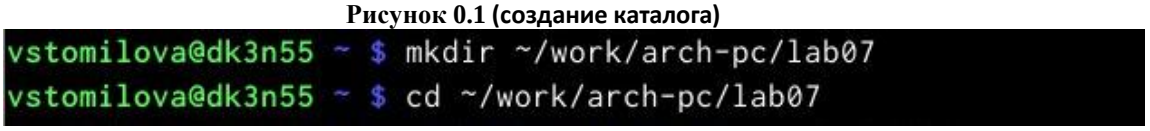

Рисунок 0.2 (переход в каталог)

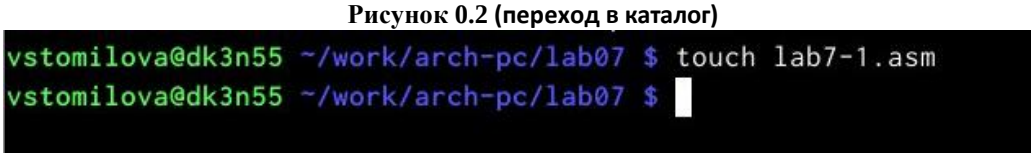

Рисунок 0.3 (создаем файл)

Введем в файл lab7-1.asm текст программы из листинга (рисунок 0.4)

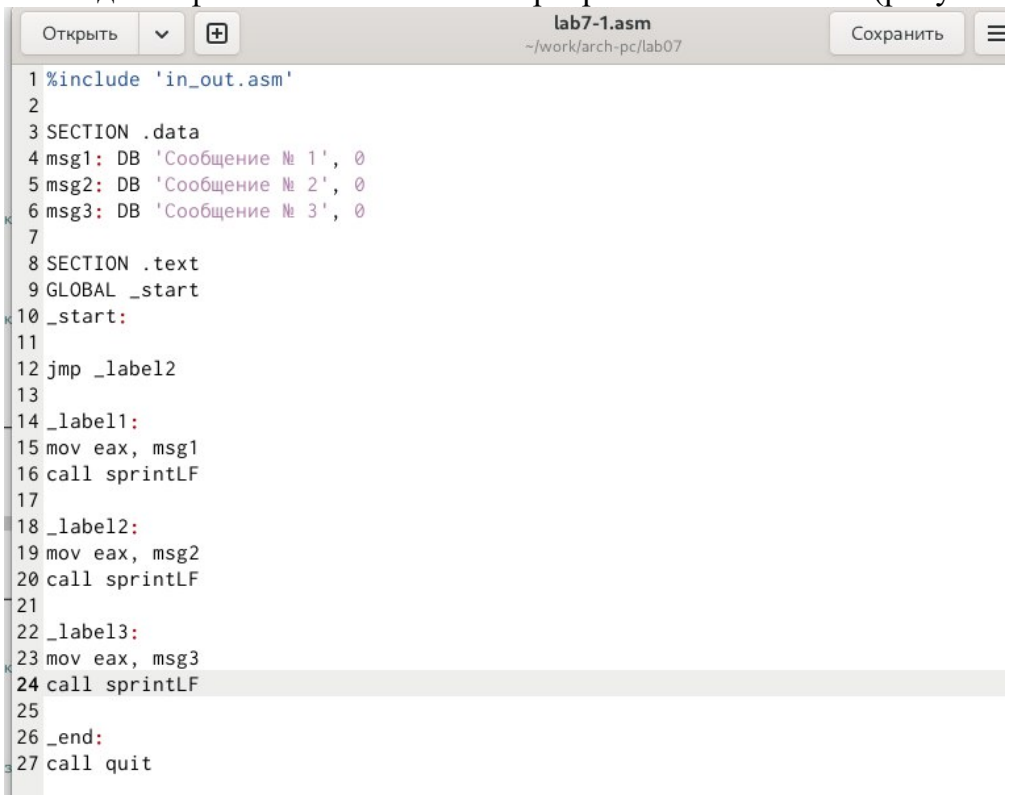

Рисунок 0.4 (текст программы)

Создадим исполняемый файл и запустим его (рисунок 0.5-0.6)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.5 (создание исполняемого файла)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
```

Рисунок 0.6 (запуск исполняемого файла)

Результат работы данной программы будет следующим (рисунок 0.7)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.7 (результат работы файла)

Изменим текст программы в соответствии с листингом (рисунок 0.8)

Открыть | lab7-1.asm ~/work/arch-pc/lab07 | Сохранить

```
%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 0.8 (измененный текст программы)

Создадим исполняемый файл и проверим его работу (рисунок 0.9)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.9 (создание и запуск исполняемого файла)

Изменим текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим (рисунок 0.10)

Открыть | lab7-1.asm ~/work/arch-pc/lab07 | Сохранить

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label3
13
14 _label1:
15 mov eax, msg1
16 call sprintLF
17 jmp _end
18
19 _label2:
20 mov eax, msg2
21 call sprintLF
22 jmp _label1
23
24 _label3:
25 mov eax, msg3
26 call sprintLF
27 jmp _label2
28
29 _end:
30 call quit
```

Рисунок 0.10 (измененный текст программы)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-1.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
```

Рисунок 0.11 (создание исполняемого файла)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.12(запуск исполняемого файла)

Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучим текст программы из листинга 7.3 и введем в lab7-2.asm (рисунок 0.13-0.15)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-2.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.13(создание файла)

Открыть | lab7-2.asm ~/work/arch-pc/lab07 | Сохранить

```
%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'
```

Рисунок 0.14 (текст программы)

```
; ---------- 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:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рисунок 0.15 (текст программы)

Создадим исполняемый файл и проверим его работу для разных значений B (рисунок 0.16-0.18)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-2.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 25
Наибольшее число: 50
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.16 (создание исполняемого файла и проверка работы)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 80
Наибольшее число: 80
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.17(создание исполняемого файла и проверка работы)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 5
Наибольшее число: 50
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.18(создание исполняемого файла и проверка работы)

Изучение структуры файлы листинга

Создадим файл листинга для программы из файла lab7-2.asm (рисунок 0.19-

0.21)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.19 (создание файла листинга)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst
```

Рисунок 0.20 (открытие листинга)

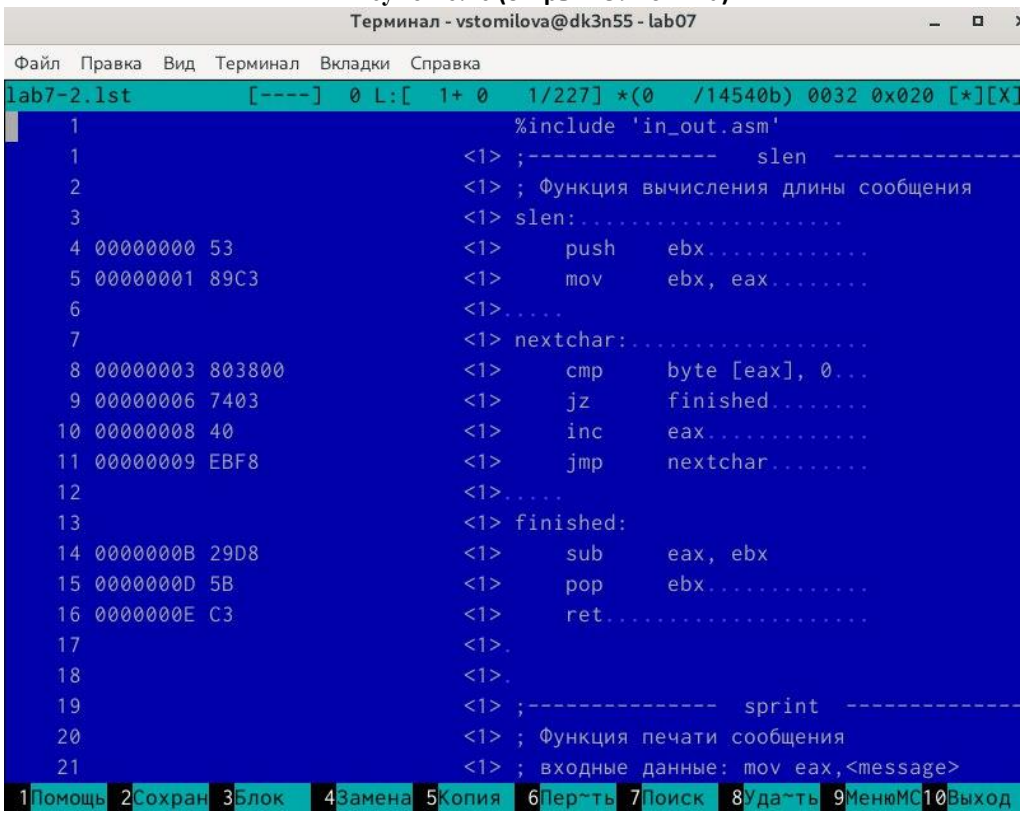

Рисунок 0.21 (открытие листинга)

Объяснение строк листинга:

Самое первое значение в файле листинга – номер строки—это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы)

Второе вхождение- адрес—это смещение машинного кода от начала текущего сегмента

Далее идут машинный код, представляющий собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, и исходный текст программы — это просто строка исходной программы вместе с комментариями

Откроем файл с программой lab7-2.asm и в любой инструкции с двумя операндами

удалим один операнд. Выполним трансляцию с получением файла листинга (рисунок 0.22-0.24)

```
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:
mov eax,
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рисунок 0.22 (текст программы с удалением операнды)

В 47 строке удалим один операнд

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-2.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:47: error: invalid combination of opcode and operands
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.23 (создание файла листинга)

```
45                                      ; ---------- Вывод результата
46                                      fin:
47                                      mov eax,
47                ******************     error: invalid combination of opcode an
48 00000159 E8B1FEFFFF                 call sprint ; Вывод сообщения 'Наибольше
49 0000015E A1[00000000]                mov eax,[max]
50 00000163 E81EFFFFFF                  call iprintLF ; Вывод 'max(A,B,C)'
51 00000168 E86EFFFFFF                  call quit ; Выход

1Помощь 2Сохран 3Блок   4Замена 5Копия  6Пер~ть 7Поиск  8Уда~ть 9МенюMC 10Выход
```

Рисунок 0.24 (открытие листинга)

В новом файле листинга показывает ошибку, возникшую при попытке трансляции файла. При этом никакие выходные файлы, помимо файла листинга, не появляются

Задание для самостоятельной работы

Напишем программу нахождения наименьшей из 3 целочисленных

переменных *a*, *b* и c. Значения переменных 95,2,61 (рисунок 0.25-0.27)

Открыть *lab7-3.asm ~/work/arch-pc/lab07

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd 95
C dd 61

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
```

Рисунок 0.25 (текст программы)

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

mov eax, [A]
mov [min], eax

mov eax, [min]
cmp eax, [C]
jle check_B
mov eax, [C]
mov [min], eax

```

Рисунок 0.26 (текст программы)

```
check_B:
mov eax, [min]
cmp eax, [B]
jle print_result
mov eax, [B]
mov [min], eax

print_result:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF

call quit
```

Matlab Ширина та

Рисунок 0.27 (текст программы)

Создайте исполняемый файл и проверьте его работу (рисунок 0.28)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-3.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-3.lst lab7-3.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-3
Введите B: 2
Наименьшее число: 2
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.28 (создание исполняемого файла и проверка его работы)

Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ (x-a,x>=a;

5, x<a)(рисунок 0.29-0.30)

Открыть | *lab7-4.asm ~/work/arch-pc/lab07 | Сохранить

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg_x: DB 'Введите значение переменной x: ', 0
4 msg_a: DB 'Введите значение переменной a: ', 0
5 res: DB 'Результат: ', 0
6 SECTION .bss
7 x: RESB 80
8 a: RESB 80
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg_x
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax, x
18 call atoi
19 mov edi, eax
20
21 mov eax, msg_a
22 call sprint
23 mov ecx, a
24 mov edx, 80
25 call sread
26 mov eax, a
27 call atoi
28 mov esi, eax
```

Рисунок 0.29 (текст программы)

```
29
30 cmp edi, esi
31 jl less_case
32 mov eax, edi
33 sub eax, esi
34 jmp print_result
35
36 less_case:
37 mov eax, 5
38
39 print_result:
40 mov edi, eax
41 mov eax, res
42 call sprint
43 mov eax, edi
44 call iprintLF
45 call quit
```

Рисунок 0.30 (текст программы)

Создадим исполняемый файл и проверим его работу для значений x и a (рисунок 0.31-0.35)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-4.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ gedit lab7-4.asm
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
```

Рисунок 0.31 (создание исполняемого файла)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
```

Рисунок 0.32 (создание исполняемого файла)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение переменной x:
```

Рисунок 0.33 (запуск программы)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение переменной x: 1
Введите значение переменной a: 2
Результат: 5
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.34 (проверка результатов)

```
vstomilova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение переменной x: 2
Введите значение переменной a: 1
Результат: 1
vstomilova@dk3n55 ~/work/arch-pc/lab07 $
```

Рисунок 0.35 (проверка результатов)

5 Выводы

Мы изучили команды условного и безусловного перехода, приобрели навыки написания программ с использованием переходов, ознакомились с назначением и структурой файла листинга

Список литературы

1. GDB:TheGNUProjectDebugger.—URL:https://www.gnu.org/software/gdb/.

2. GNUBashManual.—2016.—URL:https://www.gnu.org/software/bash/manual/.

3. Midnight CommanderDevelopment Center.—2021.—URL: https://midnight-commander. org/.

4. NASMAssemblyLanguageTutorials.—2021.—URL: https://asmtutor.com/.

5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 c. —(In a Nutshell). —ISBN 0596009658.—URL: http://www.amazon.com/Learning bash-Shell-Programming-Nutshell/dp/0596009658.

6. RobbinsA. Bash Pocket Reference.—O'Reilly Media,2016.—156 c.—ISBN 978-1491941591.

7. TheNASMdocumentation.—2021.—URL:https://www.nasm.us/docs.php.

8. Zarrelli G. Mastering Bash.—Packt Publishing,2017.—502 c.—ISBN 9781784396879.

9. Колдаев В. Д.,Лупин С. А.Архитектура ЭВМ.—М. : Форум,2018.

10. Куляс О. Л.,Никитин К. А. Курс программирования наASSEMBLER.—М. : Солон-Пресс, 2017.

11. НовожиловО.П.Архитектура ЭВМисистем.—М.:Юрайт,2016.

12. Расширенный ассемблер: NASM. — 2021. — URL:https://www.opennet.ru/docs/RUS/nasm/.

13. Робачевский А.,Немнюгин С.,Стесик О. Операционная система UNIX.—2-е изд.—БХВ Петербург, 2010.—656 с.—ISBN 978-5-94157-538-1.

14. СтоляровА.Программированиенаязыкеассемблера NASMдляОСUnix.—2-еизд.— М. : МАКСПресс,2011.—URL:http://www.stolyarov.info/books/asm_unix.

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

16. Таненбаум Э.,Бос Х. Современные операционные системы.—4-е изд.—СПб.: Питер, 2015. —1120 с.—(Классика Computer Science)