Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Абакумов Тимофей Александрович

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

**Порядок выполнения лабораторной работы**

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Выполнение заданий для самостоятельной работы.

# 3 Выполнение лабораторной работы

## 3.1 Реализация переходов в NASM

1. Для начала создадим каталог для программ лабораторной работы № 7, перейдём в него и создадим файл lab7-1.asm (рис. 1).

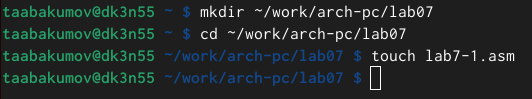


Рис. 1: Создание каталога и файла

1. Рассмотрим пример программы с использованием инструкции jmp. Введём в файл lab7-1.asm текст программы из листинга 7.1 (рис. 2).

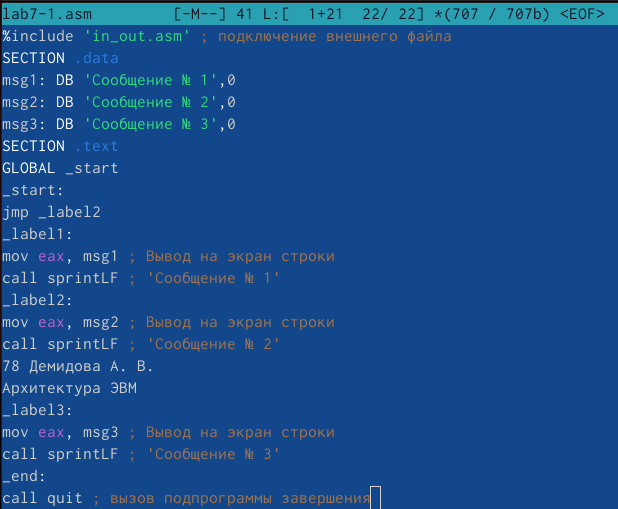


Рис. 2: Код программы

Код программы из пункта 2:

%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

1. Создадим исполняемый файл и запустим его. Результат работы данной программы будет следующим (рис. 3).

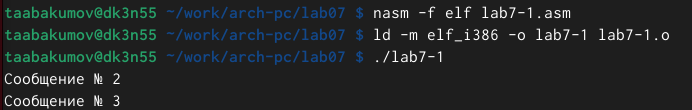


Рис. 3: Работа программы

Таким образом, использование инструкции jmp \_label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки \_label2, пропустив вывод первого сообщения.

1. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Изменим текст программы в соответствии с листингом 7.2 (рис. 4).

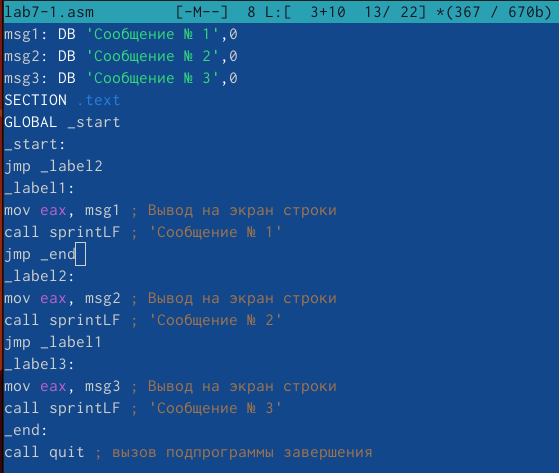


Рис. 4: Код программы

Код программы из пункта 4:

%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

\_label2:

mov eax, msg2 ; Вывод на экран строки

call sprintLF ; ‘Сообщение № 2’

jmp \_label1

\_label3:

mov eax, msg3 ; Вывод на экран строки

call sprintLF ; ‘Сообщение № 3’

\_end:

call quit ; вызов подпрограммы завершения

1. Создадим исполняемый файл и проверим его работу (рис. 5).

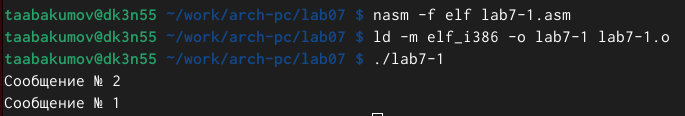


Рис. 5: Работа программы

1. Измените текст программы добавив или изменив инструкции jmp, чтобы вывод програм- мы был следующим: user@dk4n31:~$ ./lab7-1 Сообщение № 3 Сообщение № 2 Сообщение № 1 user@dk4n31:~$ (рис. 6).

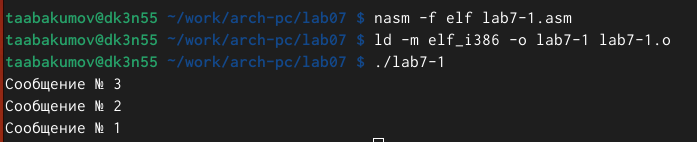


Рис. 6: Работа программы

(рис. 7).

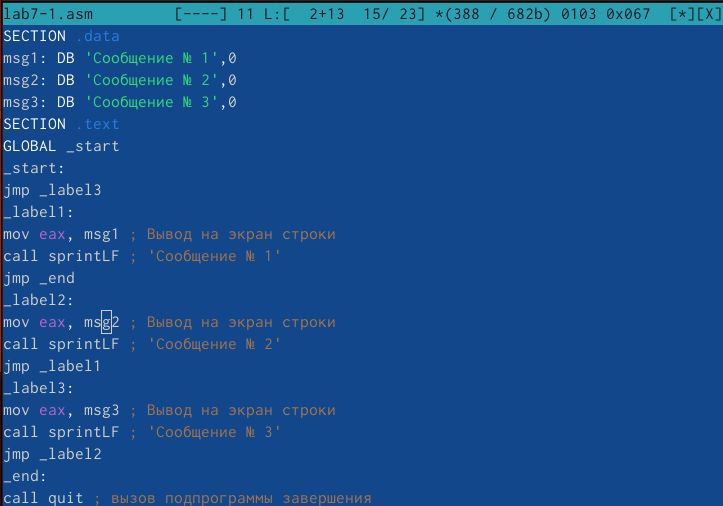


Рис. 7: Код программы

Код программы из пункта 6:

%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 \_label3

\_label1:

mov eax, msg1 ; Вывод на экран строки

call sprintLF ; ‘Сообщение № 1’

jmp \_end

\_label2:

mov eax, msg2 ; Вывод на экран строки

call sprintLF ; ‘Сообщение № 2’

jmp \_label1

\_label3:

mov eax, msg3 ; Вывод на экран строки

call sprintLF ; ‘Сообщение № 3’

jmp \_label2

\_end:

call quit ; вызов подпрограммы завершения

1. Далее создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучим текст программы из листинга 7.3 и введём в lab7-2.asm (рис. 8).

Рис. 8: Создание файла

Рис. 8: Создание файла

Код программы из пункта 7:

%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 ‘С’ (как символы)

cmp ecx,[C] ; Сравниваем ‘A’ и ‘С’

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

1. Создадим исполняемый файл и проверим его работу для разных значений (рис. 9).

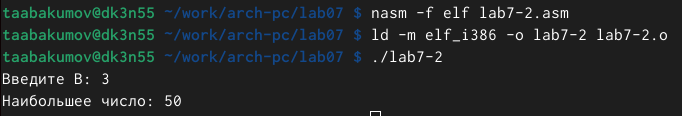


Рис. 9: Работа программы

Обратим внимание, в данном примере переменные A и С сравниваются как символы, а переменная B и максимум из A и С как числа (для этого используется функция atoi преобразования символа в число).

## 3.2 Изучение структуры файлы листинга

1. Создадим файл листинга для программы из файла lab7-2.asm, а затем откроем файл листинга lab7-2.lst с помощью текстового редактора mcedit:(рис. 10).

Рис. 10: Создание файла

Рис. 10: Создание файла

1. Откроем файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалим один операнд, после чего выполним трансляцию с получением файла листинг (рис. 11).

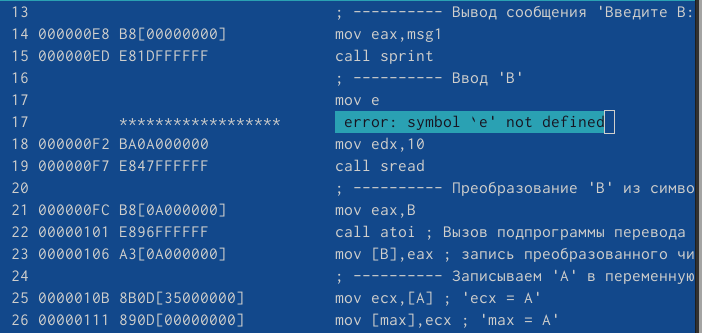


Рис. 11: Ошибка в листинге

## 3.3 Выполнение заданий для самостоятельной работы

1. 1 Задание: Написать программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы №6. Создать исполняемый файл и проверить его работу.
2. Для начала создадим файл lab7-3-1.asm. Мой вариант из прошлолй лабораторной работы - 9, соответственно программа должна выводить миниальное число среди чисел: 24,98,15 (рис. 12).

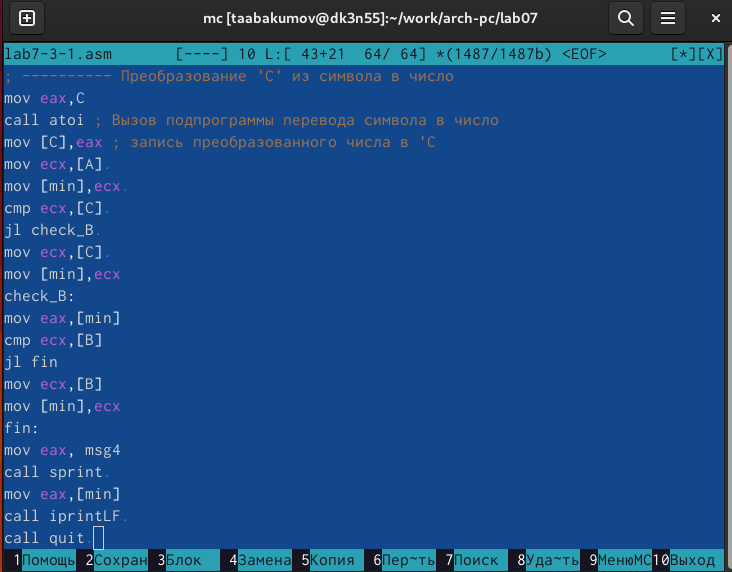


Рис. 12: Код программы

Код программы из пункта 12:

include ‘in\_out.asm’

section .data

msg1 db ‘Введите A:’,0h

msg2 db ‘Введите В:’,0h

msg3 db ‘Введите С:’,0h

msg4 db “Наименьшее число:”,0h

section .bss

min resb 10

A resb 10

B resb 10

C resb 10

section .text

global \_start

\_start:

; ———- Вывод сообщения ‘Введите A:’

mov eax,msg1

call sprint

; ———- Ввод ‘A’

mov ecx,A

mov edx,10

call sread

; ———- Преобразование ‘A’ из символа в число

mov eax,A

call atoi ; Вызов подпрограммы перевода символа в число

mov [A],eax ; запись преобразованного числа в ‘B’

; ———- Вывод сообщения ‘Введите B:’

mov eax,msg2

call sprint

; ———- Ввод ‘B’

mov ecx,B

mov edx,10

call sread

mov eax,B

call atoi

mov [B],eax

; ———- Вывод сообщения ‘Введите C:’

mov eax,msg3

call sprint

; ———- Ввод ‘C’

mov ecx,C

mov edx,10

call sread

; ———- Преобразование ‘C’ из символа в число

mov eax,C

call atoi ; Вызов подпрограммы перевода символа в число

mov [C],eax ; запись преобразованного числа в ’C

mov ecx,[A]

mov [min],ecx

cmp ecx,[C]

jl check\_B

mov ecx,[C]

mov [min],ecx

check\_B:

mov eax,[min]

cmp ecx,[B]

jl fin

mov ecx,[B]

mov [min],ecx

fin:

mov eax, msg4

call sprint

mov eax,[min]

call iprintLF

call quit

1. Создадим исполняемый файл и проверим его работу (рис. 13).

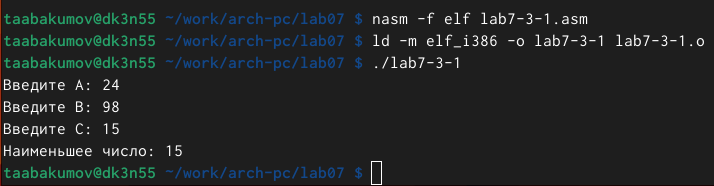


Рис. 13: Работа программы

1. 2 Задание: Написать программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. После этого необходимо создать исполняемый файл и проверьте его работу для значений x и a из 7.6.

Для начала создадим файл lab7-3-2.asm (рис. 14).

Рис. 14: Создание файла

Рис. 14: Создание файла

1. Мой вариант из прошлолй лабораторной работы - 9, соответственно при вводе числа 5, должно выводиться число 7, и точно также при вводе числа 6, должно выводиться число 4 (рис. 15).

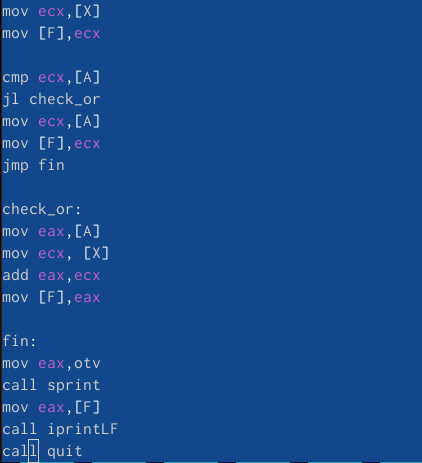


Рис. 15: Код программы

Код программы из пункта 15:

%include ‘in\_out.asm’

SECTION .data

prim1 DB ‘a+x ,x<=a’ ,0

prim2 DB ‘a, 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]

jl check\_or

mov ecx,[A]

mov [F],ecx

jmp fin

check\_or:

mov eax,[A]

mov ecx, [X]

add eax,ecx

mov [F],eax

fin:

mov eax,otv

call sprint

mov eax,[F]

call iprintLF

call quit

1. После написания программы создадим исполняемый файл (рис. 16).

Рис. 16: Создание исполняемого файла

Рис. 16: Создание исполняемого файла

1. Проверими работу программы (рис. 17).

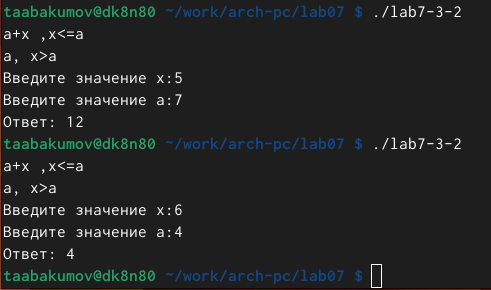


Рис. 17: Работа программы

# 4 Выводы

Были изучены основные принципы работы с условным и безусловным переходом в assembler и изучены основы чтения файлов листинга.