

Отчет по лабораторной работе №7

Простейший вариант

Чигладзе Майя Владиславовна

Содержание

1	Цель работы	5
2	Порядок выполнения лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.1.1	Задание 1	6
2.1.2	Задание 2	6
2.2	Задание 3	10
2.3	Задание 4	11
3	Задание для самостоятельной работы	13
3.1	Задание 1	13
3.2	Задание 2	15
4	Выводы	17
	Список литературы	18

Список иллюстраций

2.1	Создание каталога	6
2.2	Заполняем файл	7
2.3	Запускаем файл	7
2.4	Заполняем файл	8
2.5	Запускаем файл	8
2.6	Заполняем файл	9
2.7	Запускаем файл	10
2.8	Создаем и запускаем файл	10
2.9	Создаем файл листинга и открываем	11
2.10	Листинг	11
2.11	Удаление операнда	12
2.12	Запуск файла	12
3.1	Редактированная программа	14
3.2	Результат	14
3.3	Результат	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 Порядок выполнения лабораторной работы

2.1 Реализация переходов в NASM

2.1.1 Задание 1

Создадим каталог для программ лабораторной работы No 7, перейдем в него и создадим файл lab6-7.asm (рис. [2.1])

```
mvchigladze@dk6n54 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc $ mkdir ~/work/study/2023-2024/Архитектура\ компьютера/study_2023-2024_arhpc/arch-pc/lab07
mvchigladze@dk6n54 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc $ cd ~/work/study/2023-2024/Архитектура\ компьютера/study_2023-2024_arhpc/arch-pc/lab07
mvchigladze@dk6n54 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/arch-pc/lab07 $ touch lab7-1.asm
mvchigladze@dk6n54 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/arch-pc/lab07 $ ls
lab7-1.asm
mvchigladze@dk6n54 ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc/arch-pc/lab07 $
```

Рис. 2.1: Создание каталога

2.1.2 Задание 2

Заполняем файл по листингу (рис. [2.2]), и создаем и запускаем исполняемый файл (рис. [2.3])

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения
```

1Помощь 2Разверн 3Выход 4Нех 5Перейти 6

Рис. 2.2: Заполняем файл

```
mvchigladze@dk6n54: ~/work/study/2023-2024/Ar...
07 $ nasm -f elf lab7-1.asm
mvchigladze@dk6n54: ~/work/study/2023-2024/Ar...
07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
mvchigladze@dk6n54: ~/work/study/2023-2024/Ar...
07 $ ./lab7-1
Сообщение No 2
Сообщение No 3

```

Рис. 2.3: Запускаем файл

Изменим порядок запуска с помощью команд, заполним файл по листингу (рис. [2.4]), и запускаем исполняемый файл (рис. [2.5])

```

lab7-1.asm      [----]  8 L:[  8+14  22
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 2.4: Заполняем файл

```

y_2023-2024_arhpc/arch-pc/lab07 $ ld -m elf_
mvchigladze@dk3n64 ~/work/study/2023-2024/Ar
y_2023-2024_arhpc/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1
mvchigladze@dk3n64 ~/work/study/2023-2024/Ar
y_2023-2024_arhpc/arch-pc/lab07 $ 

```

Рис. 2.5: Запускаем файл

Изменим порядок запуска с помощью команд, добавив после старта прыжок к 3, а у 3 прыжок ко 2 (рис. [2.6]), и запускаем исполняемый файл (рис. [2.7])


```

_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на э
call sprintfLF ; 'Сообщение
jmp _end
_label2:
mov eax, msg2 ; Вывод на э
call sprintfLF ; 'Сообщение
jmp _label1
_label3:
mov eax, msg3 ; Вывод на э
call sprintfLF ; 'Сообщение
jmp _label2
_end:

```

1По~щъ 2Со~ть 3Блок 4За~

Рис. 2.6: Заполняем файл

```

mvchigladze@dk3n64 ~/work/study/
y_2023-2024_arhpc/arch-pc/lab07
mvchigladze@dk3n64 ~/work/study/
y_2023-2024_arhpc/arch-pc/lab07
Сообщение No 3
Сообщение No 2
Сообщение No 1
mvchigladze@dk3n64 ~/work/study/
y_2023-2024_arhpc/arch-pc/lab07

```

Рис. 2.7: Запускаем файл

2.2 Задание 3

Создаем файл asm командой touch, заполняем по листингу 7.3 и запускаем исполняемый файл (рис. [2.8])

```

touch lab7-2.asm
^C
mvchigladze@dk3n64 ~/work/study/2023-2024/Архитектура компьютера/stud
y_2023-2024_arhpc/arch-pc/lab07 $ nasm -f elf lab7-2.asm
mvchigladze@dk3n64 ~/work/study/2023-2024/Архитектура компьютера/stud
y_2023-2024_arhpc/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
mvchigladze@dk3n64 ~/work/study/2023-2024/Архитектура компьютера/stud
y_2023-2024_arhpc/arch-pc/lab07 $ ./lab7-2
Введите B: 14
Наибольшее число: 50
mvchigladze@dk3n64 ~/work/study/2023-2024/Архитектура компьютера/stud
y_2023-2024_arhpc/arch-pc/lab07 $

```

Рис. 2.8: Создаем и запускаем файл

2.3 Задание 4

Создам файл листинга для программы из файла lab7-2.asm Открою файл листинга lab7-2.lst с помощью любого текстового редактора mcedit lab7-2.lst (рис. [2.9]) Подробно объясню содержимое трёх строк файла листинга (рис. [2.10]). Строка 21 - мы записываем в переменную eax значение переменной B Строка 22 - преобразуем из вида строки в числовой вид Строка 23 - Записываем полученное обратно в число B Открою файл с программой lab7-2.asm и удалю операнд B (рис. [2.11]). Выполню трансляцию с получением файла листинга, получилась ошибка, которая записалась в листинге (рис. [2.12]).

```
mvchigladze@dk3n64 ~/work/study/2023-2024/Архитектура компьютерных систем/2023-2024_arhpc/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
mvchigladze@dk3n64 ~/work/study/2023-2024/Архитектура компьютерных систем/2023-2024_arhpc/arch-pc/lab07 $ mcedit lab7-2.lst
mvchigladze@dk3n64 ~/work/study/2023-2024/Архитектура компьютерных систем/2023-2024_arhpc/arch-pc/lab07 $
```

Рис. 2.9: Создаем файл листинга и открываем

```
11      global _start
12      _start:
13      ; ----- Вывод сообщения 'Введите B: '
14      000000E8 B8[00000000]    mov eax,msg1
15      000000ED E81DFFFFFF    call sprint
16      ; ----- Ввод 'B'
17      000000F2 B9[0A000000]    mov ecx,B
18      000000F7 BA0A000000    mov edx,10
19      000000FC E842FFFFFF    call sread
20      ; ----- Преобразование 'B' из символа в число
21      00000101 B8[0A000000]    mov eax,B
22      00000106 E891FFFFFF    call atoi ; Вызов подпрограммы перевода символа в число
23      0000010B A3[0A000000]    mov [B],eax ; запись преобразованного числа в 'B'
24      ; ----- Записываем 'A' в переменную 'max'
25      00000110 8B0D[35000000]    mov ecx,[A] ; 'ecx = A'
26      00000116 890D[00000000]    mov [max],ecx ; 'max = A'
27      ; ----- Сравниваем 'A' и 'C' (как символы)
28      0000011C 3B0D[39000000]    cmp ecx,[C] ; Сравниваем 'A' и 'C'
29      00000122 7F0C            jg check_B ; если 'A>C', то переход на метку 'check_B',
30      00000124 8B0D[39000000]    mov ecx,[C] ; иначе 'ecx = C'
31      0000012A 890D[00000000]    mov [max],ecx ; 'max = C'
32      ; ----- Преобразование 'max(A,C)' из символа в число
; -----
```

Рис. 2.10: Листинг

```

; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax
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'

```

Рис. 2.11: Удаление операнда

19 000000FC E842FFFFFF	call sread
20	; ----- Преобразование 'B' из символа в число
21	mov eax
21	error: invalid combination of opcode and operands
22 00000101 E896FFFFFF	call atoi ; Вызов подпрограммы перевода символа в число
23 00000106 A3[0A000000]	mov [B],eax ; запись преобразованного числа в 'B'
24	; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D[35000000]	mov ecx,[A] ; 'ecx = A'

Рис. 2.12: Запуск файла

3 Задание для самостоятельной работы

3.1 Задание 1

Написала программу нахождения наименьшей из 3 целочисленных переменных x, y и z . Значения переменных 92, 2, 61 (рис. [3.1]). Создам исполняемый файл и проверю его работу (рис. [3.2]).

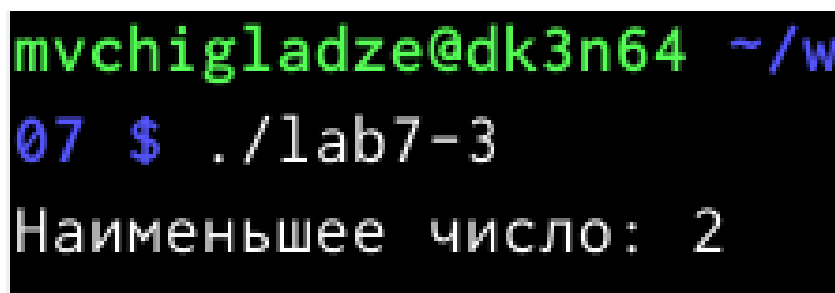
```

2 section .data
3 msg2 db "Наименьшее число: ",0h
4 A dd '95'
5 B dd '2'
6 C dd '61'
7 section .bss
8 min resb 10
9 section .text
0 global _start
1 _start:
2 ; ----- Записываем 'A' в переменную 'min'
3 mov ecx,[A] ; 'ecx = A'
4 mov [min],ecx ; 'min = A'
5 ; ----- Сравниваем 'A' и 'C' (как символы)
6 cmp ecx,[C] ; Сравниваем 'A' и 'C'
7 jl check_B ; если 'A<C', то переход на метку 'check_B',
8 mov ecx,[C] ; иначе 'ecx = C'
9 mov [min],ecx ; 'min = C'
0 ; ----- Преобразование 'min(A,C)' из символа в число
1 check_B:
2 ;mov eax,min
3 ;call atoi ; Вызов подпрограммы перевода символа в число
4 ;mov [min],eax ; запись преобразованного числа в 'min'
5 ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
6 mov ecx,[min]
7 cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
8 jl fin ; если 'min(A,C)>B', то переход на 'fin',|
9 mov ecx,[B] ; иначе 'ecx = B'
0 mov [min],ecx
1 ; ----- Вывод результата
2 fin:
3 mov eax, msg2
4 call sprint ; Вывод сообщения 'Наименьшее число: '
5 mov eax,min
6 call atoi
7 call iprintLF ; Вывод 'min(A,B,C)'
8 call quit ; Выход

```

Matlab ▾ Ширина табуляции: 8 ▾ Стр 28, Стлб 49

Рис. 3.1: Редактированная программа



```

mvchigladze@dk3n64 ~/w
07 $ ./lab7-3
Наименьшее число: 2

```

Рис. 3.2: Результат

3.2 Задание 2

Написала программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Моя система в варианте 20. Ввела данные по заданию варианты значений и их результат (рис. [??]). Мой код программы (рис. [3.3]).

```
mvchigladze@dk6n65 ~/work/study/2023-2024/Архитек
ab07 $ ./lab7-4
Введите X: 1
Введите A: 2
5
mvchigladze@dk6n65 ~/work/study/2023-2024/Архитек
ab07 $ ./lab7-4
Введите X: 2
Введите A: 1
Значение функции: 1
```

```

_start:
; ----- Вывод сообщения 'Введите x: '
mov eax,msg1 ;в переменную пишем msg1
call sprint
; ----- Ввод 'x'
mov ecx,x
mov edx,10
call sread
; ----- Преобразование 'x' из символа в число
mov eax,x
call atoi ; Вызов подпрограммы перевода символа в число
mov [x],eax ; запись преобразованного числа в 'x'
; ----- Вывод сообщения 'Введите a: '
mov eax, msg2
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 'a'
; ----- Сравниваем 'x' и 'a' (как числа)
mov eax, [x]
cmp eax, [a] ; Сравниваем 'x' и 'a'
jge make_X ; если 'x>a', то переход на 'make_x',

mov eax, 5 ; иначе 'ecx = 5'
call iprintLF
call quit
; -----
make_X:
mov eax, [x]
sub eax, [a]
mov [result], eax
; ----- Вывод результата
print_result:
mov eax, msg3
call sprint
mov eax, [result]
call iprintLF
call quit ; Выход

```

Рис. 3.3: Результат

4 Выводы

В ходе лабораторной работы, я изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. Позанкомилась с назначением и структурой файла

Список литературы