

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Хоюгбан Ганчыыр Анатольевич

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выполнение самостоятельной работы	13
5	Выводы	16

Список иллюстраций

3.1	Создание файла lab7-1.asm	7
3.2	Текст программы файла lab7-1.asm	8
3.3	Исполнение файла lab7-1.asm	8
3.4	Измененный текст программы файла lab7-1.asm	9
3.5	Исполнение измененного файла lab7-1.asm	9
3.6	Текст программы файла lab7-2.asm	10
3.7	Исполнение файла программы lab7-2.asm	11
3.8	Открытие файла листинга lab7-2.asm	11
3.9	Ошибка в файле lab7-2.asm	12
4.1	Текст программы для пункта 1 файла lab7-3.asm	14
4.2	Исполнение программы файла lab7-3.asm для 1 пункта	15

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 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.

3 Выполнение лабораторной работы

Для начала я зашел в терминал, перешел в свой каталог по пути ~/work/arch-pc, создал в нем каталог lab07, а в нем уже создал файл lab7-1.asm(рис. 3.1).

```
gakhoyugban@dk4n65 ~ $ cd work/arch-pc
gakhoyugban@dk4n65 ~/work/arch-pc $ ls
lab05  lab06
gakhoyugban@dk4n65 ~/work/arch-pc $ mkdir lab07
gakhoyugban@dk4n65 ~/work/arch-pc $ touch lab7-1.asm
gakhoyugban@dk4n65 ~/work/arch-pc $ ls
lab05  lab06  lab07  lab7-1.asm
gakhoyugban@dk4n65 ~/work/arch-pc $ cd lab07
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ls
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 3.1: Создание файла lab7-1.asm

В этом файле я написал текст программы с использованием инструкции jmp(рис. 3.2).

```

%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
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit

```

Рис. 3.2: Текст программы файла lab7-1.asm

Я оттранслировал данный файл lab7-1.asm в объектный, выполнил компоновку и отправил на исполнение. В результате мне выдало правильный ответ по условиям лабораторной работы, что мне сначала выдало 2 сообщение, а затем 3(рис. 3.3).

```

gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $

```

Рис. 3.3: Исполнение файла lab7-1.asm

Я изменил текст файла lab7-1.asm и написал такой текст программы, чтобы мне поочередно выдали сообщения 3, 2, 1(рис. 3.4).


```

#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'
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

```

Рис. 3.4: Измененный текст программы файла lab7-1.asm

Я оттранслировал данный измененный файл lab7-1.asm в объектный, выполнил компоновку и отправил на исполнение. В результате мне выдало правильный ответ по условиям лабораторной работы, что мне сначала выдало 3 сообщение, а затем 2 и потом уже 1 сообщение(рис. 3.5).

```

gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1

```

Рис. 3.5: Исполнение измененного файла lab7-1.asm

Я создал файл lab7-2.asm, ввел в него текст программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C(рис. 3.6).

```

; -----; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintL; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintL Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintL%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

```

Рис. 3.6: Текст программы файла lab7-2.asm

Я оттранслировал данный файл lab7-2.asm в объектный, выполнил компоновку и отправил на исполнение. Затем я ввел пару значений, в ходе которых я пришел к такому выводу, что все введенные числа больше 50 будут наибольшими. Если ввести число меньше 50, то наибольшим для него будет 50(рис. 3.7).

```

gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 100
Наибольшее число: 100
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 1
Наибольшее число: 50
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 8
Наибольшее число: 50
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 101
Наибольшее число: 101
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 51
Наибольшее число: 51
gakhoyugban@dk4n65 ~/work/arch-pc/lab07 $

```

Рис. 3.7: Исполнение файла программы lab7-2.asm

Я указал ключ -l и задал имя файла листинга в командной строке, создал файл листинга lab7-2.lst и открыл его с помощью команды mcedit(рис. 3.8).

```

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 E8E8FFFF <1>      call    slen

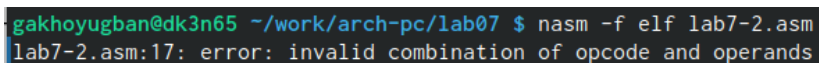
```

Рис. 3.8: Открытие файла листинга lab7-2.asm

По условию лабораторной мне надо объяснить 3 строки из файла листинга lab7-2.asm: 1)В строке 11 содержится номер сторки [11], адресс [00000008], машинный код [40] и содержимое строки кода [inc eax] 2)В строке 9 содержится собствен но номер сторки [9], адресс [00000003], машинный код [803800] и содержимое строки кода [cmp byte [eax], 0] 3)В строке 24 содержится номер сторки [24], адресс

[0000000F], машинный код [52] и содержимое строки кода [push edx]

По условию лабораторной мне надо было в файле lab7-2.asm найти строку, где производится действия над двумя операндами и удалить один из них. Затем надо оттранслировать, отправить на компоноку и отправить на исполнение. Но, так как я допустил ошибку выше, мне выдаст ошибку, которую я демонстрирую на рисунке(рис. 3.9).

A screenshot of a terminal window with a dark background. The prompt is 'gakhoyugban@dk3n65 ~/work/arch-pc/lab07 \$'. The command entered is 'nasm -f elf lab7-2.asm'. The output is 'lab7-2.asm:17: error: invalid combination of opcode and operands'.

```
gakhoyugban@dk3n65 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
lab7-2.asm:17: error: invalid combination of opcode and operands
```

Рис. 3.9: Ошибка в файле lab7-2.asm

4 Выполнение самостоятельной работы

Для начала я создал файл lab7-3.asm, а затем напечатал для первого пункта программу нахождения наименьшей из 3 целочисленных переменных x, y и z . Текст программы на рисунке(рис. 4.1).

```

%include 'in_out.asm'

section .data
    msg1 db "Наименьшее число:"
    a dd 95
    b dd 2
    c dd 61

section .bss
    min resb 10

section .text
global _start

_start:
    mov eax, msg1
    call sprint

    mov ecx, [a]
    mov [min], ecx ; 'min = A'
    ; ----- Сравниваем 'A' и 'C' (как числа)
    cmp ecx, [c] ; Сравниваем 'A' и 'C'
    jl check_B ; если 'A<C', то переход на метку 'check_B',
    mov ecx, [c] ; иначе 'ecx = C'
    mov [min], ecx ; 'min = C'
    ; ----- Преобразование 'min(A,C)' из символа в число

check_B:
    ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
    mov ecx, [min]
    cmp ecx, [b] ; Сравниваем 'min(A,C)' и 'B'
    jl fin ; если 'min(A,C)>B', то переход на 'fin',
    mov ecx, [b] ; иначе 'ecx = B'

    mov [min], ecx

; ----- Вывод результата
fin:
    mov eax, [min]
    call iprintLF ; Вывод 'min(A,B,C)'
    call quit ; Выход

```

Рис. 4.1: Текст программы для пункта 1 файла lab7-3.asm

Затем я оттранслировал файл lab7-3.asm в объектный, выполнил компоновку и отправил на исполнении. В результате мне выдало правильное значение, а именно 2, так как среди 95, 2, 61 оно будет самым наименьшим(рис. 4.2).

```
gakhoyugban@dk3n65 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
gakhoyugban@dk3n65 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
gakhoyugban@dk3n65 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: _2
```

Рис. 4.2: Исполнение программы файла lab7-3.asm для 1 пункта

5 Выводы

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