

Отчёт по лабораторной работе %8

Дисциплина: 'архитектура компьютеров'

Бабенко Роман Игоревич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	18

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Перепечатаывам текст программы	7
2.3	Запуск программы	8
2.4	Проверка изменённой программы	8
2.5	Изменяем программу	9
2.6	Проверка вывода	10
2.7	Создание ещё одного файла	10
2.8	Проверка программы 8.2	10
2.9	Проверка программы 8.2	10
2.10	Создание файла листинга	10
2.11	Файл листинга	11
2.12	Трансляция файла листинга после удаления операнда	12
2.13	Текстовый редактор mcedit	12
2.14	Программа	13
2.15	Результат программы	14
2.16	Написанная програма(1)	15
2.17	Написанная програма(2)	16
2.18	Проверка программы	17

Список таблиц

1 Цель работы

Изучить команды условного и безусловного переходов, приобрести навыки написания программ с использованием переходов, а также познакомиться с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

Создаём каталог lab08 и файл lab8-1.asm (рис. 2.1)

```
ribabenko@dk1n22 ~ $ mkdir ~/work/arch-pc/lab08  
ribabenko@dk1n22 ~ $ cd ~/work/arch-pc/lab08  
ribabenko@dk1n22 ~/work/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 2.1: Создание каталога и файла

Вводим в файл текст программы из листинга 8.1(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 printf

_label2:
    mov eax, msg2
    call printf

_label3:
    mov eax, msg3
    call printf

_end:
    call quit
```

Рис. 2.2: Перепечатаываю текст программы

Создаём исполняемый файл и запускаем его (рис. 2.3)

```
ribabenko@dk1n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
ribabenko@dk1n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
ribabenko@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение №2
Сообщение №3
ribabenko@dk1n22 ~/work/arch-pc/lab08 $
```

Рис. 2.3: Запуск программы

Изменим программу в соответствии с листингом 8.2 и проверяем его работу(рис. 2.4)

```
nasm -f elf lab8-1.asm
ribabenko@dk1n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
ribabenko@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение №2
Сообщение №1
ribabenko@dk1n22 ~/work/arch-pc/lab08 $
```

Рис. 2.4: Проверка изменённой программы

Снова изменяем текст программы так: (рис. 2.5)


```
%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 _label3

_label1:
    mov eax, msg1
    call sprintf
    jmp _end

_label2:
    mov eax, msg2
    call sprintf
    jmp _label1

_label3:
    mov eax, msg3
    call sprintf
    jmp _label2

_end:
    call quit
```

Рис. 2.5: Изменяем программу

Проверяем нашу программу (рис. 2.6)

```
ribabenko@dk1n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
ribabenko@dk1n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
ribabenko@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение №3
Сообщение №2
Сообщение №1
ribabenko@dk1n22 ~/work/arch-pc/lab08 $
```

Рис. 2.6: Проверка вывода

Создаём файл lab8-2.asm (рис. 2.7)

```
touch lab8-2.asm
ribabenko@dk1n22 ~/work/arch-pc/lab08 $
```

Рис. 2.7: Создание ещё одного файла

Переписываем в созданный файл программу из листинга 8.3 и проверяем его работу (рис. 2.8),(рис. 2.9)

```
ribabenko@dk1n22 ~/work/arch-pc/lab08 $ ./lab8-2
Введите В: 10
Наибольшее число: 50
ribabenko@dk1n22 ~/work/arch-pc/lab08 $
```

Рис. 2.8: Проверка программы 8.2

```
ribabenko@dk5n55 ~/work/arch-pc/lab08 $ ./lab8-2
Введите В: 100
Наибольшее число: 100
ribabenko@dk5n55 ~/work/arch-pc/lab08 $
```

Рис. 2.9: Проверка программы 8.2

Создаём файл листинга и открываем его с помощью текстового редактора (рис. 2.10)

```
ribabenko@dk5n55 ~/work/arch-pc/lab08 $ nasm -f elf -l lab8-1.lst lab8-1.asm
ribabenko@dk5n55 ~/work/arch-pc/lab08 $ mcedit lab8-2.lst
```

Рис. 2.10: Создание файла листинга

Изучаем формат и содержимое файла (рис. 2.11)

```
lab9-2.lst [----] 0 L: [ 1+ 0 1/225] *(0 /14458b) 0032 0x020
1      %include 'in_out.asm'
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:
5      00000000 53      <1> push    ebx
6      00000001 89C3    <1> mov     ebx, eax
7      <1>
8      <1> nextchar:
9      00000003 803800  <1> cmp     byte [eax], 0
10     00000006 7403    <1> jz      finished
11     00000008 40      <1> inc     eax
12     00000009 EBF8    <1> jmp     nextchar
13     <1>
14     <1> finished:
15     0000000B 29D8    <1> sub     eax, ebx
16     0000000D 5B      <1> pop     ebx
17     0000000E C3      <1> ret
18     <1>
19     <1>
20     <1> ;----- sprint -----
21     <1> ; Функция печати сообщения
22     <1> ; входные данные: mov eax,<message>
23     <1> sprint:
24     0000000F 52      <1> push    edx
25     00000010 51      <1> push    ecx
26     00000011 53      <1> push    ebx
27     00000012 50      <1> push    eax
28     00000013 E8E8FFFF <1> call    slen
29     <1>
30     00000018 89C2    <1> mov     edx, eax
31     0000001A 58      <1> pop     eax
32     <1>
33     0000001B 89C1    <1> mov     ecx, eax
34     0000001D BB01000000 <1> mov     ebx, 1
```

Рис. 2.11: Файл листинга

Открываем файл с программой lab8-2.asm и на строке 39 удаляем операнд.Выполняем трансляцию с получением файла листинга (рис. 2.12)

```

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
33                          check_B:
34 00000130 B8[00000000]    mov eax,max
35 00000135 E862FFFFFF      call atoi ; Вызов подпрограммы перевода символа в число
36 0000013A A3[00000000]    mov [max],eax ; запись преобразованного числа в 'max'
37                          ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000]  mov ecx,[max]
39                          cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
39 *****
error: invalid combination of opcode and operands
40 00000145 7F0C            jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 00000147 8B0D[0A000000]  mov ecx,[B] ; иначе 'ecx = B'
42 0000014D 890D[00000000]  mov [max],ecx
43                          ; ----- Вывод результата
44                          fin:
45 00000153 B8[13000000]    mov eax, msg2

```

Рис. 2.12: Трансляция файла листинга после удаления операнда

Открываем текстовый редактор mcedit. (рис. 2.13)

```

ribabenko@dk5n55 ~/work/arch-pc/lab08 $ nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:39: error: invalid combination of opcode and operands
ribabenko@dk5n55 ~/work/arch-pc/lab08 $ mcedit lab8-2.lst
ribabenko@dk5n55 ~/work/arch-pc/lab08 $

```

Рис. 2.13: Текстовый редактор mcedit

#Задания для самостоятельной работы

Создаём файл lab8-3.asm и напишем программу для нахождения наименьшей из трёх переменных (рис. 2.14)

```

%include 'in_out.asm'
section .data
msg2 db "Наименьшее число: ",0h
A dd '45'
B dd '67'
C dd '15'
section .bss
min resb 10
section .text
global _start

_start:
mov ecx,[A]
mov [min],ecx
cmp ecx,[C]
jl check_B
mov ecx,[C]
mov [min],ecx

check_B:
mov eax,min
call atoi
mov [min],eax
mov ecx,[min]
cmp ecx,[B]
jl fin
mov ecx,[B]
mov [min],ecx

fin:
mov eax, msg2
call sprint
mov eax,[min]
call iprintLF
call quit

```

Рис. 2.14: Программа

Проверка написанной программы (рис. 2.15)

```
nasm -f elf lab8-3.asm
ribabenko@dk5n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
ribabenko@dk5n55 ~/work/arch-pc/lab08 $ ./lab8-3
Наименьшее число: 15
```

Рис. 2.15: Результат программы

Напишем программу для вычисления значения функции по 7 варианту (рис. [-fig. 2.16) и (рис. 2.17)

```

%include 'in_out.asm'
section .data
vv1 db "Введите x: ",0h
vv2 db "Введите a: ",0h
clishe db "Результат: ",0h

section .bss
x resb 10
a resb 10

section .text
global _start

_start:

mov eax,vv1
call sprint

mov ecx,x
mov edx,10
call sread

mov eax,x
call atoi
mov [x],eax

mov eax,vv2
call sprint

mov ecx,a
mov edx,10
call sread

mov eax,a
call atoi
mov [a],eax

mov ecx,[x]
cmp ecx,[a]
je check_N

```

Рис. 2.16: Написанная програма(1)

```

mov eax,x
call atoi
mov [x],eax

mov eax,vv2
call sprint

mov ecx,a
mov edx,10
call sread

mov eax,a
call atoi
mov [a],eax

mov ecx,[x]
cmp ecx,[a]
je check_N
mov eax,clishe
call sprint
mov eax,[x]
mov ebx,[a]
add eax,ebx
mov edi,eax
call iprintLF
jmp end

check_N:
mov eax,clishe
call sprint
mov eax,[a]
mov ebx,6
mul ebx
call iprintLF
jmp end

end:
call quit

```

Рис. 2.17: Написанная програма(2)

Проверяем результат выполнения написанной программы (рис. 2.18)

```
ribabenko@dk5n55 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
ribabenko@dk5n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
ribabenko@dk5n55 ~/work/arch-pc/lab08 $ ./lab8-4
Введите x: 1
Введите a: 1
Результат: 6
ribabenko@dk5n55 ~/work/arch-pc/lab08 $ ./lab8-4
Введите x: 2
Введите a: 1
Результат: 3
```

Рис. 2.18: Проверка программы

3 Выводы

В ходе выполнения лабораторной работы я научился пользоваться командами условного и безусловного переходов, приобрёл навыки написания программ с использованием переходов, а также познакомился с назначением и структурой файла листинга.