Отчет по лабораторной работе №8

Борунов Семён Сергеевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить основы работы комманд усовного и безусловного перехода в assembler.

2 Выполнение лабораторной работы

Создадим рабочую папку и рабочий файл (рис. **??**)

```
ssborunov@dk4n64 ~/work/arc-pc $ mkdir lab08
ssborunov@dk4n64 ~/work/arc-pc $ cd lab08
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ touch lab08.asm
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ gedit lab08.asm
```

Запишем в файл код, проассемблируем его, запустим(рис. 2.1)

```
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ nasm -f elf lab8-1.asm
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 3
ssborunov@dk4n64 ~/work/arc-pc/lab08 $
```

Рис. 2.1: jmp работает

изменим программу так, чтобы она выводила второе,затем первое сообщение и завершала работу. Её код (рис. 2.2) и работа (рис. 2.3)

```
%include 'in_out.asm'

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
call sprintLF
jmp _end

_label2:
mov eax,msg2
call sprintLF
jmp _label1

_label3:
mov eax,msg3
call sprintLF

_end:
call quit
```

Рис. 2.2: Код программы

```
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ nasm -f elf lab8-1.asm
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 1
ssborunov@dk4n64 ~/work/arc-pc/lab08 $
```

Рис. 2.3: Её работа

Еще напишем программу, выводящую сообщения в обратном порядке(рис. 2.4 и рис. 2.5)

```
%include 'in_out.asm'

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 _label3

_label1:
mov eax,msg1
call sprintLF
jmp _end

_label2:
mov eax,msg2
call sprintLF
jmp _label1

_label3:
mov eax,msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 2.4: Код программы

```
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ nasm -f elf lab8-1.asm
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ ./lab8-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
ssborunov@dk4n64 ~/work/arc-pc/lab08 $
```

Рис. 2.5: Её работа

Напишем программу с условным переходом(рис. 2.6)

```
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ ./lab8-2
Введите B: 100
Наибольшее число: 100
ssborunov@dk4n64 ~/work/arc-pc/lab08 $ ./lab8-2
Введите B: 1
Наибольшее число: 50
ssborunov@dk4n64 ~/work/arc-pc/lab08 $
```

Рис. 2.6: Работа программы

```
  1                                     %include 'in_out.asm'
  2                                <1> ;---------------   slen  ---
  3                                <1> ; Функция вычисления длины с
  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,<m
 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 E8E8FFFFFF            <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
```

Рассмотрим файл листинга одной из программ(рис. **??**)

в строке 9 содержится собственно номер сторки [9], адресс [00000003], машинный код [803800] и содержимое строки кода [cmp byte [eax], 0] в строке 11 содержится номер сторки [11], адресс [00000008], машинный код [40] и содержимое строки кода [inc eax] в строке 24 содержится номер сторки [24], адресс [0000000F], машинный код [52] и содержимое строки кода [push edx]

Если в коде появляется ошибка, то ее описание появится в файле листинга(рис. 2.7)

```
 9                                  section .bss
10 00000000 <res Ah>                max resb 10
11 0000000A <res Ah>                B resb 10
12
13                                  section .text
14
15                                  global _start
16                                  _start:
17                                  mov eax
17          ******************       error: invalid combination of opcode and operands
18 000000E8 E822FFFFFF              call sprint
19
20 000000ED B9[0A000000]            mov ecx, B
21 000000F2 BA0A000000              mov edx,10
22 000000F7 E847FFFFFF              call sread
23
24 000000FC B8[0A000000]            mov eax, B
25 00000101 E896FFFFFF              call atoi
26 00000106 A3[0A000000]            mov [B], eax
27
28 0000010B 8B0D[35000000]          mov ecx, [A]
```

Рис. 2.7: ошибка в файле листинга

3 Задания для самостоятельной работы

(Вар 17)

программа для сравнения трех заранее известных чисел(рис. **??**) и ее работа(рис. **??**)

```
%include 'in_out.asm'
section .data
msg2 db "Наибольшее число: ",0h
A dd '26'
B dd '12'
C dd '68'

section .bss
max resb 10

section .text

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

```
ssborunov@dk2n22 ~/work/arc-pc/lab08 $ gedit lab08-
ssborunov@dk2n22 ~/work/arc-pc/lab08 $ nasm -f elf
 -o lab08-3 lab08-3.o && ./lab08-3
Наибольшее число: 68
ssborunov@dk2n22 ~/work/arc-pc/lab08 $
```

Программа для вычисления выражения в зависимости от условия на одну из

вводимых переменных(рис. **??**) и ее работа(рис. **??**)

```
%include 'in_out.asm'

SECTION .data
input1 db "Введите x: ",0h
input2 db "Ввведите a: ",0h

SECTION .bss
max resb 10
x resb 10
a resb 10

SECTION .text
GLOBAL _start

_start:
mov eax,input1
call sprint

mov ecx,x
mov edx,10
call sread

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

mov eax,input2
call sprint

mov ecx,a
mov edx,10
call sread

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

mov ebx, 8
cmp [a], ebx
jge check

mov eax, [a]
mov ebx, 8
add eax, ebx
call iprintLF
```

```
9ssborunov@dk2n22 ~/work/arc-pc/lab08 $ nasm -f elf
6 -o lab08-4 lab08-4.o && ./lab08-4
Введите x: 3
Ввведите a: 4
12
ssborunov@dk2n22 ~/work/arc-pc/lab08 $ nasm -f elf l
 -o lab08-4 lab08-4.o && ./lab08-4
2Введите x:
Ввведите a: 9
18
```

4 Выводы

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