

Отчёт по лабораторной работе №8

Программирование ветвлений

Мулин Иван Владимирович

Содержание

1	Цель работы	4
2	Ход работы	5
2.1	Выполнение лабораторной работы	5
2.2	Выполнение заданий для самостоятельной работы	6
3	Листинги написанных программ	8
4	Заключение	15

Список иллюстраций

2.1	Запуск программы 1, дубль 1	5
2.2	Запуск изменённой программы 1	5
2.3	Запуск программы 2, дубль 1	6
2.4	Запуск программы 2, дубль 2	6
2.5	Поиск наименьшего числа	7
2.6	Вычисление значения функции от двух переменных	7

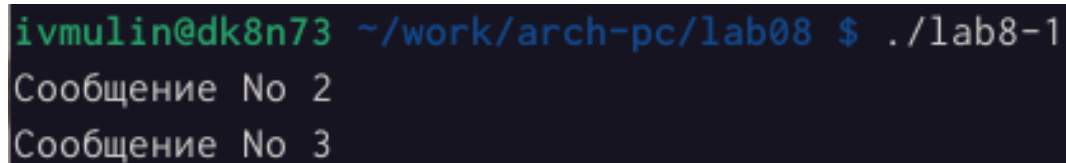
1 Цель работы

В ходе выполнения данной лабораторной работы необходимо изучить программирование ветвлений в языке ассемблера NASM. Репозиторий github расположен по адресу https://github.com/ivmulin/study_2022-2023_arch-pc.

2 Ход работы

2.1 Выполнение лабораторной работы

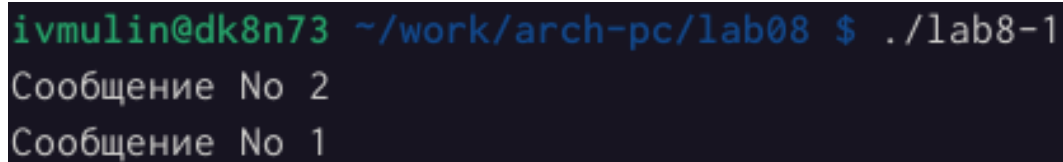
Программа `lab8-1.asm` реализует алгоритм выполнения ветвлений при помощи команды `jmp`, выводя сначала второе сообщение, затем третье:



```
ivmulin@dk8n73 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 3
```

Рис. 2.1: Запуск программы 1, дубль 1

Изменим программу так, что она выводит первое сообщение вслед за вторым:



```
ivmulin@dk8n73 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 1
```

Рис. 2.2: Запуск изменённой программы 1

Вторая программа находит наибольшее из трёх чисел:

```

ivmulin@dk8n73 ~/work/arch-pc/lab08 $ ./lab8-2
Введите В: 37
Наибольшее число: 50
ivmulin@dk8n73 ~/work/arch-pc/lab08 $ ./lab8-2
Введите В: 69
Наибольшее число: 69

```

Рис. 2.3: Запуск программы 2, дубль 1

При сборке программы был создан файл листинга. Рассмотрим следующий отрывок из него:

```

20 000000F2 B9[0A000000]      mov ecx, B
21 000000F7 BA0A000000      mov edx, 10
22 000000FC E842FFFFFF      call sread

```

Команда B9[0A000000] (mov eax, B) располагается по смещению 000000F2. По аналогии команды BA0A000000 (mov edx, 10) и E842FFFFFF (call sread) находятся по смещениям 000000F7 и 000000FC соответственно.

Удалим второй операнд в команде mov eax, max и попробуем собрать программу. В итоге создастся файл листинга со следующей ошибкой в соответствующем месте кода:

```

38                               mov eax
38 ***** error: invalid combination of opcode and operands

```

Рис. 2.4: Запуск программы 2, дубль 2

Объектный файл в этом случае не будет создан.

2.2 Выполнение заданий для самостоятельной работы

Выполняя задания самостоятельной работы, необходимо написать две программы. Первая программа вычисляет наименьшее из чисел 21, 28, 34:

```
ivmulin@dk8n73 ~/work/arch-pc/lab08 $ ./lab8-3
Наименьшее число - 20
```

Рис. 2.5: Поиск наименьшего числа

Вторая программа выводит значение функции

$$f(x, a) = 4a, x = 0$$

$$f(x, a) = 4a + x, x \neq 0$$

от двух введённых аргументов:

```
ivmulin@dk8n73 ~/work/arch-pc/lab08 $ ./lab8-4
x = 1
a = 2
9
ivmulin@dk8n73 ~/work/arch-pc/lab08 $ ./lab8-4
x = 0
a = 3
12
```

Рис. 2.6: Вычисление значения функции от двух переменных

Программы, как видно, работают исправно.

3 Листинги написанных программ

1. lab8-1.asm

```
%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 printf
    jmp _end

_label2:
    mov eax, msg2
    call printf
    jmp _label1
```



```
_label3:
    mov eax, msg3
    call sprintf
```

```
_end:
    call quit
```

2. lab8-2.asm

```
%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:
    mov eax, msg1
    call sprintf
    ; ----- 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 '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
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

```

3. lab8-3.asm

```
%include 'in_out.asm'
```

```
; 21, 28, 34
```

section .data

```

msg1 db "Наименьшее число - "
a dd 20
b dd 28
c dd 50

```

section .bss

```
min resb 10
```

section .text

```
global _start
```

```
_start:
```

```
; ----- печатаем "min(a, b, c) = "
```

```

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

```

4. lab8-4.asm

```

%include 'in_out.asm'

```

```

section .data

```

```

msgX db 'x = ',0h
msgA db "a = ",0h

```

```

section .bss

```

```

x resb 10
a resb 10
f resb 10

section .text
global _start

_start:
    ; ----- Ввод 'B'
    mov eax, msgX
    call sprint
    mov ecx, x
    mov edx, 10
    call sread

    ; ----- Ввод 'x'
    mov eax, msgA
    call sprint
    mov ecx, a
    mov edx, 10
    call sread

    ; ----- Преобразование 'x' из символа в число
    mov eax, x
    call atoi
    mov [x], eax

    ; ----- Преобразование 'a' из символа в число
    mov eax, a

```

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

```
mov ecx, [x]
cmp ecx, 0
```

```
je vadin
```

```
mov eax, [a]
mov ebx, 4
mul ebx
add eax, [x]
jmp fin
```

vadin:

```
mov eax, [a]
mov ebx, 4
mul ebx
```

fin:

```
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

4 Заключение

В ходе выполнения лабораторной работы №8 было изучено программирование условного и безусловного переходов и ветвлений в языке ассемблера NASM, а значит, цель данной лабораторной работы была достигнута.