

Отчет по лабораторной работа №8

Группа НПИбд-02-22

Стариков Данила Андреевич

Содержание

1	Цель работы	3
2	Основная часть	4
2.1	Выполнение лабораторной работы	4
2.1.1	Реализация переходов в NASM	4
2.1.2	Изучение структуры файла листинга	6
2.2	Выполнение заданий для самостоятельной работы.	8
2.2.1	Задание 1.	8
2.2.2	Задание 2.	8
3	Выводы	15

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

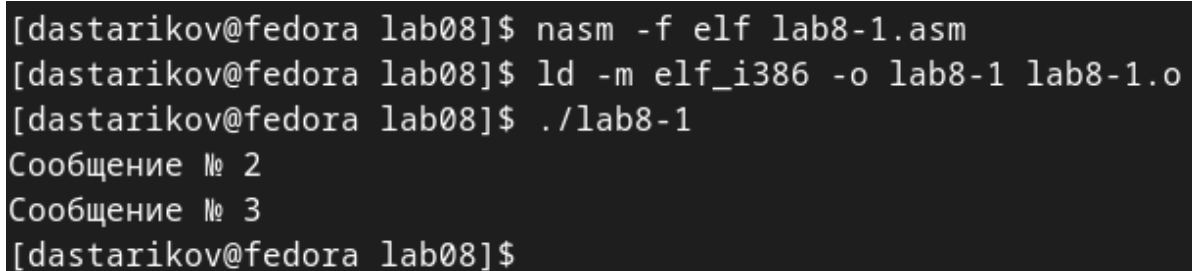
2 Основная часть

2.1 Выполнение лабораторной работы

2.1.1 Реализация переходов в NASM

Для выполнения лабораторной работы создали каталог `~/work/study/arch-rc/lab08`. В нем создали файл `lab7-8.asm` и ввели текст из Листинга 2.1. Также положили в каталог файл `in_out.asm`, использованный в лабораторной работе №6.

Создали исполняемый файл и запустили его (Рис. 2.1). Инструкция `jmp _label2` изменяет порядок выполнения программы, пропуская вывод первого сообщения. Далее изменили текст программы (Листинг 2.2), выводится второе, а затем первое сообщения, третье игнорируется. ((Рис. 2.2)).



```
[dastarikov@fedora lab08]$ nasm -f elf lab8-1.asm
[dastarikov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[dastarikov@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[dastarikov@fedora lab08]$
```

Рис. 2.1: Результат запуска файла `lab8-1`.

Листинг 2.1 Программа с использованием инструкции `jmp`

```
%include 'in_out.asm' ; подключение внешнего файла
```

```
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 ; Вывод на экран строки
```

```
call printf ; 'Сообщение № 1'
```

```
_label2:
```

```
mov eax, msg2 ; Вывод на экран строки
```

```
call printf ; 'Сообщение № 2'
```

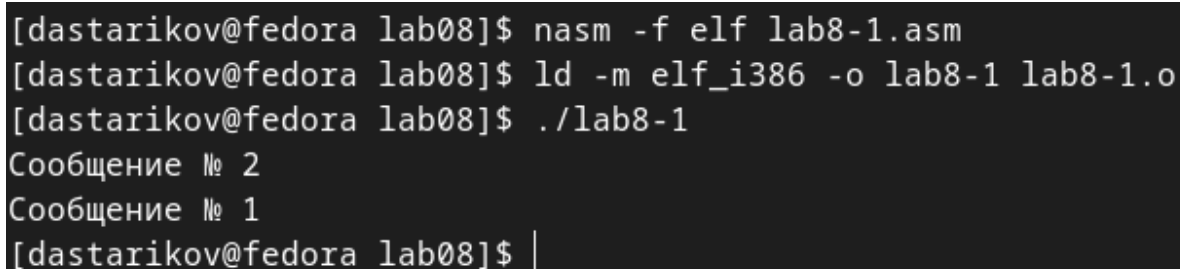
```
_label3:
```

```
mov eax, msg3 ; Вывод на экран строки
```

```
call printf ; 'Сообщение № 3'
```

```
_end:
```

```
call _quit ; вызов подпрограммы завершения
```



```
[dastarikov@fedora lab08]$ nasm -f elf lab8-1.asm
[dastarikov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[dastarikov@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[dastarikov@fedora lab08]$ |
```

Рис. 2.2: Порядок вывода сообщений изменился.

Затем изменили программу, что сообщения выводились в обратном порядке в

соответствии с Листингом 2.3 (Рис. 2.3)

```
[dastarikov@fedora lab07]$ nasm -f elf lab7-1.asm
[dastarikov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[dastarikov@fedora lab07]$ ./lab7-1

[dastarikov@fedora lab07]$ |
```

Рис. 2.3: Сообщения выводятся в обратном порядке.

Рассмотрели команды условного перехода на примере программы lab8-2.asm, вычисляющей наибольшее из 3х целых чисел А, В и С (Листинг 2.4). А и С заданы заранее, а В вводится с клавиатуры, при этом А и С для демонстрации сравниваются как символы, а наибольшее из них конвертируется в число и с сравнивается с В как число (Рис. 2.4).

```
[dastarikov@fedora lab07]$ nasm -f elf lab7-2.asm
[dastarikov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[dastarikov@fedora lab07]$ ./lab7-2
106
[dastarikov@fedora lab07]$
```

Рис. 2.4: Результат работы программы lab8-2 при вводе разных чисел.

2.1.2 Изучение структуры файла листинга

Обычно nasm создает только объектный файл, для создания файла листинга необходимо указать ключ -l в командной строке.

```
nasm -f elf lab8-2.asm -l lab8-2.lst
```

Проанализировали полученный файл на примере 3 строк:

```
20                                     ; ----- Ввод 'В'
21 000000F2 B9[0A000000]               mov ecx,B
```

22	000000F7 BA0A000000	mov edx,10
23	000000FC E842FFFFFF	call sread

Первый столбец содержит номер строки тексте программы: 21, 22, 23. Затем идет адрес команды в текущем сегменте кода: так инструкция `mov ecx, B` начинается по виртуальному адресу `000000F2`. Виртуальный адрес - это число из абстрактного виртуального адресного пространства, характерной конкретной программе, и не всегда соответствующее адресу физической памяти. Далее идет машинное представление инструкции: `mov ecx` ассемблируется как `B9`, а так как переменная `B` обозначает виртуальный адрес, где хранится ее значение, то инструкция `[0A000000]` означает, что нужно взять данные по адресу `0000000A` (адрес в сегменте `.bss`). Далее идет исходный текст программы. Далее 22 строка хранится по адресу `000000F7` - сдвиг на 5 байт от предыдущей, так как 1 байт занимает инструкция `mov ecx` и 4 байта - адрес памяти переменной `B`. `BA` - `move edx` и `0A000000` - число 10 в шестнадцатеричном представлении (Разрядность увеличивается справа налево). Затем строка 23: перешли еще на 5 байтов, `E842FFFFFF` соответствует вызову инструкции `call sread`.

Попробовали изменить программу, чтобы вызвать ошибку: команда `mov` всегда принимает 2 операнда (Рис. 2.5). При попытке трансляции с флагом создания файла листинга происходит ошибка, при этом файл листинга создается с указанием, где произошла ошибка (Рис. 2.6).

```
[dastarikov@fedora lab07]$ nasm -f elf lab7-2.asm
[dastarikov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[dastarikov@fedora lab07]$ ./lab7-2
10
[dastarikov@fedora lab07]$ |
```

Рис. 2.5: Измененная часть программы.

```
[dastarikov@fedora lab07]$ nasm -f elf lab7-2.asm
[dastarikov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[dastarikov@fedora lab07]$ ./lab7-2
10[dastarikov@fedora lab07]$
```

Рис. 2.6: Файл листинга указывает, где произошла ошибка.

2.2 Выполнение заданий для самостоятельной работы.

Для выполнения заданий выбран вариант 12, полученный при выполнении лабораторной работы №7.

2.2.1 Задание 1.

Написали программу (Листинг 2.5), сравнивающую 3 целых числа a, b и c. Числа вводятся с клавиатуры, в результаты выводится наибольшее из них. Проверили работу программы на числах 99, 29, 26 (Рис. 2.7).

```
[dastarikov@fedora lab08]$ ./lab8-n1-var12
Введите A: 99
Введите B: 29
Введите C: 26
Наибольшее число: 99
```

Рис. 2.7: Результат работы программы из задания 1 (Вариант 12).

2.2.2 Задание 2.

Написали программу (Листинг 2.6), вычисляющую значение выражения:

$$f(x) = \begin{cases} a * x, & x < 5 \\ x - 5, & x \geq 5 \end{cases}$$

Проверили работу программы на двух примерах: $x = 3, a = 7$; $x = 6, a = 4$ (Рис. 2.8).

```
[dastarikov@fedora lab07]$ touch variant.asm
[dastarikov@fedora lab07]$ nasm -f elf variant.asm
[dastarikov@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[dastarikov@fedora lab07]$ ./variant
Введите № студенческого билета:
1132226531
Ваш вариант: 12
[dastarikov@fedora lab07]$
```

Рис. 2.8: Результат работы программы из задания 2 (Вариант 12)

Созданные файлы *.asm скопировали в каталог ~/work/study/2022-2023/“Архитектура компьютера”/archpc/labs/lab08/ и загрузили на Github.

Листинг 2.2 Программа с использованием инструкции `jmp`

```
%include 'in_out.asm' ; подключение внешнего файла
```

```
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 ; Вывод на экран строки
```

```
call sprintf ; 'Сообщение № 1'
```

```
jmp _end
```

```
_label2:
```

```
mov eax, msg2 ; Вывод на экран строки
```

```
call sprintf ; 'Сообщение № 2'
```

```
jmp _label1
```

```
_label3:
```

```
mov eax, msg3 ; Вывод на экран строки
```

```
call sprintf ; 'Сообщение № 3'
```

```
_end:
```

```
call quit ; вызов подпрограммы завершения
```

Листинг 2.3 Программа с использованием инструкции `jmp`

```
%include 'in_out.asm' ; подключение внешнего файла
```

```
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 ; Вывод на экран строки
```

```
call sprintf ; 'Сообщение № 1'
```

```
jmp _end
```

```
_label2:
```

```
mov eax, msg2 ; Вывод на экран строки
```

```
call sprintf ; 'Сообщение № 2'
```

```
jmp _label1
```

```
_label3:
```

```
mov eax, msg3 ; Вывод на экран строки
```

```
call sprintf ; 'Сообщение № 3'
```

```
jmp _label2
```

```
_end:
```

```
call quit ; вызов подпрограммы завершения
```

Листинг 2.4 Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C.

```
%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:
; ----- Вывод сообщения 'Введите B: '
    mov eax, msg1
    call sprint
; ----- 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, 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: '
```

Листинг 2.5 Программа находит наибольшее из 3 чисел.

```
%include 'in_out.asm'

section .data
    msgA db 'Введите A: ',0h
    msgB db 'Введите B: ',0h
    msgC db 'Введите C: ',0h

    msg2 db "Наибольшее число: ",0h

section .bss
    max resb 10
    A resb 10
    B resb 10
    C resb 10

section .text
    global _start

_start:
; ----- Получение переменной A
    mov eax, msgA
    call sprint

    mov ecx,A
    mov edx,10
    call sread

    mov eax,A
    call atoi ; Вызов подпрограммы перевода символа в число
    mov [A],eax ; запись преобразованного числа в 'A'

; ----- Получение переменной B
    mov eax, msgB
    call sprint

    mov ecx,B
    mov edx,10
    call sread

    mov eax,B
    call atoi ; Вызов подпрограммы перевода символа в число
    mov [B],eax ; запись преобразованного числа в 'B'

; ----- Получение переменной C
    mov eax, msgC
    call sprint

    mov ecx,C
    mov edx,10
    call sread

    mov eax,C
```

Листинг 2.6 Программа вычисления выражения $f(x)$

```
%include 'in_out.asm'

section .data
    msgf db 'f(x) = a*x, x<5',10, 9, 'x-5, ', 'x>=5',0h
    msgx db 'Введите x: ',0h
    msga db 'Введите a: ',0h
    msg2 db 'f(x) = ',0h

section .bss
    res resb 10
    x resb 10
    a resb 10

section .text
    global _start

_start:
; ----- Вывод функции
    mov eax, msgf
    call sprintLF

; ----- Получение переменной x
    mov eax, msgx
    call sprint

    mov ecx,x
    mov edx,10
    call sread

    mov eax,x
    call atoi ; Вызов подпрограммы перевода символа в число
    mov [x],eax ; запись преобразованного числа в 'x'

; ----- Получение переменной a
    mov eax, msga
    call sprint

    mov ecx,a
    mov edx,10
    call sread

    mov eax,a
    call atoi ; Вызов подпрограммы перевода символа в число
    mov [a],eax ; запись преобразованного числа в 'a'

; ----- Проверка значения x (x<5 или part_2: x>=5)
    mov ecx,[x] ; 'ecx = x'

    cmp ecx,5 ; Сравниваем 'x' и '5'
    jge part_2 ; если 'x>=5', то переход на метку 'part_2',
    mov eax,[x] ; иначе вычисляем выражение a*x
    mov ebx,[a]
```

3 Выводы

В рамках лабораторной работы получили практические навыки использования условных и безусловных переходов, анализа файла листинга.