

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Лаптев Тимофей Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файла листинга	10
4.3	Задания для самостоятельной работы	11
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Сохранение программы	8
4.2	Запуск программы	9
4.3	Изменение программы	9
4.4	Проверка изменений	9
4.5	Проверка программы из листинга	10
4.6	Просмотр ошибки в файле листинга	11
4.7	Первая программа самостоятельной работы	12
4.8	Проверка работы первой программы	14
4.9	Вторая программа самостоятельной работы	15
4.10	Проверка работы второй программы	18

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 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы

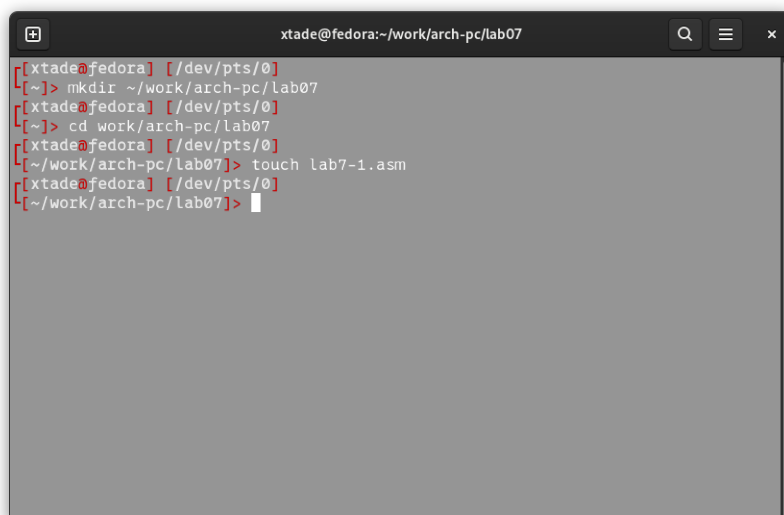
3 Теоретическое введение

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.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7. Копирую код из листинга в файл будущей программы. (рис. -fig. 4.1).



```
xtade@fedora: ~/work/arch-pc/lab07
[xtade@fedora] [/dev/pts/0]
[~]> mkdir ~/work/arch-pc/lab07
[xtade@fedora] [/dev/pts/0]
[~]> cd work/arch-pc/lab07
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> touch lab7-1.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]>
```

Рис. 4.1: Сохранение программы

При запуске программы я убедился в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. -fig. 4.2).


```

[xtade@fedora] [/dev/pts/0] [127]
[~/work/arch-pc/lab07]> nasm -f elf lab7-1.asm
ld -m elf_i386 -o lab7-1 lab7-1.o
./lab7-1
Сообщение № 2
Сообщение № 3
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]>

```

Рис. 4.2: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций. Запускаю программу и проверяю, что примененные изменения верны (рис. -fig. 4.3).

```

[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> nasm -f elf lab7-1.asm
ld -m elf_i386 -o lab7-1 lab7-1.o
./lab7-1
Сообщение № 2
Сообщение № 1
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]>

```

Рис. 4.3: Изменение программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке. Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. -fig. 4.4).

```

[~/work/arch-pc/lab07]> nvim lab7-1.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> nasm -f elf lab7-1.asm
ld -m elf_i386 -o lab7-1 lab7-1.o
./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 4.4: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга.

Программа выводит значение переменной с максимальным значением, проверяя работу программы с разными входными данными (рис. -fig. 4.5).

```

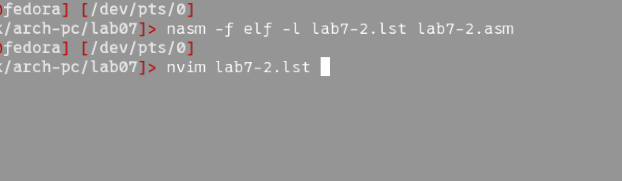
xtade@fedora:~/work/arch-pc/lab07
[xtade@fedora] [/dev/pts/0] [130]
[~/work/arch-pc/lab07]> nasm -f elf lab7-2.asm
ld -m elf_i386 -o lab7-2 lab7-2.o
./lab7-2
Введите B: 15
Наибольшее число: 50
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> ./lab7-2
Введите B: 51
Наибольшее число: 51
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> ./lab7-2
Введите B: -20
Наибольшее число: 50
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]>

```

Рис. 4.5: Проверка программы из листинга

4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm (рис. -fig. ??). Открываю его с помощью текстового редактора neovim (рис. -fig. ??).



```
xtade@fedora:~/work/arch-pc/lab07
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> nasm -f elf -l lab7-2.lst lab7-2.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> nvim lab7-2.lst
```

[illegible]

Первое значение в файле листинга - номер строки, и он может вовсе не совпа-

дать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем. В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла (рис. -fig. 4.8). Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. -fig. 4.9).

```
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:39: error: invalid combination of opcode and operands
[xtade@fedora] [/dev/pts/0] [1]
[~/work/arch-pc/lab07]>
```

```
38 0000013F 8B0D[00000000]      mov ecx,[max]
39                               cmp ecx ; Сравниваем 'max(A,C)' и 'B'.
39                               error: invalid combination of opcode and operands
40 00000145 7E0C      jg fin ; если 'max(A,C)>B', то переход на 'fin',
```

Рис. 4.6: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Выполнил вариант прошлой лабораторной работы, а именно, второй (рис. -fig. 4.7).

```

1  %include 'in_out.asm'
2
3  SECTION .data
4  msg1 db 'Введите B: ', 0h
5  msg2 db 'Наименьшее число: ', 0h
6  A dd '82'
7  C dd '61'
8
9  SECTION .bss
10 min resb 10
11 B resb 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 1: mov eax, msg1
18 2: call printf
19 3:
20 4: mov ecx, 0
21 5: mov edx, 10
22 6: call scanf
23 7:
24 8: mov eax, 0
25 9: call atoi
26 10: mov [B], eax
27 11:
28 12: mov ecx, [A]
29 13: mov [min], ecx
30 14:
31 15: cmp ecx, [C]
32 16: jb check_0
33 17: mov ecx, [C]
34 18: mov [min], ecx
35 19:
36 20: check_0:
37 21: mov eax, min
38 22: call atoi
39 23: mov [min], eax
40 24:
41 25: mov ecx, [min]
42 26: cmp ecx, [B]
43 27: jb fin
44 28: mov ecx, [B]
45 29: mov [min], ecx
46 30:
47 31: fin:
48 32: mov eax, msg2
49 33: call printf
50 34: mov eax, [min]
51 35: call printf
52 36: call quit

```

Рис. 4.7: Первая программа самостоятельной работы

Код первой программы:

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

```
SECTION .data
```

```
msg1 db 'Введите B: ', 0h
```

```
msg2 db 'Наименьшее число: ', 0h
```

```
A dd '82'
```

```
C dd '61'
```

```
SECTION .bss
```

```
min resb 10
```

```
B resb 10
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg1
```

```
call sprint
```

```
mov ecx, B
```

```
mov edx, 10
```

```
call sread
```

```
mov eax, B
```

```
call atoi
```

```
mov [B], eax
```

```
mov ecx, [A]
```

```
mov [min], ecx
```

```
cmp ecx, [C]
```

```
jg check_B
```

```
mov ecx, [C]
```

```
mov [min], ecx
```

```
check_B:
```

```
mov eax, min
```

```
call atoi
```

```
mov [min], eax
```

```
mov ecx, [min]
```

```
cmp ecx, [B]
```

```
jb fin
```

```
mov ecx, [B]
```

```
mov [min], ecx
```

```

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

```

Проверяю корректность написания первой программы (рис. -fig. 4.8).

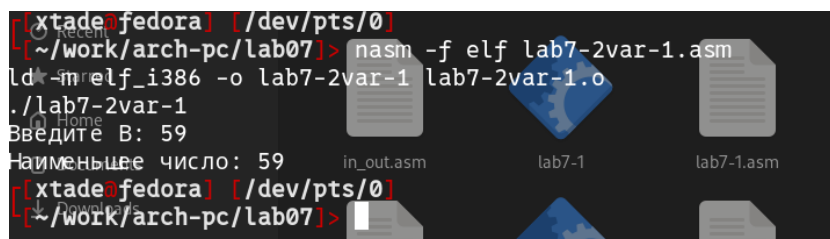


Рис. 4.8: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных а и х (рис. -fig. 4.9).

```

#include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0

SECTION .bss
x: RESB 80
a: RESB 80

SECTION .text
GLOBAL _start

_start:
    ; Запрос значения x
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    mov edi, eax ; Сохраняем значение x в edi

    ; Запрос значения a
    mov eax, msg_a
    call sprint
    mov ecx, a
    mov edx, 80
    call sread
    mov eax, a
    call atoi
    mov esi, eax ; Сохраняем значение a в esi

    ; Сравниваем x и a
    cmp edi, esi
    jl subtract_one_a ; Если x < a, переходим к subtract_one_a

    ; Если x >= a, вычитаем 1 из x
    dec edi ; edi = x - 1
    jmp print_result

subtract_one_a:
    ; Вычитаем 1 из a
    dec esi ; esi = a - 1
    mov edi, esi ; Перемещаем результат в edi для вывода

print_result:
    ; Выводим результат
    mov ecx, res
    call sprint
    mov eax, edi
    call iprintLF
    call quit

```

Рис. 4.9: Вторая программа самостоятельной работы

Код второй программы:

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

```
SECTION .data
```

```
msg_x: DB 'Введите значение переменной x: ', 0
```

```
msg_a: DB 'Введите значение переменной a: ', 0
```

```
res: DB 'Результат: ', 0
```

```
SECTION .bss
```

```
x: RESB 80
```

```
a: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
    ; Запрос значения x
```

```
    mov eax, msg_x
```

```
    call sprint
```

```
    mov ecx, x
```

```
    mov edx, 80
```

```
    call sread
```

```
    mov eax, x
```

```
    call atoi
```

```
    mov edi, eax ; Сохраняем значение x в edi
```

```
    ; Запрос значения a
```

```
    mov eax, msg_a
```

```
    call sprint
```



```

mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax ; Сохраняем значение a в esi

; Сравниваем x и a
cmp edi, esi
jl subtract_one_a ; Если x < a, переходим к subtract_one_a

; Если x >= a, вычитаем 1 из x
dec edi ; edi = x - 1
jmp print_result

subtract_one_a:
; Вычитаем 1 из a
dec esi ; esi = a - 1
mov edi, esi ; Перемещаем результат в edi для вывода

print_result:
; Выводим результат
mov ecx, res
call sprint
mov eax, edi
call iprintLF
call quit

```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений a и x (рис. -fig. 4.10).

```
xtade@fedora:~/work/arch-pc/lab07
zsh: no such file or directory: ./lab7-2var-2
[xtade@fedora] [/dev/pts/0] [127]
[~/work/arch-pc/lab07]> nasm -f elf lab7-2var-2.asm
ld -m elf_i386 -o lab7-2var-2 lab7-2var-2.o
./lab7-2var-2
Введите значение переменной x: 13
Введите значение переменной a: 12
Результат: 12
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> ./lab7-2var-2
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> ./lab7-2var-2
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab07]> 
```

Рис. 4.10: Проверка работы второй программы

5 Выводы

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.