

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Кириянова Екатерина Андреевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файла листинга	11
4.3	Задание для самостоятельной работы	13
5	Вывод	15
6	Список литературы	16

Список иллюстраций

4.1	Создание	8
4.2	Программа	8
4.3	Запуск	9
4.4	Редактирование	9
4.5	Запуск	9
4.6	Редактирование	10
4.7	Запуск	10
4.8	Создание	10
4.9	Редактирование	11
4.10	Проверка	11
4.11	Листинг	11
4.12	Открытие листинга	12
4.13	Трансляция	12
4.14	Ошибка	12
4.15	Программа 1	13
4.16	Запуск	13
4.17	Программа 2	14
4.18	Запуск	14

1 Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. Познакомиться с назначением и структурой файла листинга.

2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Задание для самостоятельной 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.

Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление: `jmp` Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода.

Для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора.

Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания: `cmp` , Команда `cmp`, так же как и команда вычитания, выполняет вычитание - , но результат вычитания никуда не

записывается и единственным результатом команды сравнения является формирование флагов.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию. Структура листинга: номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы); адрес — это смещение машинного кода от начала текущего сегмента; машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция `int 80h` начинается по смещению `00000020` в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция `int 80h` ассемблируется в `CD80` (в шестнадцатеричном представлении); `CD80` — это инструкция на машинном языке, вызывающая прерывание ядра); исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю новый каталог и файл в нем (рис. 4.1).

```
eakirjyanova@user:~$ mkdir -p /work/arch-pc/lab07
eakirjyanova@user:~$ cd /work/arch-pc/lab07
eakirjyanova@user:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 4.1: Создание

Ввожу текст программы из листинга 7.1 (рис. 4.2)

```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab07/lab7-1.asm *
%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'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Программа

Транслирую текст программы в объектный файл, выполняю компоновку объектного файла и запускаю исполняемый файл (рис. 4.3).


```
eakirjyanova@user:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
eakirjyanova@user:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
eakirjyanova@user:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 4.3: Запуск

Изменяю текст программы в соответствии с листингом 7.2 (рис. 4.4).

```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab07/lab7-1.asm *
%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:
```

Рис. 4.4: Редактирование

Создаю обновленный исполняемый файл и запускаю его (рис. 4.5).

```
eakirjyanova@user:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
eakirjyanova@user:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
eakirjyanova@user:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
eakirjyanova@user:~/work/arch-pc/lab07$
```

Рис. 4.5: Запуск

Меняю текст программы для нужного вывода программы (рис. 4.6).

```

GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab07/lab7-1.asm *
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

```

Рис. 4.6: Редактирование

Транслирую текст программы в объектный файл, выполняю компоновку объектного файла и запускаю исполняемый файл (рис. 4.7).

```

eakirjyanova@user:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
eakirjyanova@user:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
eakirjyanova@user:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 4.7: Запуск

Создаю новый файл lab7-2.asm (рис. 4.8).

```

eakirjyanova@user:~/work/arch-pc/lab07$ touch lab7-2.asm
eakirjyanova@user:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm

```

Рис. 4.8: Создание

Ввожу текст программы из листинга 7.3 (рис. 4.9).

```

GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab07/lab7-2.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:
; ----- 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'

```

Рис. 4.9: Редактирование

Создаю исполняемый файл и проверяю его работу для разных значений B (рис. 4.10).

```

eakirjyanova@user:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
eakirjyanova@user:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
eakirjyanova@user:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 27
Наибольшее число: 50
eakirjyanova@user:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 113
Наибольшее число: 113
eakirjyanova@user:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
eakirjyanova@user:~/work/arch-pc/lab07$

```

Рис. 4.10: Проверка

4.2 Изучение структуры файла листинга

Создаю файл листинга для программы из файла lab7-2.asm (рис. 4.11).

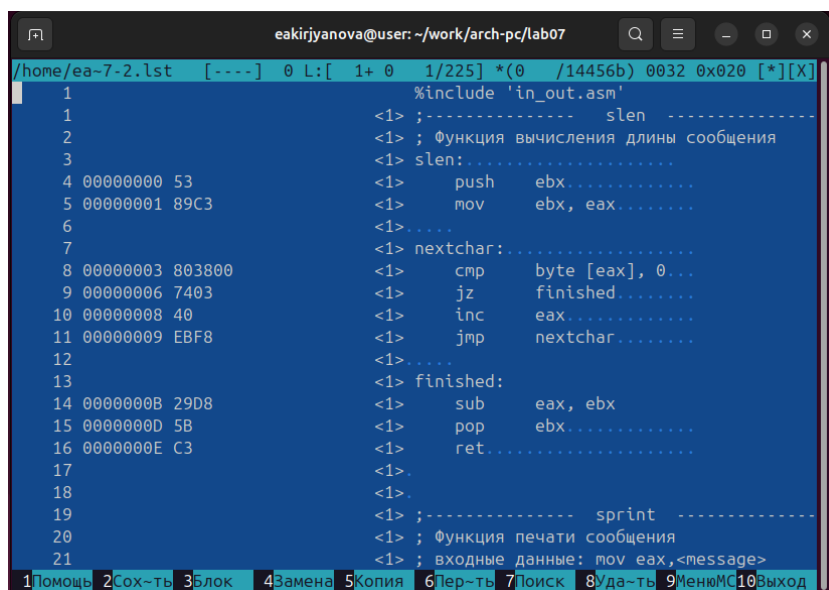
```

eakirjyanova@user:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
eakirjyanova@user:~/work/arch-pc/lab07$

```

Рис. 4.11: Листинг

Открываю файл листинга с помощью mcedit (рис. 4.12).

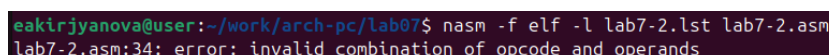


```
1 %include 'in_out.asm'
2 <1> ;----- slen -----
3 <1> ; Функция вычисления длины сообщения
4 <1> slen:.....
5 00000000 53 <1> push ebx.....
6 00000001 89C3 <1> mov ebx, eax.....
7 <1>.....
8 <1> nextchar:.....
9 00000003 803800 <1> cmp byte [eax], 0...
10 00000006 7403 <1> jz finished.....
11 00000008 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
<1> ; входные данные: mov eax, <message>
```

Рис. 4.12: Открытие листинга

Строка на 14 месте имеет адрес '0000000B' и машинный код 29D8, sub eax, ebx выполняет вычитание значение ebx из значения eax. Строка на 15 месте имеет адрес '0000000D' и машинный код 5B, pop ebx сохраняет значение в регистре ebx. Строка на 16 месте '0000000E' и машинный код C3, ret используется для возврата из подпрограммы.

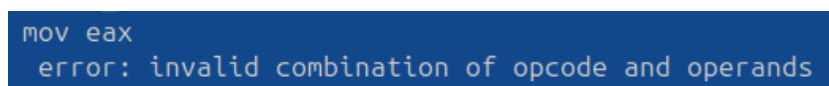
Убираю один операнд из двух и трансляция не работает, указывая на отсутствие операнда (рис. 4.13).



```
eakirjyanova@user:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
```

Рис. 4.13: Трансляция

В листинге отображается ошибка (рис. 4.14).

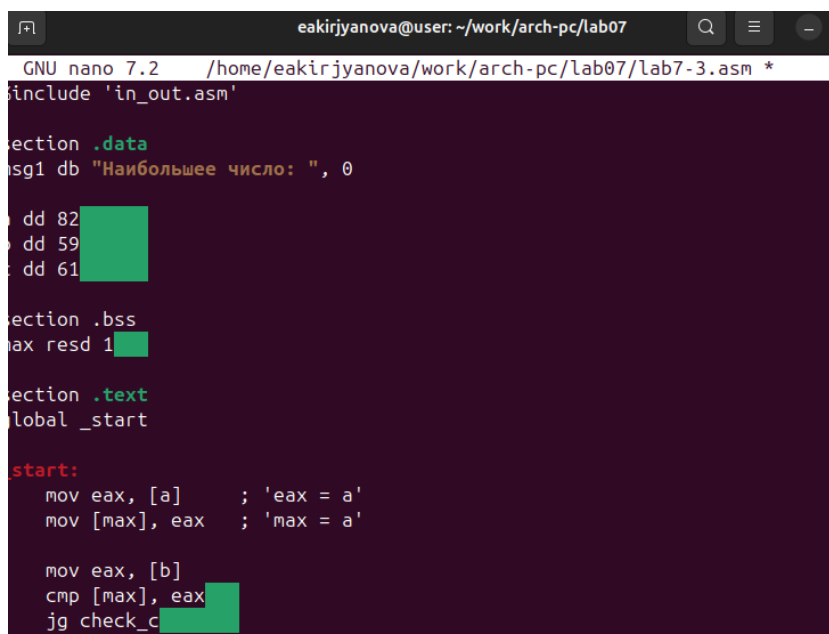


```
mov eax
error: invalid combination of opcode and operands
```

Рис. 4.14: Ошибка

4.3 Задание для самостоятельной работы

Пишу программу для нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбираю согласно 2 варианту (рис. 4.15).



```
eakirjyanova@user: ~/work/arch-pc/lab07
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab07/lab7-3.asm *
#include 'in_out.asm'

section .data
msg1 db "Наибольшее число: ", 0

dd 82
dd 59
dd 61

section .bss
max resd 1

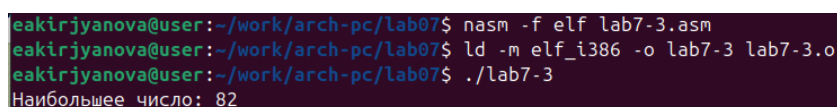
section .text
global _start

_start:
mov eax, [a] ; 'eax = a'
mov [max], eax ; 'max = a'

mov eax, [b]
cmp [max], eax
jg check_c
```

Рис. 4.15: Программа 1

Создаю исполняемый файл и запускаю. Программа работает корректно (рис. 4.16).



```
eakirjyanova@user:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
eakirjyanova@user:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
eakirjyanova@user:~/work/arch-pc/lab07$ ./lab7-3
Наибольшее число: 82
```

Рис. 4.16: Запуск

Пишу программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции выбираю из 2 варианта (рис. 4.17).

```
eakirjyanova@user: ~/work/arch-pc/lab07
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab07/lab7-4.asm
#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:
    ; Ввод значения переменной x
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
```

Рис. 4.17: Программа 2

Создаю исполняемый файл и запускаю. Программа работает корректно (рис. 4.18).

```
eakirjyanova@user:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
eakirjyanova@user:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
eakirjyanova@user:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 6
eakirjyanova@user:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 5
```

Рис. 4.18: Запуск

5 Вывод

В ходе выполнения данной лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов и познакомилась с назначением и структурой файла листинга.

6 Список литературы

1. Лабораторная работа №7