Отчёта по лабораторной работе №7

Дисциплина: архитектура компьютера

Худдыева Дженнет

Содержание

Список иллюстраций

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 файлы листинга.

3.Задания для самостоятельной работы.

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды переходов.Можно выделить 2 типа переходов:

- условный переход-выполнение или не выполнение переходав определённую точку программы в зависимости от проверки условия.
- безусловный переход-выполнение передачи управления в определённую точку программы без каких-либо условий.

Безусловный переход выполняется инструкцией jmp.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 команды вычитания.

Листинг (в аппарате NASM) - это один из выходных файлов, так как кроме строк самой программы он содержит дополнительную информайию

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю каталог для программы лабораторной работы №7,перехожу в него и создаю файл lab7-1.asm (рис. [4.11]).

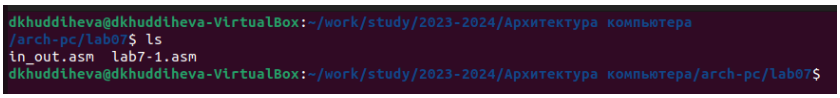

Рис. 4.1: Создание файла

Ввожу файл lab7-1.asm текст программы из листинга 7.1. (рис. [4.2]).

```
%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 sprintLF    ; Сообщение №1

 _label2:
mov eax, msg2   ; Вывод на экран строки
call sprintLF   ; Сообщение №2

 _label3:
mov eax, msg3   ; Вывод на экран строки
call sprintLF    ; Сообщение №3

 _end:
   call quit
```

Рис. 4.2: Ввод текста

Создаю исполняемый файл и запускаю его (рис. [4.3]).

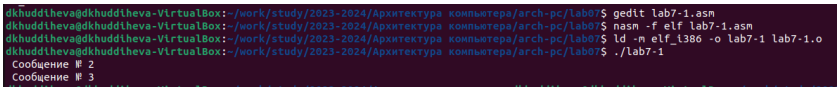

Рис. 4.3: Запуск исполняемого кода

Таким образом,использование инструкции jmp _label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки _label2, пропустив вывод первого сообщения.

Изменяю программу таким образом,чтобы она выводила сначала ‘Сообщение №2’, потом ‘Сообщение №1’ и завершала работу.Для этого изменяю текст программы в соответствии с листингом 7.2 (рис. [4.4]).

```
1 %include 'in_out.asm'      ; Подключение внешнего файла
2 SECTION .data
3 msg1:  DB 'Сообщение № 1',0
4 msg2:  DB 'Сообщение № 2',0
5 msg3:  DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8  _start:
9 jmp _label2
10
11  _label1:
12 mov eax, msg1    ; Вывод на экран строки
13 call sprintLF    ; Сообщение №1
14 jmp _end
15
16  _label2:
17 mov eax, msg2   ; Вывод на экран строки
18 call sprintLF   ; Сообщение №2
19 jmp _label1
20
21  _label3:
22 mov eax, msg3   ; Вывод на экран строки
23 call sprintLF    ; Сообщение №3
24
25  _end:
26    call quit
```

Рис. 4.4: Изменение текста программы

Создаю исполняемый файл и проверяю его.(рис. [4.5]).

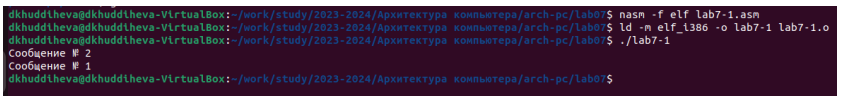

Рис. 4.5: Создание исполняемого файла

Изменяю текст программы, добавив в начале программы jmp _label3, jmp _label2 в конце метки jmp _label3, jmp _label1 добавляю в конце метки jmp _label2, и добавляю jmp _end в конце метки jmp _label1 (рис. [4.6]).

```
%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 sprintLF    ; Сообщение №1
jmp _end

 _label2:
mov eax, msg2   ; Вывод на экран строки
call sprintLF   ; Сообщение №2
jmp _label1

 _label3:
mov eax, msg3   ; Вывод на экран строки
call sprintLF    ; Сообщение №3
jmp _label2

 _end:
   call quit
```

Рис. 4.6: изменение текста программы

Вывод программы будет таким образом:(рис. [4.7]).

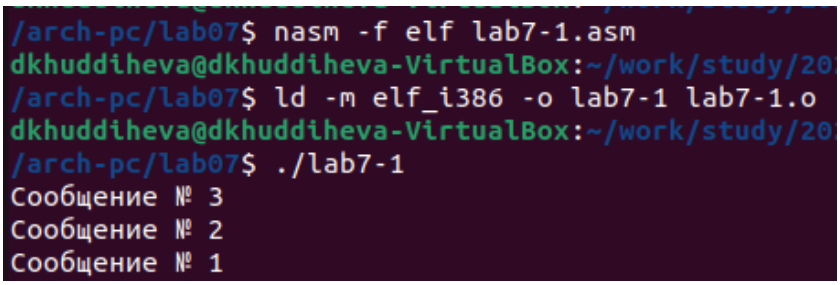

Рис. 4.7: Вывод программы

Рассмотрим программу,которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B,C.Значение для A и C задаются в программе, значение B вводиться с клавиатуры.Создаю файл lab7-2.asm (рис. [4.8]).

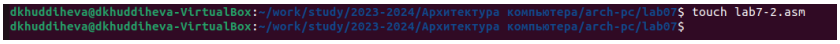

Рис. 4.8: Создание файла

Текст программы из листинга 7.3 ввожу в lab7-2.asm (рис. [4.9]).

```
%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)
```

Рис. 4.9: Ввод текста программы

Создаю исполняемый файл и проверяю его (рис. [4.10]).

```
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 55
Наибольшее число: 55
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.10: Проверка работы файла

4.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm (рис. [**??**]).

```
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ mcedit lab7-2.lst
```

Рис. 4.11: Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора и внимательно изучаю его формат и содержимое.((рис. [4.12]).

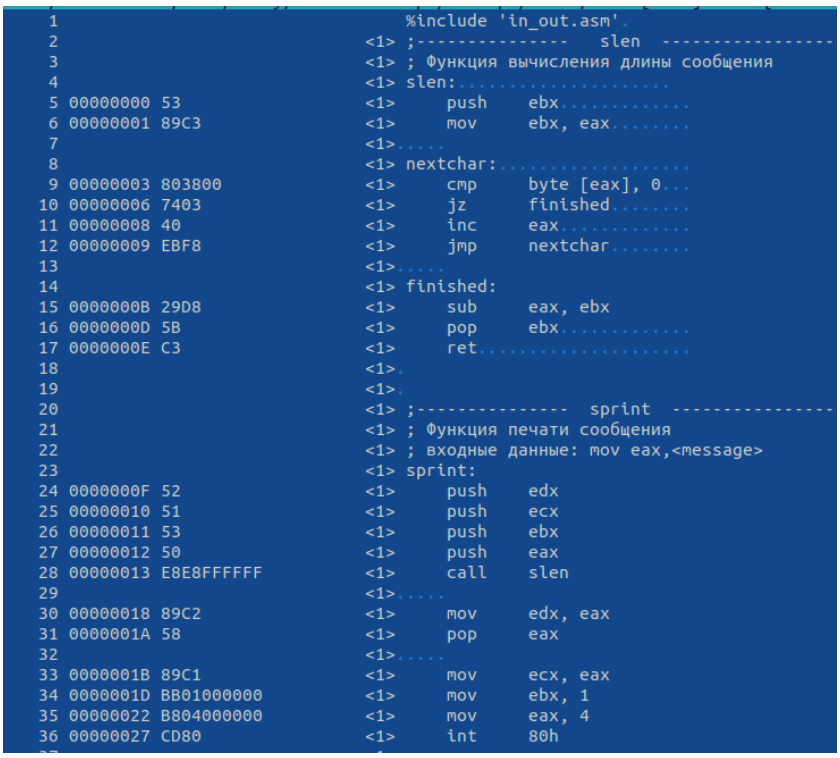

Рис. 4.12: Изучение файла листинга

В представленных трёх строчках содержаться следующие данные: (рис. [4.13]).

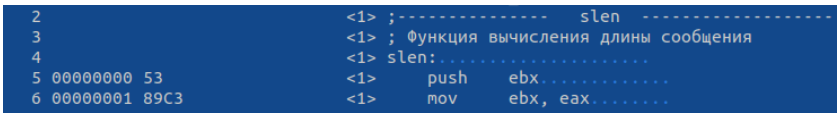

Рис. 4.13: Выбранные строки файла

“3”-номер строки кода, ;Функция вычисления длинны сообщения - комментарий к коду,не имеет адреса и машинного кода.

“4”-номер строки кода,“slen”- название функции, не имеет адреса и машинного кода.

“5”-номер строки кода, “00000000” -адрес строки,“53” - машинный код, “push

ebx”- исходный текст программы, инструкция “push” помещает операнд “ebx” в стек.

Открываю файл с программой lab7-2.asm и выбранной мной инструкции с двумя операндами удаляю выделенный операнд.(рис. [4.14]).

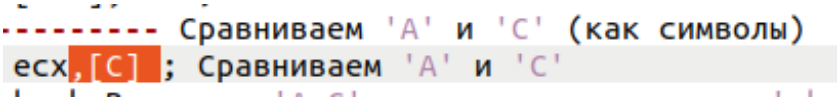

Рис. 4.14: Удаление выделенного операнда из кода

Выполняю трансляцию с получением файла с листинга (рис. [4.15]).

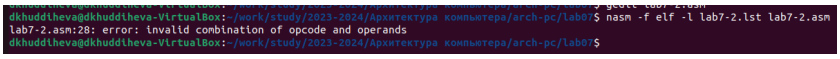

Рис. 4.15: Получение файла листинга

4.3 Задания для самостоятельной работы

1.Пишу программу нахождения наименьшей из 3 целочисленных переменных a,b,c.Значения переменных выбираю из табл. 7.5 в соотвествии с вариантом, полученным при выполнении лабораторной работы №7.Мой вариант под номером 17. (рис. [4.17]).

```
%include 'in_out.asm'
section .data
msg db "Наименьшее число: ",0h
A dd '12'
B dd '26'
C dd '68'
section .bss
min resb 10
section .text
global _start
_start:
; - - - - - - Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx=A'
mov [min],ecx  ; 'min=A'
; - - - - - - Сравниваем 'A' и 'C'(как символы)
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:
mov eax,min
call atoi  ;Вызов подпрограммы перевода символа в число
mov [min],eax  ; запись преобразованного числа  в 'min'
; - - - - - - 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, msg
call sprint
mov eax,[min]
call iprintLF
call quit
```

Рис. 4.16: Написание программы

Создаю исполняемый файл и проверяю его работу (рис. [**??**]).

```
Наименьшее число: 26
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/
/arch-pc/lab07$ nasm -f elf task1.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/
/arch-pc/lab07$ ld -m elf_i386 -o task1 task1.o
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/study/
/arch-pc/lab07$ ./task1
Наименьшее число: 12
```

Рис. 4.17: Запуск файла и проверка

5 Выводы

Я изучила команды условного и безусловного переходов и приобрела навыки написания программ с использованием переходов, а также познакомилась с назначением и структурой файла листинга.

Список литературы

Лабораторная работа №7