Отчёт по лабораторной работе №7

Дисциплина: Комьютерные технологии и технологии программирования

Дымовой Д.Д.

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 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: 1) условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. 2) безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

3 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы №7, перехожу в него и создаю файл lab7-1.asm (рис. 3.1).

```
dddihmova@dk3n33 ~ $ mkdir ~/work/arch-pc/lab07
dddihmova@dk3n33 ~ $ cd ~/work/arch-pc/lab07
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 3.1: Создание каталога и файла

Ввожу текст программы листинга 7.1 (рис. 3.2).

```
/afs/.dk.sci.pfu.edu.ru/home/d/d/dd~mova/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 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 ; вызов подпрограммы завершения
```

Рис. 3.2: Листинг 7.1

Создаю исполняемый файл и запускаю его (рис. 3.3).

```
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 3.3: Трансляция и компановка

Ввожу текст программы листинга 7.2 (рис. 3.4).

```
lab7-1.asm          [----]  0 L:[  1+22  23/ 29] *(572 / 776b) 0010 0x00A
%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'
    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'

    _end:
    call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Листинг 7.2

Создаю исполняемый файл и запускаю его (рис. 3.5).

```
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 3.5: Трансляция и компановка

Изменяю программы согласно условиям задачи (рис. 3.6).

```
/afs/.dk.sci.pfu.edu.ru/home/d/d/dd~mova/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 _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 ; вызов подпрограммы завершения
```

Рис. 3.6: Мои изменения

Создаю исполняемый файли запускаю его (рис. 3.7).

```
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 3.7: Трансляция и компановка

Создаю с помощью команды touch файл lab7-2.asm (рис. 3.8).

```
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ touch lab7-2.asm
```

Рис. 3.8: Создание файла

Ввожу текст программы листинга 7.3 (рис. 3.9).

```
/afs/.dk.sci.pfu.edu.ru/home/d/d/dddihmova/work/arch-pc/lab07/lab7-2.
%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: '
    mov eax,[max]
    call iprintLF ; Вывод 'max(A,B,C)'
    call quit ; Выход
```

Рис. 3.9: Листинг 7.3

Создаю исполняемый файли запускаю его, проверяю работу программы на различных числах (рис. 3.10).

```
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 1
Наибольшее число: 50
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 30
Наибольшее число: 50
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 54
Наибольшее число: 54
```

Рис. 3.10: Трансляция и компановка, проверка работы

Получаю файл листинга, открываю его (рис. **??**).

```
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
dddihmova@dk3n33 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst
```

Рис. 3.11: Создание файла листинга

Я попробую объснить этот фрагмент программы (рис. 3.12).

```
   12                              <1>
   13                              <1> finished:
   14 0000000B 29D8                <1>     sub     eax, ebx
   15 0000000D 5B                  <1>     pop     ebx
   16 0000000E C3                  <1>     ret
   17                              <1>
   18                              <1>
```

Рис. 3.12: Фрагмент листинга

Первая цифра это номер строки, далее идёт адрес, а затем машинный код в 16-ричной последовательности, после этого исходный текст программы.

В моём фрагменте: 14 - номер строки, 0000000B - адрес, 29D8 - машинный код программы sub eax, ebx.

15 - номер строки, 0000000D - адрес, 5B - машинный код программы pop ebx.

16 - номер строки, 0000000E - адрес, C3 - машинный код программы ret.

В данном фрагменте (mov [max], eax) удаляю eax (рис. 3.13).

```
    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], ; запись преобразованного числа в 'max'
; ---------- Сравниваем 'max(A,C)' и 'B' (как числа)
    mov ecx,[max]
```

Рис. 3.13: Фрагмент программы

Выполняю трансляцию с получением файла листинга (рис. 3.14).

```
dddihmova@dk3n62 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:37: error: invalid combination of opcode and operands
```

Рис. 3.14: Трансляция

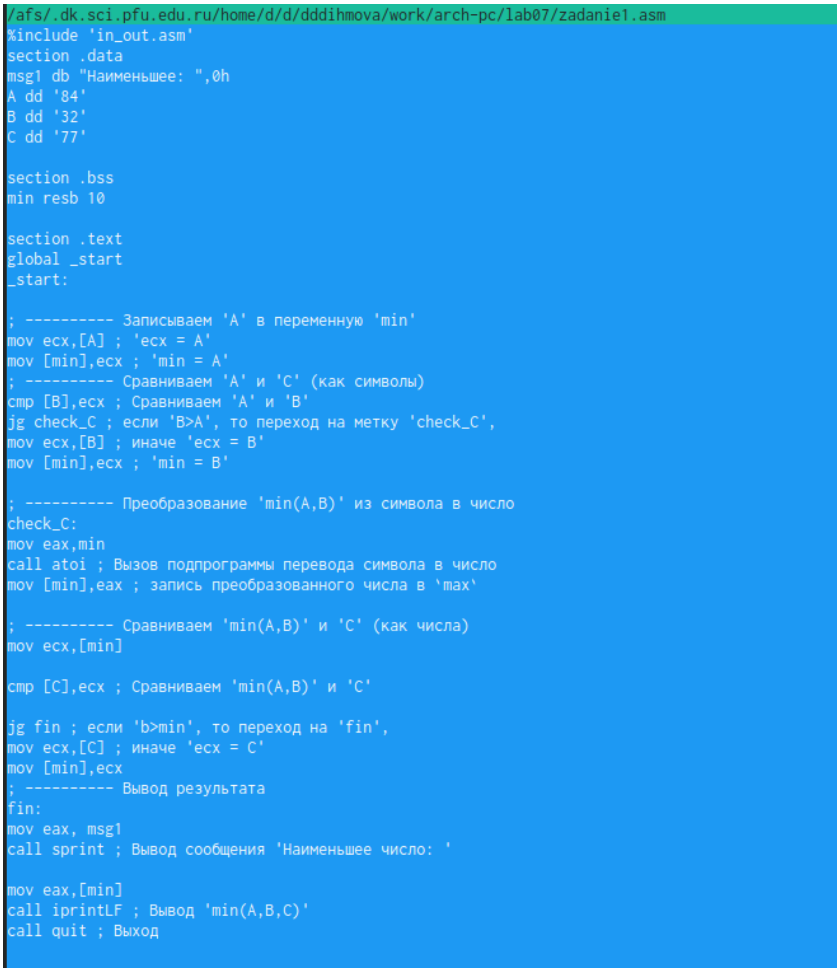
Выходным файлом будет толкьо файл с листингом. В листинге добавляется сообщение об ошибке (рис. 3.15).

```
34                              check_B:
35 00000130 B8[00000000]            mov eax,max
36 00000135 E862FFFFFF              call atoi ; Вызов подпрограммы перевода символа в число
37                                  mov [max] ; запись преобразованного числа в `max`
37          ****************       error: invalid combination of opcode and operands
```

Рис. 3.15: Ошибка

#Задание для самостоятельной работы

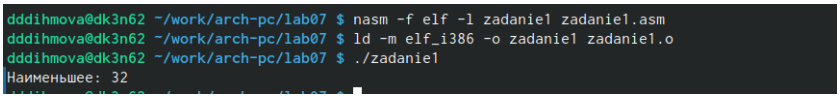
Пишу программу согласно условиям 13 варианта (рис. 3.16).



```
/afs/.dk.sci.pfu.edu.ru/home/d/d/dddihmova/work/arch-pc/lab07/zadanie1.asm
%include 'in_out.asm'
section .data
msg1 db "Наименьшее: ",0h
A dd '84'
B dd '32'
C dd '77'

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 [B],ecx ; Сравниваем 'A' и 'B'
jg check_C ; если 'B>A', то переход на метку 'check_C',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx ; 'min = B'

; ---------- Преобразование 'min(A,B)' из символа в число
check_C:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в `max`

; ---------- Сравниваем 'min(A,B)' и 'C' (как числа)
mov ecx,[min]

cmp [C],ecx ; Сравниваем 'min(A,B)' и 'C'

jg fin ; если 'b>min', то переход на 'fin',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx
; ---------- Вывод результата
fin:
mov eax, msg1
call sprint ; Вывод сообщения 'Наименьшее число: '

mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход
```

Рис. 3.16: Программа

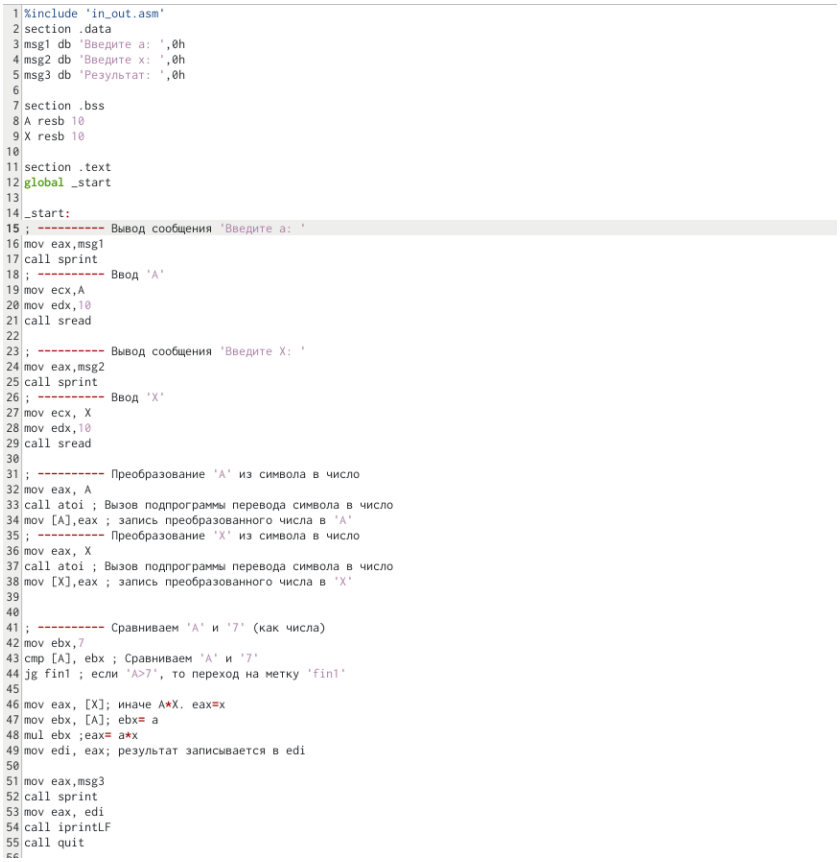
Создаю исполняемый файл и запускаю его (рис. 3.17).



```
dddihmova@dk3n62 ~/work/arch-pc/lab07 $ nasm -f elf -l zadanie1 zadanie1.asm
dddihmova@dk3n62 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o zadanie1 zadanie1.o
dddihmova@dk3n62 ~/work/arch-pc/lab07 $ ./zadanie1
Наименьшее: 32
```

Рис. 3.17: Запуск файла

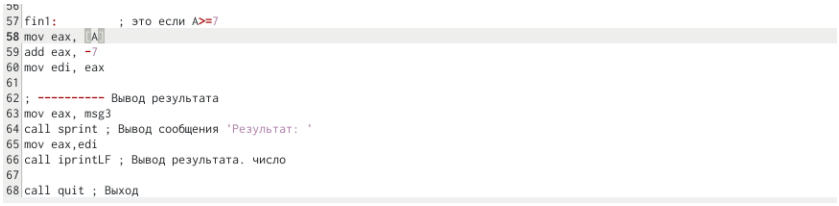
Пишу программу согласно условиям 13 варианта (рис. 3.18).



```
%include 'in_out.asm'
section .data
msg1 db 'Введите a: ',0h
msg2 db 'Введите x: ',0h
msg3 db 'Результат: ',0h

section .bss
A resb 10
X resb 10

section .text
global _start

_start:
; ---------- Вывод сообщения 'Введите a: '
mov eax,msg1
call sprint
; ---------- Ввод 'A'
mov ecx,A
mov edx,10
call sread

; ---------- Вывод сообщения 'Введите X: '
mov eax,msg2
call sprint
; ---------- Ввод 'X'
mov ecx, X
mov edx,10
call sread

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


; ---------- Сравниваем 'A' и '7' (как числа)
mov ebx,7
cmp [A], ebx ; Сравниваем 'A' и '7'
jg fin1 ; если 'A>7', то переход на метку 'fin1'

mov eax, [X]; иначе A*X. eax=x
mov ebx, [A]; ebx= a
mul ebx ;eax= a*x
mov edi, eax; результат записывается в edi

mov eax,msg3
call sprint
mov eax, edi
call iprintLF
call quit
```

Рис. 3.18: Первая часть программы

Продолжение программы (рис. 3.19).



```

fin1:         ; это если A>=7
mov eax, [A]
add eax, -7
mov edi, eax

; ---------- Вывод результата
mov eax, msg3
call sprint ; Вывод сообщения 'Результат: '
mov eax,edi
call iprintLF ; Вывод результата. число

call quit ; Выход
```

Рис. 3.19: Вторая часть программы

Создаю исполняемый файл и запускаю его с указанными значениями x и a (рис. 3.20).

Рис. 3.20: Запуск файла

Сохраняю все данные и отправляю результаты на git hub.

4 Выводы

Я изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. Познакомилась с назначением и структурой файла листинга.

Список литературы