Лабораторная работа №7

Дисциплина: Архитектура компьютера

Савостин Олег

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целью данной лабораторной работы является 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. Выполнение заданий 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

Как отмечалось выше, 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 условного перехода имеет вид j label Мнемоника перехода связана со

значением анализируемых флагов или со способом фор- мирования этих флагов

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Сперва создаю нужный для работы файл lab7-1.asm(рис. 4.1).

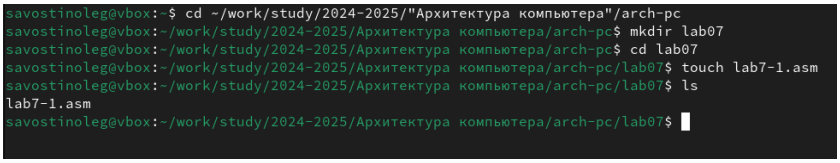

Рис. 4.1: Создание нужного файла.

Вставляю в него листинг 7.1 из файла на ТУИС(рис. 4.2). Создаю исполняемый файл и запускаю его. (рис. 4.3).

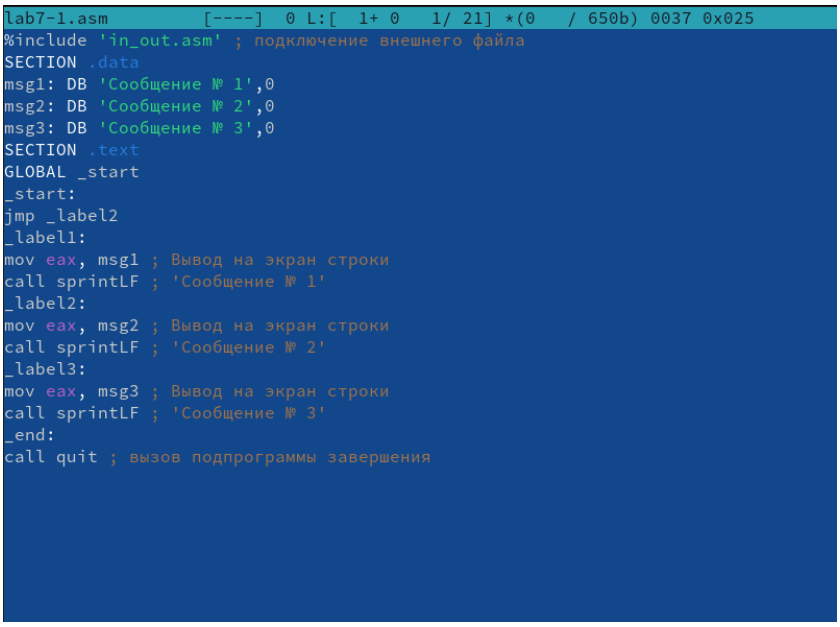

Рис. 4.2: Текст Листинга в файле

```
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.3: Исполняемый файл Листинга 7.1

Изменяю текст файла так, чтобы выводились: 2, 1. Вставляю текст в файл из Листинга 7.2 (рис. 4.4). Создаю исполняемый файл и запускаю его. (рис. 4.5).

```
lab7-1.asm          [----]  0 L:[  1+20  21/ 23] *(596 / 671b) 0095 0x05F
%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. 4.4: Текст Листинга 7.2 в файле

```
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 4.5: Исполняемый файл Листинга 7.2

Теперь изменяю файл так, чтобы выводились все сообщения по порядку возрастания. Ввожу код (рис. 4.6). Затем создаю исполняемый файл и проверяю на правильность (рис. 4.7).

```
lab7-1.asm          [----]  0 L:[  1+20  21/ 24] *(596 / 683b) 0106 0x06A
%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. 4.6: Новый код.

```
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 4.7: Проверка на правильность кода

Создаю новый файл и вставляю в него текст из Листинга 7.3 (рис. 4.8).

```
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch lab7-2.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07 — mcedit lab7-2.asm
lab7-2.asm          [----] 11 L:[  1+ 7   8/ 50] *(159 /1744b) 0010 0x00A          [*][X]
%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`
```

Рис. 4.8: Lab7-3.asm и текст из Листинга 7.3

Создаю исполняемый файл(рис. 4.9) и проверяю что он делает (рис. 4.10).

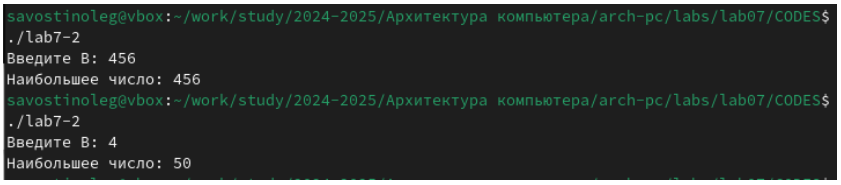

Рис. 4.9: Создание исполняемого файла

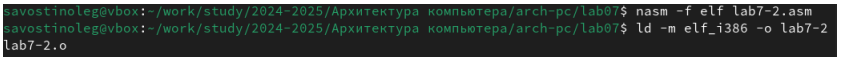

Рис. 4.10: Запуск данного файла

Для углубленного изучения файла, я создаю lab7-2.lst и изучаю содержимое(рис. 4.11).

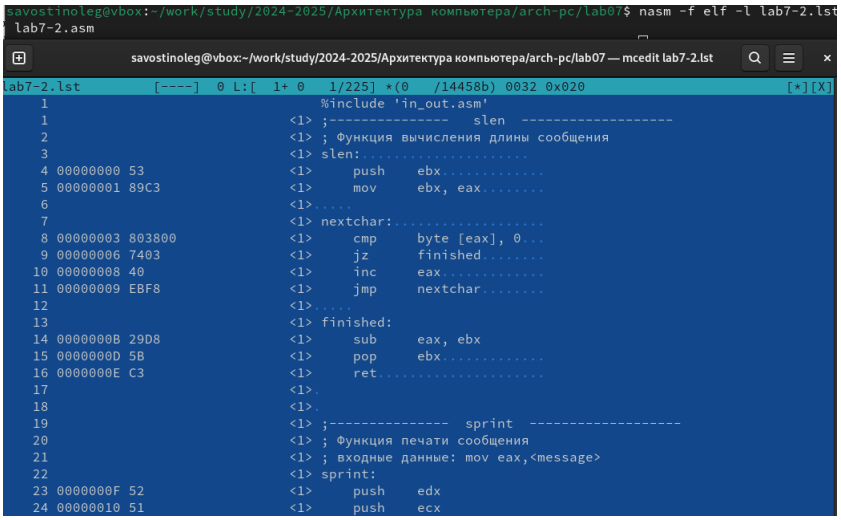

Рис. 4.11: lab7-2.lst

В данной строчке указан регистр на строчке 21. Её адрес - 00000101. Машинный код - B8[0A000000]. строчка содержит код mov eax,B, который переводит значение B в регистр (рис. 4.12).

Рис. 4.12: Первая строчка.

В данной строчке указан регистр на строчке 23. Её адрес - 0000010B. Машинный код - A3[0A000000]. строчка содержит код mov [B], который переводит значение регистра в [B] (рис. 4.13).

```
  23 0000010B A3[0A000000]                mov [B],eax ;
```

Рис. 4.13: Вторая строчка.

В данной строчке указан регистр на строчке 22. Её адрес - 00000106. Машинный код - E891FFFFFF. строчка содержит код call atoi который переводит символ, лежащий выше, в число(рис. 4.14).

```
  22 00000106 E891FFFFFF                  call atoi ;
```

Рис. 4.14: Третья строчка.

Удаляю строчку mov ecx,[max] (рис. 4.15). Пытаюсь создать исполняемый файл и .lst и получаю ошибку (рис. 4.16). Создается только lst. Открываю lst файл и нахожу ошибку, которая добавилась.(рис. 4.17).

Рис. 4.15: Строчка, которая будет удалена

```
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst
 lab7-2.asm
lab7-2.asm:38: error: invalid combination of opcode and operands
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.16: Попытка создать исполняемый файл

```
38                              mov ecx,
38          ******************      error: invalid combination of opcode and operands
```

Рис. 4.17: Ошибка в листинге

4.2 Выполнение заданий для самостоятельной работы

1. Сперва, пишу код который будет находить наименьшее число.(рис. 4.18).

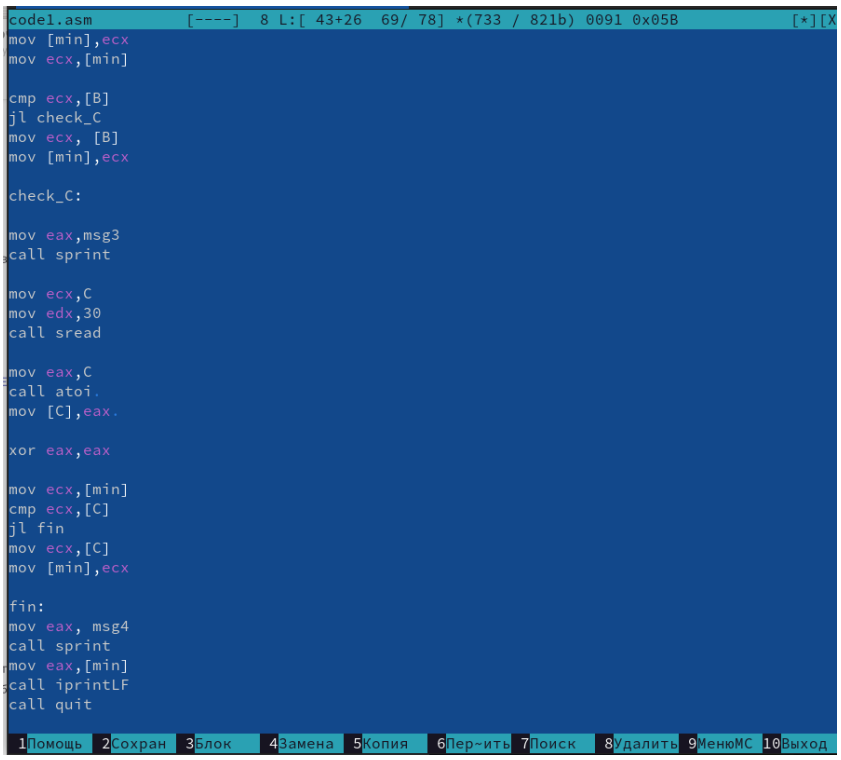

Рис. 4.18: Первый код

Создаю исполняемый, всё верно. (рис. 4.19).

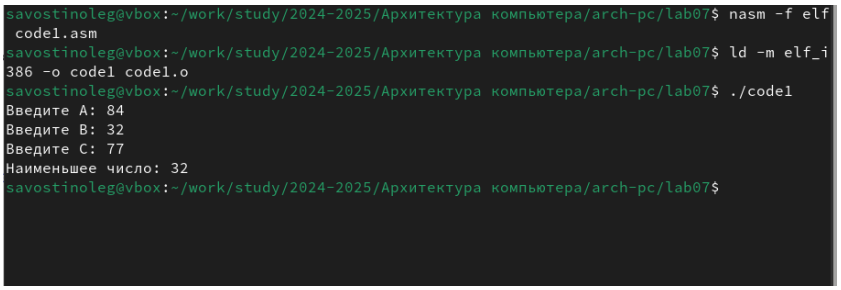

Рис. 4.19: Запуск исполняемого файла

КОД %include ‘in_out.asm’

section .data

msg1 db ‘Введите A:’,0h

msg2 db ‘Введите B:’,0h

msg3 db ‘Введите C:’,0h

msg4 db “Наименьшее число:”,0h

```
section .bss
min resb 30
A resb 30
B resb 30
C resb 30
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,A
mov edx,30
call sread
mov eax,A
call atoi
mov [A],eax
xor eax,eax
mov eax,msg2
call sprint
mov ecx,B
mov edx,30
call sread
mov eax,B
call atoi
mov [B],eax
xor eax,eax
mov ecx, [A]
mov [min],ecx
mov ecx,[min]
```

```
cmp ecx,[B]
jl check_C
mov ecx, [B]
mov [min],ecx
check_C:
mov eax,msg3
call sprint
mov ecx,C
mov edx,30
call sread
mov eax,C
call atoi
mov [C],eax
xor eax,eax
mov ecx,[min]
cmp ecx,[C]
jl fin
mov ecx,[C]
mov [min],ecx
fin:
mov eax, msg4
call sprint
mov eax,[min]
call iprintLF
call quit
```

2. Теперь записываю код для следующей функции: a-7 a>=7 | ax a<7 (рис. 4.20).

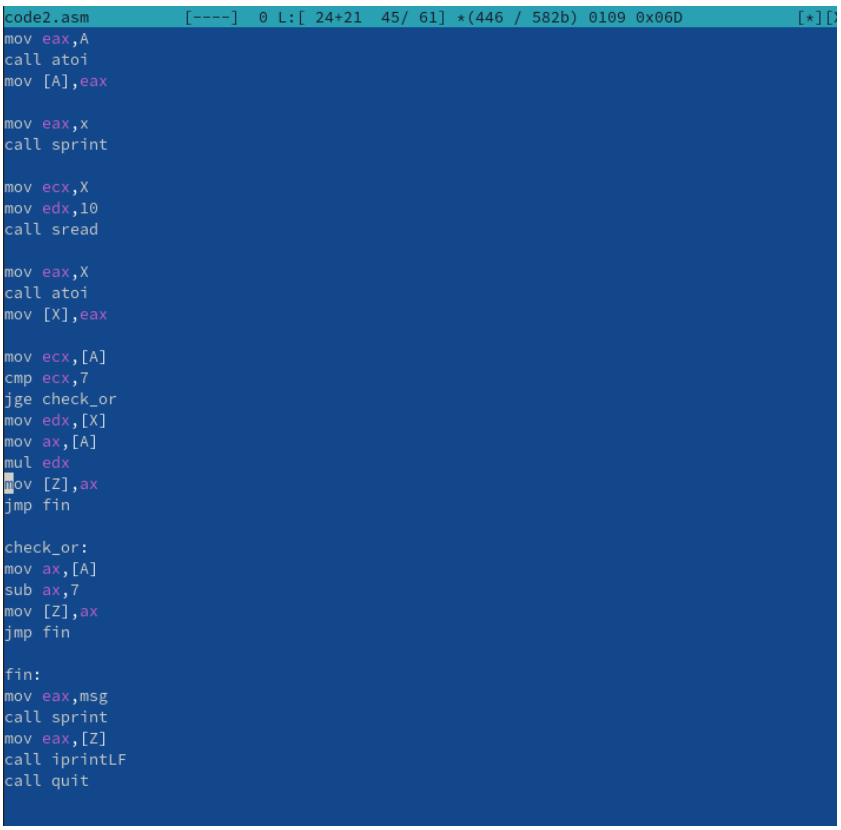

Рис. 4.20: Код второй

Проверяю на правильность написанного кода. Всё верно (рис. 4.21).

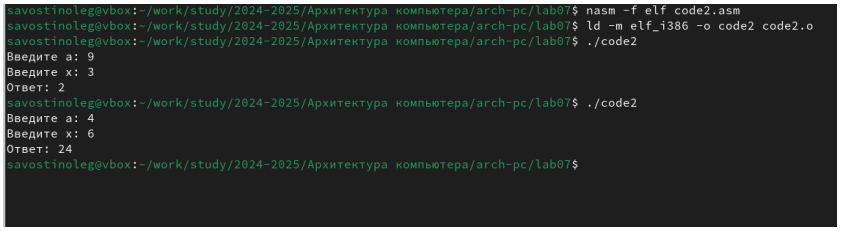

Рис. 4.21: Запуск исполняемого файла второго кода

КОД

%include ‘in_out.asm’

section .data

a db ‘Введите a:’,0h

x db ‘Введите x:’,0h

msg db “Ответ:”,0h

```
section .bss
X RESB 10
A RESB 10
Z RESB 10
section .text
global _start
_start:
mov eax,a
call sprint
mov ecx,A
mov edx,10
call sread
mov eax,A
call atoi
mov [A],eax
mov eax,x
call sprint
mov ecx,X
mov edx,10
call sread
mov eax,X
call atoi
mov [X],eax
mov ecx,[A]
cmp ecx,7
jge check_or mov edx,[X]
mov ax,[A]
mul edx
mov [Z],ax
```

```
jmp fin
check_or:
mov ax,[A]
sub ax,7
mov [Z],ax
jmp fin
fin:
mov eax,msg
call sprint
mov eax,[Z]
call iprintLF
call quit
```

5 Вывод.

В заключении я изучил команды условного и безусловного переходов. Приобрел навыков написания программ с использованием переходов. Ознакомился с назначением и структурой файла листинга.

Список литературы

Лабораторная работа №7

::: ::