РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: Архитектура компьютера

Студент: Нурмухаметова Каролина Марселевна

Группа: НКАбд-05-25

МОСКВА

2025 г.

Содержание

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Порядок выполнения лабораторной работы

2.1 Реализация переходов в NASM

Создадим каталог для программ лабораторной работы № 7, перейдём в него и создадим файл *lab7-1.asm*:

```
kmnurmukhametov@dk6n01 ~ $ mkdir ~/work/arch-pc/lab07
kmnurmukhametov@dk6n01 ~ $ cd ~/work/arch-pc/lab07
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ touch lab7-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $
```

Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введём в файл lab7-1.asm текст программы из листинга 7.1.

Открыть ▾ lab7-1.asm ~/work/arch-pc/lab07

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

Создадим исполняемый файл и запустим его:

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $
```

Инструкция jmp позволяет осуществлять переходы не только вперёд но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию jmp с меткой _label1 (т.е. переход к инструкциям вывода сообщения № 1) и после вывода

сообщения № 1 добавим инструкцию jmp с меткой _end (т.е. переход к инструкции call quit).
Создадим файл *lab7-01.asm* и введём в него текст программы из листинга 7.2.:

Открыть ▾ lab7-1.asm ~/work/arch-pc/lab07

```
%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
```

Теперь запустим файл.

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $
```

Использование инструкции jmp приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры.
Создадим *файл lab7-2.asm* в каталоге *~/work/arch-pc/lab07.*
Внимательно изучим текст программы из листинга 7.3 и введём в *lab7-2.asm:*

Открыть

*lab7-2.asm
~/work/arch-pc/lab07

lab7-1.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'
; ----------  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: '
```

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ touch lab7-2.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 5
Наибольшее число: 50
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 3
Наибольшее число: 50
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 16
Наибольшее число: 50
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 63
Наибольшее число: 63
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $
```

2.2. Изучение структуры файлы листинга

Обычно **nasm** создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Создадим файл листинга для программы из файла *lab7-2.asm.*

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Откроем файл листинга *lab7-2.lst*.. Подробно объясним содержимое трёх строк файла листинга:

```
143 000000BB 80EB30                <1>     sub     bl, 48
144 000000BE 01D8                  <1>     add     eax, ebx
145 000000C0 BB0A000000            <1>     mov     ebx, 10
```

Строка 143: sub bl, 48

- ✧ **Операция**: Вычитание
- ✧ **Что делает**: Преобразует символ цифры в соответствующее числовое значение.
  - ➢ В кодировке ASCII символ '0' имеет код 48 (0x30)
  - ➢ Вычитая 48 из значения в регистре BL (младший байт EBX), мы преобразуем символ цифры в фактическое число (0-9)
  - ➢ Например: символ '5' (код 53) → 53 - 48 = 510

Строка 144: add eax, ebx

- ✧ **Операция**: Сложение
- ✧ **Что делает**: Добавляет преобразованную цифру к накапливаемому результату.
  - ➢ Регистр EAX обычно содержит текущее накопленное число
  - ➢ К нему добавляется значение из EBX (которое теперь содержит цифру 0-9 после преобразования)

Строка 145: mov ebx, 10

- ✧ **Операция**: Перемещение (присваивание)
- ✧ **Что делает**: Устанавливает значение 10 в регистр EBX.
  - ➢ Подготавливает регистр EBX для следующей операции

Откроем файл с программой *lab7-2.asm* и в инструкции с двумя операндами удалим один операнд.

```
1 %include 'in_out.asm'
2 section        .data
3    msg1 db 'Введите B: ',0h
4    msg2 db "Наибольшее число: ",0h
5    A dd '20'
6    C dd '50'
7 section .bss
8    max resb 10
9    B resb 10
10 section        .text
11   global _start
12 _start:
13 ; ----------   Вывод сообщения 'Введите B: '
14   mov eax,msg1
15   call sprint
16 ; ----------   Ввод 'B'
17   mov ecx,B
18   mov edx,10
19   call sread
20 ; ----------   Преобразование 'B' из символа в число
21   mov eax,B
22   call atoi ; Вызов подпрограммы перевода символа в число
23   mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----------  Записываем 'A' в переменную 'max'
25   mov ecx,[A] ; 'ecx = A'
26   mov [max],ecx ; 'max = A'
27 ; ---------- Сравниваем 'A' и 'C' (как символы)
28   cmp ecx,[C] ; Сравниваем 'A' и 'C'
29   jg check_B ; если 'A>C', то переход на метку 'check_B',
30   mov ecx,[C] ; иначе 'ecx = C'
31   mov [max],ecx ; 'max = C'
[kmnurmukhametov@dk6n01 - la...
```

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:17: error: invalid combination of opcode and operands
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab07 $
```

Если выполнить трансляцию после удаления операнда, то листинг создастся лишь частично (до строки с ошибкой), а объектный файл не создастся вовсе.

2.3. Задание для самостоятельной работы

1. Напишем программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выберем из табл. 7.5 в соответствии с вариантом № 16.

Таблица 7.5. Значения a, b, c для задания №1.

Номер варианта	Значения a, b, c	Номер варианта	Значения a, b, c
1	17,23,45	**11**	21,28,34
2	82,59,61	**12**	99,29,26
3	94,5,58	**13**	84,32,77
4	8,88,68	**14**	81,22,72
5	54,62,87	**15**	32,6,54
6	79,83,41	**16**	44,74,17
7	45,67,15	**17**	26,12,68
8	52,33,40	**18**	83,73,30
9	24,98,15	**19**	46,32,74
10	41,62,35	**20**	95,2,61

Создадим файл *lab7-3.asm*, впишем в него наш код, преобразуем его в исполняемый файл и проверим корректность выполнения программы.

```
kmnurmukhametov@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
kmnurmukhametov@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
kmnurmukhametov@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-3
Введите a: 44
Введите b: 74
Введите c: 17
Наименьшее число: 17
```

```
%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
   a resb 10
   b resb 10
   c resb 10
   min resb 10

section  .text
   global _start
_start:
; Ввод переменной a
   mov eax, msg1
   call sprint
   mov ecx, a
   mov edx,10
   call sread
   mov eax, a
   call atoi
   mov [a], eax
; Ввод переменной b
   mov eax, msg2
   call sprint
   mov ecx, b
   mov edx,10
   call sread
   mov eax, b
   call atoi
   mov [b], eax
; Ввод переменной c
   mov eax, msg3
   call sprint
   mov ecx, c
   mov edx,10
   call sread
   mov eax, c
   call atoi
   mov [c], eax
; Находим наименьшее число
   mov eax, [a]
   cmp eax, [b]
   jl check_c
   mov eax, [b]
check_c:
   cmp eax, [c]
   jl print_min
   mov eax, [c]
print_min:
   mov [min], eax
   mov eax, msg4
   call sprint
   mov eax, [min]
   call iprintLF
```

2. Напишем программу, которая для введённых с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выберем из таблицы 7.6 в соответствии с вариантом № 16.

16

$$\begin{cases} x+4, & x<4 \\ ax, & x \geq 4 \end{cases}$$

Создадим файл *lab7-4.asm,* преобразуем его в исполняемый файл и проверим корректность выполнения программы для значений x и a из таблицы 7.6.

```
kmnurmukhametov@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-4.asm
kmnurmukhametov@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
kmnurmukhametov@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
kmnurmukhametov@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 1
Введите a: 1
f(x)= 5
kmnurmukhametov@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите x: 7
Введите a: 1
f(x)= 7
```

```asm
%include 'in_out.asm'
section        .data
   msg1 db 'Введите x: ',0h
   msg2 db 'Введите a: ',0h
   msg3 db 'f(x)= ',0h
   newline db 0xA,0
section .bss
    x resb 10
    a resb 10
    result resb 10
section .text
global _start
_start:
    ; Ввод переменной x
    mov eax, msg1
    call sprint
    mov ecx, x
    mov edx, 10
    call sread
    mov eax, x
    call atoi
    mov [x], eax
    ; Ввод переменной a
    mov eax, msg2
    call sprint
    mov ecx, a
    mov edx, 10
    call sread
    mov eax, a
    call atoi
    mov [a], eax
    ; Вычисление функции f(x)
    mov eax, [x]      ; загружаем x в eax
    cmp eax, 4        ; сравниваем x с 4
    jl case_x_less    ; если x < 4, переходим к первому случаю
    ; Случай 2: x ≥ 4, f(x) = a*x
case_x_ge:
    mov eax, [a]      ; загружаем a
    mov ebx, [x]      ; загружаем x
    imul eax, ebx     ; eax = a * x
    jmp print_result
    ; Случай 1: x < 4, f(x) = x + 4
case_x_less:
    mov eax, [x]      ; загружаем x
    add eax, 4        ; eax = x + 4
print_result:
    ; Сохраняем результат
    mov [result], eax
    ; Вывод результата
    mov eax, msg3
    call sprint
    mov eax, [result]
    call iprintLF
    ; Завершение программы
    call quit
```

3 Выводы

Изучила команды условного и безусловного переходов.
Приобрела навыки написания программ с использованием переходов.
Ознакомилась с назначением и структурой файла листинга.

4 Список используемой литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.
3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander.org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. *Newham C.* Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658.
6. *Robbins A.* Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. *Zarrelli G.* Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. *Колдаев В. Д., Лупин С. А.* Архитектура ЭВМ. — М. : Форум, 2018.
10. *Куляс О. Л., Никитин К. А.* Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. *Новожилов О. П.* Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.
13. *Робачевский А., Немнюгин С., Стесик О.* Операционная система UNIX. — 2-е изд. — БХВ-Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. *Столяров А.* Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
15. *Таненбаум Э.* Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. *Таненбаум Э., Бос Х.* Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).