

Лабораторная работа No7.

Команды безусловного и условного переходов в Nasm.

Бурыкина Софья Дмитриевна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Самостоятельная работа	10
5	Выводы	14
	Список литературы	15

Список иллюстраций

3.1	Программа с использованием инструкции jmp	7
3.2	Результат работы	7
3.3	Результат работы	7
3.4	Изменённая программа	8
3.5	Работа программы	9
3.6	Ошибка	9
3.7	ВЫБРАННЫЕ ТРИ СТРОКИ	9
4.1	Результат работы 1	10
4.2	Программа 1	11
4.3	Программа 2	12
4.4	Программа 2	13
4.5	РАБОТА ПРОГРАММЫ 2	13

Список таблиц

1 Цель работы

Изучение команд условного и безусловного переходов.

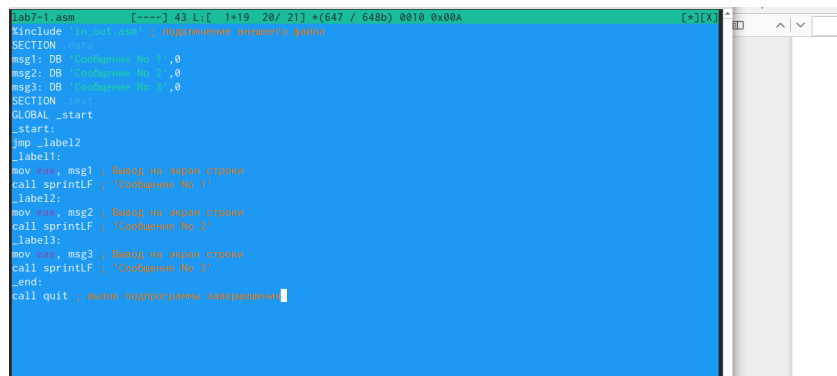
2 Теоретическое введение

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

Более подробно об Unix см. в [1–6].

3 Выполнение лабораторной работы

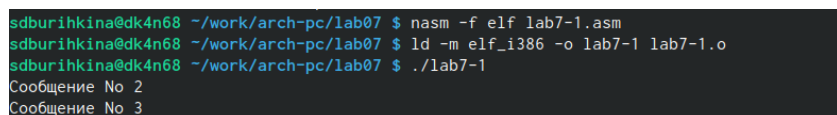
Создала каталог, рассмотрела пример программы (рис. 3.1).



```
lab7-1.asm [----] 43 L: [ 1*19 20/ 21] *(647 / 648b) 0010 0x00A
Xinclude: [lab7-1.asm] ; подключение внешнего файла.
SECTION: .text
msg1: DB "Сообщение No 1",0
msg2: DB "Сообщение No 2",0
msg3: DB "Сообщение No 3",0
SECTION: .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; "Сообщение No 1"
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; "Сообщение No 2"
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; "Сообщение No 3"
_end:
call quit ; Выход подпрограммы завершающей
```

Рис. 3.1: Программа с использованием инструкции jmp

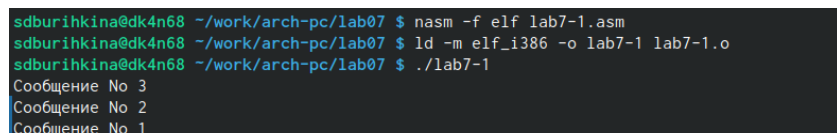
Создала исполняемый файл и запустила его (рис. 3.2).



```
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
```

Рис. 3.2: Результат работы

Изменила текст программы (рис. 3.3).



```
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
```

Рис. 3.3: Результат работы

Создала файл lab7-2.asm проверила его работу для разных значений В (рис. ??).

```
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 6
Наибольшее число: 50
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: ^[[B^[[B
Наибольшее число: 50
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 22
Наибольшее число: 50
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 55
Наибольшее число: 55
sdburikhkina@dk4n68 ~/work/arch-pc/lab07 $
```

Создала файл изменила в нём программу (рис. 3.4).

```
lab7-2.asm [-M--] 4 L: [ 8+ 6 14/ 50] *(274 /1741b) 0044 0x02C
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите В: '
mov msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- Записываем 'А' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
```

Рис. 3.4: Изменённая программа

Работа изменённого файла (рис. 3.5).


```
sdburikhina@dk4n68 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: expression syntax error
sdburikhina@dk4n68 ~/work/arch-pc/lab07 $
```

Рис. 3.5: Работа программы

Получиласт в файле листинга ошибка (рис. 3.6).

```
14          *****          mov eax
15 000000E8 E822FFFFFF      error: invalid combination of opcode and operands
                           call sprint
```

Рис. 3.6: Ошибка

Влистинге добавляются ***

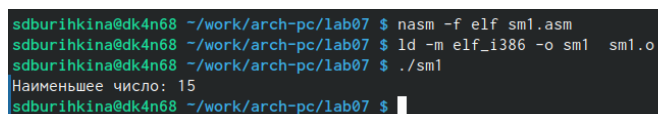
Строки листинга 23,24,25 номер строки второй столбец это адрес, третий столбец это машинный код (рис. 3.7).

```
23 0000000F 52          <1>    push    edx
24 00000010 51          <1>    push    ecx
25 00000011 53          <1>    push    ebx
```

Рис. 3.7: ВЫБРАННЫЕ ТРИ СТРОКИ

4 Самостоятельная работа

Написала программу нахождения наименьшей из 3 целочисленных переменных из таблицы (рис. 4.1).



```
sdburihkina@dk4n68 ~/work/arch-pc/lab07 $ nasm -f elf sm1.asm
sdburihkina@dk4n68 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o sm1 sm1.o
sdburihkina@dk4n68 ~/work/arch-pc/lab07 $ ./sm1
Наименьшее число: 15
sdburihkina@dk4n68 ~/work/arch-pc/lab07 $
```

Рис. 4.1: Результат работы 1

Програма для нахождения наименьшей из 3 целочисленных переменных из таблицы (рис. 4.2).

```

%include 'in_out.asm'
section .data
msg1 db "Наименьшее число: ",0h
A dd '45'
B dd '67'
C dd '15'
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 'B' (как символы)
cmp [C], ecx ; Сравниваем 'A' и 'B'
jg check_B ; если 'A>B', то переход на метку 'check_C',
mov ecx, [C] ; иначе 'ecx = B'
mov [min],ecx ; 'min = B'
; ----- Преобразование 'max(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 'max'
; ----- Сравниваем 'min(A,B)' и 'C' (как числа)
mov ecx,[min]
cmp [B], ecx ; Сравниваем 'min(B,A)' и 'C'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = C'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg1
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

Рис. 4.2: Программа 1

Написала программу (рис. 4.3).

```

/afs/.dk.sci.pfu.edu.ru/home/s/d/sdburikhina/work/arch-pc/lab07/sm2.asm 1
#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 'X' (как символы)
mov edx, [X]
mov ebx, [A]

```

Рис. 4.3: Программа 2

Продолжение (рис. 4.4).

```

; ----- 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 'X' (как символы)
mov edx, [X]
mov ebx, [A]
cmp ebx,edx ; Сравниваем 'A' и 'X'
je check ; если 'A=X', то переход на метку ''

mov eax, [A]
mov ebx, [X]
add eax,ebx
mov edi,eax

mov eax,msg3
call sprint
mov eax, edi
call iprintlnLF
call quit

check:

mov eax,[A]
mov ebx,6
mul ebx
mov edi, eax

mov eax,msg3
call sprint
mov eax,edi
call iprintlnLF
call quit

```

Рис. 4.4: Программа 2

Создала исполняемый файл и проверила его работу (рис. 4.5).

```

sdburikhkina@dk8n77 ~/work/arch-pc/lab07 $ nasm -f elf sm2.asm
sdburikhkina@dk8n77 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o sm2 sm2.o
sdburikhkina@dk8n77 ~/work/arch-pc/lab07 $ ./sm2
Введите a: 1
Введите x: 2
Результат: 3
sdburikhkina@dk8n77 ~/work/arch-pc/lab07 $ ./sm2
Введите a: 1
Введите x: 1
Результат: 6

```

Рис. 4.5: РАБОТА ПРОГРАММЫ 2

5 Выводы

Приобрела навыки написания программ с использованием переходов. Познакомилась с назначением и структурой файла листинга.

Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016.
URL: <https://www.gnu.org/software/bash/manual/>.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.