

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Буриева Шахзода Акмаловна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
4	Задание для самостоятельной работы	14
5	Вывод	16

Список иллюстраций

3.1	Создание каталога lab7,переход в него и создание файла lab7-1.asm	8
3.2	Ввод текста программы в файл lab7-1.asm	8
3.3	Создание исполняемого файла и его запуск	9
3.4	Изменение текста программы lab7-1.asm	9
3.5	Создание исполняемого файла и его запуск	9
3.6	Изменение текста программы lab7-1.asm	10
3.7	Создание исполняемого файла и его запуск	10
3.8	Создание файла lab7-2.asm	10
3.9	Ввод текста программы в файл lab7-2.asm	11
3.10	Создание исполняемого файла и его запуск	11
3.11	Создание файла для программы из файла lab7-2.asm	11
3.12	Открытие файла листинга с помощью любого текстового редактора	12
3.13	Удаление одного из операндов	13
4.1	Создание файла lab7-3.asm	14
4.2	Написание программы для нахождения наименьшего числа из 3 данных числе	14
4.3	Создание исполняемого файла и его запуск	15
4.4	Выполнение второй части самостоятельного задания	15

Список таблиц

1 Цель работы

Целью работы изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. Познакомиться с назначением и структурой файла листинга.

2 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление. Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре. Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

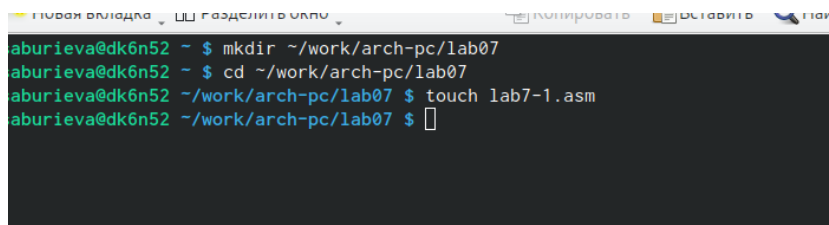
Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора. В следующей таблице указано положение битовых флагов в регистре флагов. 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.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную

информацию.

3 Выполнение лабораторной работы

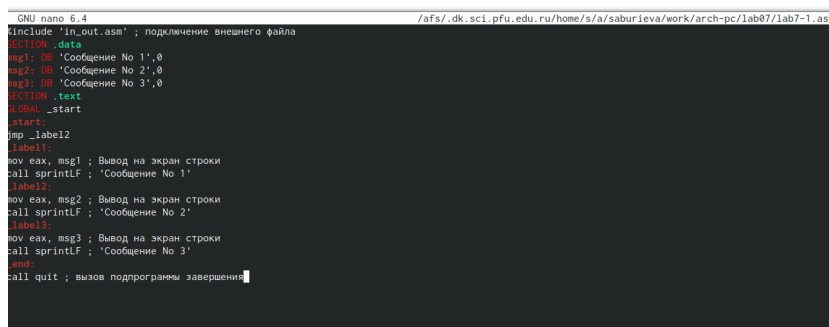
Создала каталог для программ лабораторной работы № 7, перешла в него и создала файл lab7-1.asm



```
aburieva@dk6n52 ~ $ mkdir ~/work/arch-pc/lab07
aburieva@dk6n52 ~ $ cd ~/work/arch-pc/lab07
aburieva@dk6n52 ~/work/arch-pc/lab07 $ touch lab7-1.asm
aburieva@dk6n52 ~/work/arch-pc/lab07 $
```

Рис. 3.1: Создание каталога lab7, переход в него и создание файла lab7-1.asm

Рассмотрела пример программы с использованием инструкции jmp. Ввела в файл lab7-1.asm текст программы из листинга.



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data
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 ; вызов подпрограммы завершения
```

Рис. 3.2: Ввод текста программы в файл lab7-1.asm

Создала исполняемый файл и запустила его.


```
saburieva@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
saburieva@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
saburieva@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 3
saburieva@dk6n52 ~/work/arch-pc/lab07 $
```

Рис. 3.3: Создание исполняемого файла и его запуск

Изменила программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавила инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавила инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Измените текст программы в соответствии с листингом.

```
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data
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'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Изменение текста программы lab7-1.asm

Создала исполняемый файл и проверила его работу.

```
saburieva@dk6n52 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
saburieva@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
saburieva@dk6n52 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1
saburieva@dk6n52 ~/work/arch-pc/lab07 $
```

Рис. 3.5: Создание исполняемого файла и его запуск

Изменила текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы сначала выводил на экран сообщение 3, потом сообщение 2 и

затем сообщение 1.

```
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab07/lab7-1.asm
#include 'in_out.asm'; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.6: Изменение текста программы lab7-1.asm

Создала исполняемый файл и проверила его работу.

```
saburieva@dk3n63 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
saburieva@dk3n63 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
saburieva@dk3n63 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение No 3
Сообщение No 2
Сообщение No 1
saburieva@dk3n63 ~/work/arch-pc/lab07 $
```

Рис. 3.7: Создание исполняемого файла и его запуск

Создала файл lab7-2.asm в каталоге ~/work/arch-pc/lab07.

```
saburieva@dk3n63 ~/work/arch-pc/lab07 $ touch lab7-2.asm
saburieva@dk3n63 ~/work/arch-pc/lab07 $
```

Рис. 3.8: Создание файла lab7-2.asm

Внимательно изучила текст программы из листинга и ввела в lab7-2.asm.

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab07/lab7-2.asm
#include "in_out.h"
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 read
; ----- 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)
str 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]
str ecx,[B] ; Сравниваем 'max(A,C)' и 'B'

```

Рис. 3.9: Ввод текста программы в файл lab7-2.asm

Создала исполняемый файл и проверила его работу для разных значений B.

```

saburieva@dk3n37 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
saburieva@dk3n37 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
saburieva@dk3n37 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 10
Наибольшее число: 50
saburieva@dk3n37 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 20
Наибольшее число: 50
saburieva@dk3n37 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 50
Наибольшее число: 50
saburieva@dk3n37 ~/work/arch-pc/lab07 $

```

Рис. 3.10: Создание исполняемого файла и его запуск

Создала файл листинга для программы из файла lab7-2.asm.

```

saburieva@dk3n63 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
saburieva@dk3n63 ~/work/arch-pc/lab07 $

```

Рис. 3.11: Создание файла для программы из файла lab7-2.asm

Открыла файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit.

```

lab7-2.lst  [-----]  0  L:  1+ 0  /1225  +0  /14450b  0032  0x020
1          include 'in_out.asm'
2          <!--> -----  len -----
3          <!--> ; функция вычисления длины сообщения
4          00000000 53          <!--> len: -----
5          00000001 85C3         <!--> push  ebx -----
6          <!--> mov  ebx, eax -----
7          <!--> nextchar: -----
8          00000003 803800       <!--> cmp   byte [eax], 0
9          00000006 7403         <!--> jz    finished -----
10         00000008 40          <!--> inc   eax -----
11         00000009 EBF8         <!--> jmp   nextchar -----
12         <!--> -----
13         <!--> finished: -----
14         0000000B 2908         <!--> sub   eax, ebx -----
15         0000000D 5B          <!--> pop   ebx -----
16         0000000E C3          <!--> ret   -----
17         <!--> -----
18         <!--> -----  sprint -----
19         <!--> -----
20         <!--> ; функция печати сообщения
21         <!--> ; входные данные: mov  eax, <message>
22         <!--> sprint: -----
23         0000000F 52          <!--> push  edx -----
24         00000010 51          <!--> push  ecx -----
25         00000011 53          <!--> push  ebx -----
26         00000012 58          <!--> push  eax -----
27         00000013 E8E0FFFFFF   <!--> call  len -----
28         <!--> -----
29         00000018 89C2         <!--> mov   edx, eax -----
30         0000001A 58          <!--> pop   eax -----
31         <!--> -----
32         0000001B 89C1         <!--> mov   ecx, eax -----
33         0000001D 0B01000000    <!--> mov   ebx, 1 -----
34         00000022 B804000000    <!--> mov   eax, 4 -----
35         00000027 CD80         <!--> int   80h -----
36         <!--> -----
37         00000029 5B          <!--> pop   ebx -----
38         0000002A 59          <!--> pop   ecx -----

```

Рис. 3.12: Открытие файла листинга с помощью любого текстового редактора

В строке 17 содержится номер строки [17], адрес строки [000000F2], машинный код [B9] и исходный текст программы [mov ecx, B]. В строке 18 содержится номер строки [18], адрес строки [000000F7], машинный код [BA0A000000] и исходный текст программы [mov edx, 10]. В строке номер 19 содержится номер строки [19], адрес строки [000000FC], машинный код [E842FFFFFF] и исходный текст программы [call sread].

Открыла файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалила один операнд. Выполнила трансляцию с получением файла листинга.

```

12 2 section .data
13 3 00000000 D092D082D0B5D0B4D0- msg1 db "Введите B: ",0h
14 3 00000009 B8D182D0B520423A20-
15 3 00000012 00
16 4 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
17 4 0000001C BED0BBD18CD185D0B5-
18 4 00000025 D0B520D187D0B8D181-
19 4 0000002E D0B8D0B3A20000
20 5 00000035 32300000 A dd '20'
21 6 00000039 35300000 C dd '50'
22 7 section .bss
23 8 00000000 <res Ah> max resb 10
24 9 0000000A <res Ah> B resb 10
25 10 section .text
26 11 global _start
27 12 _start:
28 13 ; ----- Вывод сообщения 'Введите B: '
29 13 ***** error: invalid combination of opcode and operands
30 14 000000E8 B8[00000000] mov eax,msg1
31 15 000000ED E81DFFFFFF call sprintf
32 16 ; ----- Ввод 'B'
33 17 000000F2 B9[0A000000] mov ecx,B
34 18 000000F7 BA0A000000 mov edx,10
35 19 000000FC E842FFFFFF call sread
36 20 ; ----- Преобразование 'B' из символа в число
37 21 00000101 B8[0A000000] mov eax,B
38 22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
39 23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
40 24 ; ----- Записываем 'A' в переменную 'max'
41 25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
42 26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
43 27 ; ----- Сравниваем 'A' и 'C' (как символы)
44 28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
45 29 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
46 30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
47 31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
48 32 ; ----- Преобразование 'max(A,C)' из символа в число
49 33 check_B:
50 34 00000130 B8[00000000] mov eax,max
51 35 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
52 36 0000013A A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
53 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
54 38 0000013F 8B0D[00000000] mov ecx,[max]
55 39 00000145 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
56 40 0000014B 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',
57 41 0000014D 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'
58 42 00000153 8B0D[0A000000] mov fin:

```

Рис. 3.13: Удаление одного из операндов

4 Задание для самостоятельной работы

Спрева создала файл lab7-3.asm для написания программы задания самостоятельного выполнения

```
saburieva@dk2n26 ~/work/arch-pc/lab07 $ touch lab7-3.asm
saburieva@dk2n26 ~/work/arch-pc/lab07 $
```

Рис. 4.1: Создание файла lab7-3.asm

Написала программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрала из первой таблицы в соответствии с вариантом, полученным при выполнении лабораторной работы No 6 (вариант 4).

```
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab07/lab7-3.asm
#include "in_out.asm"
section .data
msg1 db "Наименьшее число: "
a dd 8
b dd 68
c dd 88
section .bss
min resb 10
section .text
global _start
_start:
mov eax, msg1
call sprint

mov ecx, [a]
mov [min], ecx ; 'minA'
; ----- Сравниваем 'A' и 'C' (как числа)
; ----- Сравниваем 'A' и 'C'
спр 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:
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx, [min]
спр 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, [min]
call iprintlf ; Вывод 'min(A,B,C)'
call quit ; Выход
```

Рис. 4.2: Написание программы для нахождения наименьшего числа из 3 данных числе

Создала исполняемый файл и проверила его работу.

```
saburieva@dk8n69 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
saburieva@dk8n69 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
saburieva@dk8n69 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 8
saburieva@dk8n69 ~/work/arch-pc/lab07 $
```

Рис. 4.3: Создание исполняемого файла и его запуск

Создала файл lab7-4.asm. Написала программу которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Создала исполняемый файл и проверила его работу.

```
GNU nano 6.4 /root/.ssh/pf0.edu.ru/home/s/saburieva/work/arch-pc/lab07/lab7-4.asm
#include "in_out.asm"
section .data
msgx db "Введите x: ",0h
msga db "Введите a: ",0h

section .bss
resb 10
resb 10
resb 10

section .text
global _start

_start:
; ----- Ввод 'x'
mov eax, msgx
call sprint
mov ecx, x
mov edx, 10
call read
; ----- Ввод 'a'
mov eax, msga
call sprint
mov ecx, a
mov edx, 10
call read
; ----- Преобразование 'x' из символа в число
mov eax, x
call atoi
mov [x], eax

; ----- Преобразование 'a' из символа в число
mov eax, [a]
call atoi
mov [a], eax

mov ecx, [x]
```

Рис. 4.4: Выполнение второй части самостоятельного задания

5 Вывод

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