

Отчет по лабораторной работе № 8

Дисциплина: архитектура компьютера

Никулина Ксения Ильинична

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Вывод	15

Список иллюстраций

4.1	Создание файла	7
4.2	Текст	7
4.3	Проверка файла	8
4.4	Текст	8
4.5	Вывод программы	8
4.6	Текст	9
4.7	Вывод программы	9
4.8	Создание файла	9
4.9	Текст	10
4.10	Вывод программы	10
4.11	Удаление	11
4.12	Ошибка из-за отсутствия операнда	11
4.13	Текст	12
4.14	Проверка файла	12
4.15	Текст	13
4.16	Ответы	14

1 Цель работы

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

2 Задание

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

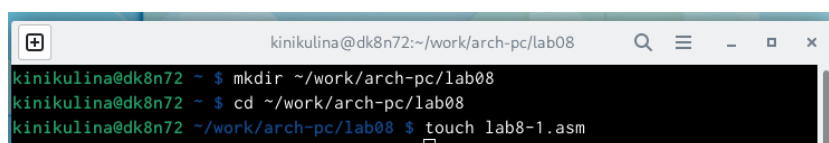
3 Теоретическое введение

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.

Все ошибки и предупреждения, обнаруженные при ассемблировании, транслятор выводит на экран, и файл листинга не создаётся. Итак, структура листинга: • номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы); • адрес — это смещение машинного кода от начала текущего сегмента; • машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция `int 80h` начинается по смещению `00000020` в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция `int 80h` ассемблируется в `CD80` (в шестнадцатеричном представлении); `CD80` — это инструкция на машинном языке, вызывающая прерывание ядра); • исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

4 Выполнение лабораторной работы

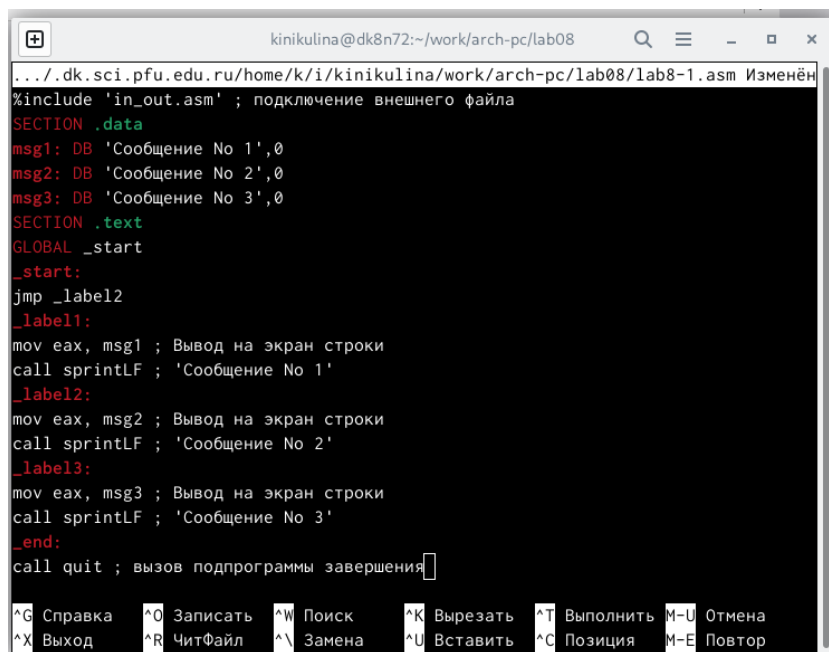
1. Создала каталог для программ лабораторной работы No 8, перешла в него и создала файл lab8-1.asm(рис. 4.1)



```
kinikulina@dk8n72:~/work/arch-pc/lab08
kinikulina@dk8n72 ~$ mkdir ~/work/arch-pc/lab08
kinikulina@dk8n72 ~$ cd ~/work/arch-pc/lab08
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 4.1: Создание файла

2. Ввела в файл lab8-1.asm текст программы из листинга 8.1.(рис. 4.2)



```
.../.dk.sci.pfu.edu.ru/home/k/i/kinikulina/work/arch-pc/lab08/lab8-1.asm Изменён
#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. 4.2: Текст

3. Создала исполняемый файл и проверила его работу(рис. 4.3)

```
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 3
```

Рис. 4.3: Проверка файла

4. Изменила текст программы в соответствии с листингом 8.2. Создала исполняемый файл и проверила его работу. (рис. 4.4,рис. 4.5)

```
kinikulina@dk8n72:~/work/arch-pc/lab08
.../.dk.sci.pfu.edu.ru/home/k/i/kinikulina/work/arch-pc/lab08/lab8-1.asm Из
%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'
```

Рис. 4.4: Текст

```
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ ./lab8-1
Сообщение No 2
Сообщение No 1
```

Рис. 4.5: Вывод программы

5. Изменила текст программы, чтобы вывод программы был следующим: (рис. 4.6, рис. 4.7)

```
jmp _end
_label12:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label13:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
jmp _label12
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.6: Текст

```
Сообщение No 3
Сообщение No 2
Сообщение No 1
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ ./lab8-1
```

Рис. 4.7: Вывод программы

6. Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 (рис. 4.8)

```
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ touch lab8-2.asm
kinikulina@dk8n72 ~/work/arch-pc/lab08 $
```

Рис. 4.8: Создание файла

7. Ввела в файл текст из листинга 8.3, создала исполняемый файл и профилировала работу (рис. 4.9, рис. 4.10)

```

mc [kinikulina@dk8n72]:~/work/arch-pc/lab08
kinikulina@dk8n72:~/work/arch-pc/lab08 x mc [kinikulina@dk8n72
.../.dk.sci.pfu.edu.ru/home/k/i/kinikulina/work/arch-pc/la
%include 'in_out.asm'
section .data
msg1 db 'Введите В: ',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 В: '
mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Вып
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^C Поз

```

Рис. 4.9: Текст

```

kinikulina@dk8n72 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
kinikulina@dk8n72 ~/work/arch-pc/lab08 $ ./lab8-2
Введите В: 36
Наибольшее число: 50

```

Рис. 4.10: Вывод программы

8. Открыла файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалила один операнд, выполнила трансляцию с получением файла листинга (рис. 4.11, рис. 4.12)

```

.../.dk.sci.pfu.edu.ru/home/k/i/kinikulina/work/arch-pc/lab08/lab8-2.asm Изменён
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. 4.11: Удаление


```

kinikulina@dk8n72 ~/work/arch-pc/lab08 $ nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:39: error: invalid combination of opcode and operands

```

Рис. 4.12: Ошибка из-за отсутствия операнда

#Самостоятельная работа (Вариант № 8) 1. Написала программу нахождения наименьшей из 3 целочисленных переменных a, b и c (52,33,40) (Ответ: 33) (рис. 4.13)



```
mc [kinikulina@dk3n38]:~/work/arch-pc/lab08
/afs/.dk.sci.pfu.edu.ru/home/k/i/kinikulina/work/arch-pc/lab08/lab8-1
%include 'in_out.asm'
section .data
msg1 db 'Введите B', 0h
msg db 'Наименьшее число', 0h
A dd '52'
C dd '40'

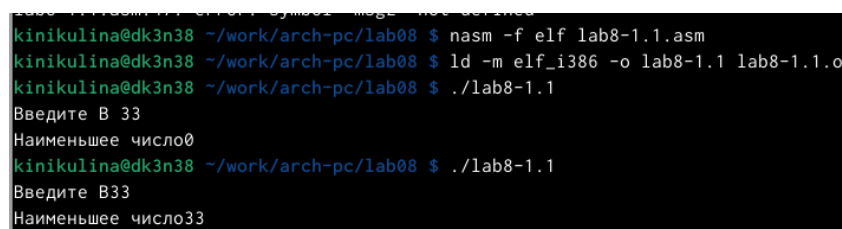
section .bss
min resb 10
B resb 10
section .text

global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10

[ Прочитано 52 строки ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить M-U От
```

Рис. 4.13: Текст

2. Создала исполняемый файл и проверила его работу(рис. 4.14)



```
kinikulina@dk3n38 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.1.asm
kinikulina@dk3n38 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1.1 lab8-1.1.o
kinikulina@dk3n38 ~/work/arch-pc/lab08 $ ./lab8-1.1
Введите B 33
Наименьшее число0
kinikulina@dk3n38 ~/work/arch-pc/lab08 $ ./lab8-1.1
Введите B33
Наименьшее число33
```

Рис. 4.14: Проверка файла

3. Написала программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. (рис. 4.15)

```

ASM sam_rabota2.2.asm
ASM sam_rabota2.2.asm
1  %include 'in_out.asm'
2
3  SECTION .data
4
5  x1: DB 'Введите значение x: ',0
6  a1: DB 'Введите значение a: ',0
7  otv1: DB 'Ответ при x, a:',0
8
9  SECTION .bss
10 x: RESB 10
11 a: RESB 10
12 rez:RESB 10
13
14 SECTION .text
15 Global _start
16
17 _start:
18
19     mov eax, x1
20     call sprintf
21
22     mov ecx,x
23     mov edx,80
24     call sread
25
26     mov eax, x
27     call atoi
28     mov x, eax
29
30     mov eax,a1
31     call sprintf
32
33     mov ecx,a
34     mov edx,80
35     call sread
36
37     mov eax, a
38     call atoi
39     mov a, eax
40
41     mov ebx, 3
42     cmp a, ebx ; compare
43     jb where
44
45     mov eax, x
46     add eax, 1
47

```

Рис. 4.15: Текст

4. Создала исполняемый файл и проверила его работу(рис. 4.16)

```
kinikulina@dk3n37:~/work/arch-pc/lab08
kinikulina@dk3n37 ~$ cd ~/work/arch-pc/lab08
kinikulina@dk3n37 ~/work/arch-pc/lab08 $ nasm -f elf sam_rabota2.2.asm
kinikulina@dk3n37 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o sam_rabota2.2 sam_rabota2.2.o
kinikulina@dk3n37 ~/work/arch-pc/lab08 $ ./sam_rabota2.2
Введите значение x:
1
Введите значение a:
4
2
kinikulina@dk3n37 ~/work/arch-pc/lab08 $ ./sam_rabota2.2
Введите значение x:
1
Введите значение a:
2
6
```

Рис. 4.16: Ответы

5 Вывод

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