

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Горобцова Арина Романовна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	14

Список иллюстраций

3.1	Создание файла	7
3.2	Ввод данных	7
3.3	Создание и запуск файла	8
3.4	Название рисунка	8
3.5	Создание и запуск файла	8
3.6	Создание и запуск файла	9
3.7	Создание файла	9
3.8	Создание и запуск файла	9
3.9	Создание и открытие файла	9
3.10	Рассмотрение файла листинга одной из программ	10
3.11	Ошибка в файле листинга	11
3.12	Программа для сравнения трех заранее известных чисел	12
3.13	Создание и запуск файла	12
3.14	Название рисунка	13
3.15	Создание и запуск файла	13

Список таблиц

1 Цель работы

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

2 Задание

1. Реализация переходов в NASM
2. Структурирование файла листинга
3. Задание для самостоятельной работы

3 Выполнение лабораторной работы

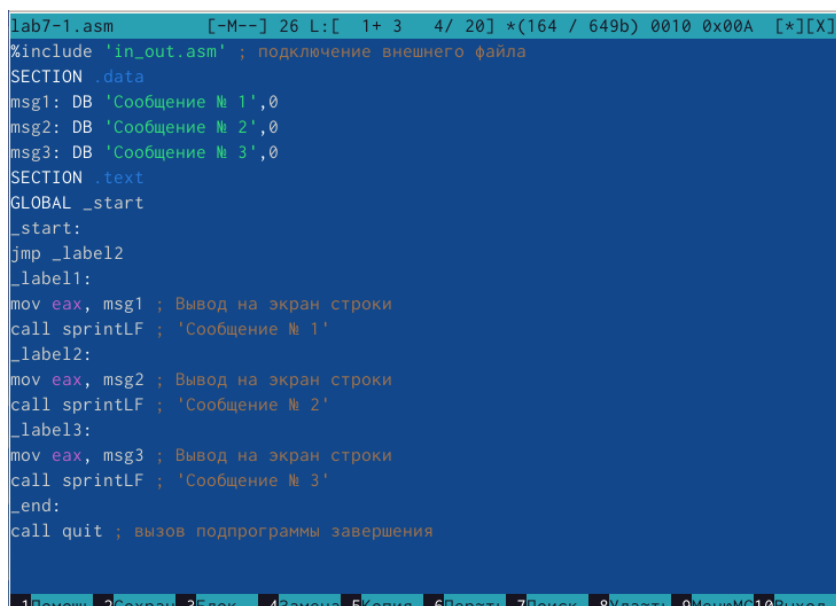
1. Реализация переходов в NASM

Создаем каталог для программ лабораторной работы № 7, переходим в него и создаем файл lab7-1.asm (рис. 3.1).

```
argorobcova@dk3n35 ~ $ mkdir ~/work/arch-pc/lab07
argorobcova@dk3n35 ~ $ cd ~/work/arch-pc/lab07
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ touch lab7-1.asm
argorobcova@dk3n35 ~/work/arch-pc/lab07 $
```

Рис. 3.1: Создание файла

Вводим в файл lab7-1.asm текст программы из листинга (рис. 3.2).



```
lab7-1.asm      [-M--] 26 L: [ 1+ 3 4/ 20] *(164 / 649b) 0010 0x00A  [*][X]
%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 sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.2: Ввод данных

Создаем исполняемый файл и запускаем его (рис. 3.3).

```

argorobcova@dk3n35 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ █

```

Рис. 3.3: Создание и запуск файла

Изменяем программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. (рис. 3.4).

```

lab7-1.asm      [-M--] 31 L:[ 1+15 16/ 22] *(475 / 670b) 0010 0x00A [*][X]
%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 sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМСТ10Выход

```

Рис. 3.4: Название рисунка

Создаем исполняемый файл и запускаем его (рис. 3.5).

```

argorobcova@dk3n35 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ █

```

Рис. 3.5: Создание и запуск файла

Изменяем текст программы, чтобы вывод программы был сначала ‘Сообщение №3’, потом ‘Сообщение №2’ и ‘Сообщение №1’ (рис. 3.6).


```
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
argorobcova@dk3n35 ~/work/arch-pc/lab07 $
```

Рис. 3.6: Создание и запуск файла

Создаем файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и вводи текст программы (рис. 3.7).

```
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ touch lab7-2.asm
```

Рис. 3.7: Создание файла

Создаем исполняемый файл и проверяем его работу для разных значений В (рис. 3.8).

```
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 100
Наибольшее число: 100
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ./lab7-2
Введите В: 1
Наибольшее число: 50
argorobcova@dk3n35 ~/work/arch-pc/lab07 $
```

Рис. 3.8: Создание и запуск файла

2. Структурирование файла листинга

Создайте файл листинга для программы из файла lab7-2.asm и открываем его с помощью команды mcedit (рис. 3.9).

```
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ mcedit lab7-2.lst

argorobcova@dk3n35 ~/work/arch-pc/lab07 $
```

Рис. 3.9: Создание и открытие файла

Объясним содержимое трёх строк файла листинга по выбору (рис. 3.10). В строке 9 содержится собственно номер сторки [9], адрес [00000003], машинный код [803800] и содержимое строки кода [стр byte [eax], 0] в строке 11 содержится

номер строки [11], адрес [00000008], машинный код [40] и содержимое строки кода [inc eax] в строке 24 содержится номер строки [24], адрес [0000000F], машинный код [52] и содержимое строки кода [push edx].

```

lab7-2.lst      [----]  0 L:  1+ 0  1/225] *(0  /14458b) 0032 0x020 [*][X]
1               %include 'in_out.asm'
1               <1> ;----- slen -----
2               <1> ; Функция вычисления длины сообщения
3               <1> slen:.....
4 00000000 53    <1>      push    ebx.....
5 00000001 89C3  <1>      mov     ebx, eax.....
6               <1>.....
7               <1> nextchar:.....
8 00000003 803800 <1>      cmp     byte [eax], 0...
9 00000006 7403   <1>      jz      finished.....
10 00000008 40    <1>      inc     eax.....
11 00000009 EBF8  <1>      jmp     nextchar.....
12               <1>.....
13               <1> finished:
14 0000000B 29D8  <1>      sub     eax, ebx
15 0000000D 5B    <1>      pop     ebx.....
16 0000000E C3   <1>      ret.....
17               <1>
18               <1>
19               <1> ;----- sprint -----
20               <1> ; Функция печати сообщения
21               <1> ; входные данные: mov eax,<message>
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Переть 7Поиск 8Удалить 9МенюMC10Выход

```

Рис. 3.10: Рассмотрение файла листинга одной из программ

Если в коде появляется ошибка, то ее описание появится в файле листинга (рис. 3.11).

```

lab7-2.lst      [----]  0 L:[179+ 0 179/210] *(10993/13264b) 0032 0x020[*][X]
4 0000002E D0BBD0BE3A2000....      A dd '20'
5 00000035 32300000                      C dd '50'
6 00000039 35300000                      section .bss
7                                     max resb 10
8 00000000 <res Ah>                      B resb 10
9 0000000A <res Ah>                      section .text
10                                    global _start
11                                    _star:
12                                    ; ----- Записываем 'A'
13                                    mov ecx
14                                    ***** error: invalid combination of opcode and
15 000000E8 890D[00000000]              mov [max],ecx ; 'max = A'
16                                    ; ----- Сравниваем 'A' и 'C' (как с
17 000000EE 3B0D[39000000]              cmp ecx,[C] ; Сравниваем 'A' и 'C'
18 000000F4 7F0C                      jg check_B ; если 'A>C', то переход на м
19 000000F6 8B0D[39000000]              mov ecx,[C] ; иначе 'ecx = C'
20 000000FC 890D[00000000]              mov [max],ecx ; 'max = C'
21                                    ; ----- Преобразование 'max(A,C)' и
22                                    check_B:
23 00000102 B8[00000000]              mov eax,max
24 00000107 E890FFFFFF              call atoi ; Вызов подпрограммы перевода
25 0000010C A3[00000000]              mov [max],eax ; запись преобразованного
26                                    ; ----- Сравниваем 'max(A,C)' (как
27                                    ; ----- Вывод результата
28                                    fin:
1 Помощь 2 Сохран 3 Блок 4 Замена 5 Копия 6 Пер-ть 7 Поиск 8 Уда-ть 9 МенюМС 10 Выход

```

Рис. 3.11: Ошибка в файле листинга

3. Задание для самостоятельной работы

Программа нахождения наименьшей из 3 целочисленных переменных a,b. Значения переменных 20 варината, полученного при выполнении лабораторной работы № 6. (рис. 3.12).

```

lab7-3.asm          [----] 29 L:[ 12+10  22/ 41] *(733 /1516b) 0010 0x00A  [X][X]
_start:
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp 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:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp 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,msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[min]
call iprintf ; Вывод 'max(A,B,C)'
call quit ; Выход
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ить 7Поиск 8Удалить 9МенюМС 10Выход

```

Рис. 3.12: Программа для сравнения трех заранее ихвестных чисел

Создаем исполняемый файл и проверяем его работу (рис. 3.13).

```

argorobcova@dk3n35 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ ./lab7-3
Наименьшее число: 2
argorobcova@dk3n35 ~/work/arch-pc/lab07 $ █

```

Рис. 3.13: Создание и запуск файла

Программа, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции 20 варианта, полученный при выполнении лабораторной работы № 6 (рис. 3.14).

```

lab7-4.asm [----] 4 L: [ 1+ 6 7/ 51] *(127 / 612b) 0032 0x020
#include 'in_out.asm'
SECTION .data
A1 db 'x-a ,x=>a' ,0
X1 db '5, x<a' ,0
msg1 DB 'Введите значение X:',0
msg2 DB 'Введите значее a:',0
otv DB 'Ответ: ',0

SECTION .bss
X resb 10
A resb 10
F resb 10

SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint
mov ecx, A
mov edx, 80
call sread
mov eax, A
call atoi
mov [A], eax

mov eax, msg2
call sprint

```

Рис. 3.14: Название рисунка

Создаем исполняемый файл и проверяем его работу для значений x и a из таблицы (рис. 3.15).

```

argorobcova@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
argorobcova@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
argorobcova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение X:1
Введите значее a:2
1
argorobcova@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение X:2
Введите значее a:1
5
argorobcova@dk3n55 ~/work/arch-pc/lab07 $ █

```

Рис. 3.15: Создание и запуск файла

4 Выводы

Изучили команды условного и безусловного переходов. Приобрели навыки написания программ с использованием переходов. Ознакомились с назначением и структурой файла листинга.