

Отчет по лабораторной работе No8

Дисциплина: архитектура компьютера

Уточкина Ульяна Андреевна

Содержание

1	Цель работы	3
2	Задание	4
3	Выполнение лабораторной работы	5
4	Самостоятельная работа	9
5	Выводы	11

1 Цель работы

Познакомиться с программой ветвлений, изучить команды условного и безусловного перехода

2 Задание

Написать программу с использованием ветвлений

3 Выполнение лабораторной работы

1. Создаём файл lab8-1.asm и открываем его.

```
[uautochkina@fedora ~]$ mkdir ~/work/arch-pc/lab08  
[uautochkina@fedora ~]$ cd ~/work/arch-pc/lab08  
[uautochkina@fedora lab08]$ touch lab8-1.asm
```

Рис. 3.1: Создание каталога

2. Записываем в файл код из листинга, откомпилируем и запускаем файл

```
[uautochkina@fedora lab08]$ nasm -f elf lab8-1.asm  
[uautochkina@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o  
[uautochkina@fedora lab08]$ ./lab8-1  
Сообщение No 2  
Сообщение No 3  
[uautochkina@fedora lab08]$
```

Рис. 3.2: Запуск кода

3. Теперь изменяем код программы и смотрим результат

```

%include 'in_out.asm' ;
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 ;
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ;
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ;
call sprintf ; 'Сообщение No 3'
_end:
call quit ;|

```

Рис. 3.3: Измененный код из листинга

```

[uautochkina@fedora lab08]$ ./lab8-1
Сообщение No 2
Сообщение No 1

```

Рис. 3.4: Результат программы

4. Создаём новый файл lab8-2.asm, записываем в него предложенный код и смотрим на результат при разных значениях переменных

```

[uautochkina@fedora lab08]$ nasm -f elf lab8-2.asm
[uautochkina@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[uautochkina@fedora lab08]$ ./lab8-2
Введите B: 7
Наибольшее число: 50
[uautochkina@fedora lab08]$ ./lab8-2
Введите B: 70
Наибольшее число: 70
[uautochkina@fedora lab08]$ ./lab8-2
Введите B: 108
Наибольшее число: 108
[uautochkina@fedora lab08]$

```

Рис. 3.5: Нахождение максимального числа

5. С помощью команды `nasm -f elf -l lab8-2.lst lab8-2.asm` создаём файл листинга

```

lab8-2.lst  [----]  0 L:  1+ 0  1/225] *(0  /13507b) 0032 0x020  [*][X]
1          %include 'in_out.asm'
2          <1> ;----- slen -----
3          <1> ; Функция вычисления длины сообщения
4          <1> slen:.....
5 00000000 53          <1>  push  ebx.....
6 00000001 89C3        <1>  mov   ebx, eax.....
7          <1>.....
8          <1> nextchar:.....
9 00000003 803800      <1>  cmp   byte [eax], 0...
10 00000006 7403       <1>  jz    finished.....
11 00000008 40          <1>  inc   eax.....
12 00000009 EBF8       <1>  jmp   nextchar.....
13          <1>.....
14          <1> finished:
15 0000000B 29D8        <1>  sub   eax, ebx
16 0000000D 5B          <1>  pop   ebx.....
17 0000000E C3         <1>  ret.....
18          <1>.
19          <1>.
20          <1> ;----- sprint -----
21          <1> ; Функция печати сообщения
22          <1> ; входные данные: mov eax,<message>
23          <1> sprint:
24 0000000F 52          <1>  push  edx
25 00000010 51          <1>  push  ecx
26 00000011 53          <1>  push  ebx
27 00000012 50          <1>  push  eax
28 00000013 E8E8FFFF   <1>  call  slen

```

Рис. 3.6: Вывод файла листинга

6. Изменяем код и видим, что выводится ошибка

```
,  
mov ecx,[A] ; 'ecx = A'  
mov [max],ecx ; 'max = A'
```

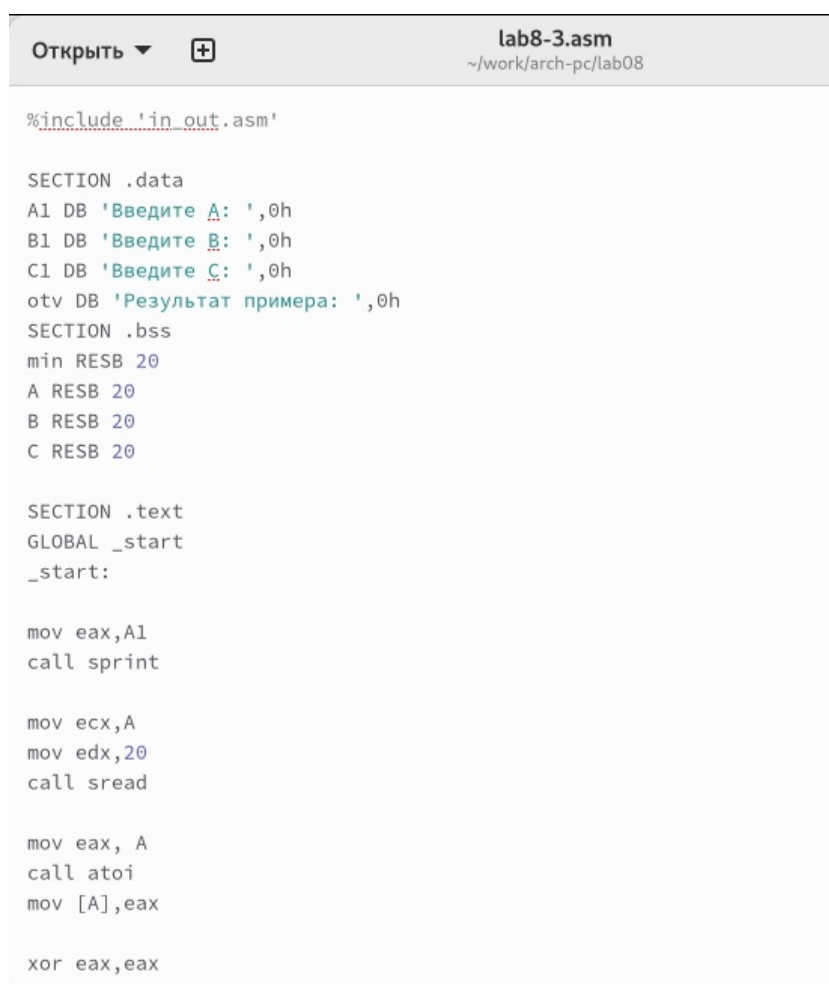
Рис. 3.7: Удаление

```
[uautochkina@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm  
lab8-2.asm:25: error: invalid combination of opcode and operands
```

Рис. 3.8: Вывод ошибки

4 Самостоятельная работа

1. Мне выпал вариант 17, создаем файлы lab8-3.asm и lab8-4.asm, пишем код и выводим результат



```
lab8-3.asm
~/work/arch-pc/lab08

%include 'in_out.asm'

SECTION .data
A1 DB 'Введите A: ',0h
B1 DB 'Введите B: ',0h
C1 DB 'Введите C: ',0h
otv DB 'Результат примера: ',0h
SECTION .bss
min RESB 20
A RESB 20
B RESB 20
C RESB 20

SECTION .text
GLOBAL _start
_start:

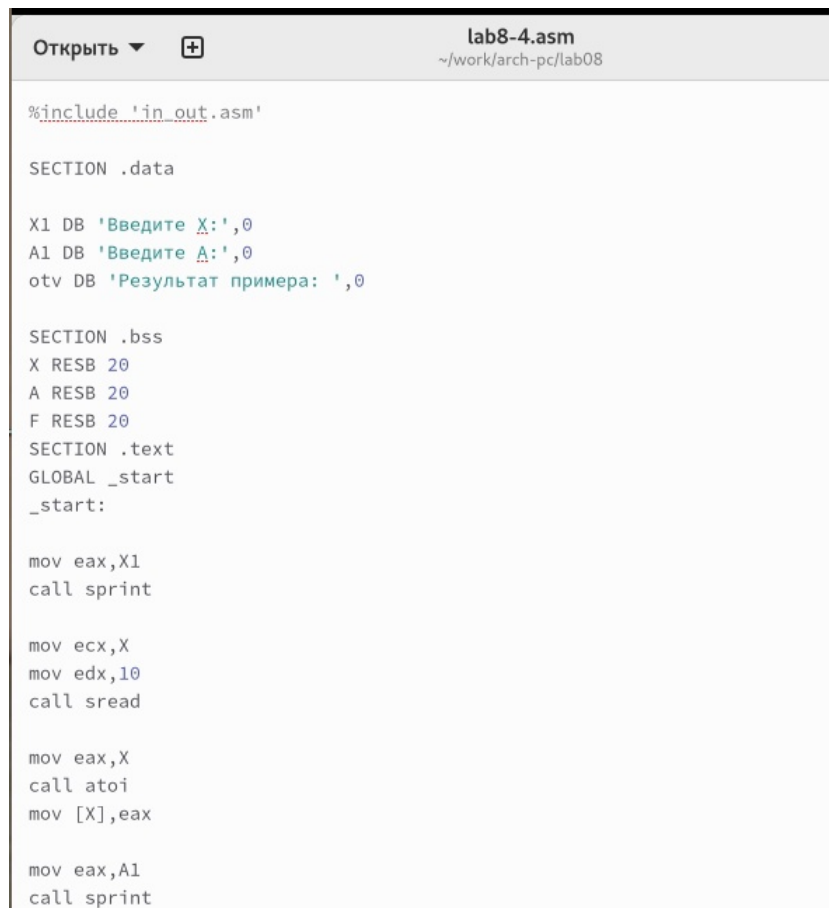
mov eax,A1
call sprint

mov ecx,A
mov edx,20
call sread

mov eax, A
call atoi
mov [A],eax

xor eax,eax
```

Рис. 4.1: Первая задача



```
Открыть ▾ + lab8-4.asm
~/work/arch-pc/lab08

%include 'in_out.asm'

SECTION .data

X1 DB 'Введите X:',0
A1 DB 'Введите A:',0
otv DB 'Результат примера: ',0

SECTION .bss
X RESB 20
A RESB 20
F RESB 20

SECTION .text
GLOBAL _start
_start:

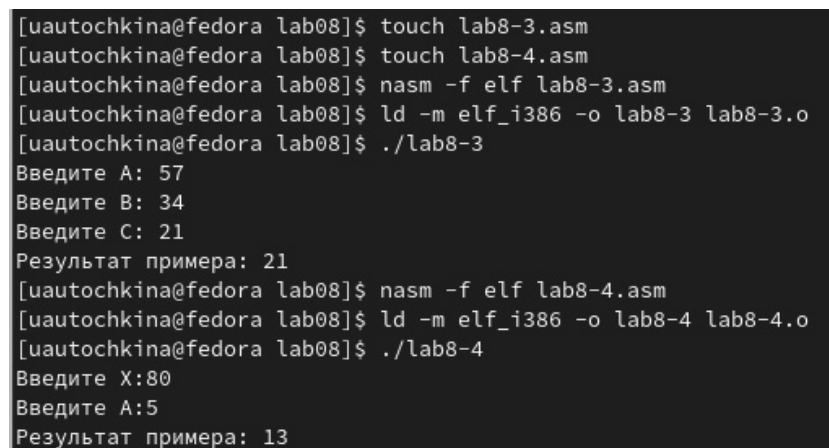
mov eax,X1
call sprint

mov ecx,X
mov edx,10
call sread

mov eax,X
call atoi
mov [X],eax

mov eax,A1
call sprint
```

Рис. 4.2: Вторая задача



```
[uautochkina@fedora lab08]$ touch lab8-3.asm
[uautochkina@fedora lab08]$ touch lab8-4.asm
[uautochkina@fedora lab08]$ nasm -f elf lab8-3.asm
[uautochkina@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[uautochkina@fedora lab08]$ ./lab8-3
Введите A: 57
Введите B: 34
Введите C: 21
Результат примера: 21
[uautochkina@fedora lab08]$ nasm -f elf lab8-4.asm
[uautochkina@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[uautochkina@fedora lab08]$ ./lab8-4
Введите X:80
Введите A:5
Результат примера: 13
```

Рис. 4.3: Результаты

5 Выводы

В этой работе я познакомилась с условным и безусловным переходом. А также написала программы с использованием ветвления.