Лабораторная работа №4. Ветвления: условные и безусловные переходы. Сравнения. Стек. Вызовы процедур.

Макаров П. А.

1 Краткая теория

1. x86 Assembly Language — Википедия;
2. R. Toal. x86 Assembly Language Programming;
3. D. Evans. x86 Assembly Guide;
4. A. Ferrari. A Tiny Guide to Programming in 32-bit x86 Assembly Language;
5. `http://www.techhelpmanual.com`
6. DOS API — Википедия;
7. Стек — Википедия;
8. x86 Disassembly/The Stack — Wikibooks;
9. Yung-Yu Chuang. Procedure;

2 Задания для самостоятельного решения

1. Короткие переходы.

```
-a 100
072E:0100 jmp 103
072E:0102 nop
072E:0103 jmp 100
072E:0105
-d 100 104
```

```
-u 100 104
-p=100
-p
-p
-p
```

2. Длинные переходы.

```
-a 100
072E:0100 jmp 200
072E:0103
-a 200
072E:0200 jmp 100
072E:0203
-d 100 102
-d 200 202
-p=100
-p
-p
-p
```

3. Переходы по результатам сравнения.

```
-a 100
072E:0100 mov ah, 09
072E:0102 mov dx, 114
072E:0105 int 21
072E:0107 mov ah, 01
072E:0109 int 21
072E:010B cmp al, 30
072E:010D jne 107
072E:010F mov ax, 4C00
072E:0112 int 21
072E:0114 db 'Input zero: $'
072E:0121
-d 100 120
-g=100
```

4. Напишите программу, завершающую свою работу только в том случае, когда пользователь нажмёт на клавиатуре клавишу, соответствующую десятичной цифре (любой символ в диапазоне от 0 до 9).

5. Исследуйте в отладчике работу следующей программы. На каждом шаге программы отслеживайте состояние регистра SP. Также проанализируйте память программы (следите за состоянием как переменных, так и стека).

```
-a 100
072E:0100 xor ax, ax
072E:0102 mov al, [119]
072E:0105 push ax
072E:0106 mov ax, [11A]
072E:0109 push ax
072E:010A xor ax, ax
072E:010C pop ax
072E:010D mov [119], al
072E:0110 pop ax
072E:0111 mov [11A], ax
072E:0114 mov ax, 4C00
072E:0117 int 21
072E:0119 db 77
072E:011A dw 4321
072E:011C
-p=100
-p
-p
-d fffc
-p
-p
-d fffa
-p
-p
-p
-d 119 119
-p
-p
-d 11A 11B
-p
-p
```

6. Изучите теорию по использованию стека при организации вызовов процедур. Закрепите теорию на практике, исследуя данную программу.

```
-a 100
```

```
072E:0100 mov si, 134
072E:0103 mov dx, 135
072E:0106 push dx
072E:0107 call 159
072E:010A call 167
072E:010D mov [si], al
072E:010F mov dx, 13F
072E:0112 push dx
072E:0113 call 159
072E:0116 call 167
072E:0119 and [si], al
072E:011B mov dx, 14B
072E:011E push dx
072E:011F call 159
072E:0122 mov ah, 02
072E:0124 mov dl, [si]
072E:0126 int 21
072E:0128 mov dx, 156
072E:012B push dx
072E:012C call 159
072E:012F mov ax, 4c00
072E:0132 int 21
072E:0134 db '?'
072E:0135 db 'Input x: $'
072E:013F db 0A, 0D, 'Input y: $'
072E:014B db 0A, 0D, 'x & y = $'
072E:0156 db 0A, 0D, '$'
072E:0159 push bp
072E:015A mov bp, sp
072E:015C mov ah, 09
072E:015E mov dx, [bp+4]
072E:0161 int 21
072E:0163 pop bp
072E:0164 ret 2
072E:0167 mov ah, 01
072E:0169 int 21
072E:016B ret
072E:016C
-n conj2chr.com
-r cx 6C
-w
```

```
-q
C:\>conj2chr
```