Университет ИТМО

Факультет программной инженерии и компьютерной техники
Кафедра вычислительной техники

Лабораторная работа № 2 по дисциплине "Организация ЭВМ и систем"

Выполнили: Айтуганов Д. А.
Чебыкин И. Б.

Группа: P3301

Проверяющий: Скорубский В. И.

СПб, 2016

Цель работы

1. Обретение базовых навыков работы с MCS51 в среде Keil.
2. Изучение архитектуры MCS51.
3. Изучение C51 и A51.

Задание

Разработать программу ввода и вывода целых чисел со знаком в C51 и A51 через порты.

Разработать программу ввода и вывода дробных двоичных чисел со знаком в C51 и A51 через порты.

Исходный код

Целые числа

```
#include <reg51.h>
typedef unsigned char uint8_t;

uint8_t bin_to_bcd(uint8_t x){
  uint8_t mask = 0;
  if( (char)x < 0 ){
    // Sign bit
    mask = 0x80;
    x = -x;
  }
  return ((x / 10) << 4) | (x % 10) | mask;
}

uint8_t bcd_to_bin(uint8_t x){
    return ( ((x>>4) & 0x07) * 10 + (x & 0x0f) ) * (x & 0x80 ? -1 : 1);
}

void main(){
  uint8_t num = bcd_to_bin(P0);
    P1 = bin_to_bcd(num);
    while(1);
}
```

Листинг 1: src/int.c

Дробные числа

```
#include <reg51.h>
typedef unsigned char uint8_t;
typedef unsigned int uint16_t;

uint16_t bin_to_bcd( uint16_t x ) {
  // Sign bit
    uint8_t mask = x & 0x80 ? 0x80 : 0;
    x *= 10;
    return ((x & 0x700) >> 4) | ((((x & 0xff) * 10) & 0xf00) >> 8) | mask;
}

uint16_t bcd_to_bin( uint16_t x ) {
    return (((((x & 0x70) >> 4) * 10 + (x & 0x0f)) << 8) / 100)*(x & 0x80 ? -1 : 1);
}

void main() {
    uint16_t num = bcd_to_bin(P0);
    P1 = bin_to_bcd(num);
    while(1);
```

```
}
```

Листинг 2: src/float.c

Реализация на ассемблере

```
Xseg at 20
buf: ds 2
Cseg at 0

mov DPTR, #40H; ;buffer

sjmp check_sign

input:
check_sign:
    mov a, P1
    anl a, #0xf0        ; (xA>>4)
    swap a            ; (xA>>4)
    subb a, #0x0f       ; x-13

    jnz pos
    mov r3, #1          ; neg flag

pos:
    mov a, P1         ; lower half of P1
    anl a, #0x0f
    mov b, #100
    mul ab
    mov r6, a

    mov a, P0         ; upper half of P1
    swap a
    anl a, #0x0f
    mov b, #10
    mul ab
    mov b, r6
    add a, b
    mov r6, a

    mov a, P0         ; lower half of P1
    anl a, #0x0f
    mov b, r6
    add a, b
    mov r6, a

test_sign:
    mov a, r3
    subb a, #1
    jnz to_memory

    mov a, #0xff        ; two's complement
    subb a, r6
    inc a

to_memory:
    movx @dptr, a


output:
    mov r3, #0
    mov a, #0
    movx a, @dptr
    mov  r6, a
    jnb ACC.7, pos_out

neg_out:
    mov a, #0xFF        ; inverse number back
    subb a, r6
    inc a
    mov r6, a
    mov r3, #1

pos_out:
    mov a, r6
    mov b, #10
```

```
    div ab
    mov r5, b

    mov b, #10
    div ab
    mov r4, a
    mov a, b
    swap a
    orl a, r5

    mov P2, a

    mov a, r4
    mov b, #10
    div ab

    clr a
    add a, r3
    jz done
    mov a, #0xf0

done:
    orl a, b
    mov P3, a

    jmp $
end
```

Листинг 3: src/asm.asm

Вывод