Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

ПИиКТ

# Лабораторная работа 2

по дисциплине

«Архитектура компьютера»

Выполнили: Студент группы P33113

Мансуров Б.Б.

Преподаватель: Тропченко А.Ю.

Санкт-Петербург

2020г

## Задание

Познакомиться с двоично-десятичным и двоичным представлением целых и дробных чисел. Совместить перевод из 10 в 2 и из 2 в 10 в одной программе для целых и дробных чисел и разработать программы на C51 и в Ассемблере A51 для ввода и вывода двузначных чисел. Сравнить листинги .lst программ в C51 и A51 и пояснить различия в программах.

## Исходный текст программы на C51

```
#include <reg51.h>


void bcd_to_bin_int() {
	unsigned char bin, bcd;
	bin = (P1 >> 4) * 10 + (P1 & 0x0f);
	P2 = bin;


	bcd = (((bin / 10) % 10) << 4) | (bin % 10);
	P3 = bcd;
}


void bcd_to_bin_fixed_point() {
	unsigned int bin;
	unsigned char bcd;
	bin = (P1 >> 4) * 10 + (P1 & 0x0f);
	bin <<= 8;
	bin = (bin % 100 > 50) ? bin / 100 + 1 : bin / 100;
	P2 = bin;


	bin *= 10;
	bcd = (bin & 0x0f00) >> 4;
	bcd |= (((bin & 0xff) * 10) & 0xf00) >> 8;
	P3 = bcd;
}


int main() {
	if (P0 == 0) bcd_to_bin_int();
	else bcd_to_bin_fixed_point();
```

```
        return 0;
}
```

## Исходный текст программы на A51

ASSEMBLY LISTING OF GENERATED OBJECT CODE

```
        ; FUNCTION bcd_to_bin_int (BEGIN)
                                ; SOURCE LINE # 3
                                ; SOURCE LINE # 6
0000 E590           MOV     A,P1
0002 C4             SWAP    A
0003 540F           ANL     A,#0FH
0005 75F00A         MOV     B,#0AH
0008 A4             MUL     AB
0009 FF             MOV     R7,A
000A E590           MOV     A,P1
000C 540F           ANL     A,#0FH
000E 2F             ADD     A,R7
000F FF             MOV     R7,A
;---- Variable 'bin' assigned to Register 'R7' ----
                                ; SOURCE LINE # 7
0010 F5A0           MOV     P2,A
                                ; SOURCE LINE # 9
0012 75F00A         MOV     B,#0AH
0015 84             DIV     AB
0016 75F00A         MOV     B,#0AH
0019 84             DIV     AB
001A AEF0           MOV     R6,B
001C EE             MOV     A,R6
001D C4             SWAP    A
001E 54F0           ANL     A,#0F0H
0020 FE             MOV     R6,A
0021 EF             MOV     A,R7
0022 75F00A         MOV     B,#0AH
```

```
0025 84             DIV     AB
0026 E5F0           MOV     A,B
0028 4E             ORL     A,R6
;---- Variable 'bcd' assigned to Register 'R7' ----
                            ; SOURCE LINE # 10
0029 F5B0           MOV     P3,A
                            ; SOURCE LINE # 11
002B 22             RET
        ; FUNCTION bcd_to_bin_int (END)

        ; FUNCTION bcd_to_bin_fixed_point (BEGIN)
                            ; SOURCE LINE # 13
                            ; SOURCE LINE # 17
0000 E590           MOV     A,P1
0002 C4             SWAP    A
0003 540F           ANL     A,#0FH
0005 75F00A         MOV     B,#0AH
0008 A4             MUL     AB
0009 FF             MOV     R7,A
000A E590           MOV     A,P1
000C 540F           ANL     A,#0FH
000E 7C00           MOV     R4,#00H
0010 2F             ADD     A,R7
0011 FF             MOV     R7,A
0012 EC             MOV     A,R4
0013 35F0           ADDC    A,B
;---- Variable 'bin' assigned to Register 'R2/R3' ----
0015 AB07           MOV     R3,AR7
0017 FA             MOV     R2,A
                            ; SOURCE LINE # 18
0018 EB             MOV     A,R3
```

```
0019 7B00          MOV    R3,#00H
001B FA            MOV    R2,A
                          ; SOURCE LINE # 19
001C 7D64          MOV    R5,#064H
001E 7F00          MOV    R7,#00H
0020 FE            MOV    R6,A
0021 120000    E   LCALL  ?C?UIDIV
0024 D3            SETB   C
0025 ED            MOV    A,R5
0026 9432          SUBB   A,#032H
0028 EC            MOV    A,R4
0029 9400          SUBB   A,#00H
002B 4014          JC     ?C0002
002D 7C00          MOV    R4,#00H
002F 7D64          MOV    R5,#064H
0031 7F00          MOV    R7,#00H
0033 AE02          MOV    R6,AR2
0035 120000    E   LCALL  ?C?UIDIV
0038 EF            MOV    A,R7
0039 2401          ADD    A,#01H
003B FF            MOV    R7,A
003C E4            CLR    A
003D 3E            ADDC   A,R6
003E FE            MOV    R6,A
003F 800B          SJMP   ?C0003
0041       ?C0002:
0041 7C00          MOV    R4,#00H
0043 7D64          MOV    R5,#064H
0045 AF03          MOV    R7,AR3
0047 AE02          MOV    R6,AR2
0049 120000    E   LCALL  ?C?UIDIV
004C       ?C0003:
```

```
004C AA06           MOV     R2,AR6
004E AB07           MOV     R3,AR7
                            ; SOURCE LINE # 20
0050 8BA0           MOV     P2,R3
                            ; SOURCE LINE # 22
0052 7C00           MOV     R4,#00H
0054 7D0A           MOV     R5,#0AH
0056 AF03           MOV     R7,AR3
0058 AE02           MOV     R6,AR2
005A 120000     E   LCALL   ?C?IMUL
005D AA06           MOV     R2,AR6
005F AB07           MOV     R3,AR7
                            ; SOURCE LINE # 23
0061 EA             MOV     A,R2
0062 540F           ANL     A,#0FH
0064 C4             SWAP    A
0065 F8             MOV     R0,A
0066 54F0           ANL     A,#0F0H
0068 C8             XCH     A,R0
0069 68             XRL     A,R0
006A E4             CLR     A
006B C4             SWAP    A
006C 540F           ANL     A,#0FH
006E 48             ORL     A,R0
;---- Variable 'bcd' assigned to Register 'R1' ----
006F F9             MOV     R1,A
                            ; SOURCE LINE # 24
0070 7E00           MOV     R6,#00H
0072 AF03           MOV     R7,AR3
0074 120000     E   LCALL   ?C?IMUL
```

```
0077 EE          MOV    A,R6
0078 540F        ANL    A,#0FH
007A 4201        ORL    AR1,A
                        ; SOURCE LINE # 25
007C 89B0        MOV    P3,R1
                        ; SOURCE LINE # 26
007E 22          RET
        ; FUNCTION bcd_to_bin_fixed_point (END)

        ; FUNCTION main (BEGIN)
                        ; SOURCE LINE # 28
                        ; SOURCE LINE # 29
0000 AF80        MOV    R7,P0
0002 EF          MOV    A,R7
0003 7005        JNZ    ?C0005
0005 120000  R   LCALL  bcd_to_bin_int
0008 8003        SJMP   ?C0006
000A       ?C0005:
                        ; SOURCE LINE # 30
000A 120000  R   LCALL  bcd_to_bin_fixed_point
000D       ?C0006:
                        ; SOURCE LINE # 31
000D E4          CLR    A
000E FE          MOV    R6,A
000F FF          MOV    R7,A
                        ; SOURCE LINE # 32
0010 22          RET
        ; FUNCTION main (END)
```

MODULE INFORMATION: STATIC OVERLAYABLE

```
  CODE SIZE      =   188   ----
  CONSTANT SIZE  =   ----  ----
  XDATA SIZE     =   ----  ----
  PDATA SIZE     =   ----  ----
  DATA SIZE      =   ----  ----
  IDATA SIZE     =   ----  ----
  BIT SIZE       =   ----  ----
END OF MODULE INFORMATION.
```

## Сравнение листингов

Code size = 306 C51
Code size = 188 Assembly

В отличие от скомпилированного кода, написанный вручную ассемблерный код более эффективно использует регистры и выполняет операции.

## Вывод

Выполнив данную лабораторную работу, я понял, что писать на ассемблере это ужасно, но иногда целесообразнее написать ассемблерный код вручную, потому что мы сократим размер кода и повысим производительность. Но так как на ассемблере писать сложно и за все надо следить то из-за этого скорость разработки падает. Поэтому программисты предпочитают писать код на языках высокого уровня. Иногда конечно целесообразно переписать участок кода на ассемблере.