同济大学

TONGJI UNIVERSITY

《计算机系统实验》

实验报告

实验名称	μC/OS-II 应用程序开发
实验成员	罗劲桐（1951443）
日　　期	2022 年　6 月　27 日

1、实验目的

- 回顾编译原理、操作系统等课程；
- 贯通计算机相关课程

2、实验内容

根据自己移植操作系统，设计一个应用程序，包括图形界面。按照《自己动手写 CPU》中，对 μC/OS-II 系统的分析过程，修改 μC/OS-II 系统，完成一个使用 uart 输入输出的，带有 UI 的一位计算器，支持加减乘除计算。

实验过程中，主要做出以下修改：

- 在之前的基础上，修改 openmips.c，实现用户任务和显示 UI。
- 修改 μC/OSII 系统内核，实现 uart 输入输出功能
- 修改 bootloader 模块，以适配 CPU 频率和 UART 频率

3、实验步骤

3.1. 修改 MIPS CPU，总线模块和移植操作系统

在实验一、二中，参考《自己动手写 CPU 》，已经实现了这部分功能。

3.2. 用户任务文件修改

3.2.1. 修改 openmips.c 的用户任务函数

在 TaskStart 用户任务函数，分别实现了 UI 输出，uart 输入和输出，用户任务逻辑模块。其中 UI 输出采用 uart 输出方式实现，wart 输入输出通过调用系统函数完成。用户任务逻辑模块包括 uart 输入两个运算数字和运算符，并输出结果运算式。

```
1.  void  TaskStart (void *pdata)
2.  {
3.      INT32U count = 0;
4.      INT32U uart_in=0,stat=0,op0,op1,op2,res;
5.      pdata = pdata;           /* Prevent compiler warning                */
6.      OSInitTick();          /* don't put this function in main()       */
7.      for (;;) {
8.         if(count <= 102)
9.              {
10.          uart_putc(Info[count]);
```

```
11.          uart_putc(Info[count+1]);
12.         }
13.         gpio_out(count);
14.         count=count+2;
15.         uart_in=uart_getc();
16.         if(uart_in!=0){
17.              if(stat=0){
18.                         op0=uart_in-48;
19.                         stat+=1;
20.                         uart_print_str(Compute);
21.                         }
22.                         if(stat=1){
23.                         op1=uart_in;
24.                         stat+=1;
25.                         uart_print_str(Compute);
26.                         }
27.                         if(stat=2){
28.                         op2=uart_in-48;
29.                         stat+=1;
30.                         uart_print_str(Compute);
31.
32.                         if(op1==43)
33.                                   {
34.                                   res=op0+op2;
35.                                   }
36.                                   if(op1==45)
37.                                   {
38.                                   res=op0-op2;
39.                                   }
40.                                   if(op1==42)
41.                                   {
42.                                   res=op0*op2;
43.                                   }
```

装 订 线

```
                                if(op1==47)
                                {
                                res=op0/op2;
                                }
                                uart_putc(op0+48);
                                uart_putc(op1);
                                uart_putc(op2+48);
                                uart_putc('=');
                                uart_putc(res+48);
                                uart_print_str(Compute);
                        }
                }

        OSTimeDly(10);  /* Wait 100ms                */
    }

}
```

3.2.2. 新增 uart 读取函数到 openmips.c

uart 读取中断例程会将 uart 读取的数据放入 cp0 $7 的保留寄存器中，在 uart_getc 中，若读到非 0 的$7 寄存器，则代表 uart 经过中断读入流程，有新输入。在读取后清零$7 寄存器，等待下一个输入。

```
char Compute[]="\n---------\n|7|8|9|+|\n---------\n|4|5|6|-|\n---------\n|1|2|3|*|\n---------\n|0|  =  |\n---------\n\n";

//uart
INT32U uart_getc(void)
{
  INT32U temp = 0;
  asm volatile("mfc0   $t0,$7");
  asm volatile("sw   $t0,0($sp)");
  asm volatile("mtc0   %0,$7"   : :"r"(0x0));
  return temp;
}
```

3.2.3. 修改 openmips.h

修改 CPU 频率和 UART 频率，同时增加 uart_getc 的头文件声明。

装 订 线

```
#define IN_CLK 100000000            /* 输入时钟是 100MHz */

/*********************************************************************

                          串口相关参数、函数

*********************************************************************/

#define UART_BAUD_RATE  19200         /* 串口速率是 19200bps */
#define UART_BASE      0x10000000
#define UART_LC_REG    0x00000003    /* Line Control Register */
#define UART_IE_REG     0x00000001    /* Interrupt Enable Register */
#define UART_TH_REG    0x00000000    /* Transmitter Holding Register */
#define UART_LS_REG     0x00000005    /* Line Status Register */
#define UART_DLB1_REG   0x00000000    /* Divisor Latch Byte 1(LSB) */
#define UART_DLB2_REG   0x00000001    /* Divisor Latch Byte 2(MSB) */

/* Line Status Register 的标志位 */
#define UART_LS_TEMT  0x40      /* Transmitter empty */
#define UART_LS_THRE  0x20      /* Transmit-hold-register empty */

/* Line Control Register 的标志位 */
#define UART_LC_NO_PARITY       0x00      /* Parity Disable */
#define UART_LC_ONE_STOP        0x00      /* Stop bits: 0x00 表示 one stop bit */
#define UART_LC_WLEN8  0x03     /* Wordlength: 8 bits */

extern void uart_init(void);
extern void uart_putc(char);
extern void uart_print_str(char*);
extern void uart_print_int(unsigned int);
extern INT32U uart_getc(void);
```

3.3. μC/OSII 系统内核修改

3.3.1. os_cpu_a.S

与 TickISR 对应的添加 UartISR 中断例程，从 uart 状态寄存器读入状态，判断是否有新的 uart 输入。若没有输入，则返回，否则，将 uart 输入写入到 cp0 $7 的保留寄存器中，用于函数 uart_getc 的输入获取过程。

```
1.  /*  uart read */
2.      .ent UartISR
3.  UartISR:
4.      addiu $29,$29,-24
5.      sw $16, 0x4($29)
6.      sw $8, 0x8($29)
7.      sw $31, 0xC($29)
8.
9.     /*  move  $16, $31 */
10.
11.    lui $8,0x1000
12.    ori $8,$8,0x0005
13.    lb  $16,0x0($8)    # get line status register
14.
15.    andi $16,$16,0x01
16.    beq $16,$0,UARTISREND
17.    nop
18.
19.    lui $8,0x1000
20.    ori $8,$8,0x0000
21.    lb  $16,0x0($8)
22.    mtc0  $16,$7
23.
24. UARTISREND:
25.     /* move  $31, $16 */
26.     lw $31, 0xC($29)
27.     lw $16, 0x4($29)
28.     lw $8, 0x8($29)
29.     addiu $29,$29,24
30.     jr    $31
31.     nop
```

```
32.
33.     .end UartISR
```

3.3.2. 修改 os_cpu_c.c

与 TickISR 对应的添加 UartISR 中断跳转判断，在 cause_ip & 0x00000100!=0 是满足存在 uart 中断，运行 UartISR 中断例程，读入 uart 输入。

```
1.  /*
2.  *********************************************************************************************************
3.  *                                          BSP_Interrupt_Handler
4.  *
5.  * Description: 中断发生时会调用本函数处理具体的中断事宜
6.  *
7.  * Arguments  : None
8.  *
9.  * Note(s)    : 1) Interrupts may or may not be ENABLED during this call.
10. *********************************************************************************************************
11. */
12. void  BSP_Interrupt_Handler (void)
13. {
14.     INT32U   cause_val;
15.     INT32U   cause_reg;
16.     INT32U   cause_ip;
17.     asm ("mfc0   %0,$13"   : "=r"(cause_val));
18.     cause_reg  = cause_val;                    /* 得到 Exc Code       */
19.     cause_ip = cause_reg & 0x0000FF00;
20.     //uart_print_str("I am in BSP_Interrupt_Handler  ");
21.     if((cause_ip & 0x00000400) != 0 )
22.     {
23.             //uart_print_str("Ready To Call TickISR  ");
24.         TickISR(0x50000);
25.         //asm ("mfc0   %0,$13"   : "=r"(cause_val));
26.         //cause_val = cause_val & 0xfffffbff;
27.         //asm volatile("mtc0   %0,$13" : : "r"(cause_val));
```

```
28.             /*TickInterruptClear();*/
29.     }
30.     if((cause_ip & 0x00000100) != 0 )
31.     {
32.         UartISR();
33.     }
34. }
```

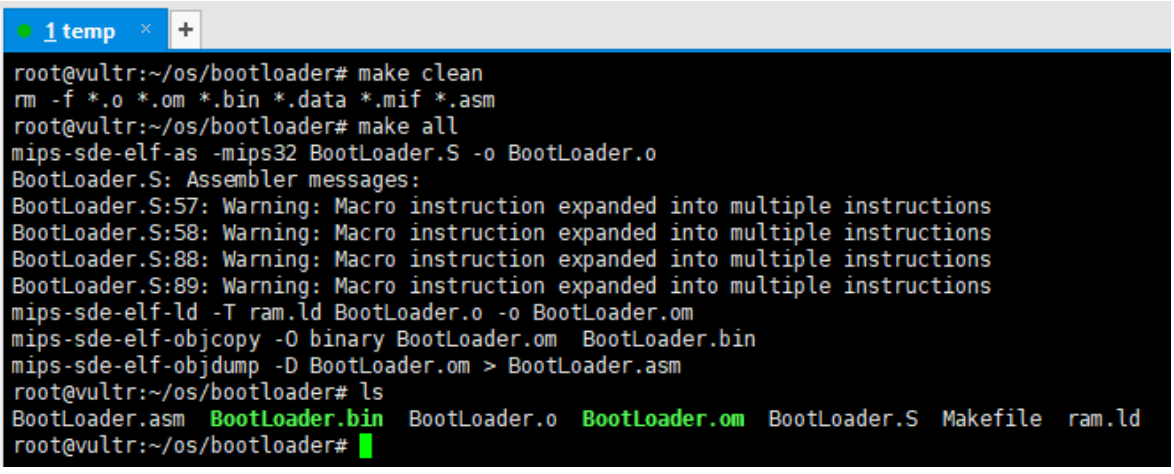
3.3.3. 修改 os_cpu.h

与 TickISR 对应的添加 UartISR 中断函数声明，声明汇编文件对应的函数。

```
1.  void       TickInterruptClear(void);
2.  void       CoreTmrInit(CPU_INT32U tmr_reload);
3.  void       TickISR(CPU_INT32U tmr_reload);
4.  void       UartISR(void);
```

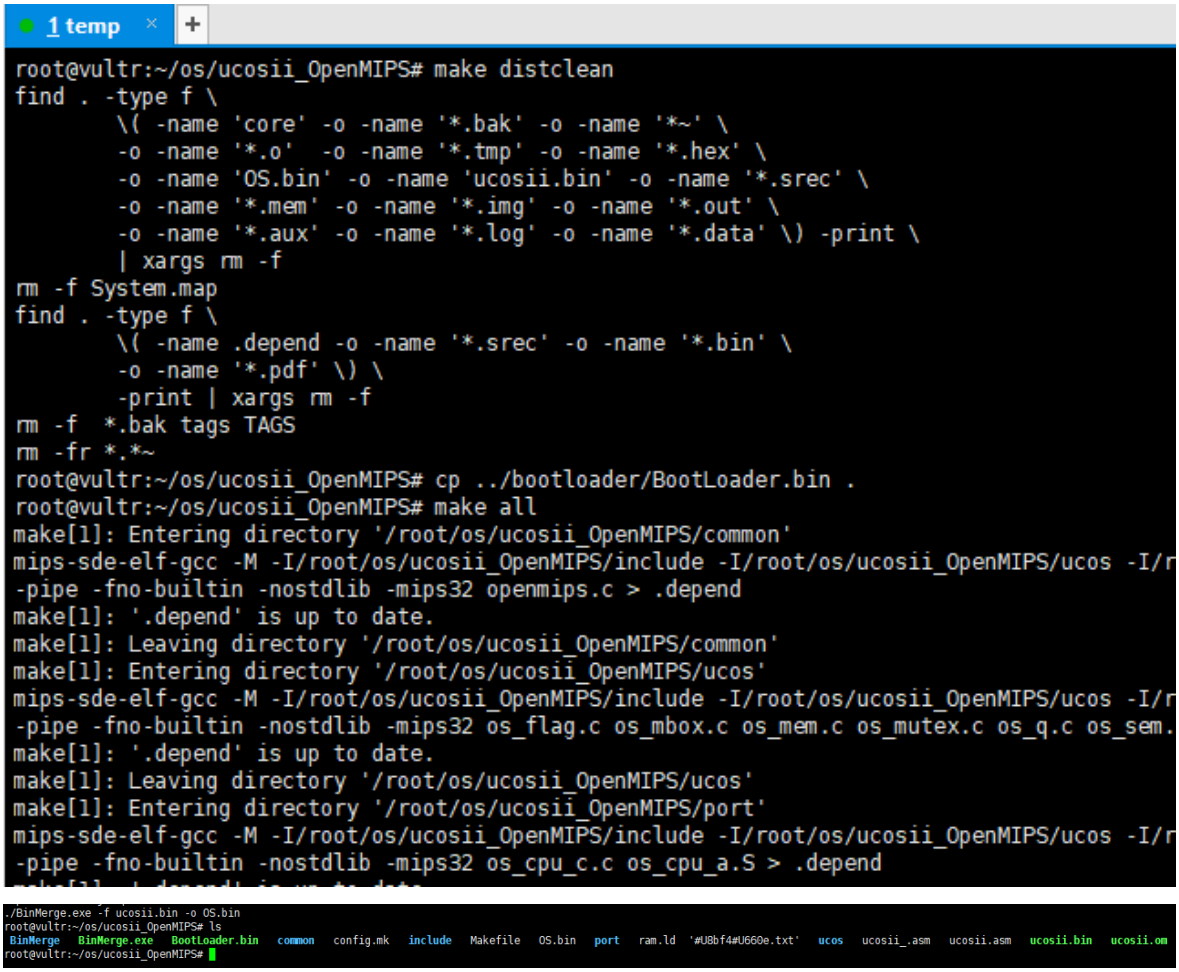
3.4. 重新编译

3.4.1. 编译 bootloader



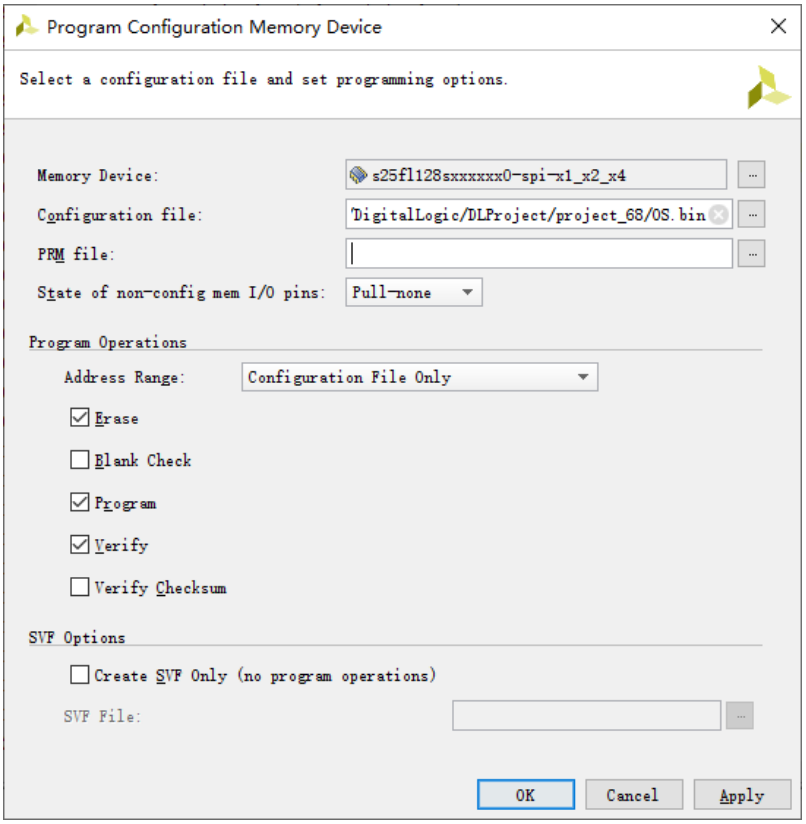
3.4.2. 编译 μC/OSII 系统

由于 makefile 文件构建.depend 文件，需要先进行 distclean，清除依赖。



4、实验结果

4.1. 配置 spi flash 数据，写入 μC/OS II 操作系统的二进制程序文件。



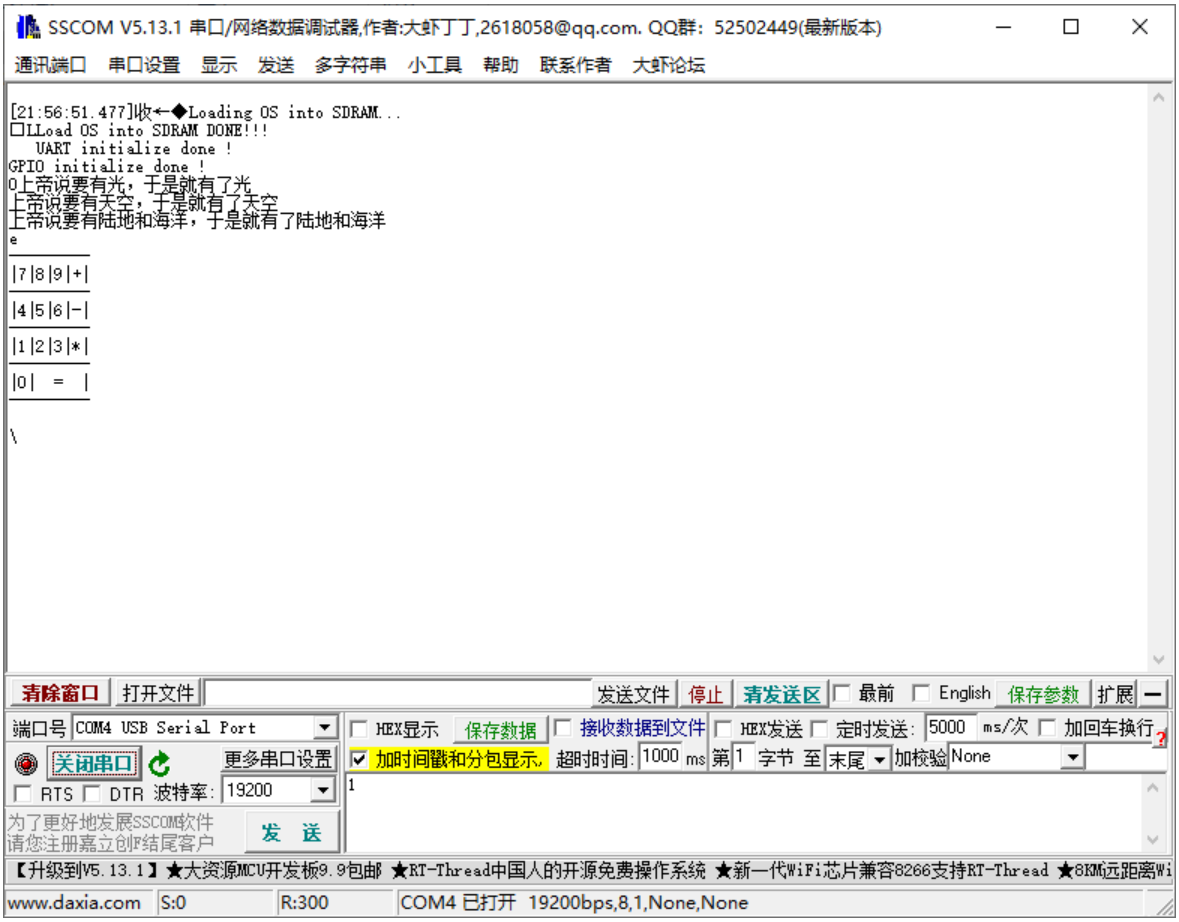
装

订

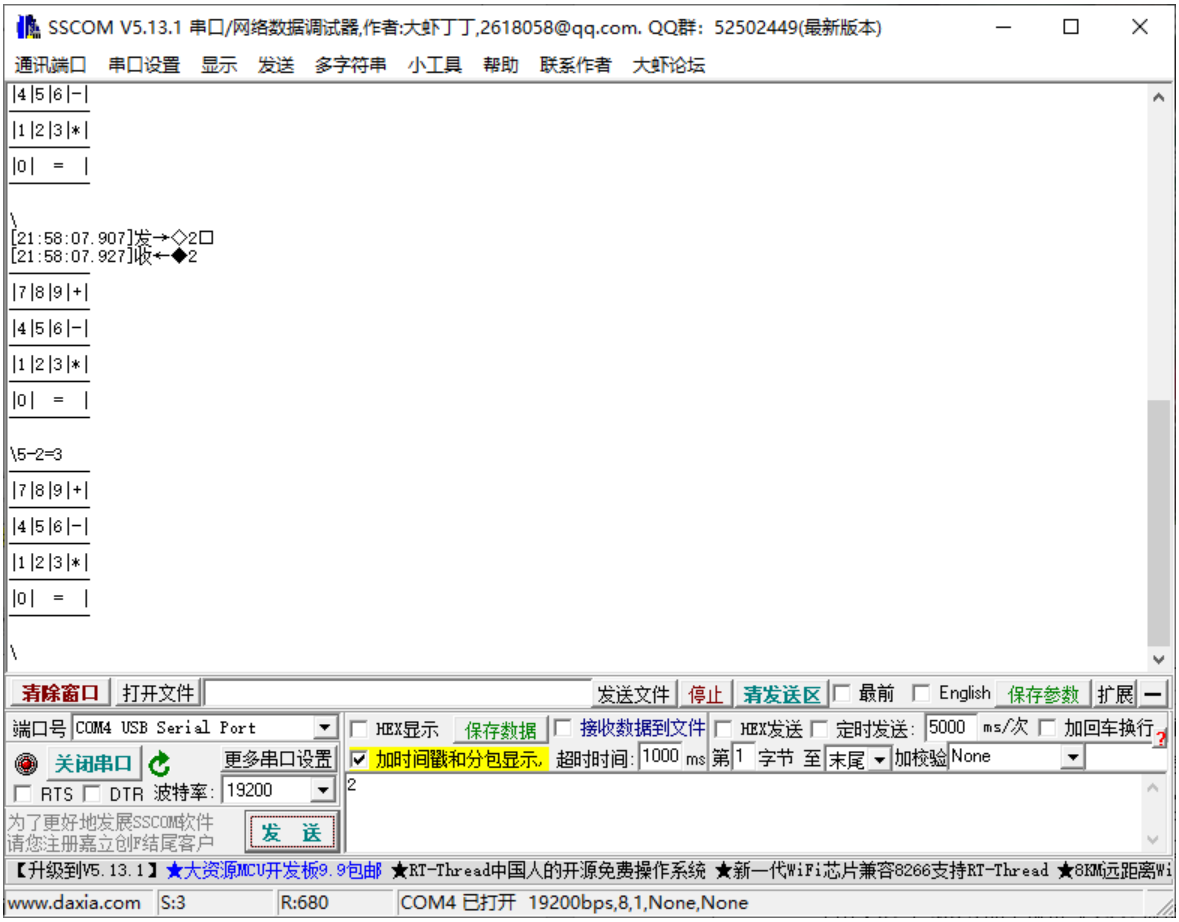
线

4.2. 串口结果

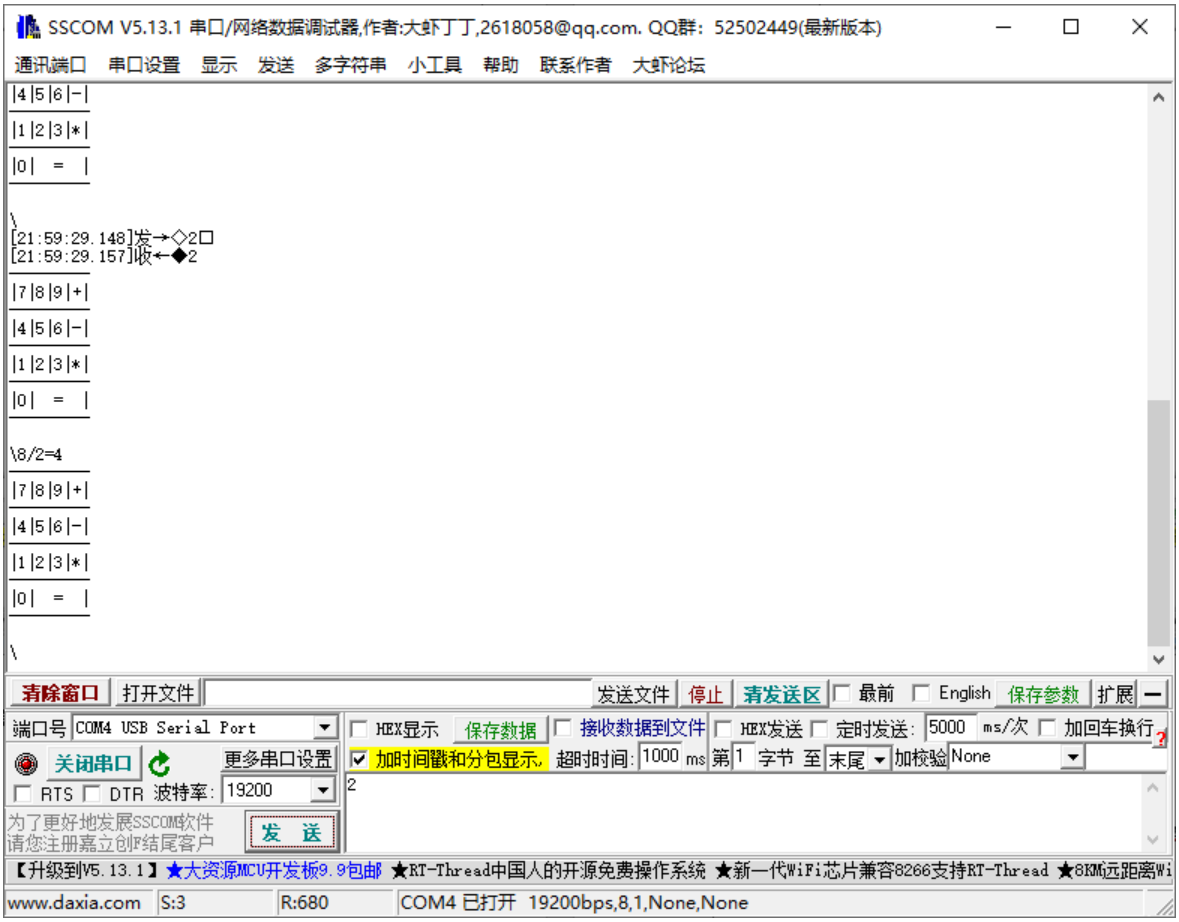
- 打开串口调试，设置波特率 19200bps，8 位数据位、没有奇偶校验位，1 位停止位。将 bit 流下板，观察串口通信



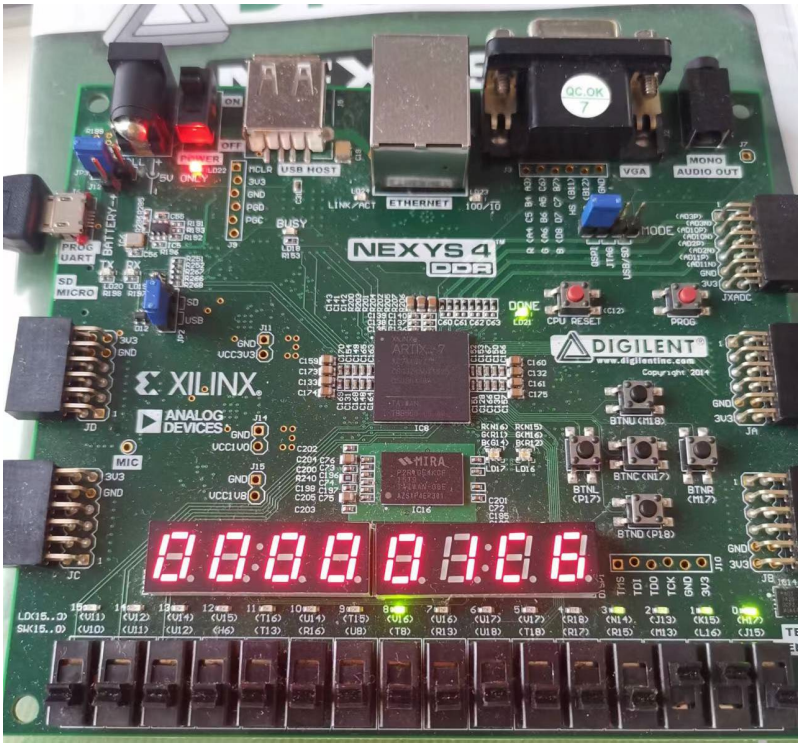
- 通过 uart，依次输入第一个运算数’5’，算符’-’，第二个运算数’2’。得到最终结果后运行结束。



- 也可以尝试其他的运算符。



4.3. 数码管 GPIO 输出，sw[3:0]=6，查看 gpio 输出。



5、实验总结

在本次μC/OS-II 应用程序开发实验中，设计并实现了使用中断实现的 uart 输入输出模块，并以此为基础完成了一个带有 UI 的简易计算器。

设计应用程序开发实验的系统过程中，由于原先的 openmips.c 中只有 uart 输出模块，却没有 uart 输入模块，因此模仿了时钟中断的处理方式实现了 uart 中断例程，能够通过 uart_getc 用户函数单此读取 1B 的 uart 输入。实现 uart 中断例程的过程中，我充分了解了μC/OS-II 操作系统内核中断的处理方式，并接用 cp0 实现了一个简单的中断处理例程，对我来说很有收获。

通过这部分的学习，方便了程序的阅读和快速修改，对于代码的可读性和健壮性有很大帮助，我从中也学到了很多。