中国地质大学（武汉）自动化学院

嵌入式实验报告

课　　程：嵌入式系统实验报告一

学　　号：20201000128

班　　级：231202

姓　　名：刘瑾瑾

指导老师：刘玮

二①二二年十月

一、给出实验二实现开发板上 4 个 LED 按二进制流水灯方式闪烁的程序，其中包含必要注释，并附 AXD 中调试成功的截图；

代码如下：

```
;汇编指令实验
;定义端口 B 寄存器预定义
GPBCON EQU 0x56000010
GPBDAT EQU 0x56000014
GPBUP EQU   0x56000018

    AREA Init,CODE,READONLY ;该伪指令定义了一个代码段，段名为 Init，属性只读
    ENTRY ;程序的入口点标识
ResetEntry
    ;下面这三条语句，主要是用来设置 GPB5--GPB8 为输出属性
    ldr r0,= GPBCON   ;将寄存器 GPBCON 的地址存放到寄存器 r0 中
    ldr r1,=0x15400
    str r1,[r0]            ;将 r1 中的数据存放到地址为 r0 的内存单元中
    ;下面这三条语句，设置 GPB5--GPB8 禁止上拉电阻
    ldr r0,= GPBUP
    ldr r1,=0xffff
    str r1,[r0]
    ldr r2,=GPBDAT    ; 将寄存器 GPBDAT 的地址存放到寄存器 r2 中
ledloop
    str r1,[r2]           ;使 GPB--GPB8 输出高电平，LED1--LED4 全灭
     sub r1,r1,#0x020   ; r1=r1-0x020，每次通过改变 r1 的值实现二进制流水灯
    bl delay              ;调用延迟子程序
    bl delay              ;调用延迟子程序
    b ledloop             ;不断的循环，LED1-LED4 将不停的闪烁
;下面是延迟子程序
delay
    ldr r3,=0xbffff ;设置延迟的时间
delay1
    sub r3,r3,#1 ;r3=r3-1
    cmp r3,#0x0 ;将 r3 的值与 0 相比较
    bne delay1 ;比较的结果不为 0（r3 不为 0），继续调用 delay1,否则执行下一条语句
    mov pc,lr ;返回
    END ;程序结束符
```

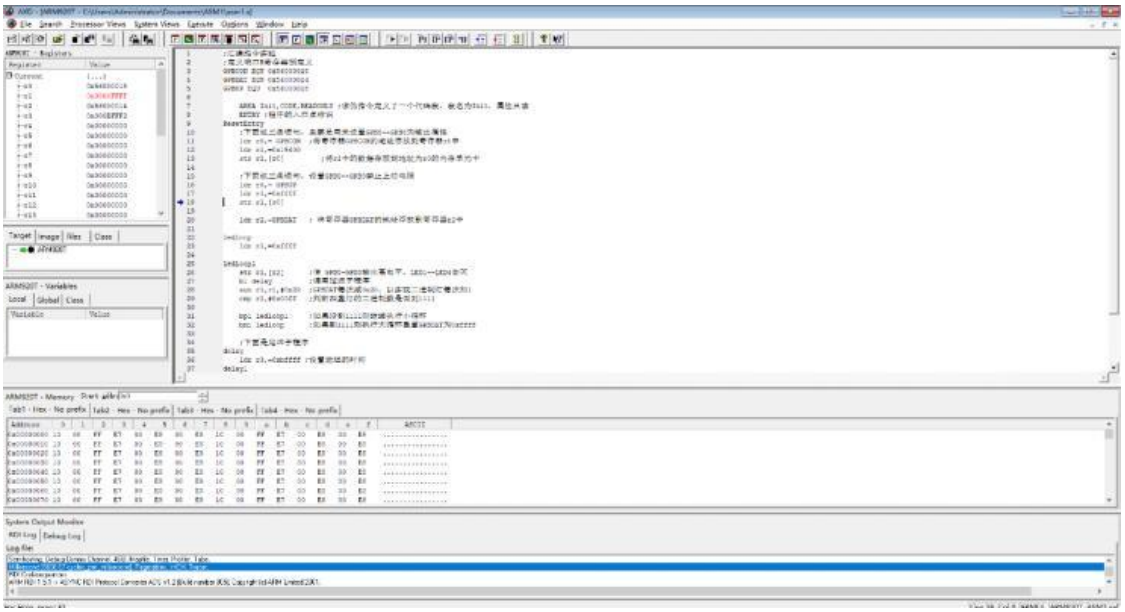

图 1 调试成功截图

二、实验三中软件实现流程图：

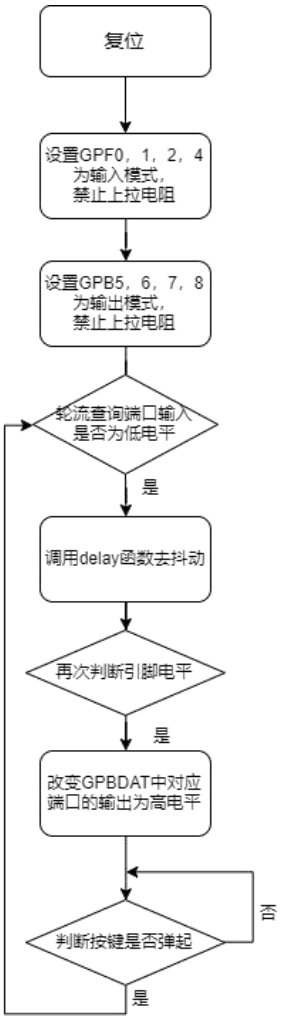

图 2 实验三软件流程图

三、实验四中软件实现流程图：

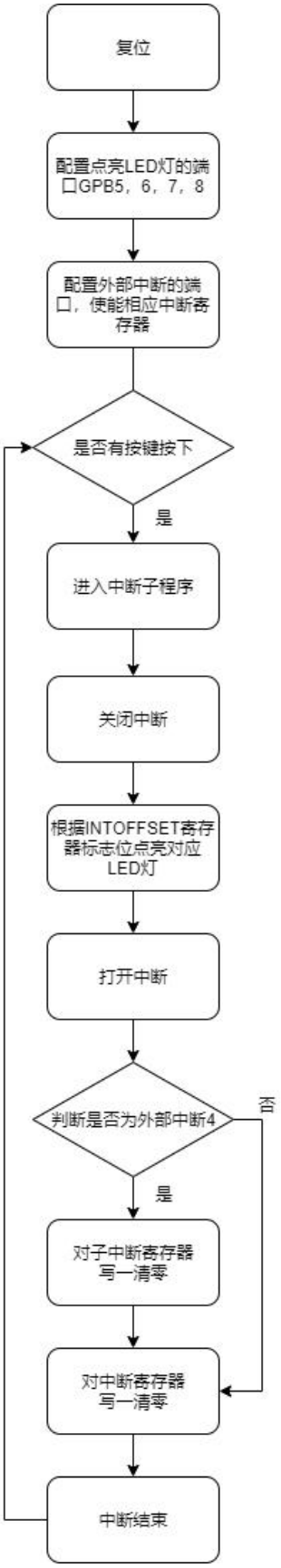

图 3 实验四软件流程图

四、**思考题**

1.在嵌入式系统编程当中，汇编语言和 C 语言分别有什么优势？是否可以完全摈弃其中一种语言？为什么？

答：汇编语言执行效率高，大多用于实时性要求高和精细处理场合，具有直接和简捷的特点，可有效地访问、控制计算机的各种硬件设备，如磁盘、存储器、CPU、I/O 端口等，且占用内存少，执行速度快，是高效的程序设计语言。但其适用范围较窄。不同的机器对应不同汇编指令，使用汇编语言能面向机器并较好地发挥机器的特性，得到质量较高的程序，但开发周期长。而 C 语言则适用范围广，可移植性强，开发周期短，但是与汇编语言比较知性效率低。两者混合使用，可结合各自优点，故不可完全摒弃其中一种语言。

2.ARM 汇编调用 C 语言以及 C 语言调用 ARM 汇编时，如何传递参数？实验二，三程序中参数是如何传递的？

答：（1）对于参数个数可变的子程序，当参数不超过 4 个时，可以使用寄存器 R0-R3 来传递参数；当参数超过 4 个，可以使用数据栈来传递参数。参数传递时，将所有参数看作是存放在连续的内存字单元中的字数据。然后，依次将各字数据传送到寄存器 R0、R1、R2、R3 中，如果参数多于 4 个，将剩余的字数据传送到数据栈中，入栈的顺序与参数顺序相反，即最后一个字数据先入栈。返回结果通过 R0-R4 传递。结果为一个 32 位的整数时，可以通过寄存器 R0 返回。结果为一个 64 位整数时，可以通过寄存器 R0 和 R1 返回，依次类推。更多位数时，使用内存传递。

（2）实验二没有通过寄存器传递参数;实验三通过 R0 传递参数。

3.C 语言中和汇编语言中是如何操作寄存器的？

答：C 语言：通过位操作如位或，位与，移位操作等改变寄存器的值，或者直接向寄存器写入立即数，进行写操作。同样，也可以通过位操作读出寄存器对应位的值。

汇编语言：通过 LDR,STR,MOV 等汇编指令，对寄存器进行读或写操作。

4.比较实验二、三和四中 ADS 下的工程设置的有何异同，并分析其理由。

答：（1）实验二、三中的 ADS 下的工程设置中，ARM Linker 里面的 Output RO Base 地址设为 0x30000000,这是 S3C2440 的 SDRAM 的首地址。但在实验四中 Output RO

base 地址改为 0x00000000,这是 S3C2440 的 NAND flash 的首地址。

理由：（2）实验二、三程序是在 Nor Flash 模式下运行，实验板的 SDRAM 挂接在 S3C2440 的 BANK6，即地址从 0x30000000 开始，由于程序要加载到 SDRAM 中运行，程序的起始地址同样要设置成 0x30000000。而实验四是在 NAND Flash 模式下，NAND Flash 启动时 S3C2440 配备了称为 Steppingstone 的内置 SRAM，Steppingstone 映射到地址空间(nGCS0)0x00000000，大小为 4K。上电后 NAND FLASH 的 前 4K 代码被拷贝至 Steppingstone 中，并开始运行。CPU 是从 0x00000000 开始执行，故程序的起始地址为 0x00000000。

5. 在中断实验中为什么要把可执行程序下载到 NAND Flash 中运行，而不是直接下载到 SDRAM 中运行？如果直接下载到 SDRAM 中运行会发生什么情况？

答：（1）中断向量表一般位于地址 0x00000000 处，如 IRQ 中断向量地址为 0x00000018，FIQ 中断向量地址为 0x0000001c；而 SDRAM 一般是映射到地址 0x30000000 以后。NAND Flash 启动时 S3C2440 配备了称为 Steppingstone 的内置 SRAM，Steppingstone 映射到地址空间(nGCS0)0x00000000，大小为 4K。上电后 NAND FLASH 的 前 4K 代码被拷贝至 Steppingstone 中，并开始运行。CPU 是从 0x00000000 开始执行，也就是 NAND flash 里的前 4KB 内容。故要将程序下载到 NAND Flash 中运行。

（2）此程序中使用了中断，若程序下载到 SDRAM 中，则会因中断向量地址不对而找不到中断函数，则程序无法正常执行。

6.结合实验，叙述 NAND Flash 启动的流程。

答：（1）完成复位，OM[1:0]=00b,使能 NAND Flash 控制器的自动引导模式。

（2）当使能自动引导模式后，NAND Flash 的首 4 个字节被复制到内部缓存 Steppingstone。

（3）CPU 开始执行 Steppingstone 中的引导程序。在引导程序中，将程序的剩余部分拷贝到 SDRAM 中，然后，跳到 SDRAM 中运行。

五、体会和建议

体会：这四个实验让我熟悉了 ADS 的基本使用方法，通过自己在电脑上参考示例代码，设计程序，让我对 C 语言与汇编语言的使用有了进一步的了解，让我更加熟练的掌握了 C 语言和汇编语言的混合编程，在实验过程中更加深刻的理解了课

堂上学习的相关知识。

同时通过这四个实验也让我了解到了 S3C2440 部分引脚的功能及相关寄存器的使用方法，会利用查询方式和中断方式来解决相对应的问题，进一步理解了 Nor Flash 和 Nand Flash 两种启动方式的区别，也注意到了使用子中断在中断程序中不仅要对相应的中断寄存器写一清零，也要对子中断的相关寄存器写一清零。

建议：可以增加程序设计的实验，让我们自己参考之前做的实验进行拓展，体会一下 S3C2440 的使用，加深我们对嵌入式的理解和体会。