实验 3：熟悉和掌握 Keil 软件开发、调试、仿真工具

姓名：朱勇椿　学号：201411213004

2016 年 10 月 30 日

1　实验题目

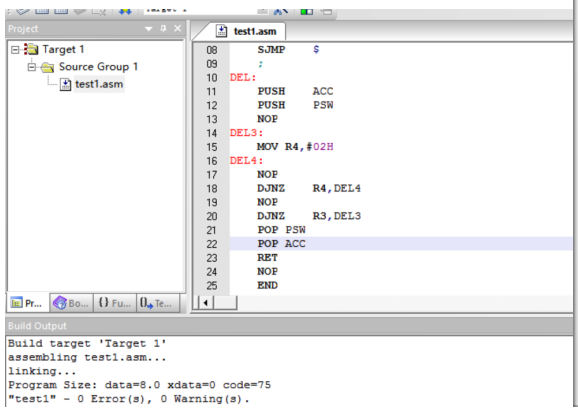
1. 熟悉 Keil 软件开发、调试、仿真工具的使用。
2. 了解 CPU 结构和各寄存器结构。
3. 熟悉汇编语言（助记符）的使用。

2　实验内容

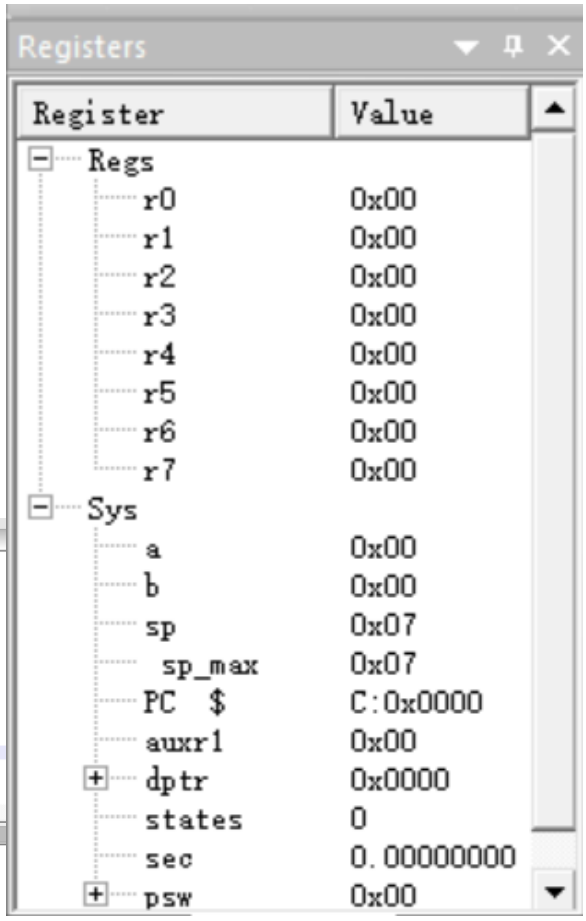
1. 安装 Keil 软件开发、调试、仿真工具。
2. 熟悉 Keil 软件查看 CPU、寄存器、RAM、ROM 等数据，修改数据。
3. 编写一小段完整的汇编语言程序（或使用下述延时程序），编译和运行该程序。
4. 在上题的基础上逐步（逐条）跟踪运行程序，每执行一步查看个数据窗口中的数据，包括 CPU 寄存器（A 累加器、PSW 程序状态字、DPTR、DPH 和 DPL 数据指针、SP 堆栈指针）以及 RAM 和 ROM 数据区等。
5. 调试和编写程序如下：
   (a) 调试下述延时程序示例，注意观察程序计数器 PC 和堆栈寄存器 SP 的变化，查看该程序汇编后的机器码，描述执行过程（方框图，标注 PC 和 SP 值），当晶振为 12MHz 时，计算延时子程序的延时时间。
   (b) 编写程序，学习随机存储器的访问，检查 RAM 中 0 到 4FH 中内容为 0 的个数，将结果存放在 50H（RAM）中，要求使用 R0（@ R0）间接寻址。
   (c) 编写程序，学习只读存储器的访问，检查 ROM 中 0 到 4FH 中内容为 0FFH 的个数，将结果存放在 50H（RAM）中，要求使用 DPTR（@A+DPTR）或 PC（@A+PC）变址寻址。

3 实验过程

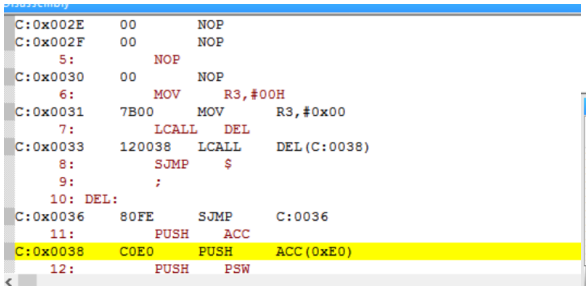
3.1 调试延时程序示例



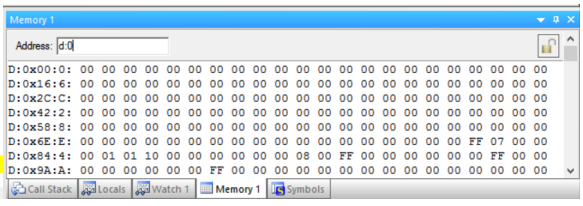
(a) 编译



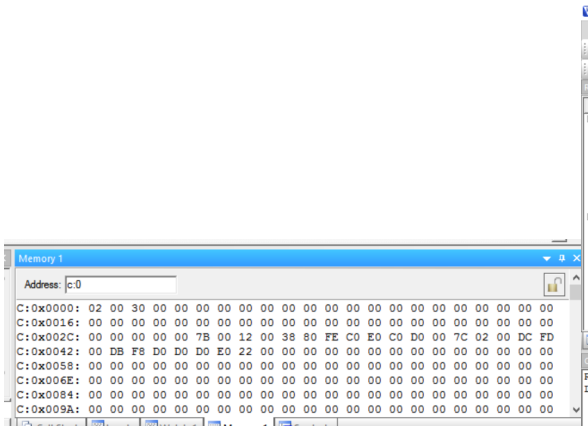
(b) 寄存器



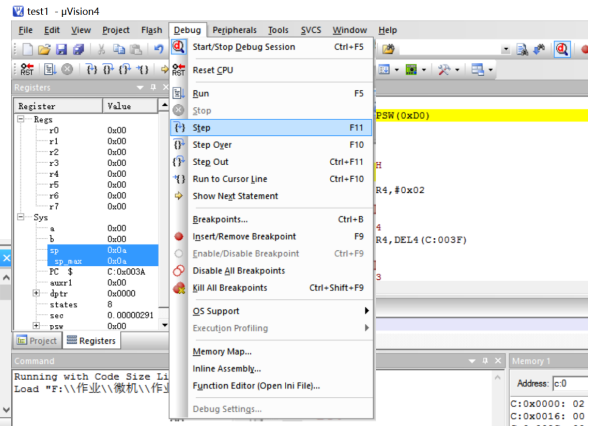
(c) 机器码



(d) RAM



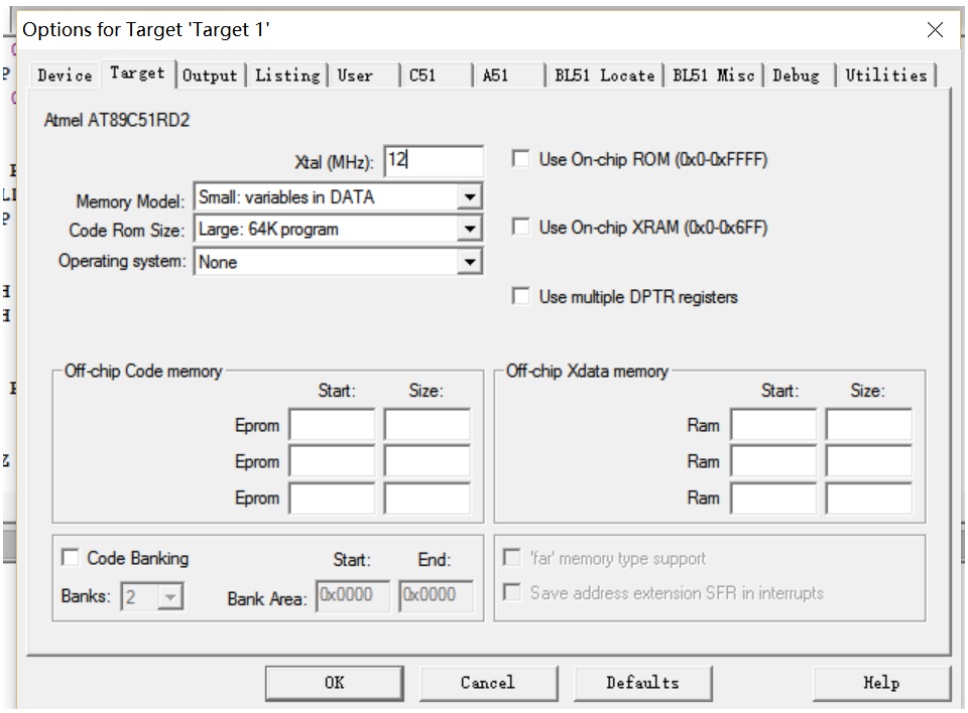
(e) ROM



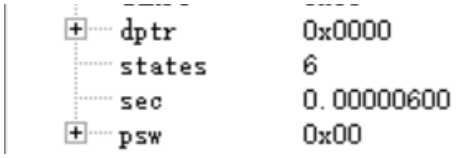
(f) 单步运行

调试延时程序

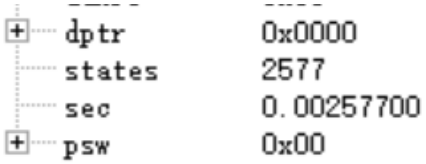
SJMP $ 语句的作用相当于一个死循环，执行这条语句执行完又回到这条指令，LJMP START 相当于执行到这里又跳转回 START 指定的位置，这两条指令共同的作用就是保证了程序一直在这个范围内运行，不会乱跳到别的地方。



设置晶振



(g) 进入延时函数前

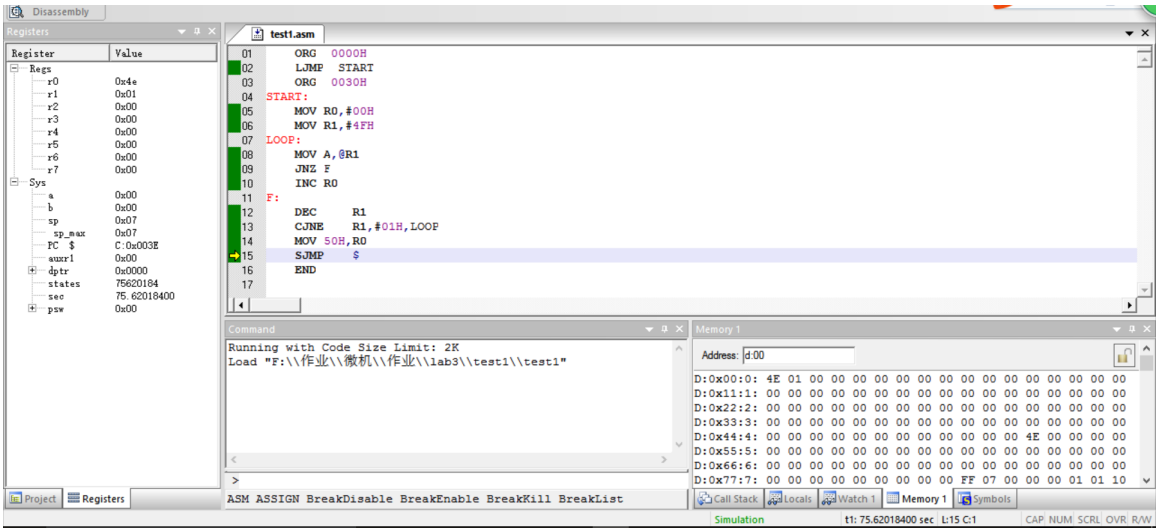


(h) 进入延时函数后

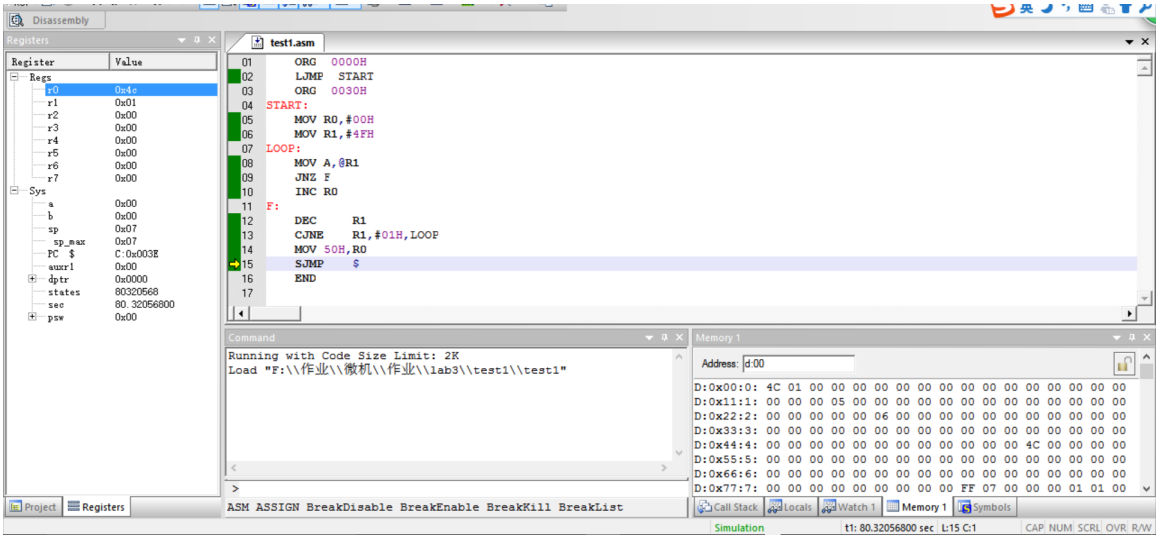
调试延时程序

延时子程序的延时时间 =0.002577-0.000006=0.002571s

3.2 检查 RAM 中 0 的个数



全部为 0 共 4E 个

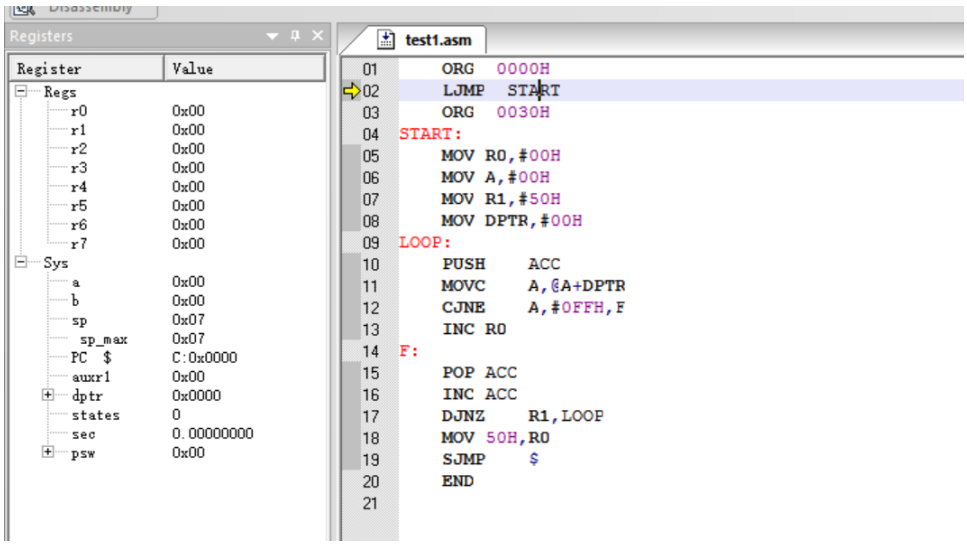


有两个不为 0 共 4C 个

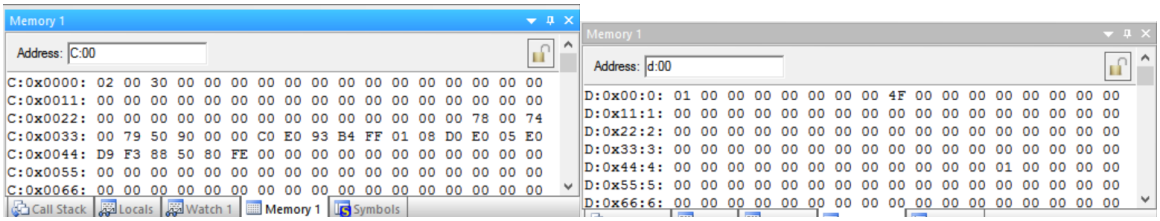
这个题目貌似有点问题，我用到 R0 和 R1 寄存器，而这两个寄存器实际上就是在 RAM 中的 01 和 02 的位置，而这个两个寄存器的值是在改变的，在计数时不应该把这两个寄存器占用的位置计算进去，我在程序中没有计算 01 和 02。

统计从 02 到 4F 总共有 4EH 个数，当中有 2 个不为 0，得到结果 4C，正确。程序中用 R0 来累加数，用 R1 来控制循环同时也表示 RAM 的

3.3 统计 ROM 中为 0FFH 的个数



代码



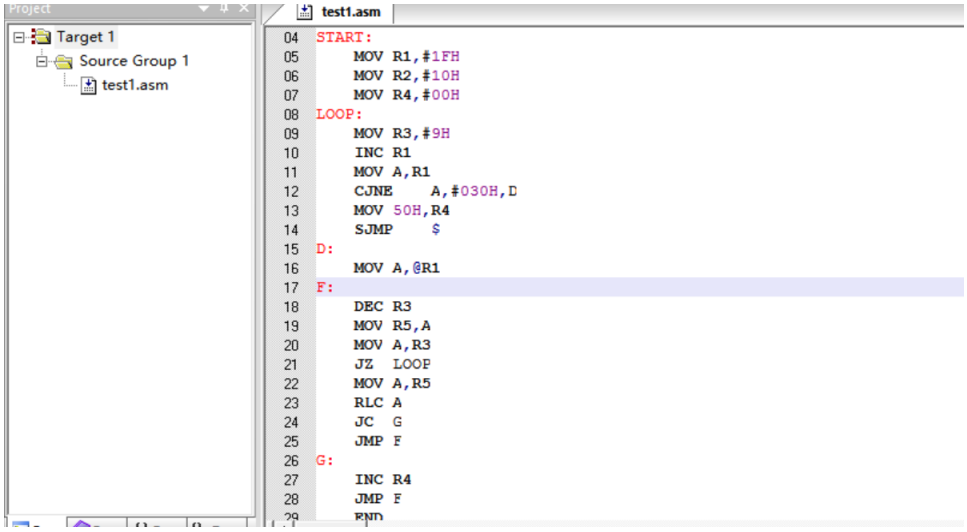
(i) ROM (j) RAM

调试延时程序

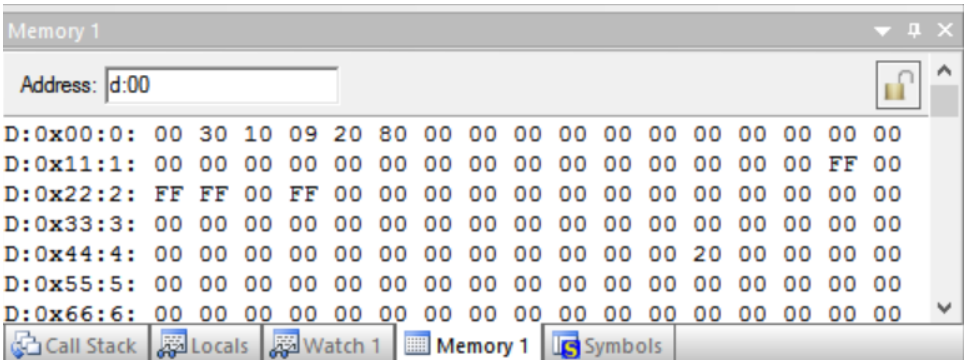
可以看到 ROM 中仅有一个 FF，最后在 RAM 中 50H 处存入了 01，故正确。

4 选做题

编写程序，学习随机位存储器的访问，检查 RAM 中 20H 到 2FH 中 BIT 位内容为 1 的个数，将结果存放在 50H(RAM) 中。



程序代码



运行结果

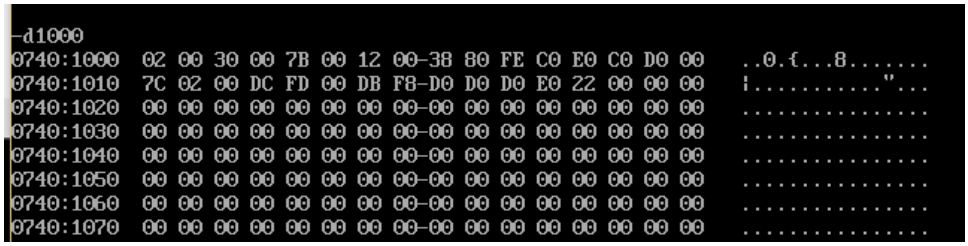
可以看到 RAM 中有 4 个 FF，那么就有 32 个 1，最后结果 20H，正确。

5 思考题

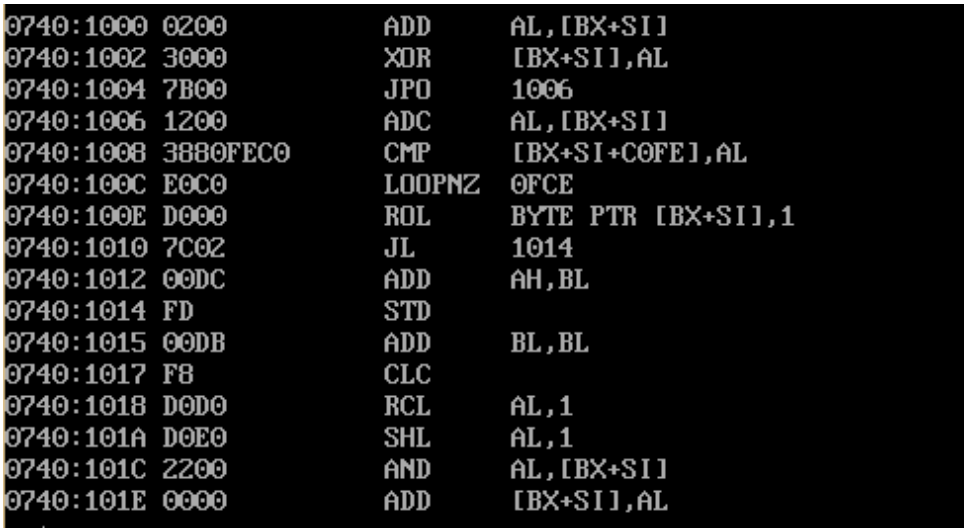
5.1 8086 汇编语言可以与 8051 汇编语言通用吗？

不能通用，汇编语言是面向 CPU 的，不同 CPU 指令集是不同的，并且 8086 是 16 位指令集，而 8051 是 8 位指令集，包含的指令都不相同，不能通用。

5.2 将延时程序示例的 8051 机器码输入到 8086（debug）RAM 数据区，将机器码在 debug 中反汇编，分析得到的汇编语言程序，与原 8051 汇编语言程序进行比较。



延时程序机器码



在 8086 中反汇编

对比延时程序源码可看出在 8086 中反汇编得到的指令是完全不同，也就是两个 CPU 的指令集不同。

6 问题反思

在前面的红字处已经提出了问题，上课时讲的 8051 的寄存器放在内存中，那么用作寄存器的内存单元不应该用来计数，在实验中我没有计算用作寄存器的 RAM 单元。

7 经验感想

本次实验的接触了 8051 的单片机，这学期也在学 8086 的汇编语言，能明显感觉到两种汇编的指令的不同之处，相比之下觉得还是 8086 的汇编语言更方便一些，可能这也和一个是 16 位的一个是 8 位的有关吧，8 位 CPU 指令不可能设置得太复杂。并且不同汇编语言之间是不通用的，表示成的机器码是不同的。