【题目 2】用汇编语言实现在程序存储器、片内基本 RAM 和片内扩展 RAM 之间的数据传输。

说明：题目 2 的设计源代码分别放在文件夹 num2 中

【设计要求】

（1）在程序存储器内初始化 20 个数据。

（2）将其同时传到片内基本 RAM 和片内扩展 RAM。

（3）将转到片内扩展 RAM 的数据，执行取反操作。

（4）将取反操作后的数据传到片内基本 RAM 区

1、设计思路

在程序存储器内初始化 20 个十进制数 1~20，同时在片内基本 RAM 和片内扩展 RAM 区提前申请好内存空间。

通过 **20 次循环**，以累加器 **A 为中间桥梁**，把数据从程序存储器传到片内基本 RAM 区和片内扩展 RAM 区，以用 R0、R1 和 DPTR 来保存对应的目标地址值，**每传值一次地址保存地址的寄存器加一，循环次数减一，直至 20 次循环结束。**

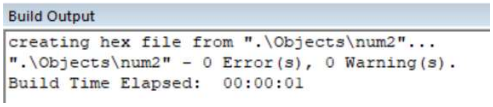
因为不能在片内扩展 RAM 区直接将数据取反，所以需把数据**先传到累加器 A 中，用指令 CPL 进行取反后再逆向传回片内扩展 RAM 区的原位置处**。同样使用 **20 次循环**来完成上取反过程。

片内基本 RAM、片内扩展 RAM 之间不能直接传值，需要**用累加器 A 作为中间桥梁，先将数据从片内扩展 RAM 区传至 A，再从 A 传至片内基本 RAM 区**。同样使用 **20 次循环**来完成。

2、调试过程及遇到问题的解决办法

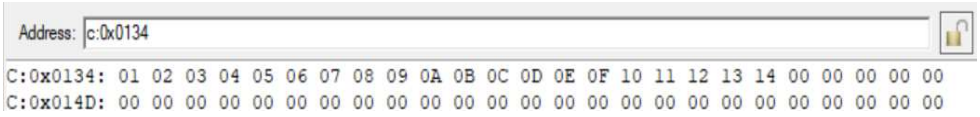
在着手这道题之前，我认真总结了书上关于在不同存储空间之间传值的指令，而且在写代码时也很注意细节，所以这道题整个过程都比较顺利，未出现大的难解决的问题。

3、实验结果：成功实现题目要求。

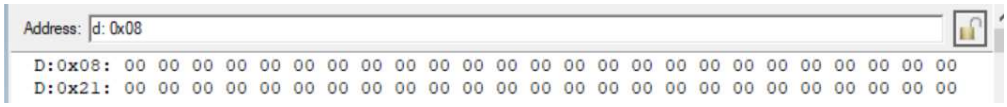


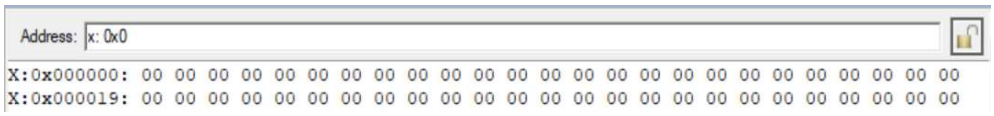
(1) 代码编译后进入 debug 模式，一步一步运行，观察数据在存储空间的变化

在程序存储器内初始化 20 个数据如下：

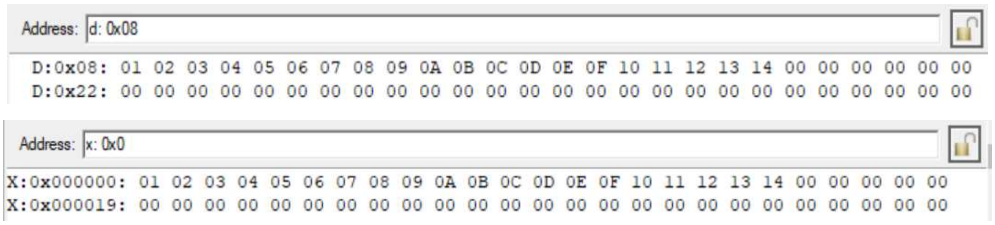


将光标定位到在基本 RAM 区的 TABLE1，显示其地址为 D:0x08，用同样的方法可看到在片内扩展 RAM 区的 TABLE2 所在地址为 X:0x000000。它们的初始值为：

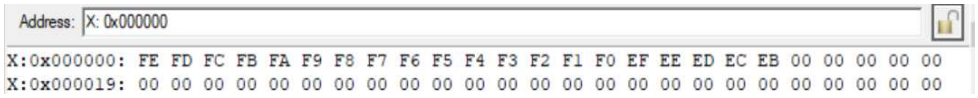




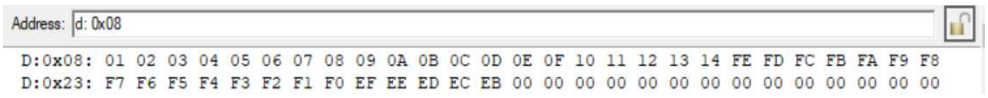
(2) 点击 单步运行程序，直至传值操作完成，分别定位到片内基本 RAM 区 TABLE1、片内扩展 RAM 区 TABLE2 所在地址空间处，可观察到位于程序存储器内的 20 个数据传值成功，如下：



(3) 点击 单步运行程序，直至取反操作完成，观察到 TABLE2 处的数值出现了如下变化，经验证，确实是之前的数据取反后的值：



(4) 点击 单步运行程序，直至在片内扩展 RAM 区取反操作后的数据完全传到片内基本 RAM 区，可观察到如下变化，即数据从离 TABLE1 起始地址 20 个地址单元处开始存入值：



4、源代码及注释

```
NAME    main
my_num SEGMENT CODE
        RSEG my_num
TABLE:  DB   1,2,3,4,5,6,7,8,9,0AH,0BH,0CH,0DH,0EH,0FH,10H,11H,12H,13H,14H
                                ;在程序存储器内初始化 20 个数据

my_num1 SEGMENT DATA
        RSEG my_num1
TABLE1: DS 40                   ;在基本 RAM 区申请 40 字节的存储空间

my_num2 SEGMENT XDATA
        RSEG my_num2
TABLE2: DS 20                   ;在片内扩展 RAM 区申请 20 字节的存储空间

myprog SEGMENT CODE
        RSEG    myprog          ;切换到代码段 myprog
        LJMP    main
```

```
        ORG     0X100           ;定位到偏移 100H 的位置
main:
        USING    0              ;使用第 0 组寄存器 R0~R7
        MOV     A,#0            ;初始化 A
        MOV     R2,#20          ;设置值传递次数为 20
        MOV     R0,#TABLE1      ;将 TABLE1 的地址赋给 R0
        MOV     R1,#TABLE2      ;将 TABLE2 的地址赋给 R1
        MOV     DPTR,#TABLE     ;将 TABLE 的地址赋给 DPTR
INIT:   MOVC    A,@A+DPTR       ;将数据从程序存储器传至累加器 A
        MOV     @R0,A           ;将数据从累加器 A 传至片内基本 RAM 区
        MOVX    @R1,A           ;将数据从累加器 A 传至片内扩展 RAM 区
        MOV     A,#0            ;偏移量设为 0
        INC     DPTR            ;地址加 1，指向下一个将要传值的数
        INC     R0              ;地址加 1，指向下一个存储区
        INC     R1              ;地址加 1，指向下一个存储区
        DJNZ    R2,INIT         ;当 R2 减到 0 时退出传值的循环

;//=======================================
;//将转到片内扩展 RAM 的数据，执行取反操作
;//=======================================
        MOV     R0,#20          ;执行 20 次取反操作，利用 20 次循环来完成
        MOV     DPTR,#TABLE2    ;将 TABLE2 的地址赋给 DPTR
NDIG:   MOVX    A,@DPTR         ;将数据从 DPTR 所指向的片内扩展 RAM 的数据传至累
加器 A
        CPL     A               ;对刚传进的数进行按位取反
        MOVX    @DPTR,A         ;再将刚取反的数据从累加器传至
        INC     DPTR            ;地址加 1 指向下一个数据
        DEC     R0              ;取反一次，次数就减少一次
        CJNE    R0,#0,NDIG      ;当 R0 减到 0，说明所有数据取反完成，退出循环

;//=======================================
;将取反操作后的数据传到片内基本 RAM 区
;//=======================================
        MOV     R0,#20          ;执行 20 次传值操作，设置循环次数为 20
        MOV     DPTR,#TABLE2    ;将 TABLE2 的地址赋给 DPTR
        MOV     A,#20
        ADD     A,#TABLE1       ;TABLE1 的地址值与累加器 A(为 20)相加，结果保存在
A 中。A 的值即为 TTABLE1 地址后移 20 个单元后的地址值，这是准备接收数据的起始地址
        MOV     R1,A            ;将结果（接收数据的起始地址值）赋给 R1
TRANS:  MOVX    A,@DPTR         ;将由 DPTR 所指向的片内扩展 RAM 区里被取反后数据
传到 A 中
        MOV     @R1,A           ;将上一步传到 A 中的值再通过 A 传到 R1 所指向的片
内基本 RAM 区，在这里 A 起到一个桥梁作用
        INC     DPTR            ;DPTR 加 1,是下一个待传数据单元的地址
```

```
        INC     R1              ;R1 加 1,指向下一个接收数据的内存单元
        DEC     R0              ;传值 1 次，R0 减 1
        CJNE    R0,#0,TRANS     ;当 R0 减到 0 时，传值结束，退出循环

END
```