

计算机工程学院

**汇编语言与接口技术**

**实验报告**

学年学期：  **2022 - 2023 学年 第1学期**

实验名称：  **实验四 冒泡排序算法的实现**

班级： 计算2114

学号： 202121331104

姓名： 庄佳强

日期： 11.07

成绩：

**一、 实验要求**

(1)学习使用汇编语言实现算法设计

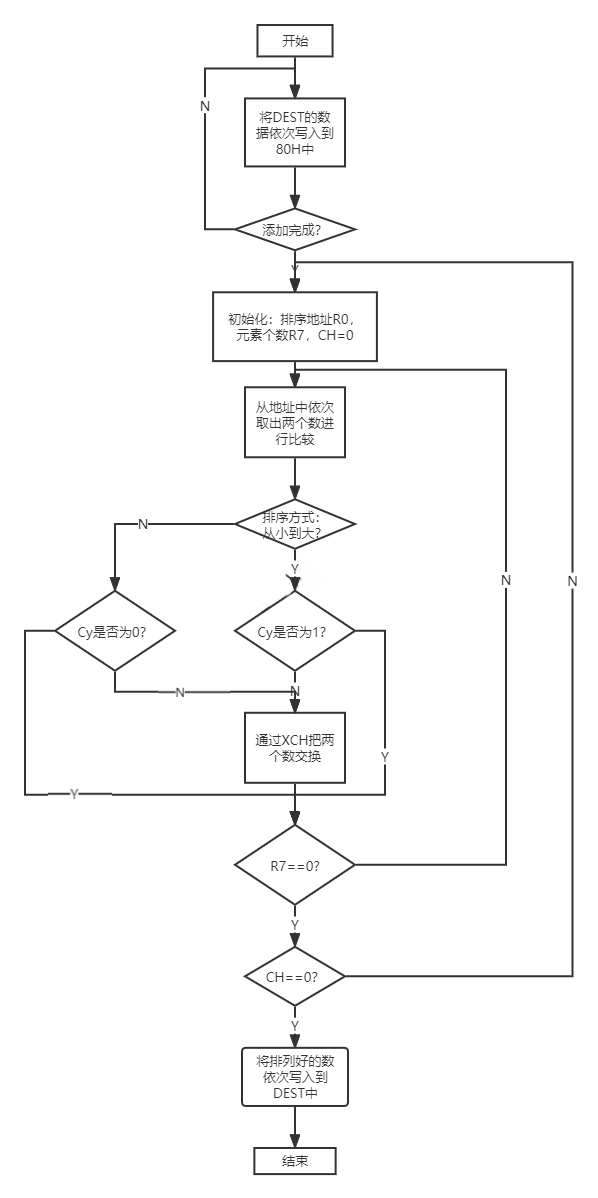
(2)熟练掌握单片机外部存储空间的访问方法

**二、 实验设计**

**1.整体思路**

先把DEST中的数据用过循环读入到片内存储中。在把数据依次读入到A,B中，通过CJNZ进行判断,跳转到交换部分，交换部分时，通过JB 7AH来判断是什么排序来转移到不同的子程序。而使用CH来识别本次循环是否有交换元素。当CH=0时，表示全部排序完成，这时在通过MOVX存回到DEST中。

**2.流程图**



**3.主要模块设计思路及分析**

1. 写出读入/子程序

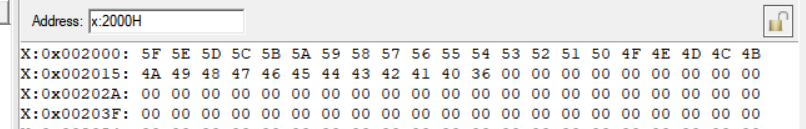
先初始化写出或读入的地址和元素的个数，这里是80H，在写入时就先把DEST的地址存在DPTR中，通过MOVX读入或写出元素，DPTR++,R0++,元素个数--到0跳出循环。

1. 冒泡排序子程序

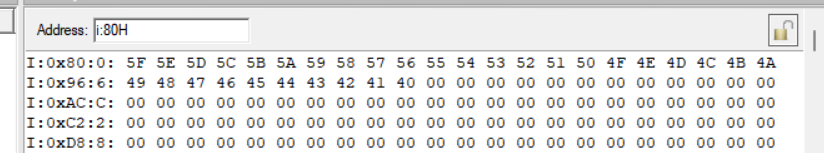
先初始化排序的地址R0和元素个数R7，在通过MOV 取出两个数到A,B中，这是R0只加1，通过CJNZ比较A B,A>B时cy=0,A<B时，cy=1。再判断7AH为0或。当7AH为0时跳转到从小到大排序，当7AH为1时跳转到从大到小排序。排序前还需要判断cy状态，当从小到大排序时，先判断cy是否为1，为1不用交换跳转到NEXT，不然就交换两个数，CH为1。当从大到小排序时，先判断cy是否为0，为0不用交换跳转到NEXT，不然就交换两个数，CH为1。NEXT中判断R7是否为0，不为0跳转到之前取数继续排序。接着判断CH是否为0,CH为1表示还没有排序完成，就要跳转到最开始 进行初始化重新排序，知道完全排序完成。全部排序后，就把80H中的数重新依次存入到DEST中。

1. **实现效果**

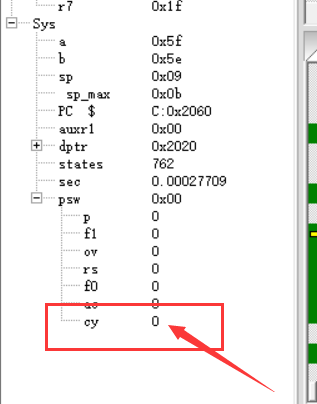
一开始DEST中的数据。

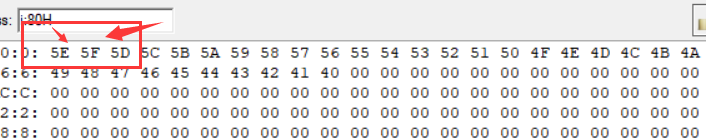


读入到80H中的数。

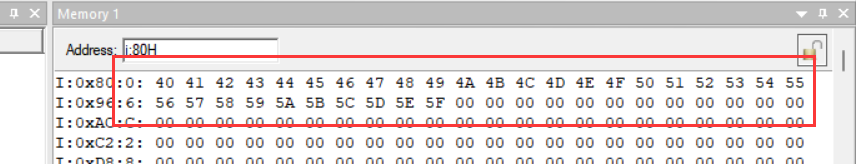


把5F和5E进行比较，5F>5E cy=0,进行排序

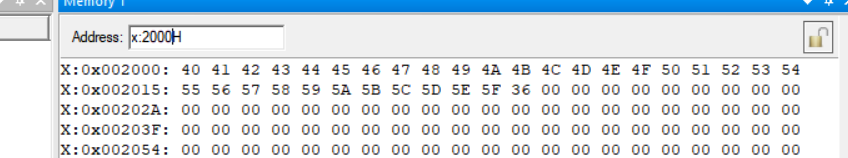




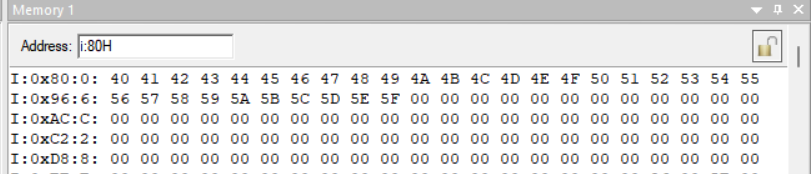
全部完成后的结果



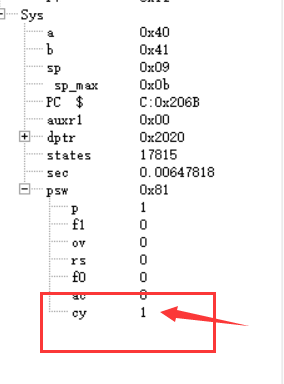
DEST中的数据



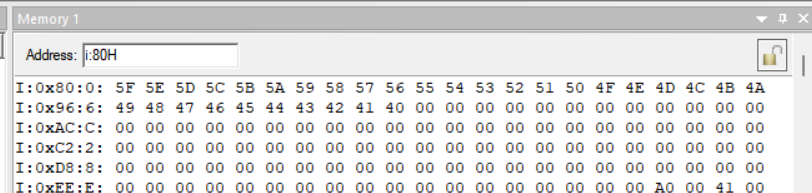
重新初始化80H



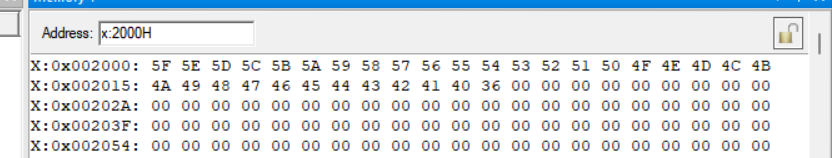
40H和41H比较 40H>41H cy=1



全部完成后的结果



DEST中的数据



1. **总结**

1.冒泡排序是C语言中必备的一个算法，但移植到8051中就变得困难，要考虑诸多因素。

2.一开始是准备直接在片外进行冒泡排序，但数据指针DPTR只能累加，不能减1，导致没有办法实现两个数的交换。最后放弃片外排序方式，转成先复制到片内冒泡排序后写出到DEST中。

**附录：**

ORG 0000H

LJMP MAIN

ORG 2000H

MAIN:

SRC DATA 30H //片内地址 //SRC表示源。

DEST XDATA 2000H //片外地址

NUM DATA 20H

CH equ 0

ACAll LOOP1 //添加子程序的转移

ACALl LOOP2 //克隆子程序的转移

CLR 7AH // 从小到大排序

ACALL ORDERS // 冒泡排序子程序

SETB 7AH // 从大到小排序

ACALL ORDERS // 冒泡排序子程序

SJMP $

LOOP1:

PUSH PSW //现场保护

PUSH Acc

MOV R2, #20H //添加的个数

MOV R0, #30H //添加到的地址

MOV R4, #40H //添加的内容

ADDS:

MOV A,R4 //把内容送到A

MOV @R0,A //A传送到间接寻址@R0

INC R4 //内容加一

INC R0 //地址加一

DJNZ R2,ADDS //判断条件

MOV @R0,#36H //$的Ascii码为36

POP Acc //现场回复

POP PSW

RET

LOOP2:

PUSH PSW

PUSH Acc

MOV R2,#20H

MOV DPTR,#2000H //因为要存储到片外，直接使用DPTR来记录地址

MOV R0,#4FH //因为只要倒序排序，所以从30H+20H-1开始

CLONE:

MOV A,@R0 //间接寻址的内容传送到A

MOVX @DPTR,A //片外使用MOVX

DEC R0 //R0减一

INC DPTR //DPTR加一

DJNZ R2,CLONE

MOV A,#36H //$的Ascii码为36

MOVX @DPTR,A

POP Acc

POP PSW

RET

ORDERS:

ADDNUM:

MOV DPTR,#2000H

MOV R0,#80H

MOV R7, #NUM

LOOP3:

MOVX A,@DPTR

MOV @R0,A

INC R0

INC DPTR

DJNZ R7,LOOP3

initialize:

MOV R0, #80H

MOV R7, #NUM-1

CLR CH

choose:

MOV A, @R0

MOV R2, A

INC R0

MOV B, @R0

CJNE A, B, arrange //A存的第一位 B存的第二位

SJMP Next

arrange:

JB 7AH, arrange1 //设置为大到小时跳转

JC Next //前小后大, 不交换

SETB CH // 前大后小, 置交换标志

XCH A, @R0

DEC R0

XCH A, @R0

INC R0 //R0到下一位（回到第二位）

JNB 7AH,NEXT

arrange1:

JNC Next //前大后小, 不交换

SETB CH // 前小后大, 置交换标志

XCH A, @R0

DEC R0

XCH A, @R0

INC R0 //R0到下一位（回到第二位）

Next:

DJNZ R7, choose

JB CH,initialize

ADDNUMS:

MOV DPTR,#DEST

MOV R0,#80H

MOV R7, #NUM

LOOP4:

MOV A,@R0

MOVX @DPTR,A

INC R0

INC DPTR

DJNZ R7,LOOP4

RET

END