# 冒泡排序算法

## 算法

冒泡排序是比较基础的排序算法之一，其思想是相邻的元素两两比较，较大的数下沉，较小的数冒起来，这样一趟比较下来，最大(小)值就会排列在一端。整个过程如同气泡冒起，因此被称作冒泡排序。

冒泡排序的步骤是比较固定的：

1>比较相邻的元素。如果第一个比第二个大，就交换他们两个。

2>每趟从第一对相邻元素开始，对每一对相邻元素作同样的工作，直到最后一对。

3>针对所有的元素重复以上的步骤，除了已排序过的元素(每趟排序后的最后一个元素)，直到没有任何一对数字需要比较。

## C语言设计

|  |
| --- |
| #include"stdio.h"  int arr[8] = {3,8,1,5,2,4,6,7};  void Bubble\_Sort(int \*arr, int begin, int end)  {  // 参数校验 begin<=end  if(begin > end)  return;    int change;    for(;begin<end;end--) // 不带 = ，end遍历至begin+1时结束  {  // 初始化 选定基准数，初始化哨兵  int i = begin+1;    for(;i<=end;i++) // 带 = ，i遍历至end时结束  {  if(arr[i]<arr[i-1]) // 如果arr[i]<arr[i-1]，交换arr[i]与arr[i-1]  {  change = arr[i];  arr[i] = arr[i-1];  arr[i-1] = change;  }  }  }  }  int main()  {  int i;  // 显示排序前的序列  for(i=0;i<=7;i++)  printf("%d ",arr[i]);  printf("\n");  // 排序  Bubble\_Sort(arr,0,7);  // 显示排序后的序列  for(i=0;i<=7;i++)  printf("%d ",arr[i]);  printf("\n");    } |

## 汇编语言设计

用LoongIDE新建一个工程。

移除core文件夹下的bsp\_start.c源文件。

移除main.c源文件。

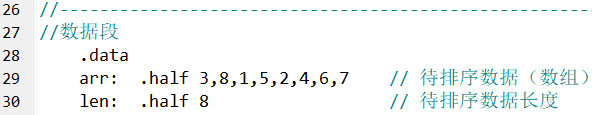
添加bsp\_start.S源文件。

在bsp\_start.S下添加如下代码：

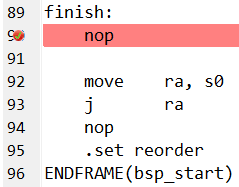
|  |
| --- |
| /\*  \* bsp\_start.S  \*  \* created: 2022/1/26  \* author: Li TianLing  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \* v0 存储器寻址辅助寄存器  \* s0 待排序数据arr基地址  \* s1 参数传递-->begin  \* s2 参数传递-->end  \* s3 跳转辅助寄存器  \* t1 哨兵i  \* t2 哨兵j（即i-1）  \* t3 哨兵i指向的值，即arr[i]  \* t4 哨兵j指向的值，即arr[j]  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include "regdef.h"  #include "cpu.h"  #include "asm.h"  //-----------------------------------------------------------------------------  //数据段  .data  arr: .half 3,8,1,5,2,4,6,7 // 待排序数据（数组）  len: .half 8 // 待排序数据长度  //-----------------------------------------------------------------------------  //代码段  .text  FRAME(bsp\_start,sp,0,ra)  .set noreorder  move s4, ra /\* 返回地址 \*/  la s0,arr // 加载待排序数据基地址（常量）  la s1,0 // begin初始化为0  la s2,len // 加载、计算end(一个半字两个字节) (8-1)\*2=14  lh s2,(s2)  sub s2,1  sll s2,1    for\_1\_begin: // end遍历至begin+1时结束，不会遍历begin  sub s3,s2,s1 // 参数校验，end<=begin,结束循环1  blez s3,for\_1\_end  nop  move t1,s1 // 哨兵i和j初始化  add t1,2  move t2,s1  for\_2\_begin: // i在begin和end之间循环遍历  sub s3,s2,t1 // 参数校验，end<i,结束循环2  bltz s3,for\_2\_end  nop  add v0,s0,t1 // 将arr[i]读入t3  lh t3,(v0)  add v0,s0,t2 // 将arr[j]读入t4  lh t4,(v0)  sub s3,t3,t4 // 如果arr[i]>arr[j],不交换  bgez s3,no\_if  nop  add v0,s0,t2 // 否则将arr[i]和arr[j]交换  sh t3,(v0)  add v0,s0,t1  sh t4,(v0)  no\_if:  add t1,2 // i+2，j+2，开始循环2  add t2,2  b for\_2\_begin  nop  for\_2\_end:  sub s2,2 // end-2,开始循环1  b for\_1\_begin  nop  for\_1\_end:  finish:  nop  move ra, s4  j ra  nop  .set reorder  ENDFRAME(bsp\_start) |

## 调试运行

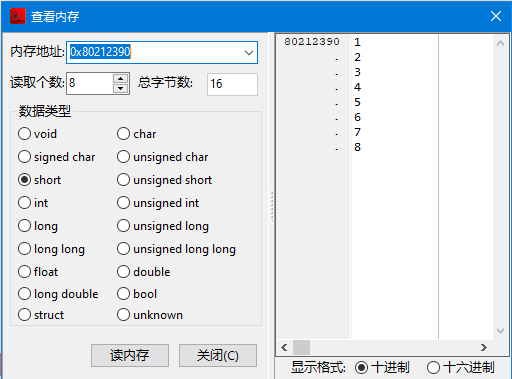
初始数据如下所示：



如下图所示，在finish后打上断点：



联合调试后，双击s0寄存器，查看对应存储器数据：



如上图所示，排序成功。

## 改进设计

加入插入排序思想的有序数组检测，先检测有序数组，再对无序数组进行冒泡排序。

## C语言设计——改进

|  |
| --- |
| #include"stdio.h"  int arr[8] = {3,8,1,5,2,4,6,7};  /\* 检测有序数组+冒泡，相当于冒泡的改进，不是插入排序 \*/  void Bubble\_Sort\_Pro(int \*arr, int begin, int end)  {  // 参数校验 begin<=end  if(begin > end)  return;    int change;  int i,j;  for(i=begin;arr[i+1]>=arr[i];i++) ; // 寻找有序数组    for(j=i+1;j<=end;j++) // 待插入数据遍历到end  {  for(i=j-1;i>=begin;i--) // 某一待插入数据  {  if(arr[i]>arr[i+1]) // 满足条件就冒泡  {  change = arr[i];  arr[i] = arr[i+1];  arr[i+1] = change;  }  }  }  }  int main()  {  int i;  // 显示排序前的序列  for(i=0;i<=7;i++)  printf("%d ",arr[i]);  printf("\n");  // 排序  Bubble\_Sort\_Pro(arr,0,7);  // 显示排序后的序列  for(i=0;i<=7;i++)  printf("%d ",arr[i]);  printf("\n");  } |

## 汇编语言设计——改进

用LoongIDE新建一个工程。

移除core文件夹下的bsp\_start.c源文件。

移除main.c源文件。

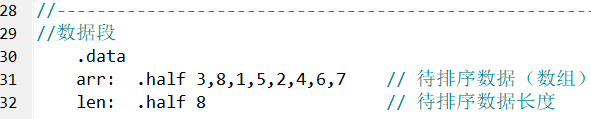
添加bsp\_start.S源文件。

在bsp\_start.S下添加如下代码：

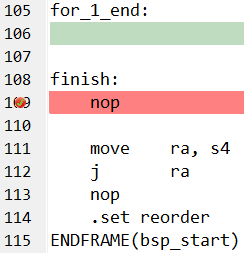
|  |
| --- |
| /\*  \* bsp\_start.S  \* 本程序是冒泡排序改进版，加入了有序数组检测，介于冒泡和插入之间  \* created: 2022/1/27  \* author: Li TianLing  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \* v0 存储器寻址辅助寄存器  \* s0 待排序数据arr基地址  \* s1 参数传递-->begin  \* s2 参数传递-->end  \* s3 跳转辅助寄存器  \* t1 哨兵i  \* t2 哨兵i+1  \* t3 哨兵i指向的值，即arr[i]  \* t4 哨兵i+1指向的值，即arr[i+1]  \* t5 哨兵j  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include "regdef.h"  #include "cpu.h"  #include "asm.h"  //-----------------------------------------------------------------------------  //数据段  .data  arr: .half 3,8,1,5,2,4,6,7 // 待排序数据（数组）  len: .half 8 // 待排序数据长度  //-----------------------------------------------------------------------------  //代码段  .text  FRAME(bsp\_start,sp,0,ra)  .set noreorder  move s4, ra /\* 返回地址 \*/  la s0,arr // 加载待排序数据基地址（常量）  la s1,0 // begin初始化为0  la s2,len // 加载、计算end(一个半字两个字节) (8-1)\*2=14  lh s2,(s2)  sub s2,1  sll s2,1  move t1,s1 // 哨兵i  add t2,t1,2 // 哨兵i+1（半字翻倍）  for\_0\_begin: // 寻找有序数组  add v0,s0,t1 // 加载哨兵i指向的值arr[i]到t3  lh t3,(v0)  add v0,s0,t2 // 加载哨兵i+1指向的值arr[i+1]到t4  lh t4,(v0)  sub s3,t3,t4 // arr[i]>arr[i+1]结束，即t3>t4结束寻找  bgtz s3,for\_0\_end  add t1,2 // 哨兵移动，开始循环  add t2,2  b for\_0\_begin  nop  for\_0\_end:  move t5,t2  for\_1\_begin: // begin遍历至end-1时结束，不会遍历最后一个数  sub s3,t5,s2 // 参数校验，t5>s2结束循环1,即待插入数据遍历到end结束  bgtz s3,for\_1\_end  nop  move t2,t5  sub t1,t2,2  for\_2\_begin: // 某一待冒泡数据  sub s3,t1,s1 // t1<s1结束，即i<begin结束循环2  bltz s3,for\_2\_end  nop  add v0,s0,t1 // 将arr[i]读入t3  lh t3,(v0)  add v0,s0,t2 // 将arr[i+1]读入t4  lh t4,(v0)  sub s3,t3,t4 // 如果arr[i]<=arr[i+1],不冒泡  blez s3,no\_if  nop  add v0,s0,t1 //否则交叉存储，用以冒泡，将t4存入arr[i]  sh t4,(v0)  add v0,s0,t2 //否则交叉存储，用以冒泡，将t3存入arr[i+1]  sh t3,(v0)  no\_if:  sub t1,2 // i-2，开始循环2  sub t2,2 // (i-1)-2  b for\_2\_begin  nop  for\_2\_end:  add t5,2 // j+2,开始循环1  b for\_1\_begin  nop  for\_1\_end:  finish:  nop  move ra, s4  j ra  nop  .set reorder  ENDFRAME(bsp\_start) |

## 调试运行

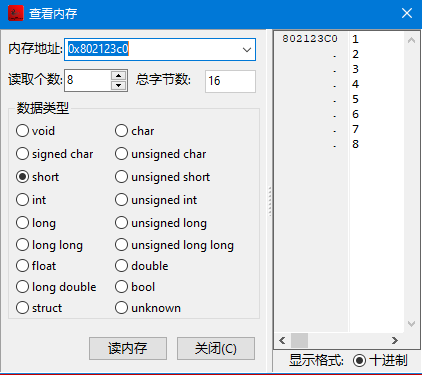
初始数据如下所示：



如下图所示，在finish后打上断点：



联合调试后，双击s0寄存器，查看对应存储器数据：



如上图所示，排序成功。