# 插入排序算法

## 算法

插入排序也是一种常见的排序算法，插入排序的思想是：将初始数据分为有序部分和无序部分，每一步将一个无序部分的数据插入到前面已经排好序的有序部分中，直到插完所有元素为止。

插入排序的步骤如下：每次从无序部分中取出一个元素，与有序部分中的元素从后向前依次进行比较，并找到合适的位置，将该元素插到有序组当中。

假如有[5,2,3,9,4,7]六个元素，下面就以排序过程中的一个步骤（此时有序部分为[2,3,5,9]，无序部分为[4,7]，接下来要把无序部分的“4”元素插入到有序部分），来展示一下插入排序的运行过程。

首先，原始的数组元素是这样的。

2 3 5 9 4 7

其中，浅绿色代表有序部分，黄色代表无序部分。

在无序部分中挑出要插入到有序部分中的元素。

2 3 5 9 4 7

将要插入的元素与左边最近的有序部分的元素进行比较。由于4 < 9，所以9向后移，4向前移。

2 3 5 4 9 7

继续将要插入的元素与左边最近的有序部分的元素进行比较。由于4 < 5，所以5向后移，4继续向前移。

2 3 4 5 9 7

继续将4与3比较。由于4 > 3，所以不再向前比较，插入到当前位置。

2 3 4 5 9 7

此时有序部分，由[2,3,5,9]变成[2,3,4,5,9]。

## C语言设计

|  |
| --- |
| #include"stdio.h"  int arr[8] = {3,8,1,5,2,4,6,7};  void Insertion\_Sort(int \*arr, int begin, int end)  {  // 参数校验 begin<=end  if(begin > end)  return;    int temp;  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  {  temp = arr[j]; // 某一待插入数据    for(i=j-1;i>=begin && arr[i]>temp;i--) // 满足条件就后移  {  arr[i+1] = arr[i];  }  arr[i+1] = temp; // 插入  }  }  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  Insertion\_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。

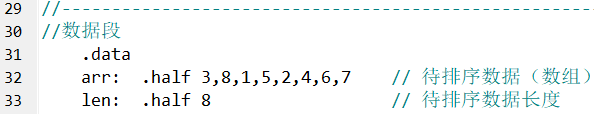
移除main.c源文件。

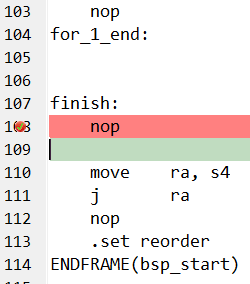
添加bsp\_start.S源文件。

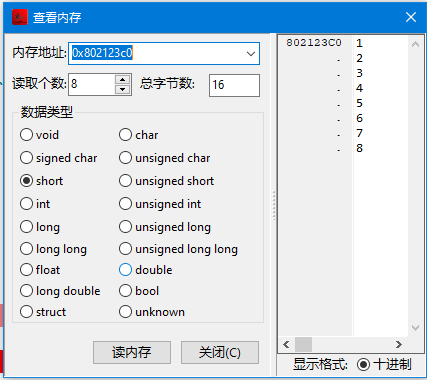
在bsp\_start.S下添加如下代码：

|  |
| --- |
| /\*  \* bsp\_start.S  \*  \* created: 2022/1/27  \* 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  \* t6 哨兵j指向的值，即arr[j]  \* t7 临时变量temp  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #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  add v0,s0,t5 // 将arr[j]直接导入temp(t7)，绕开t6  lh t7,(v0)  move t2,t5  sub t1,t2,2  for\_2\_begin:  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)  sub s3,t3,t7 // arr[i]<=temp结束，即t3<=t7结束循环2  blez s3,for\_2\_end  nop  add v0,s0,t2 // 否则将arr[i]写入arr[i+1]  sh t3,(v0)  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 v0,s0,t2 // 最后将temp插入arr[i+1]  sh t7,(v0)  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。