# 选择排序算法

## 算法

选择排序（Selection sort）是一种简单直观的排序算法。

它的工作原理：首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。以此类推，直到所有元素均排序完毕。

选择排序的主要优点与数据移动有关。如果某个元素位于正确的最终位置上，则它不会被移动。选择排序每次交换一对元素，它们当中至少有一个将被移到其最终位置上，因此对 n个元素的表进行排序总共进行至多 n-1次交换。

## C语言设计

|  |
| --- |
| #include"stdio.h"  int arr[8] = {3,8,1,5,2,4,6,7};  void Select\_Sort(int \*arr, int begin, int end)  {  // 参数校验 begin<=end  if(begin > end)  return;    int change;    for(;begin<end;begin++) // 不带 = ，begin遍历至倒数第二个数时结束  {  // 初始化 选定基准数，初始化哨兵  int i = begin;  int j = begin;    for(;i<=end;i++) // 带 = ，i遍历至最后一个数时结束  {  if(arr[i]<arr[j]) // 如果arr[i]<arr[j]，将arr[i]标记为最小值  {  j = i;  }  }  change = arr[j]; // 一轮遍历结束，一个最小值归位  arr[j] = arr[begin];  arr[begin] = 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  Select\_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。

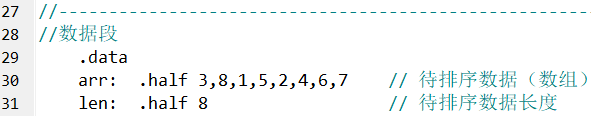
添加bsp\_start.S源文件。

在bsp\_start.S下添加如下代码：

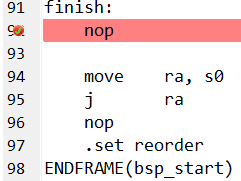
|  |
| --- |
| /\*  \* bsp\_start.S  \*  \* created: 2022/1/26  \* author: Li TianLing  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \* v0 存储器寻址辅助寄存器  \* s0 待排序数据arr基地址  \* s1 参数传递-->begin  \* s2 参数传递-->end  \* s3 跳转辅助寄存器  \* t1 哨兵i  \* t2 标志j  \* t3 哨兵i指向的值，即arr[i]  \* t4 标志j指向的值，即arr[j]  \* t5 begin指向的值，即arr[begin]  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #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: // begin遍历至end-1时结束，不会遍历最后一个数  sub s3,s2,s1 // 参数校验，end<=begin,结束循环1  blez s3,for\_1\_end  nop    move t1,s1  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],最小值不变  bgez s3,no\_if  nop    move t2,t1 // 否则将arr[i]标记为最小值    no\_if:    add t1,2 // i+2，开始循环2  b for\_2\_begin  for\_2\_end:    add v0,s0,s1 // 将arr[begin]加载到t5中  lh t5,(v0)  add v0,s0,t2 // 将arr[j]加载到t4中  lh t4,(v0)  add v0,s0,t2 // 将t5的值保存到arr[j]中  sh t5,(v0)  add v0,s0,s1 // 将t4的值保存到arr[begin]中  sh t4,(v0)    add s1,2 // begin+2,开始循环1  b for\_1\_begin  for\_1\_end:    finish:  nop  move ra, s4  j ra  nop  .set reorder  ENDFRAME(bsp\_start) |

## 调试运行

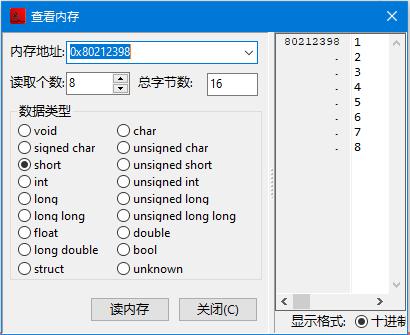
初始数据如下所示：



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。