MIPS汇编冒泡排序

PB17000002

古宜民

实现

使用MARS运行程序。

算法与C语言的冒泡排序类似，每次比较相邻两个元素，每一轮比较后比较范围内的最大元素被找出并移动到比较范围的末尾。

汇编中，数组内容放在data段，代码放在text段。其中由于数组每个元素占4个字节，除了每次加1的循环变量i和j外，还需要一个每次加4的寻址变量（当然也可以每次做乘4计算地址）。

if和for用条件跳转实现。

最后用syscall结束程序。

运行时间

使用MARS的Instruction Counter统计运行的指令条数，对于数组长度为8,16,32、完全正序和完全逆序的情况进行统计。

结果：

	8，正序	8，逆序	16，正序	16，逆序	32，正序	32，逆序
指令总条数	502	670	2030	2750	10318	11134
R-type	183	239	751	991	3759	4031
I-type	290	402	1158	1638	6062	6606
J-type	29	29	121	121	497	497

可见数组长度每增加2倍，指令条数（运行时间）增加4倍，冒泡排序的效率为O(N^2).

而数组初始状态是完全不需要处理还是完全不按顺序只会对时间产生不大的影响。

源码

```
# MIPS assembly bubble sort
# COD homework, by ustcpetergu

    .text
    .globl main
main:
    # n = 8,16,32
    #li $a0, 8
    #li $a0, 16
    li $a0, 32
```

```
    # i = 0
    li $t0, 0
loop1:
    # j = 0
    li $t1, 0
    # 4*j addr
    li $t9, 0
loop2:
    # if a[j] > a[j+1] then swap
    lw $t3, array($t9)
    lw $t4, array+4($t9)
    sub $t7, $t3, $t4
    bgtz $t7, swap
    j noswap
swap:
    sw $t3, array+4($t9)
    sw $t4, array($t9)
    j noswap
noswap:
    # j++
    add $t1, $t1, 1
    add $t9, $t9, 4
    #j < n-1-i?
    sub $t7, $a0, $t0
    sub $t7, $t7, 1
    sub $t7, $t1, $t7
    bltz $t7, loop2
    # i++
    add $t0, $t0, 1
    # i < n-1?
    sub $t7, $a0, 1
    sub $t7, $t0, $t7
    bltz $t7, loop1
    j theend
theend:
    li $v0, 10
    syscall

    .data
array:
    #.word 0x20, 0xff000011, 0x44, 0x0fffff33, 0x99, 0xffffffff, 0x01, 0x77
    #.word 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77
    #.word 0x77, 0x66, 0x55, 0x44, 0x33, 0x22, 0x11, 0x00
    #.word 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xaa,
0xbb, 0xcc, 0xdd, 0xee, 0xff
    .word 0xff, 0xee, 0xdd, 0xcc, 0xbb, 0xaa, 0x99, 0x88, 0x77, 0x66, 0x55,
0x44, 0x33, 0x22, 0x11, 0x00
    #.word 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xaa,
0xbb, 0xcc, 0xdd, 0xee, 0xff, 0xf00, 0xf11, 0xf22, 0xf33, 0xf44, 0xf55, 0xf66,
0xf77, 0xf88, 0xf99, 0xfaa, 0xfbb, 0xfcc, 0xfdd, 0xfee, 0xfff
    #.word 0xfff, 0xfee, 0xfdd, 0xfcc, 0xfbb, 0xfaa, 0xf99, 0xf88, 0xf77, 0xf66,
0xf55, 0xf44, 0xf33, 0xf22, 0xf11, 0xf00, 0xff, 0xee, 0xdd, 0xcc, 0xbb, 0xaa,
0x99, 0x88, 0x77, 0x66, 0x55, 0x44, 0x33, 0x22, 0x11, 0x00

    .align 4
```