# 计算机组成原理课程实验


> 吴骏东
>
> PB20111699


## 1. 实验内容描述

- 基于MIPS或者RISC-V汇编，设计一个冒泡排序程序，并用Debug工具调试执行。(MIPS仿真器/Ripes仿真)
- 测量冒泡排序程序的执行时间。

## 2. 实验程序设计

本实验设计了支持至多 256 个有符号数按照由小到大排序的功能。数字通过 `.data` 方式存储在内存的 `x10000000` 位置上。排序方式为冒泡排序。基本的冒泡排序 C 语言程序如下：

```c
#include <stdio.h>
#include <stdlib.h>

int n = 256;
int a[256] = {0};

int main() {
    for (int i = 0; i < n; i++) {
        a[i] = 50 - i;
    }

    for (int i = 0; i < n; i++)
        for (int j = 0; j < n - i; j++) {
            if (a[j] > a[j+1]) {
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }

    for (int i = 0; i < n; i++) {
        printf("a[%d] = %d\n", i, a[i]);
    }
    return 0;
}
```

上述程序转化为 RISC-V 代码为：

```
.data
.word 4
.word 3
.word 2
.word 1
.word -1
```

```
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
```

```
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
```

```
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
```

```
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
```

```
.word 2
.word -12
.word -13
.word 4
.word 3
.word 2
.word 1
.word -1
.word 0
.word 1
.word 1
.word 5
.word 6
.word 7
.word 8
.word 2
.word 2
.word -12
.word -13
num: .word 4            # the num of numbers

.text
lw s11, num
add s11, x0, s11        # the num of numbers
addi s2, x0, 0          # s2 = 0
lui s2, 65536           # .data starts at x10000000

# s0 saves the number of nums
# s2 saves the current address of number

addi t0, x0, 0      # i = 0
addi s11, s11, -1       # s11 = n - 1
  For1:
    beq t0, s11, Done
    addi t1, x0, 0      # j = 0
    sub s3, s11, t0     # s3 = n - i - 1

      For2:
        beq t1, s3, For1done    # if (j == n - i) break;
        addi s5, t1, 0      # s5 = j
        add s5, s5, s5
        add s5, s5, s5
        add s1, s2, s5      # s1 = s2 + 4j
        jal x1, Swap
        addi t1, t1, 1      # j++
      j For2

    For1done:
    addi t0, t0, 1  # i++
  j For1


  Swap:
    # will check the number in mem[s1] and mem[s1 + 4]
    # if mem[s1] > mem[s2] then swap them
    # reg s6 contains mem[s1]
    # reg s7 contains mem[s1+4]
```

```
    lw s6, 0(s1)
    lw s7, 4(s1)

    compare:
    blt s6, s7, noswap  # if mem[s1] < mem[s2] do not swap
    doswap:
    sw s6, 4(s1)
    sw s7, 0(s1)
    noswap:
  ret


Done: # the sort is finidhed
addi s7, s7, 1
```

## 3. 运行结果展示

以下为程序运行结果：

### 4个数

共耗时 111 个时钟周期。

**排序前**

| | |
|---|---|
| 0x1000000c | 0x00000001 |
| 0x10000008 | 0x00000002 |
| 0x10000004 | 0x00000003 |
| 0x10000000 | 0x00000004 |

**排序后**

| | |
|---|---|
| 0x1000000c | 0x00000004 |
| 0x10000008 | 0x00000003 |
| 0x10000004 | 0x00000002 |
| 0x10000000 | 0x00000001 |

### 8个数

共耗时 433 个时钟周期。

**排序前**

| | |
|---|---|
| 0x1000001c | 0x00000001 |
| 0x10000018 | 0x00000001 |
| 0x10000014 | 0x00000000 |
| 0x10000010 | 0xffffffff |
| 0x1000000c | 0x00000001 |
| 0x10000008 | 0x00000002 |
| 0x10000004 | 0x00000003 |
| 0x10000000 | 0x00000004 |

**排序后**

| | |
|---|---|
| 0x1000001c | 0x00000004 |
| 0x10000018 | 0x00000003 |
| 0x10000014 | 0x00000002 |
| 0x10000010 | 0x00000001 |
| 0x1000000c | 0x00000001 |
| 0x10000008 | 0x00000001 |
| 0x10000004 | 0x00000000 |
| 0x10000000 | 0xffffffff |

## 16个数

共耗时 1697 个时钟周期

### 排序前

| | |
|---|---|
| 0x1000003c | 0xfffffff3 |
| 0x10000038 | 0xfffffff4 |
| 0x10000034 | 0x00000002 |
| 0x10000030 | 0x00000002 |
| 0x1000002c | 0x00000008 |
| 0x10000028 | 0x00000007 |
| 0x10000024 | 0x00000006 |
| 0x10000020 | 0x00000005 |
| 0x1000001c | 0x00000001 |
| 0x10000018 | 0x00000001 |
| 0x10000014 | 0x00000000 |
| 0x10000010 | 0xffffffff |
| 0x1000000c | 0x00000001 |
| 0x10000008 | 0x00000002 |
| 0x10000004 | 0x00000003 |
| 0x10000000 | 0x00000004 |

### 排序后

| | |
|---|---|
| 0x1000003c | 0x00000008 |
| 0x10000038 | 0x00000007 |
| 0x10000034 | 0x00000006 |
| 0x10000030 | 0x00000005 |
| 0x1000002c | 0x00000004 |
| 0x10000028 | 0x00000003 |
| 0x10000024 | 0x00000002 |
| 0x10000020 | 0x00000002 |
| 0x1000001c | 0x00000002 |
| 0x10000018 | 0x00000001 |
| 0x10000014 | 0x00000001 |
| 0x10000010 | 0x00000001 |
| 0x1000000c | 0x00000000 |
| 0x10000008 | 0xffffffff |
| 0x10000004 | 0xfffffff4 |
| 0x10000000 | 0xfffffff3 |

## 更多的数字排序结果概述

**由于篇幅限制，以下仅展示程序运行时间。**

| 排序数字数目 | 运行所需时钟周期数目 |
| --- | --- |
| 32 | 6889 |
| 64 | 27819 |
| 128 | 112384 |
| 256 | 455897 |

彩蛋：跑了一个小时的程序