LABH3 Report

PB21030794 王道宇

实验目的及内容

- 本次实验需要使用`RISC-V`指令集编写一个排序程序以及指令测试程序

逻辑设计

- 排序程序：

  使用冒泡排序，数组的首位是要排的数的个数，其余是数组的内容。

  - 使用

    ```
    array: .word 5, 5, 3, 8, 4, 2   # initial array
    la a0 , array
    lw a0, 0(a0)   # a0 is the length
    la a1, array
    addi a1, a1, 4  #a1 is &array[1]
    ```

    来取数组的个数以及数组第一个数的地址，用于后续的排序操作。

  - 排序分为外部排序和内部排序，在外部排序中，保持外部变量依次加一，在内部排序中，保持外部变量不变，让内部变量依次加一，直到内部变量或外部变量达到给定的值。在排序内部给定的两个数判断大小进行比较大小，满足一定条件就进行交换：对应的c++代码如下：

```cpp
#include <iostream>
#include <vector>
#define N 5
using namespace std;
int main(){
    std::vector<int> array;
    int number = N, n = 0;
    array.push_back(number);
    for (int i = 0; i < number; ++i) {
        cin >> n;
        array.push_back(n);
    }
    for (int i = 1; i < array.size() - 1;i++ ) {
        for (int j = i; j < array.size();j++) {
            if(array[i] > array[j]) swap(array[i], array[j]);
        }
    }
}
```

- 指令测试程序：
  - 首先我们知道`beq x0, x0, Next1`一定正确。所以我们基于此，继续测试之后的指令是否正确。
  - 我测试的顺序是：

```
.text
beq x0, x0, Next1
Next1: # addi
Next3: # add
Next4: # sub
Next5: # and
Next6: # or
Next7:  # xor
Next8: # blt
Next9: # slli
Next10: # srli
Next11: # srai
Next12:  # lui
Next13: # bltu
Next14: # jal
Next15: # jalr
Next16: # lw
Next17: # sw
```

- 每一个指令正确执行时，跳转到最后的TRUE，错误时，跳转到最后的FALSE。一旦跳转则进入循环。相当于一直在TRUE和FALSE内部。

测试结果及分析

- 测试结果分析

s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000170
t5	30	0x00000001
t6	31	0x00000000
pc		0x00000168

`t5`是最后完成的标志，当`t5`为`1`时表示所有指令测试结果正确。

- 排序结果分析

  排序开始：

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00002000	0x00000005	0x00000005	0x00000003	0x00000008	0x00000004	0x00000002	0x00000000	0x00000000
0x00002020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

  排序结束:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00002000	0x00000005	0x00000002	0x00000003	0x00000004	0x00000005	0x00000008	0x00000000	0x00000000
0x00002020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

可以看到，排序结果是正确的。