**中国科学技术大学计算机学院**

**《计算机组成原理实验》报告**



实验题目：汇编程序设计

学生姓名：\_\_\_黄瑞轩\_\_\_

学生学号：\_PB20111686\_

完成日期：\_ 2022.4.3 \_

计算机实验教学中心制

2020年09月

**实验题目**

汇编程序设计

**实验目的**

* 了解汇编程序的基本结构，以及汇编程序仿真和调试的基本方法
* 熟悉RISC-V常用32位整数指令的功能，掌握简单汇编程序的设计，以及CPU下载测试方法和测试数据 (COE文件) 的生成方法

**实验环境**

* RISC-V Assembler & Runtime Simulator
* Memory-Mapped Input and Output

**实验**1: **设计汇编程序，测试指令功能**

* 对10条指令功能的逐条简单测试
* 测试逻辑

根据待测试指令与已测试指令的依赖关系确定测试先后顺序。

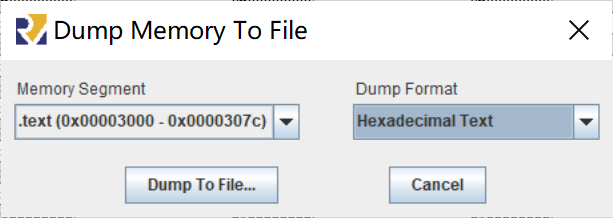
* 测试代码及逐行测试结果

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | *# test instructions*  .data  NUM1: .word 0xffffeeee  NUM2: .word 0x45677654  ADDR: .word 0x00000018  .text  *main:*  lw a0, NUM1  lw a1, NUM2  lw a5, ADDR  li s1, 0x0  li s2, 0x1  blt s2, s1, *main*  blt s1, s2, *TEST*  *START:*  add a2, a0, a1  addi a3, a0, 0x464  sub a4, a0, a1  sw a5, 0(a5)  auipc a6, 0x0  auipc a7, 0x12  *TEST:*  addi s1, s1, 0x1  beq s1, s2, *START*  *# test jal & jalr*  li t6, 1  li t5, 0  *JALBG:*  beq t6, zero, *EXIT*  addi t6, t6, -1  addi t5, t5, 1  jal *JALBG*  *EXIT:*  auipc s10, 0x0  addi s10, s10, 8  addi s10, s10, 8  addi s10, s10, 8  jalr s10  j *END*  *FUNC:*  add s8, a2, a3  ret  *END:* | 1~3条语句执行后：      lw指令测试完毕。  4~5条语句执行后：      li（实际上是特殊的addi）指令测试完毕。  6~7条语句执行后：    第6条语句执行后，未跳转，第7条语句执行后，程序跳转到TEST标记处，blt指令测试完毕。  第14条语句执行后：    addi指令测试完毕。  第15条语句执行后：    程序跳转到START标记处，beq指令测试完毕。  第8条语句执行后：    add指令测试完毕。  第10条语句执行后：    sub指令测试完毕。  第11条语句执行后：    sw指令测试完毕。  12~13条语句执行后：      auipc指令测试完毕。  第21条语句执行后：      jal指令测试完毕。  第26条语句执行后：      jalr指令测试完毕。 |

* COE文件生成

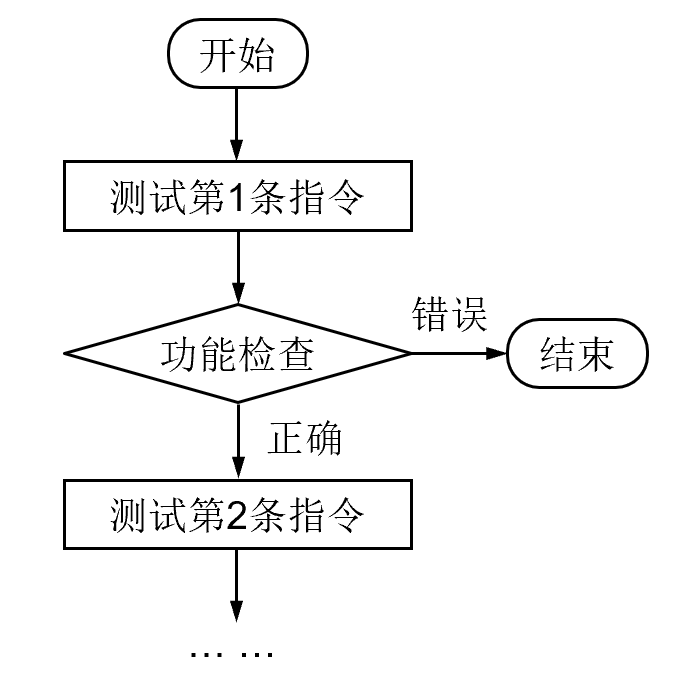
上面所用测试代码见附件step1.asm。

COE文件生成参考PPT步骤，配置如下：



生成并按PPT增加键名的COE文件见附件step1.coe。

* 选项：若干条指令功能的充分测试和自动检查
* 测试逻辑



注意：这里经人工检验过beq和lw指令是正确的，否则无法进行测试。最后结果存在s11寄存器中，寄存器内容为1则表示正确，否则，s11寄存器保存出错PC值，跳出。

* 测试汇编代码与COE文件

汇编代码见附件step1\_append.asm，COE文件见附件step1\_append.coe。

* 测试过程

运行带自动评测的第一部分代码后，结果为如下。

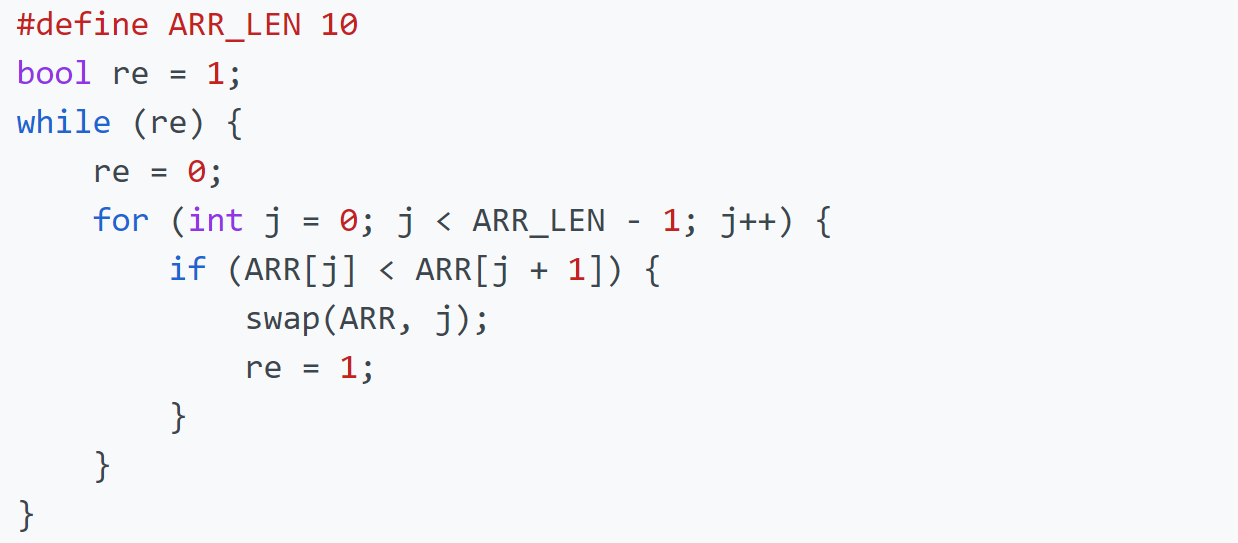


这表示自动评测正确。

实验2: 设计汇编程序实现数组排序

* 设计汇编程序实现数组排序（写死内容，输出结果到显示器）
* 设计思路

采用带Flag优化的冒泡排序法，一旦一轮检测中没有进行交换操作，说明排序已经结束。排序结束后，内存的低地址存较大的数据（有符号数）。C风格伪代码思路如下。



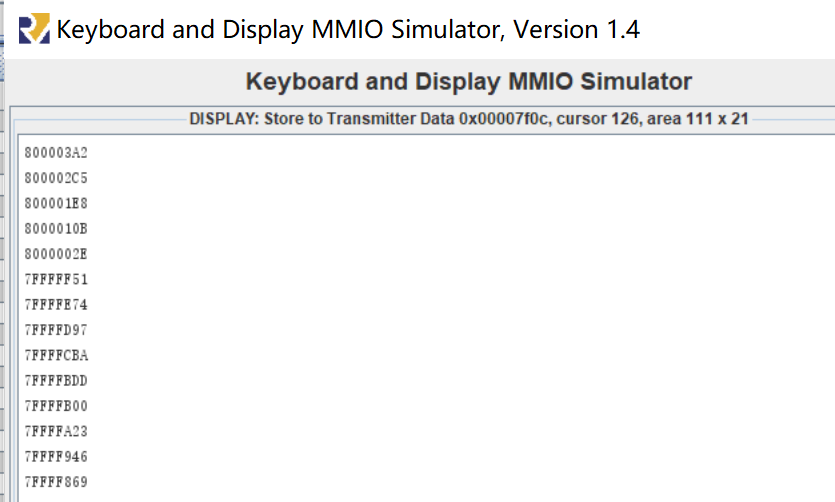
因为一开始内存里没有值，首先要向内存里写一些值，再执行排序。为了显示排序的正确性，这里取开始地址为0x100，结束地址为0x134，初始数组内容为0x800003a2，增量为-0xdd，这样中间就有正数和负数的跨越，更能显示排序正确性。

* 汇编代码与COE文件

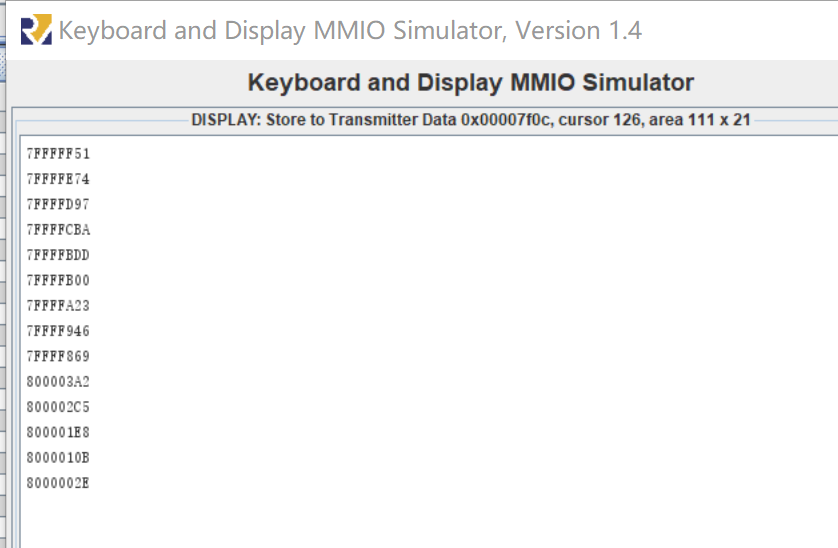
汇编代码见附件step2.asm，COE文件见附件step2.coe。

* 排序测试

先注释排序部分，利用MMIO输出数据，结果为：



再加入排序部分，利用MMIO输出数据，结果为：



排序结果与预期一致。

* 选项：设计汇编程序实现数组排序（键盘输入内容）
* 设计思路

排序思路与前面一致，主要在键盘输入MMIO上做一些处理。为了避免超出地址空间，对数组大小上限做了限制，最多为0xfff = 4095个数。输入逻辑是：在键盘上连续输入16进制数位（a~f为小写），前三位为数组大小，之后为连续的、定长的（一个数8位）数组内容。

* 汇编代码与COE文件

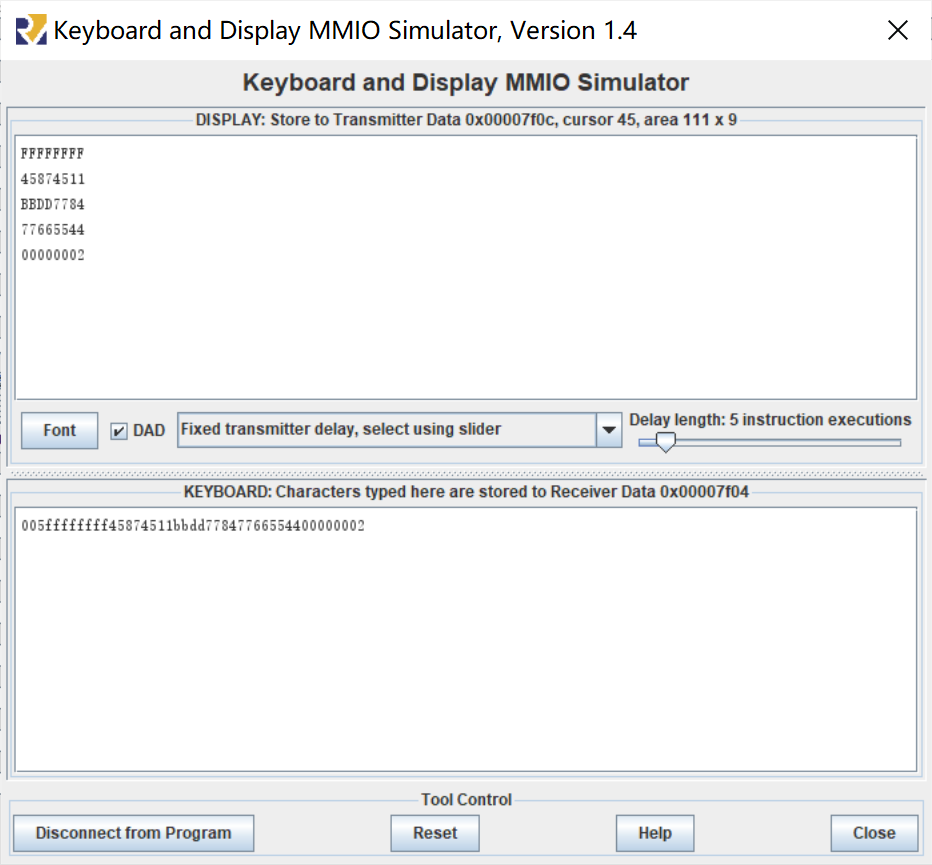
汇编代码见附件step2\_append.asm，COE文件见附件step2\_append.coe。

* 排序测试

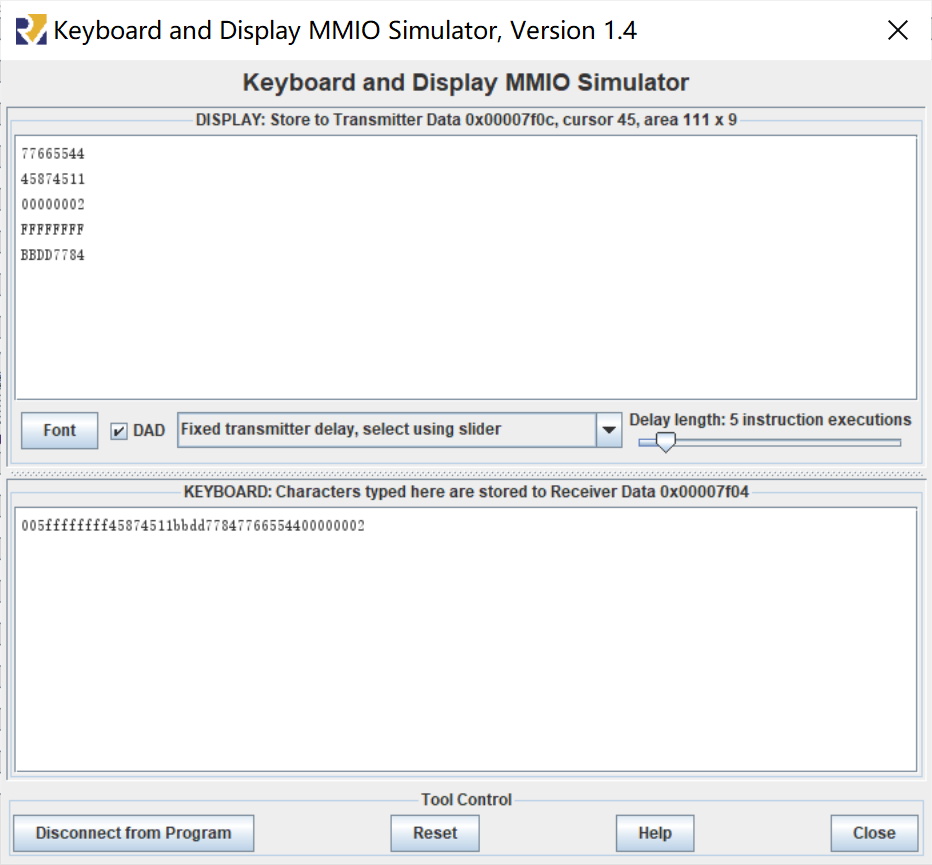
5个数字：0xffffffff，0x45874511，0xbbdd7784，0x77665544，0x00000002。

即输入：005ffffffff45874511bbdd77847766554400000002。

先去掉排序模块，输入这组数据，得到输出：



再加入排序模块，输入这组数据，得到输出：



经验证与预期结果一致。

总结篇

* 收获

这次实验我通过实际应用，熟练的掌握了RISC-V常用指令的用法。同时对在ICS课程中没有充分掌握的地址映射IO即MMIO有了更加深刻的理解和掌握。

* 建议

1. PPT对实验要求语焉不详，特别是需要达到的目标，希望改进。
2. MMIO的操作RARS没有给出详细的说明，Google上类似资料也非常少，我对MMIO的使用居然是从RARS的Github issue中学会的，希望能加入一些详细步骤，降低一些难度。