# 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*

```
    # test instructions
    .data
    NUM1: .word 0xffffeeee
    NUM2: .word 0x45677654
    ADDR: .word 0x00000018
    .text
    main:
 1      lw a0, NUM1
 2      lw a1, NUM2
 3      lw a5, ADDR
 4      li s1, 0x0
 5      li s2, 0x1
 6      blt s2, s1, main
 7      blt s1, s2, TEST
    START:
 8      add a2, a0, a1
 9      addi a3, a0, 0x464
10      sub a4, a0, a1
11      sw a5, 0(a5)
12      auipc a6, 0x0
13      auipc a7, 0x12
    TEST:
14      addi s1, s1, 0x1
15      beq s1, s2, START
    # test jal & jalr
16      li t6, 1
17      li t5, 0
    JALBG:
18      beq t6, zero, EXIT
19      addi t6, t6, -1
20      addi t5, t5, 1
21      jal JALBG
    EXIT:
22      auipc s10, 0x0
23      addi s10, s10, 8
24      addi s10, s10, 8
25      addi s10, s10, 8
26      jalr s10
27      j END
    FUNC:
28      add s8, a2, a3
29      ret
    END:
```

1~3 条语句执行后：

| | | |
|---|---|---|
| a0 | 10 | 0xffffeeee |
| a1 | 11 | 0x45677654 |
| a5 | 15 | 0x00000018 |

lw 指令测试完毕。

---

4~5 条语句执行后：

| | | |
|---|---|---|
| s1 | 9 | 0x00000000 |
| s2 | 18 | 0x00000001 |

li（实际上是特殊的 addi）指令测试完毕。

---

6~7 条语句执行后：

| | | | |
|---|---|---|---|
| 0x00003040 | 0x00148493 | addi x9, x9, 1 | 21: TEST: addi s1, s1, 0x1 |

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：

| | | |
|---|---|---|
| s1 | 9 | 0x00000001 |

addi 指令测试完毕。

---

第 15 条语句执行后：

| | | | |
|---|---|---|---|
| 0x00003028 | 0x00b50633 | add x12, x10, x11 | 15: START: add a2, a0, a1 |

程序跳转到 START 标记处，beq 指令测试完毕。

---

第 8 条语句执行后：

| | | |
|---|---|---|
| a0 | 10 | 0xffffeeee |
| a1 | 11 | 0x45677654 |
| a2 | 12 | 0x45676542 |

add 指令测试完毕。

---

第 10 条语句执行后：

| | | |
|---|---|---|
| a4 | 14 | 0xba98789a |

sub 指令测试完毕。

---

第 11 条语句执行后：

Data Segment

| Address | ... | ... | ... | ... | ... | Value (+18) |
|---|---|---|---|---|---|---|
| 0x00000000 | ... | ... | ... | ... | ... | 0x00000018 |

sw 指令测试完毕。

---

12~13 条语句执行后：

| | | | |
|---|---|---|---|
| 0x00003038 | 0x00000817 | auipc x16, 0 | 19: auipc a6, 0x0 |
| 0x0000303c | 0x00012897 | auipc x17, 18 | 20: auipc a7, 0x12 |

| | | |
|---|---|---|
| a6 | 16 | 0x00003038 |
| a7 | 17 | 0x0001503c |

auipc 指令测试完毕。

---

第 21 条语句执行后：

| | | |
|---|---|---|
| ra | 1 | 0x00003060 |

| | | | |
|---|---|---|---|
| 0x00003050 | 0x000f8863 | beq x31, x0, 0x00000010 | 26: JALBG: beq t6, zero, EXIT |

jal 指令测试完毕。

---

第 26 条语句执行后：

| | | |
|---|---|---|
| s10 | 26 | 0x00003078 |

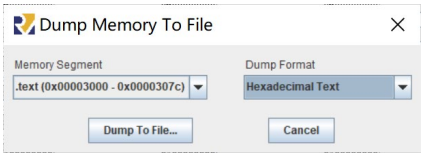
| | | | |
|---|---|---|---|
| 0x00003078 | 0x00d60c33 | add x24, x12, x13 | 36: FUNC: add s8, a2, a3 |

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。

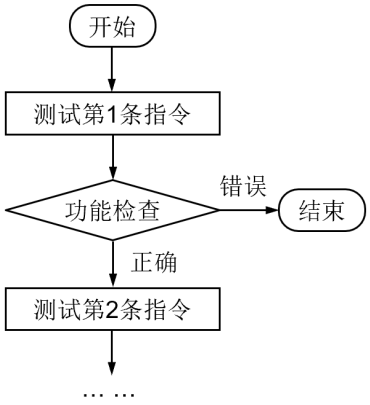
COE 文件生成参考 PPT 步骤，配置如下：




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***



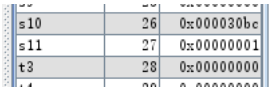

注意：这里经人工检验过 beq 和 lw 指令是正确的，否则无法进行测试。最后结果存在 s11 寄存器中，寄存器内容为 1 则表示正确，否则，s11 寄存器保存出错 PC 值，跳出。

➔ ***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。

```
#define ARR_LEN 10
bool re = 1;
while (re) {
    re = 0;
    for (int j = 0; j < ARR_LEN - 1; j++) {
        if (ARR[j] < ARR[j + 1]) {
            swap(ARR, j);
            re = 1;
        }
    }
}
```

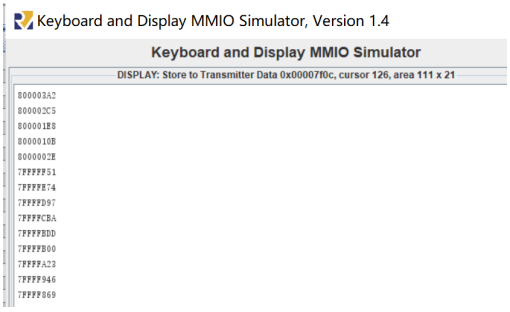
因为一开始内存里没有值，首先要向内存里写一些值，再执行排序。为了显示排序的正确性，这里取开始地址为 0x100，结束地址为 0x134，初始数组内容为 0x800003a2，增量为-0xdd，这样中间就有正数和负数的跨越，更能显示排序正确性。

➔ ***汇编代码与 COE 文件***

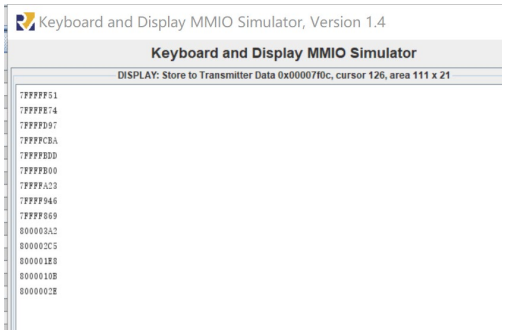
汇编代码见附件 step2.asm，COE 文件见附件 step2.coe。

➔ ***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***

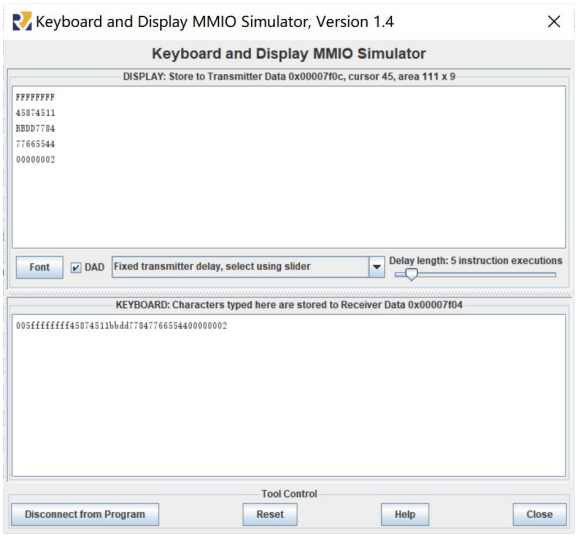
汇编代码见附件 step2_append.asm，COE 文件见附件 step2_append.coe。

➔ ***排序测试***

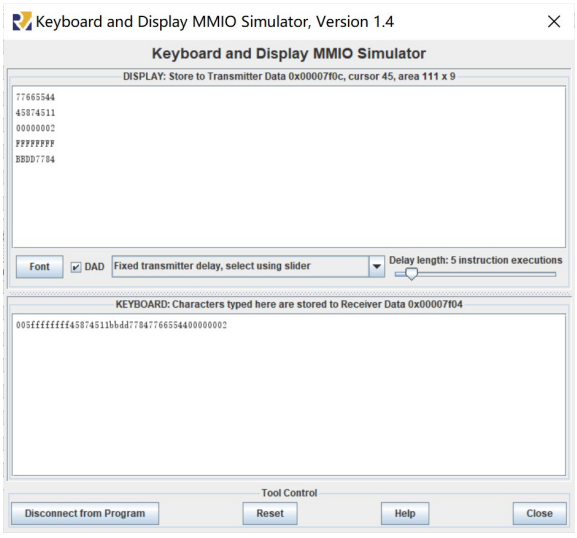
5 个数字：0xffffffff，0x45874511，0xbbdd7784，0x77665544，0x00000002。

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。