计组实验 6 报告

孙启翔 241220098

May 2025

1 控制器设计实验

1.1 实验整体方案设计

本实验目标是设计并实现 RV32I 单周期 CPU 的控制器。控制器的作用是根据指令的操作码（opcode）、功能码（funct3 和 funct7）生成相应的控制信号，驱动数据通路完成指令的执行。

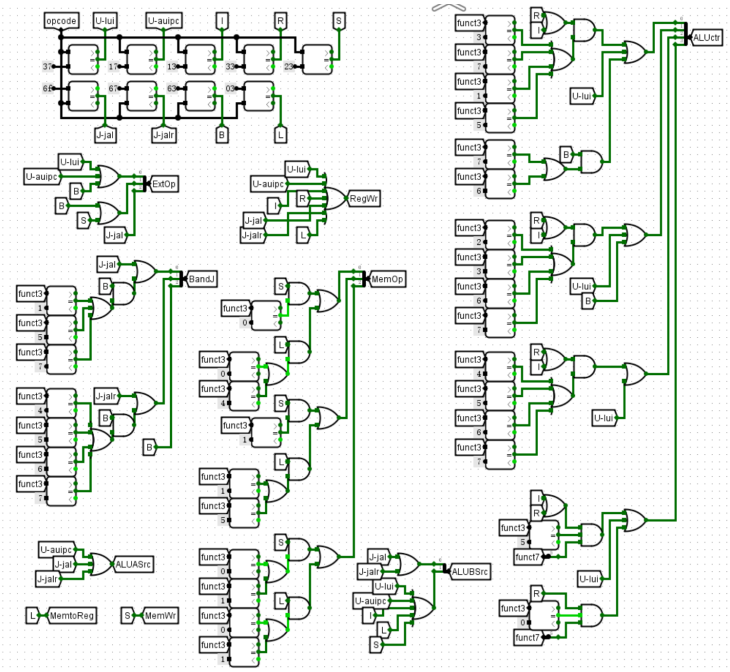
本实验需要实现 RV32I 指令集中的 37 条指令，分为三类：

- 整数运算指令（21 条）：包括加减、逻辑、比较、移位等操作；
- 控制转移指令（8 条）：包括分支跳转，如 beq、jal 等；
- 存储器访问指令（8 条）：包括 lw、sw、lb、sb 等。

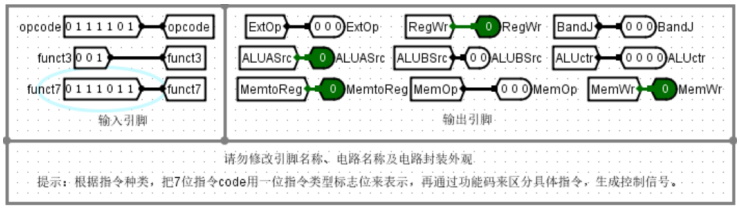
控制器输出的主要控制信号有：ExtOp、ALUASrc、ALUBSrc、ALUctr、MemOp、RegWr、MemWr、BandJ、MemtoReg、NPCASrc 和 NPCBSrc 等。

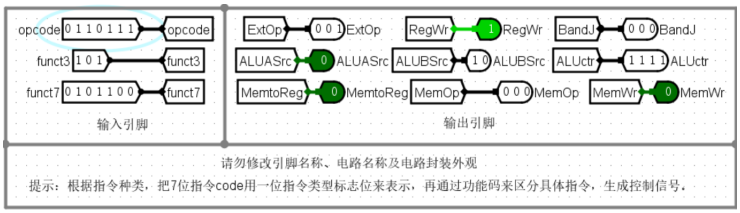
控制信号的设计基于指令类型进行划分，通过分析 opcode 和 funct3、funct7 字段，利用逻辑门电路构建控制器。在 Logisim 中完成电路搭建后，逐条测试所有 37 条指令，验证控制信号输出是否正确。

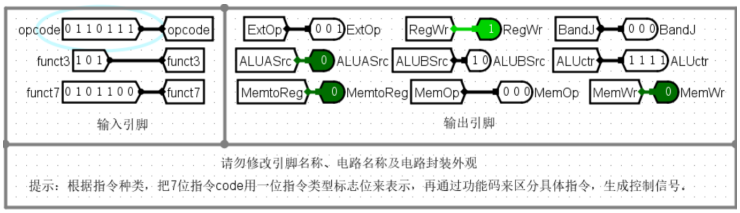
1.2 实验原理图及电路图



1.3 实验数据仿真测试图







1.4 错误现象及分析

这道题目其实就是根据给出的真值表进行判断，很麻烦，我已经尽可能对电路进行了压缩

2 单周期 CPU 设计实验

2.1 实验整体方案设计

本实验目标是基于前期已实现的各个数据通路模块，设计并搭建一个支持 RV32I 指令集的单周期 CPU，实现从取指、译码、执行、访存到写回的一体化执行流程。

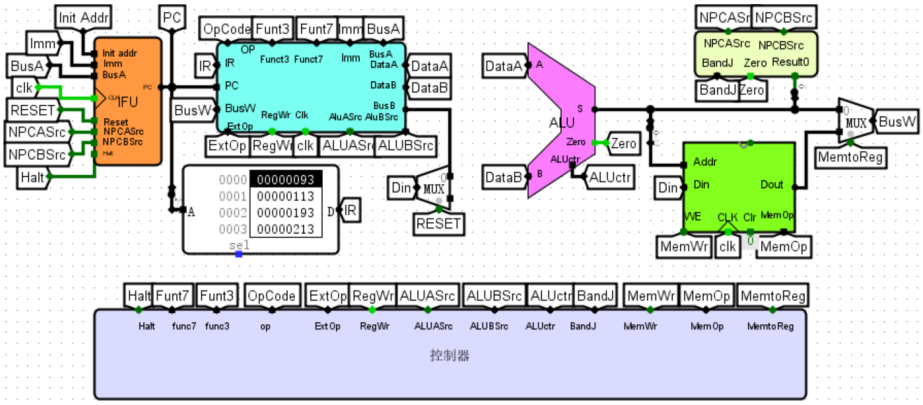
CPU 中各个部件均在一个时钟周期内完成指令执行。由于 Logisim 中 RAM 模块在上升沿写入，为保证同步，PC、寄存器堆、数据存储器等状态元件均设为上升沿触发。

控制器根据指令的 `opcode`、`funct3` 和 `funct7` 解码生成控制信号，驱动 ALU、数据存储器、寄存器堆等模块工作。指令执行终止由 `ecall` 指令控制，当识别 opcode = 0x73 时，控制器输出 Halt 信号，停止 PC 更新。

主电路结构按图示连接 IFU、ROM、控制器、ALU、DataRAM 等子模块。实验加载测试程序 `lab6.2.hex`，涵盖算术、逻辑、移位、跳转、访存等主要指令，用于验证 CPU 的功能正确性。

最终，在寄存器 x10 中若输出 `0x00c0ffee`，表示所有测试指令均通过，CPU 设计功能正确。

2.2 实验原理图及电路图



2.3 错误现象及分析

这道题目和后面的所有题目其实本质上都是从一串二进制数变成一个可以实现相应功能的高级语言程序。后面就不再给出具体实现的电路图，和这道题没有任何区别。

3 累加和程序测试实验

这道题就是一个 for 循环进行累加，将其转换成 RV32I 的指令并存入寄存器就可以了。

4 冒泡排序程序测试实验

同上，变成冒泡排序。

5 官方测试集实验

同上

6 计算机系统基础 PA 程序测试实验

同上

序号	测试指令	x1 寄存器输出值	x3 寄存器输出值	x10 寄存器输出值	时钟周期数
1	add	0x00000010	0x00000026	0x00c0ffee	458
2	addi	0x00000021	0x00000019	0x00c0ffee	235
3	and	0x11111111	0x0000001b	0x00c0ffee	478
4	andi	0x00ff00ff	0x0000000e	0x00c0ffee	191
5	auipc	0x00000000	0x00000003	0x00c0ffee	52
6	beq	0x00000003	0x00000015	0x00c0ffee	284
7	bge	0x00000003	0x00000018	0x00c0ffee	302
8	bgeu	0x00000003	0x00000018	0x00c0ffee	327
9	blt	0x00000003	0x00000015	0x00c0ffee	284
10	bltu	0x00000003	0x00000015	0x00c0ffee	309
11	bne	0x00000003	0x00000015	0x00c0ffee	284
12	jal	0x00000003	0x00000003	0x00c0ffee	48
13	jalr	0x00000000	0x00000007	0x00c0ffee	108
14	lb	0x00000020	0x00000002	0x00c0ffee	43
15	lbu	0x00002000	0x00000002	0x00c0ffee	43
16	lh	0x00002000	0x00000002	0x00c0ffee	43
17	lhu	0x00002000	0x00000002	0x00c0ffee	284
18	lui	0xfffff800	0x00000006	0x00c0ffee	58
19	lw	0x00002000	0x00000002	0x00c0ffee	44
20	or	0x11111111	0x0000001b	0x00c0ffee	481
21	ori	0x00ff00ff	0x0000000e	0x00c0ffee	198
22	sb	0x00002000	0x00000004	0x00c0ffee	63
23	sh	0x00002000	0x00000004	0x00c0ffee	65
24	sll	0x00000400	0x0000002b	0x00c0ffee	486
25	slli	0x00000021	0x00000019	0x00c0ffee	234
26	slt	0x00000010	0x00000026	0x00c0ffee	452
27	slti	0x00ff00ff	0x00000019	0x00c0ffee	230
28	sltiu	0x00ff00ff	0x00000019	0x00c0ffee	230
29	sltu	0x00000010	0x00000026	0x00c0ffee	452
30	sra	0x00000400	0x0000002b	0x00c0ffee	505
31	srai	0x00000021	0x00000019	0x00c0ffee	249
32	srl	0x00000400	0x0000002b	0x00c0ffee	499

序号	测试指令	x1 寄存器输出值	x3 寄存器输出值	x10 寄存器输出值	时钟周期数
33	srli	0x00000021	0x00000019	0x00c0ffee	243
34	sub	0x00000010	0x00000025	0x00c0ffee	450
35	sw	0x12233001	0x00000017	0x00c0ffee	483
36	xor	0x11111111	0x0000001b	0x00c0ffee	480
37	xori	0x00ff00ff	0x0000000e	0x00c0ffee	200

C-Test C 语言测试程序执行结果

b-sort.c

程序执行周期数： 1348

数据存储器 M[0x90-0xDC] 排列结果：

```
00000000 32000000 46000000 aa000000
e8030000 dc050000 401f0000 f82a0000
204e0000 905f0100 a0860100 c0d40100
20bf0200 801a0600 20d61300 c0c62d00
808d5b00 00688909 802b530b
```

q-sort.c

程序执行周期数： 999

数据存储器 M[0x314-0x360] 排列结果：

```
00000000 32000000 46000000 aa000000
e8030000 dc050000 401f0000 f82a0000
204e0000 905f0100 a0860100 c0d40100
20bf0200 801a0600 20d61300 c0c62d00
808d5b00 00688909 802b530b
```

7 思考题

7.1 如何在单 CPU 上实现多任务处理，例如同时执行计算累加和与数据排序两个程序，阐述思路。

在单 CPU 上实现多任务处理，通常采用时间片轮转调度的方法，将 CPU 时间划分为若干小时间片，按顺序快速切换运行不同任务。CPU 在每

个时间片内执行一个任务的一部分计算，保存当前任务的上下文（寄存器状态、程序计数器等），然后切换到下一个任务，恢复其上下文继续执行。通过这种快速切换，CPU 看似同时执行多个任务，实现计算累加和与数据排序程序的并发运行。

7.2 在 CPU 的基础上，如何实现键盘输入、TTY 输出部件等输入输出设备的数据访问，构建完整的计算机系统。

在 CPU 基础上，通过设计输入输出控制器并采用内存映射或端口映射方式，将键盘、TTY 等设备的寄存器映射到 CPU 地址空间，利用中断机制实现设备与 CPU 的异步通信，结合总线系统进行数据传输和协调，从而实现对输入输出设备的数据访问，构建完整的计算机系统。

7.3 阐述如何在单周期 CPU 基础上实现多周期 CPU

在单周期 CPU 基础上实现多周期 CPU，需要将指令的各个执行阶段（取指、译码、执行、访存、写回）拆分到多个时钟周期内完成，利用状态机控制器和寄存器保存各阶段的中间结果，通过多周期时钟和阶段控制信号依次驱动各功能单元工作，从而提高时钟频率和资源复用效率。

7.4 阐述如何在单周期 CPU 基础上实现流水线 CPU?

在单周期 CPU 基础上实现流水线 CPU，通过将指令执行过程划分为多个阶段（如取指、译码、执行、访存、写回），并为每个阶段设计专用的流水线寄存器，将不同指令的各阶段操作同时并行处理，实现指令的重叠执行，从而大幅提升处理器的指令吞吐率和整体性能。