实验 6：单周期 CPU 设计与测试

姜帅 221220115

一、实验目的

1. 了解指令执行过程及其与 CPU 基本结构之间的关系。
2. 掌握单周期数据通路及其控制器的设计方法。
3. 掌握 RISC-V 汇编语言程序的基本设计方法。
4. 理解汇编语言程序与机器语言代码之间的对应关系。

二、实验环境

Logisim：https://github.com/Logisim-Ita/Logisim

RISC-V 模拟器工具 RARS：https://github.com/thethirdone/rars

三、实验内容

CPU中控制指令执行的部件是控制器。控制器输入的是指令操作码op和功能码，输出的是控制信号。控制器的主要设计步骤如下。

(1)根据每条指令的功能，分析控制信号的取值，并在表中列出。

(2)根据列出的指令和控制信号之间的关系，写出每个控制信号的逻辑表达式。

(3)实现取指令部件，设计时序信号，接连模块，实现CPU的综合。

1. 控制器设计实验

RV32I 指令集中包含 47 条基础指令，涵盖了整数运算、存储器访问、控制转移和系统控制几个大类。本次实验中需要实现除了系统控制类的 10 条指令外的 37 条指令，分为整数运算指令（21 条）、控制转移指令（8 条）和存储器访问指令（8 条）。在确定具体指令后生成每个指令对应的控制信号，来控制数据通路部件进行对应的动作。控制信号生产部件根据指令代码中的操作码 opcode、功能码 func3 和功能码 func7 来生成对应的控制信号的。信号表如下：

指令	类型	op[6:0]	func3	func7[5]	ExtOp	RegWr	ALUASrc	ALUBSrc	ALUctr	Branch	MemtoReg	MemWr	MemOp
lui	U	0110111	×	×	001	1	×	10	1111	000	0	0	×
auipc	U	0010111	×	×	001	1	1	10	0000	000	0	0	×
addi	I	0010011	000	×	000	1	0	10	0000	000	0	0	×
slti	I	0010011	010	×	000	1	0	10	0010	000	0	0	×
sltiu	I	0010011	011	×	000	1	0	10	0011	000	0	0	×
xori	I	0010011	100	×	000	1	0	10	0100	000	0	0	×
ori	I	0010011	110	×	000	1	0	10	0110	000	0	0	×
andi	I	0010011	111	×	000	1	0	10	0111	000	0	0	×
slli	I	0010011	001	0	000	1	0	10	0001	000	0	0	×
srli	I	0010011	101	0	000	1	0	10	0101	000	0	0	×
srai	I	0010011	101	1	000	1	0	10	1101	000	0	0	×
add	R	0110011	000	0	×	1	0	00	0000	000	0	0	×
sub	R	0110011	000	1	×	1	0	00	1000	000	0	0	×
sll	R	0110011	001	0	×	1	0	00	0001	000	0	0	×
slt	R	0110011	010	0	×	1	0	00	0010	000	0	0	×
sltu	R	0110011	011	0	×	1	0	00	0011	000	0	0	×
xor	R	0110011	100	0	×	1	0	00	0100	000	0	0	×
srl	R	0110011	101	0	×	1	0	00	0101	000	0	0	×
sra	R	0110011	101	1	×	1	0	00	1101	000	0	0	×
or	R	0110011	110	0	×	1	0	00	0110	000	0	0	×
and	R	0110011	111	0	×	1	0	00	0111	000	0	0	×

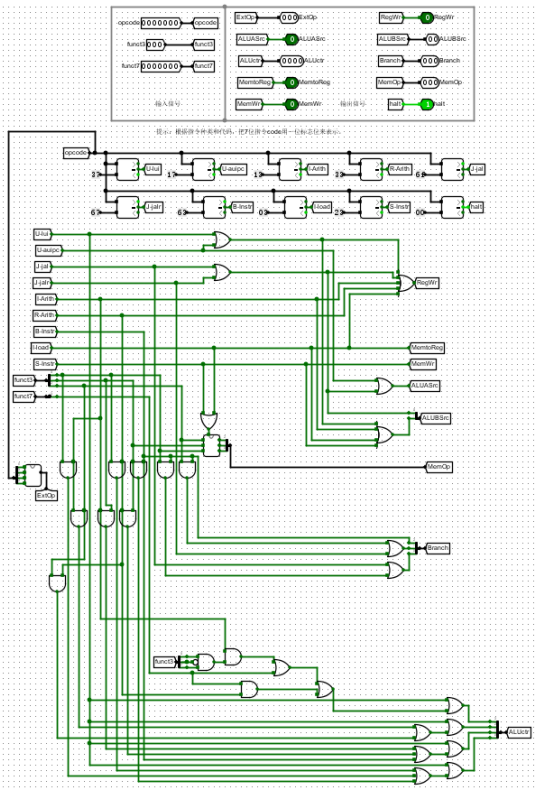
jal	J	1101111	×	×	100	1	1	01	0000	001	0	0	×
jalr	I	1100111	000	×	000	1	1	01	0000	010	0	0	×
beq	B	1100011	000	×	011	0	0	00	0010	100	×	0	×
bne	B	1100011	001	×	011	0	0	00	0010	101	×	0	×
blt	B	1100011	100	×	011	0	0	00	0010	110	×	0	×
bge	B	1100011	101	×	011	0	0	00	0010	111	×	0	×
bltu	B	1100011	110	×	011	0	0	00	0011	110	×	0	×
bgeu	B	1100011	111	×	011	0	0	00	0011	111	×	0	×
lb	I	0000011	000	×	000	1	0	10	0000	000	1	0	101
lh	I	0000011	001	×	000	1	0	10	0000	000	1	0	110
lw	I	0000011	010	×	000	1	0	10	0000	000	1	0	000
lbu	I	0000011	100	×	000	1	0	10	0000	000	1	0	001
lhu	I	0000011	101	×	000	1	0	10	0000	000	1	0	010
sb	S	0100011	000	×	010	0	0	10	0000	000	×	1	101
sh	S	0100011	001	×	010	0	0	10	0000	000	×	1	110
sw	S	0100011	010	×	010	0	0	10	0000	000	×	1	000

根据上表的定义，列出每个控制信号的逻辑表达式

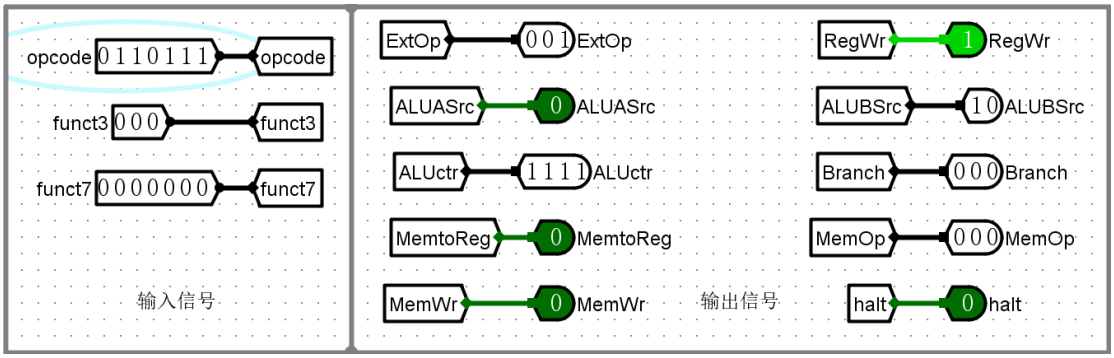
实验步骤如下：

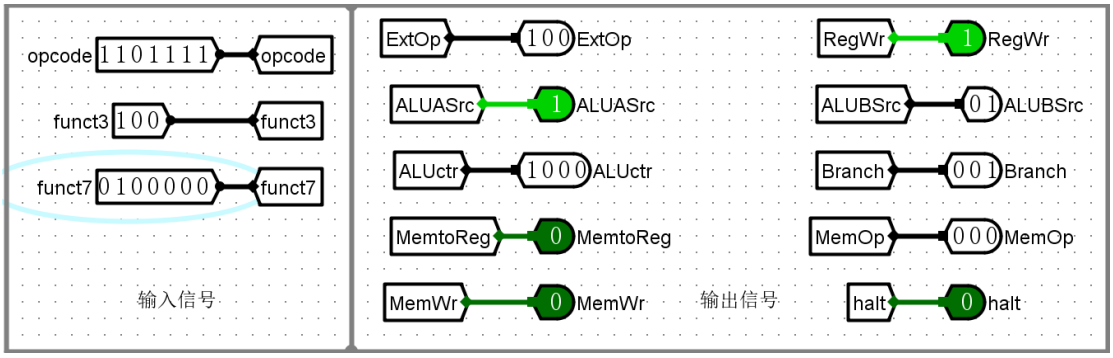
1） 创建子电路。在 Logisim 中添加一个名为“控制器”的子电路，双击该子电路名称，在右侧工作区中构建相应电路。

2） 设计子电路。在工作区中添加所需的逻辑门、输入/输出引脚。根据逻辑表达式进行线路连接，添加标识符和电路功能描述信息，得到完整的控制器电路。为了方便程序的结束执行，定义了一个输出信号 halt，当指令操作码 opcode=0000000 时，赋值为 1，反相输出到 PC 寄存器的使能端，中止程序的执行。可以根据不同类型指令操作码进行比较输出 1 位标志位。设计电路如下：



3）仿真测试输入与输出是否正确对应。





4）封装电路

2. 单周期 CPU 设计实验

在单周期 CPU 中，每条的指令都需要在一个时钟周期内完成。本次实验中，以时钟下降沿为每个时钟周期的开始，写入操作在时钟下降沿时同步实现；读取操作异步实现，只要输入有效地址后，立即输出对应数据。

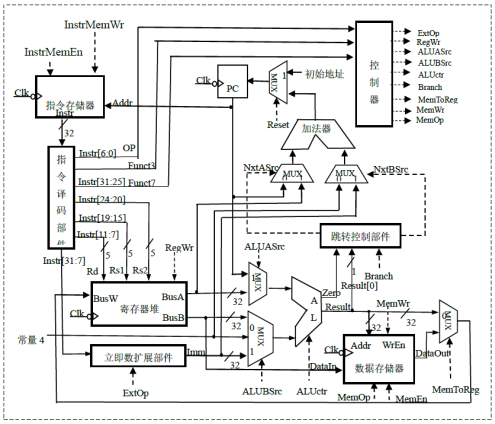
单周期 CPU 需要考虑的时序信号，有复位信号 Reset、片选信号 Sel、中止信号 halt 等。复位信号 Reset，用来初始化 CPU 中各个部件的控制信号，设置 PC 寄存器的起始址。

指令存储器和数据存储器的片选信号 Sel，高电平有效。只要 CPU 的复位信号撤销，该信号设置恒为 1。数据存储器的清零信号可与复位信号相连。写使能信号 MemWr，高电平有效，在指令执行过程中赋值。

程序执行结束后，中止信号 halt 有效，PC 寄存器使能端无效，停止输出。

CPU 由不同模块构成，分工协作完成功能。在前期的实验中已经完成全部模块的设计。lab3：寄存器堆模块；lab4：ALU 模块；lab5：数据存储器、取指令部件和数据通路模块。本次实验完成了控制器模块。

依据如下原理图，综合利用这上述模块，设计单周期 CPU。

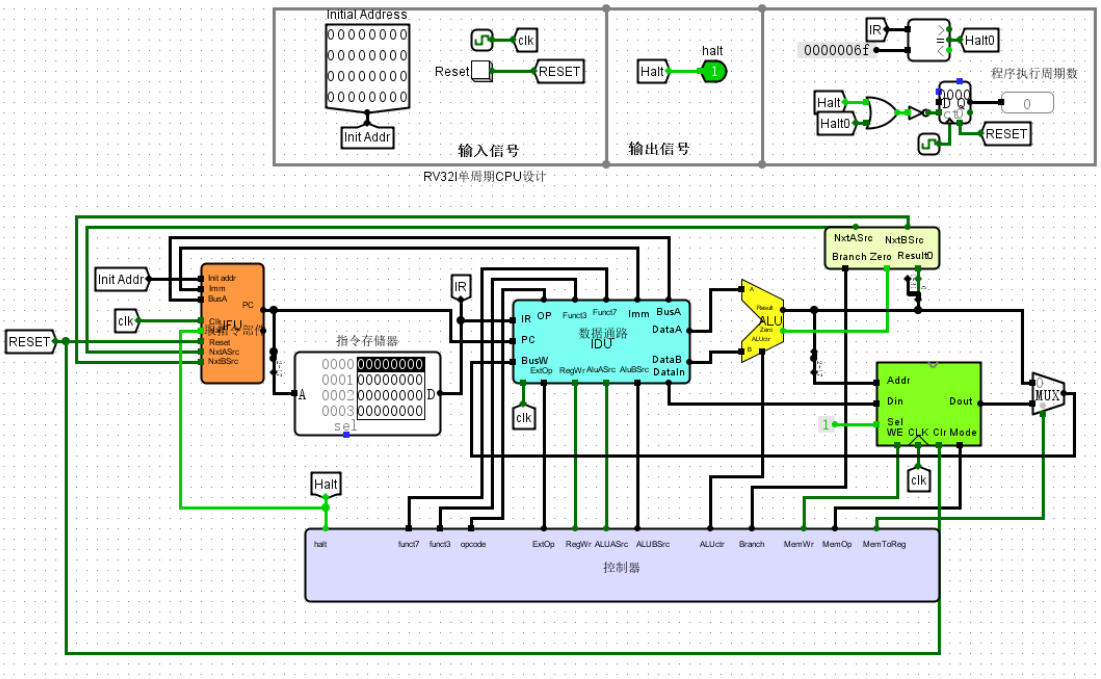


实验步骤如下：

1）创建子电路。在 Logisim 中添加一个名为“单周期 CPU”的子电路，双击该子电路名称，在右侧工作区中构建相应电路。

2）设计子电路。在 logisim 的 project 菜单下添加 logisim 库文件，把实验 3、实验 4 和实验 5 的设计文件添加到 lab6 的项目中。（为方便修改调试，本人在 lab6 中复写了部分元件）

在工作区中放置取指令部件 IFU、指令存储器 ROM、数据通路 IDU、ALU、数据存储器模块和控制器部件，以及输入输出引脚和隧道，布局引脚。根据模块中定义信号进行连接。设置指令存储器和数据存储器的属性，片选信号都定义为高电平有效。指令存储器的地址位宽设置为 16 位，数据位宽设置为 32 位。实验电路设计如下：



3）仿真测试(见以下实验)

3.用累加和程序验证 CPU 设计

（本报告略过机器码编译过程，直接使用编译好的机器码文件验证 CPU 功能）

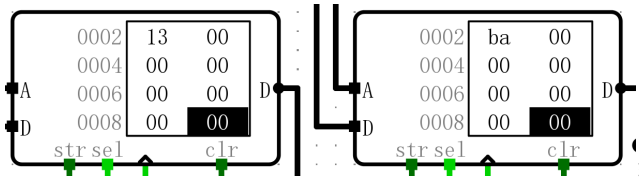
在单周期 CPU 的指令存储器（ROM）中装载计算累加和可执行机器代码镜像文件 sum.hex，在数据存储器地址 0 单元中设置初始参数，然后启动时钟信号，执行程序，观察输出结果，验证 CPU 设计的正确性。

实验步骤如下：

1） 在 Logisim 中，打开单周期 CPU 电路图，用鼠标右键点击指令存储器（ROM 组件），选择装载镜像（load image）命令，将镜像文件 sum.hex 加载到指令存储器起始地址 0 单元中。

2） 在 CPU 电路图中选中数据存储器组件，进入子电路，在最低字节的 RAM 组件地址 0x0000 处，用 16 进制编辑器写入参数 0x64。

3） 执行 CPU 电路，选择时钟连续(1kHZ) 程序结束后，进入数据储存器查看结果：

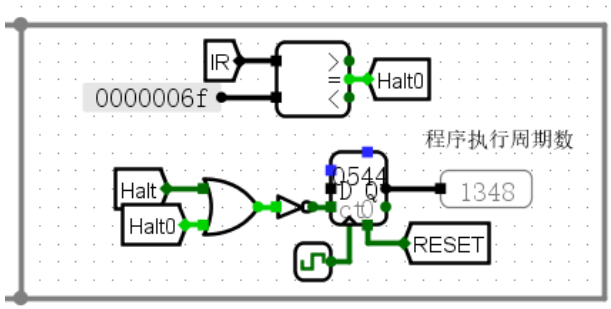


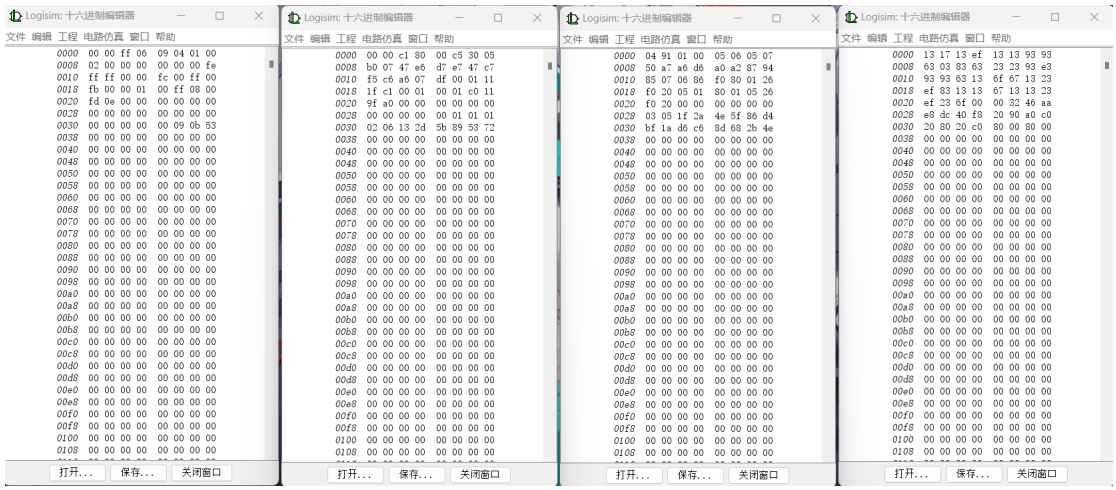
结果与预期相符

4. 排序算法测试

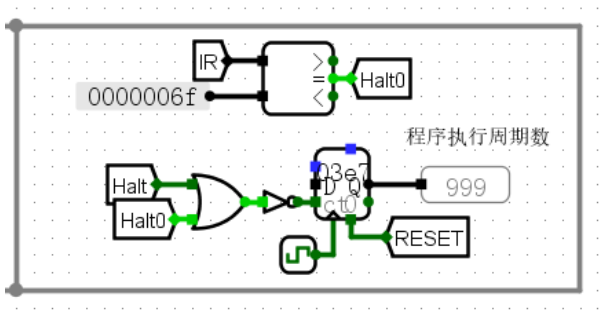
按照实验 pdf 要求分别加载冒泡排序，快速排序的实验文件，运行 CPU，查看结果

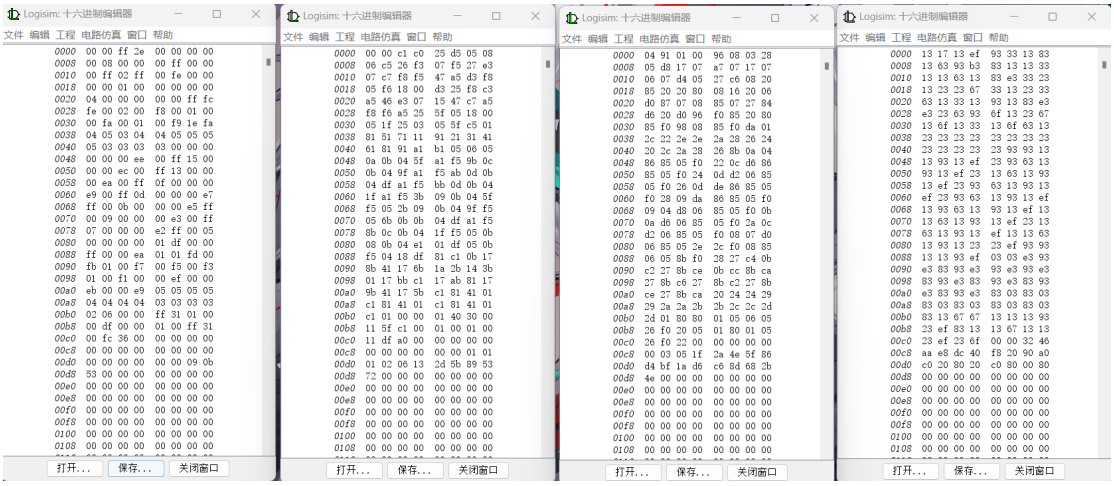
冒泡排序：





快速排序：





结果与预期相符

四、思考题

1. **在累加和计算程序中，添加溢出判读语句，并把最大的不溢出累计和以及累加序数保存到数据存储器中输出。**

溢出判断：

下一次累加结果比上一次小(或变为负数)

对应判断指令: blt r1, r2, exit (ble 为 branch ＜, r1 中为此次运算结果， r2 中为上次运算结果，exit 为跳转位置)

保存输出：

sw r3, r2, imm(保存 r2 中内容，即不溢出累计和到数据存储器相应地址中)

sw r4, r0, imm(保存 r0 中内容，即累加序数到数据存储器相应地址中)

2. **如果分支跳转指令不在 ALU 内部使用减法运算来实现，而是在 ALU 外使用独立比较器来实现，说说单周期 CPU 的电路原理图中需要做哪些修改？**

   添加额外的比较器，比较器接收 busA 和 busB 两个数据，输出 0 或 1，输出送到 ALU zero 原本送到的位置，ALU zero 输出端取消。

3. **实现单周期 CPU 后，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。**

   通过存储器(缓冲区)建立 cpu 和输入输出部件之间的联系，在 cpu 中设置 MAR（存储器地址寄存器）和 MDR（存储器数据寄存器）例如，通过键盘输入时，键入的数据会转换为二进制暂存到键盘缓冲区（一块实际的内存区域），等 CPU 从缓冲区中存取数据。

4. **如果需要实现 5 级流水线 RV32I CPU，则如何在单周期 CPU 基础上进行修改**

   一种方案：

   ① 将 CPU 的各部件按阶段划分，分别为，取指令段（IF），指令译码（ID），执行（EX），访存（M），写回（WB），在五个阶段之间添加流水段寄存器，用于存放当前流水段传输到后面所有流水段的信息。

   ② 重新设计控制信号，ID 段的控制信号有 ExtOp，EX 段有 ALUASrc，ALUBSrc， ALUctr，M 段有 MenWr， Branch， Jump，WB 段有 MemtoReg， RegWr 。

   ③ 处理冒险，为转发和动态预测等冒险处理机制添加相应的电路。