实验 5：指令读取和控制器设计

实验目的

1. 理解随机访问存储器 RAM 和只读存储器 ROM 的操作原理
2. 理解 RISC-V 指令类型和指令格式
3. 掌握使用 Logisim 软件实现取指、指令解析、立即数扩展、操作数存取的方法

实验环境

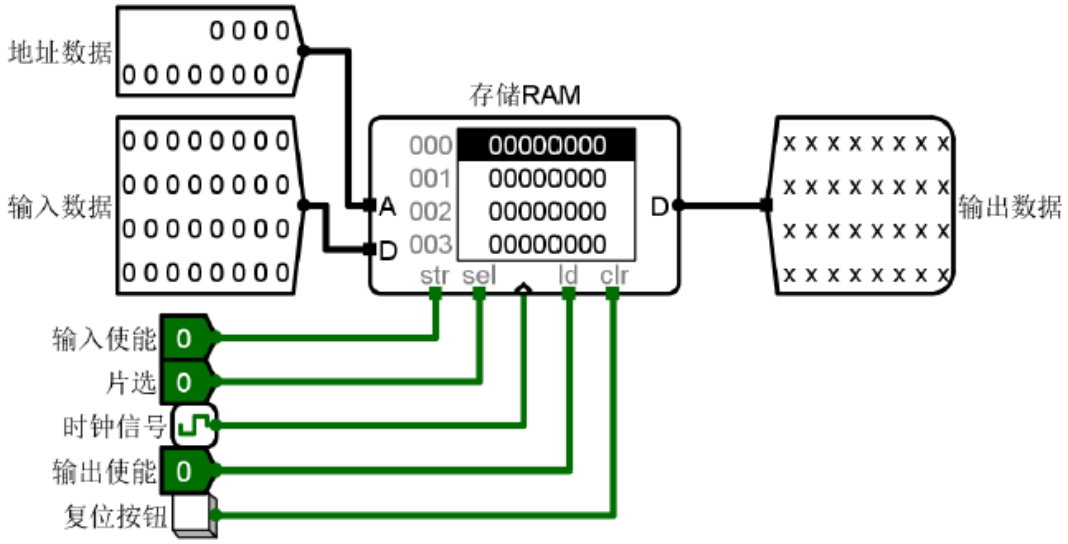
Logisim-ITA V2.16.1.2
https://sourceforge.net/projects/logisimit/

头歌线上评测平台
https://www.educoder.net/classrooms/10924/

实验内容

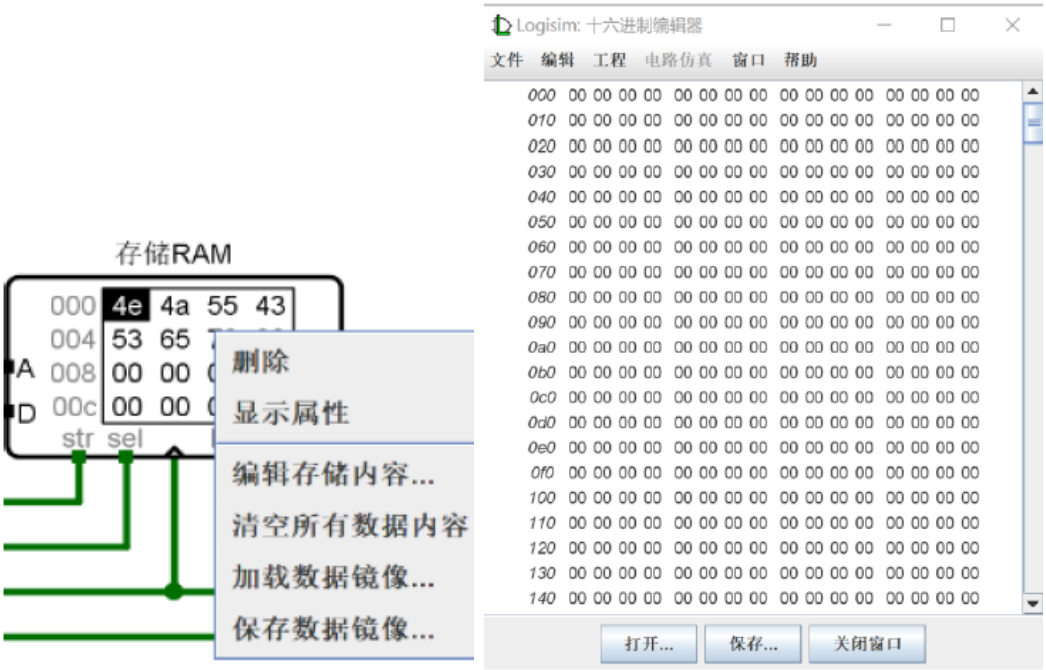
1. 存储器的写入和读取

Logisim 中 RAM 器件的地址位宽最多可设置为 24 位，数据位宽最多可设置为 32 位。在属性窗口的数据接口中有三种不同的工作模式。若设置为“分离的加载和存储引脚”模式，则有两个数据端口分别连接输入数据和输出数据（如下所示）。



Logisim 中 RAM 和 ROM 器件的数据输入可以采用 Logisim 十六进制编辑器和直接读取二进制编码文件的方法实现。把鼠标移到存储器组件上，点击鼠标右键，弹出菜单框，选中“编辑存储内容”，打开 Logisim 十六进制编辑器，可按照存储器设置的数据位宽，直接使用键盘输入数据。输入数据后，点击保存按钮，可把输入的数据保存到数据镜像文件中。当需要从数据镜像文件中加载存储时，在 RAM 器件上单击鼠标右键，菜单中选择“加载数据镜像

文件”或在 Logisim 十六进制编辑器中打开数据镜像文件直接读入文件内容到存储器。



请同学们注意，当设置数据位宽为 32 位时，RAM 采用按字编址方式（32 位），而不是采用按字节编址方式。此次实验任务要求在 RAM 存储子电路中放置一个 RAM 组件，并设置地址位宽为 12 位，数据接口模式为“分离的加载和存储引脚”模式，顶层的测试部分会自动向 RAM 中写入数据，前 16 个时钟周期为写入测试，后 16 个时钟周期读取测试。通过观察本地验证区域的 RAM 测试结果验证自己的 RAM 存储实现正确。

2. 指令读取和控制信号生成

	31 27	26 25	24 20	19 15	14 12	11 7	6 0
R	funct7		rs2	rs1	funct3	rd	opcode
I	imm[11:0]			rs1	funct3	rd	opcode
S	imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode
B	imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode
U	imm[31:12]					rd	opcode
J	imm[20|10:1|11|19:12]					rd	opcode

结合理论课内容，实验课选择了同样的 9 条 RV32I 指令作为实验目标，即

3 条 R-型指令：

```
add     rd,     rs1,    rs2
slt     rd,     rs1,    rs2
sltu    rd,     rs1,    rs2
```

2 条 I-型指令：

```
ori     rd,     rs1,    imm12
lw      rd,     rs1,    imm12
```

1 条 U-型指令：

```
lui     rd,     imm20
```

1 条 S-型指令：

sw　　rs1,　　rs2,　　imm12

1 条 B-型指令：

beq　　rs1,　　rs2,　　imm12

1 条 J-型指令：

jal　　rd,　　imm20

请同学们根据 RISC-V 的指令格式和取指令部件原理图设计 RISC-V 单周期处理器的取指令部件。其中，指令存储器使用 Logisim 内置库的 ROM 器件实现，要求指令长度位 32 位，指令存储器容量为 1KB（即在按字编址的情况下，数据位宽为 32 位，地址位宽为 10 位，假设 Logisim 中的指令存储器表示为 A[9:0]，提示：当 Logisim 中的 ROM 设置数据位宽为 32 位时，每个地址中包含 32 位信息，相当于按字节编址的 RISC-V 存储器中的 4 个单元）。在 Logisim 中读取指令存储器时，原 RISC-V 设计原理图中的 32 位指令地址 PC[31:0]，对应本次实验 PC[11:2]=A[9:0]，即 PC[31:0]其余的位（高 20 位和最低 2 位）无关。

- 本次实验的任务需要首先在指令存储器子电路中放置 ROM 并在 0 号地址开始写入以下 8 条指令:

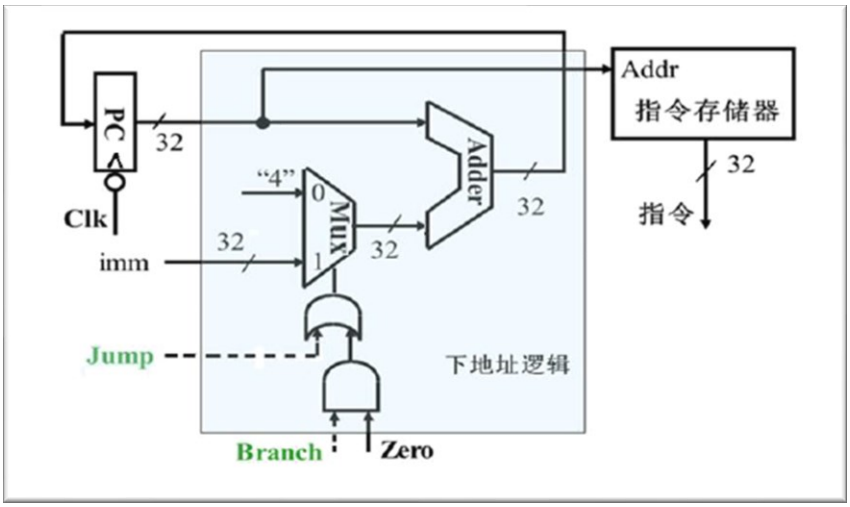
（汇编指令描述方式对应教材表 8.1，“x”代表寄存器编号，“0x”代表 16 进制数）

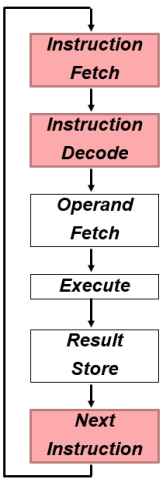
0x00a004b3　　（汇编指令 add, x10, x0, x9）
0x0020af33　　（汇编指令 slt, x30, x1, x2）
0x0020bf33　　（汇编指令 sltu, x30, x1, x2）
0x0f00ef13　　（汇编指令 ori, x30, x1, 0x0f0）
0x00412483　　（汇编指令 lw, x9, x2, 0x004）
0x00008137　　（汇编指令 lui, x2, 0x08000）
0x00912223　　（汇编指令 sw, x2, x9, 0x004）
0x7ea500e3　　（汇编指令 beq x10, x10, 0x7f0）

- 在指令存储器 Rom 的 1023 号地址写入下面这条指令:

0x818ff56f　　（汇编指令 jal, x10, 0xff80a）

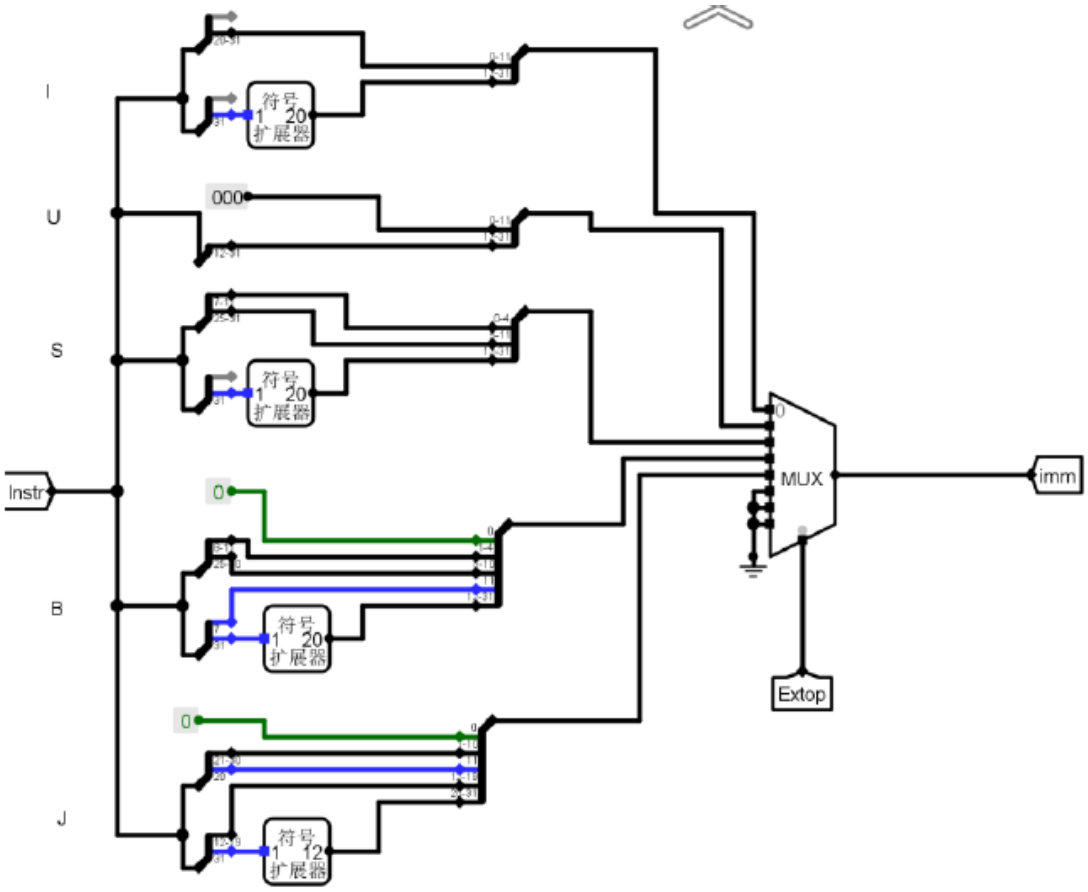
然后在指令解析测试子电路中利用 Logisim 内置库中的加法器实现指令的下地址逻辑，使得该子电路能够依次读入 9 条指令，并根据 RISC-V 指令格式将读出的指令解析为 opcode、rd、funct3、rs1、rs2、funct7 六个字段。



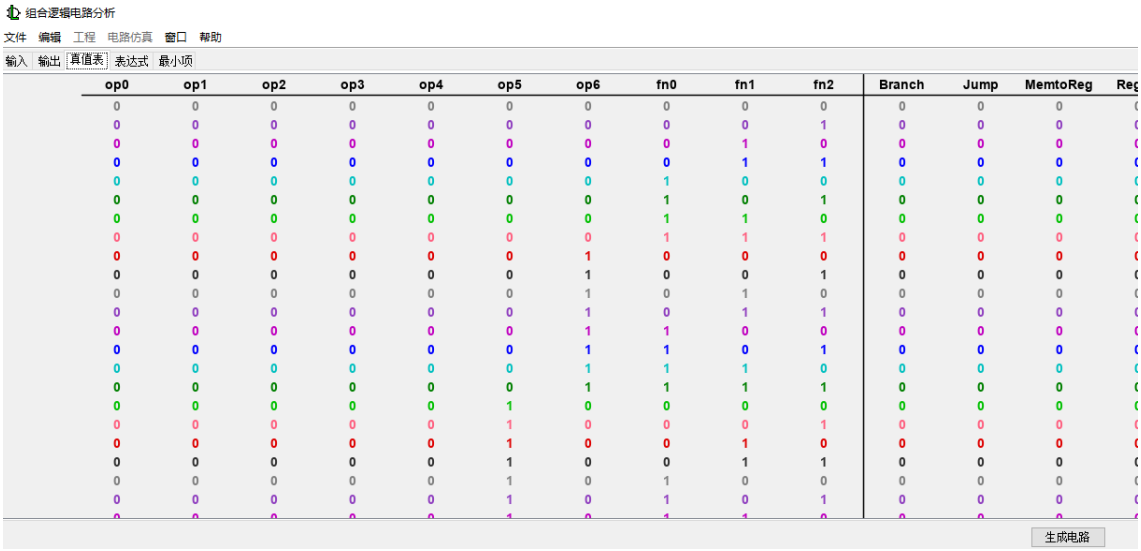


接下来，根据下面所示的控制信号功能表在控制信号子电路模块实现指令对应的控制信号生成电路，然后根据立即数扩展部件原理图，在立即数扩展器件子电路中对指令中的立即数按照需要扩展为 32 位立即数。注：在作答区文件中，我们给出了一个控制器的参考示例，但是请同学们注意该示例中的输入输出信号的高位均在左侧。

funct3 / op / 控制信号	000	010	011	110	无关	010	010	000	无关
	0110011	0110011	0110011	0010011	0110111	0000011	0100011	1100011	1101111
	add	slt	sltu	ori	lui	lw	sw	beq	jal
Branch	0	0	0	0	0	0	0	1	0
Jump	0	0	0	0	0	0	0	0	1
ALUASrc	0	0	0	0	×	0	0	0	1
ALUBSrc<1:0>	00	00	00	10	10	10	10	00	01
ALUctr<3:0>	0000（add）	0010（slt）	0011（sltu）	0110（or）	1111（srcB）	0000（add）	0000（add）	1000（sub）	0000（add）
MemtoReg	0	0	0	0	0	1	×	×	0
RegWr	1	1	1	1	1	1	0	0	1
MemWr	0	0	0	0	0	0	1	0	0
ExtOp<2:0>	×	×	×	000 immI	001 immU	000 immI	010 immS	011 immB	100 immJ



同学们注意，控制器设计涉及到多位输入和输出，这比我们之前涉及到的任何一个实验任务都要复杂，建议大家尝试使用 Logisim 中的真值表生成电路功能。在 Logisim 的菜单栏中找到“工程->分析组合逻辑电路->真值表”，可以事先在子电路画布上布好输入输出端口，也可以通过组合逻辑电路分析窗口上的“输入”、“输出”添加。真值表页面会根据输入输出端口的数量对应全部的真值组合，通过鼠标点击输出端口的值可以改变电路逻辑。当全部输出端口的值设定完毕后点击“生成电路”，使用默认选项生成即可在原有子电路中得到控制器的功能实现。提示：使用页面的“表达式”功能同样可以得到组合电路实现，同学们可以根据自己的情况自行选择。



最后，在指令解析测试子电路中，将指令的下地址逻辑、指令存储器、立即数扩展器件连接起来，使其能够产生正确的控制信号和指令跳转。