实验五

191098328 计算机科学与技术系 张世茂

191098328@smail.nju.edu.cn

2021-12-19

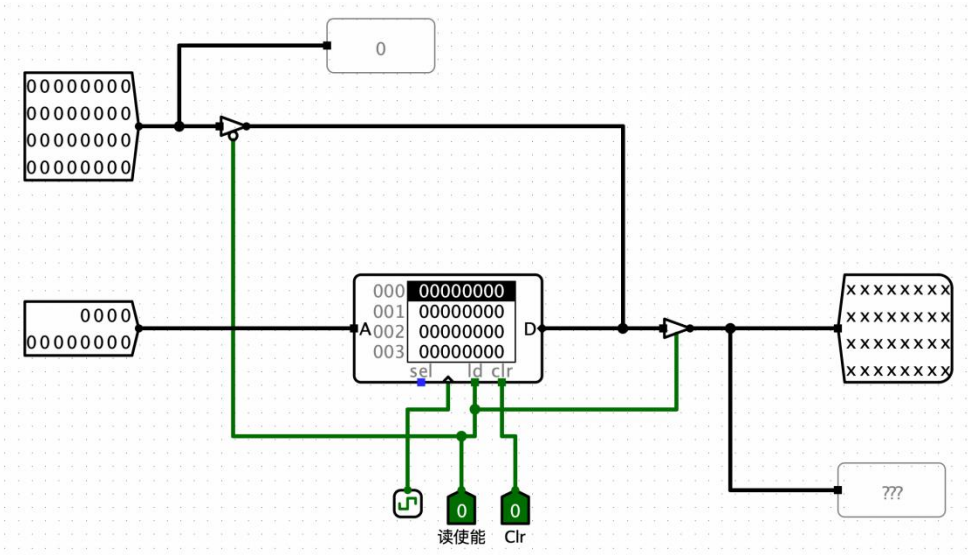
目录

1. RAM 组件存取

1.1 实验操作流程

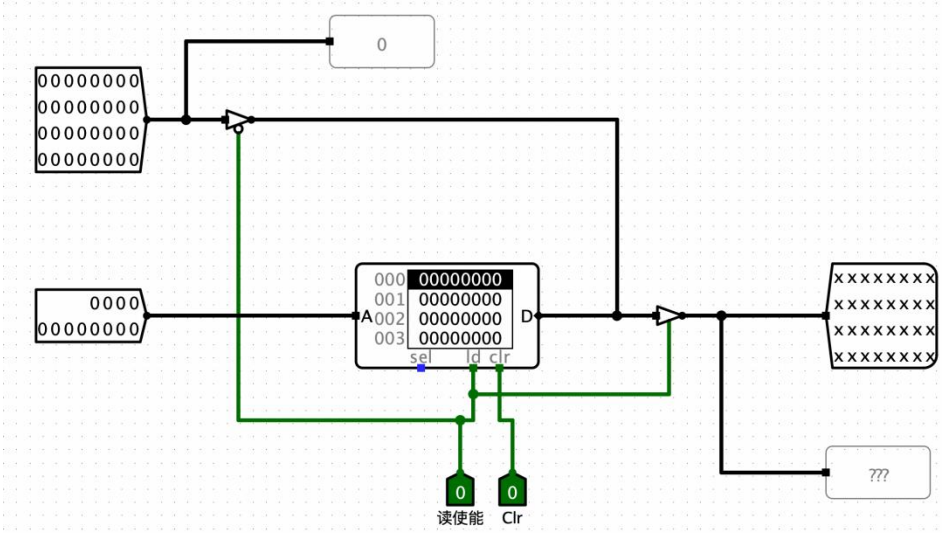
按照实验手册的要求，需要在三种 RAM 模式（同步模式、异步模式、输入输出分离同步模式）下进行数据的存取操作，其实现的电路原理图分别如下：

同步模式：



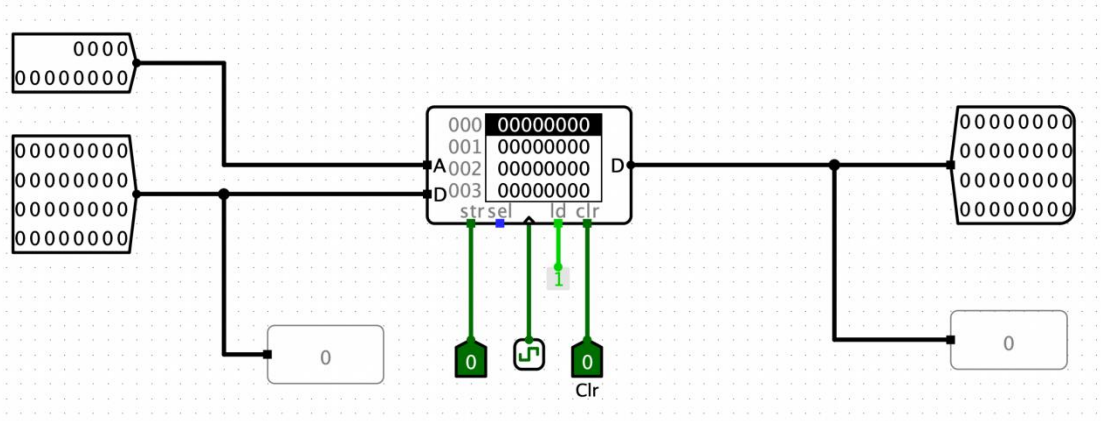
可以看到，在同步模式下，随机存储器 RAM 有一个时钟信号输入端，一个地址选择端 A，读口和写口共用一个端口 D，读使能和写使能共用一个使能端 ld。当 ld 端输入为 0 时，D 口输出当前地址对应的值，当 ld 端输入为 1 时，D 口将输入值存入当前地址对应的位置。由于输入输入共用一个端口，因此还需要通过三态门控制线路上的导通情况。

异步模式：



在异步模式下，相比同步模式除了 RAM 没有时钟信号输入端外其他基本一致。

输入输出分离同步模式：



在输入输出分离同步模式下，RAM 有一个写使能端 str、一个时钟信号输入端、一个读使能端 ld、一个清零端 clr、一个地址输入端 A、一对分离的读口和写口 D。电路中将读使能端输入设为常量 1。

在实现上述要求的电路后，就可以对电路进行进一步的仿真检验。

1.2 实验结果

首先分析电路的输入输出对应表，同步模式 RAM 输入输出对应表如下：

A	D	ld	clr
a	d	0	0
a	时钟上升沿来到后输出 RAM 中地址 a 开始存储的 32 位值	1	0
x	0	1	1
x	x	0	1

异步模式 RAM 输入输出对应表如下：

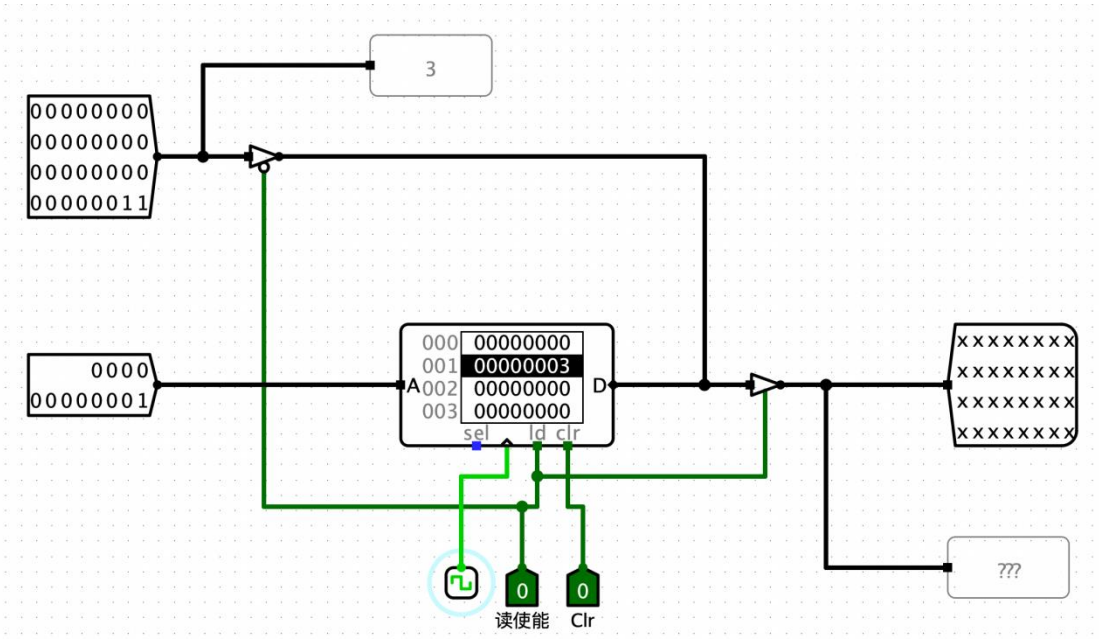
A	D	ld	clr
a	d	0	0
a	RAM 中地址 a 开始存储的 32 位值	1	0
x	0	1	1
x	x	0	1

输入输出分离同步模式的输入输出对应表如下：

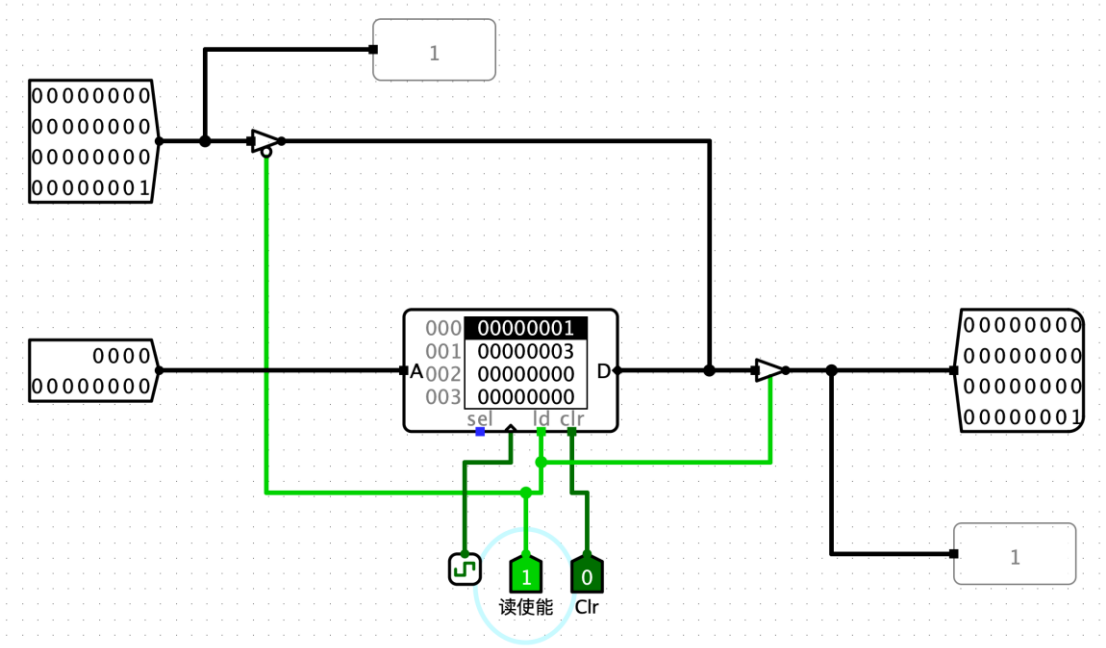
A	D_in	str	clr	D_out
x	x	x	1	0
a	d_in	0	0	RAM 中地址为 a 处开始存储的 32 位值
a	d_in	1	0	RAM 中地址为 a 处开始存储的 32 位值

即 RAM 在同步模式下且 clr 不为 1 时，写使能有效且在时钟上升沿时向 RAM 地址 a 处写入输入的 32 位数据，输出端不进行输出；在读使能有效时输出端输出 RAM 地址 a 处存储的 32 位数据。RAM 在异步模式下且 clr 不为 1 时，写使能有效时即向 RAM 地址为 a 处写入输入的 32 位数据，输出端不进行输出；在读使能有效时输出端输出 RAM 地址 a 处存储的 32 位数据。RAM 在输入输出分离同步模式下且 clr 不为 1 时，写使能有效且在时钟上升沿时向 RAM 地址 a 处写入输入的 32 位数据，在读使能有效时输出端输出 RAM 地址为 a 处存储的 32 位数据。若 clr 的值被置为 1，则将 RAM 中存储的值全部置 0。

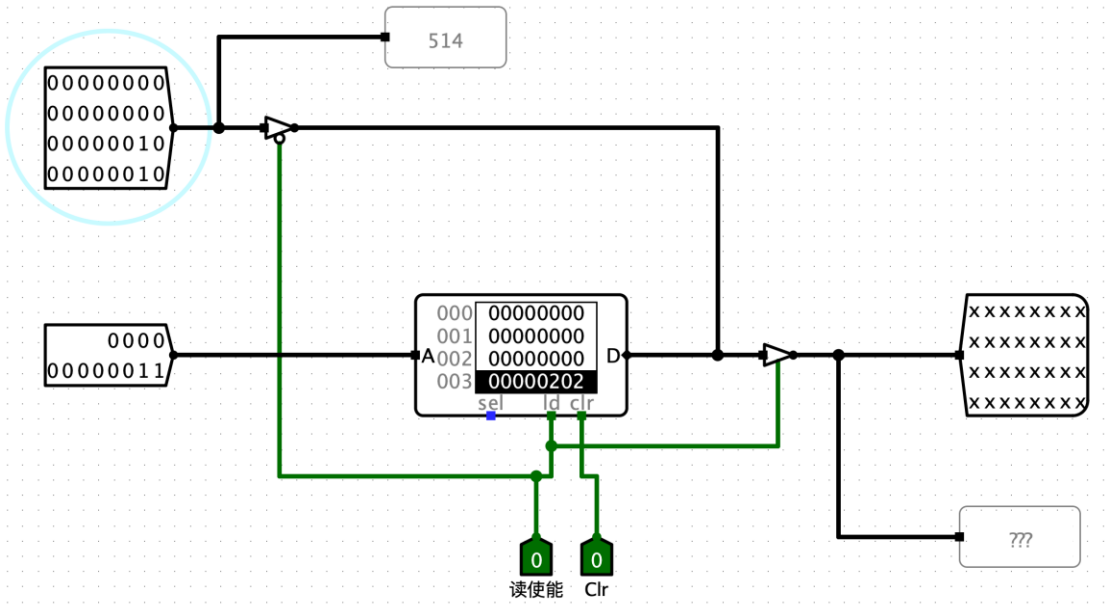
然后对电路进行仿真检验，即通过不同的输入验证加法运算结果是否正确。



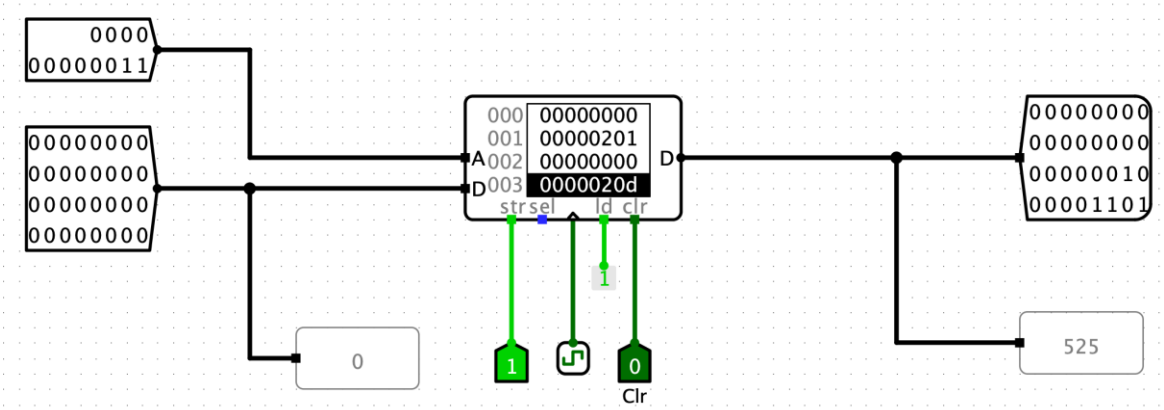
当同步模式中写使能被置为有效时，RAM 在时钟上升沿到来后在地址 0x001 处被存入值 0x00000003，输出端没有输出。



当同步模式中读使能被置为有效时，输出端输出 RAM 地址 0x000 存储的值 0x00000001。



当异步模式中写使能被置为有效时，RAM 在选择的地址 0x003 处被写入数据 0x00000202。



当输入输出分离同步模式中地址被置为 0x003 时，输出端输出 RAM 该地址处存储的数据 0x0000020d。写使能虽然有效但未到达时钟上升沿，所有不进行写入操作。

可以发现，在上面的样例中所设计的电路可以按照所预想的功能运行而输出正确的结果，一定程度上验证了所实现电路的正确性。

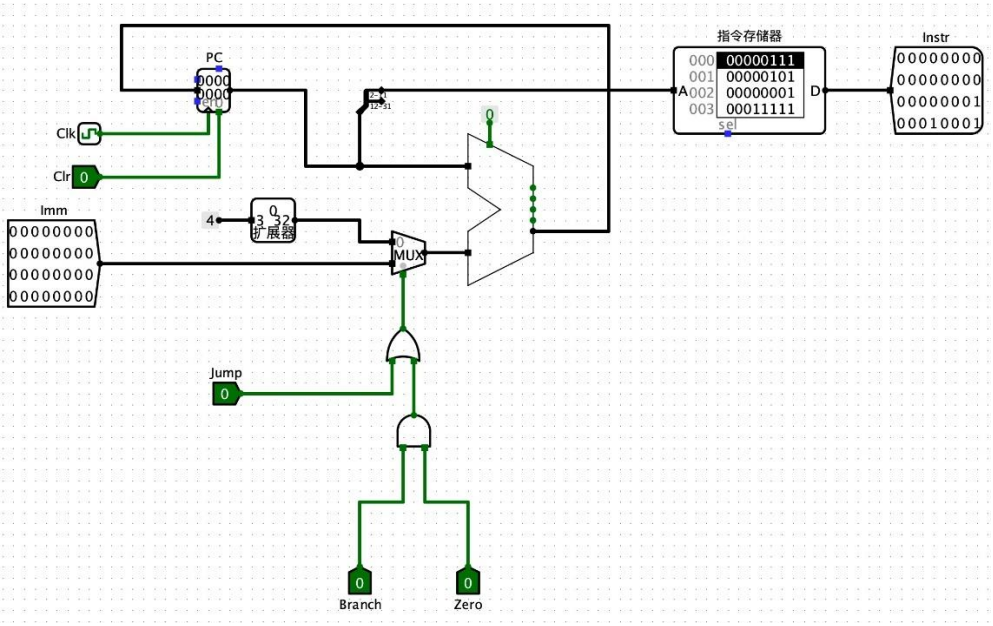
1.3 实验错误与原因分析

本阶段遇到的结果错误与实现困难主要是在于同步模式和异步模式下 RAM 具体的使用方法。一方面由于存在多处端口共用，因此需要花时间理解该器件的使用方法；另一方面因为涉及到读口写口共用一个端口，因此输入与输出会存在一段共享的线路，如果不进行特殊处理就会导致电路的混乱和错误，这一点需要通过在输入输出的线路上分别添加三态门来解决。

2. 读取任意地址中 RISC-V 指令

2.1 实验操作流程

要实现读取存储器中任意地址存储的 RISC-V 指令，则要实现 PC 的指令下地址逻辑，以及根据 PC 在地址存储器中读取对应地址的指令的电路逻辑，其电路原理图如下所示：



可以看到，左边是 PC 的下地址逻辑实现，可以根据对应控制信号选择 PC 是加 4 变化还是进行跳转，将对应的值传入加法器进行运算后传回 PC，在时钟的上升沿被存入 PC；右边是指令的读取逻辑，由于指令存储器只有 10 位地址，PC 有 32 位，因此将 PC 通过分线器取 10 位作为地址读取存在指令存储器对应地址的指令。这里需要特别注意的是，由于实际中的存储器是采用按字节编址，而这里的存储器采用每四个字节进行编址，因此现实中 PC 为了取到下一条指令进行的+4 运算在这里只需+1，而为了尽量在观感上模拟现实中的情况，在电路中仍采用+4 运算，但在取地址时取 PC 的第 2-11 位值，这样就可以模拟正确的下指令逻辑。

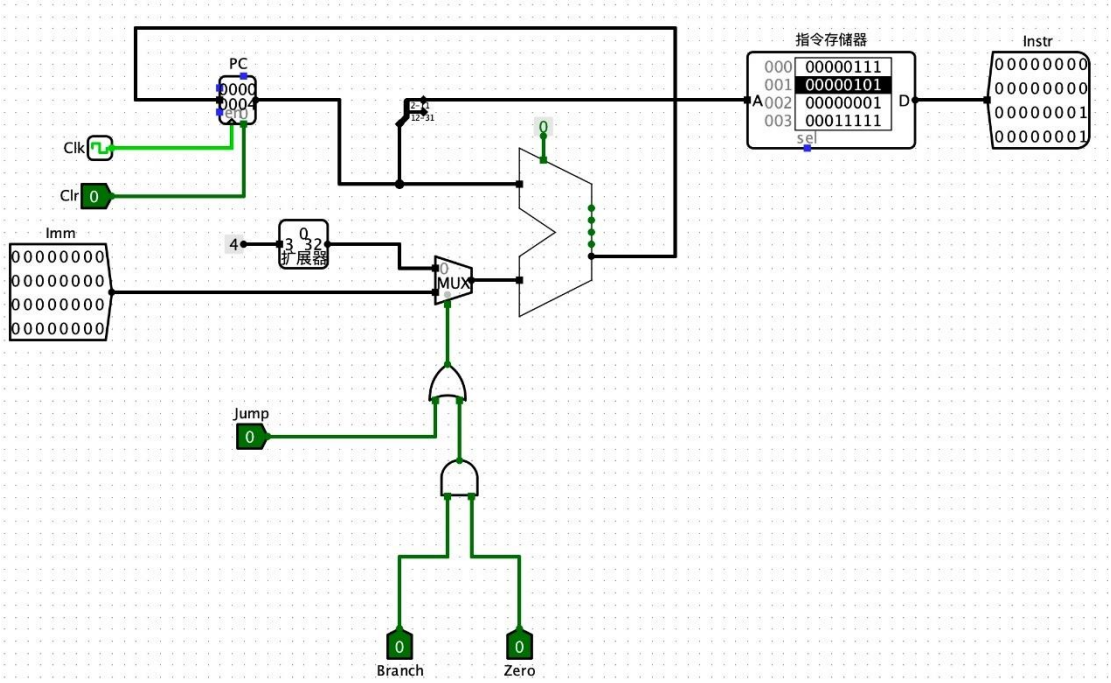
实现电路后，对电路进行进一步的观察和检验。

2.2 实验结果

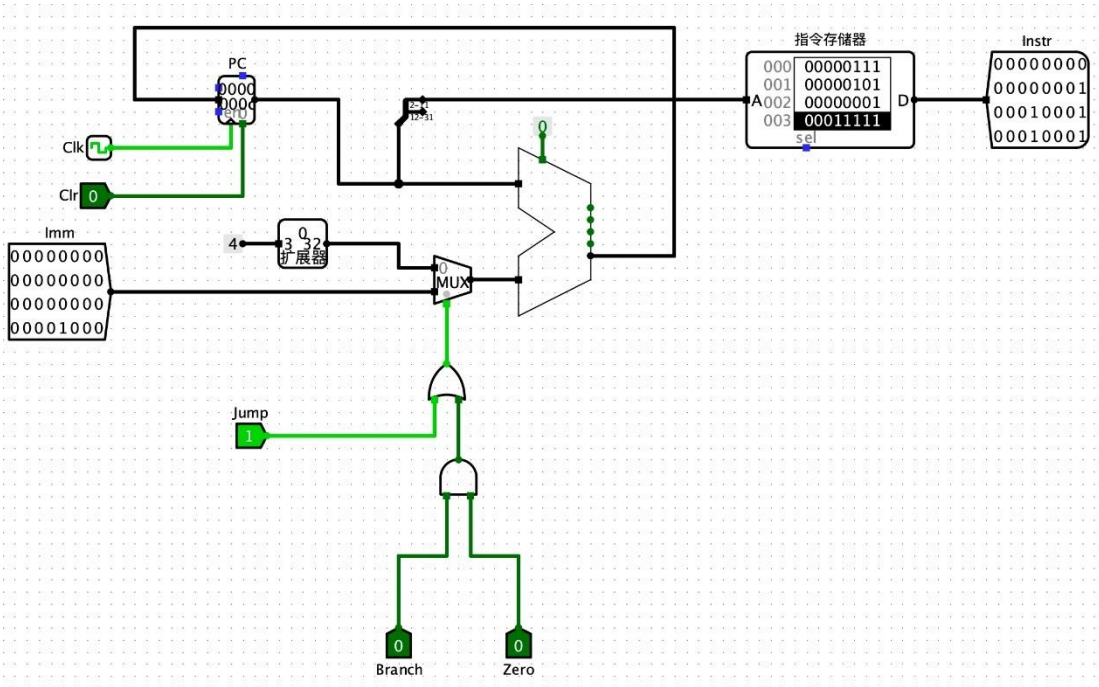
该电路的输入输出对应表如下所示：

Imm	Jump	Zero	Branch	PC	Instr
x	0	0	0	PC+4	地址 PC[11..2]处存储的指令
x	0	0	1	PC+4	地址 PC[11..2]处存储的指令
Din	x	1	1	PC+Din	地址 PC[11..2]处存储的指令
Din	1	x	x	PC+Din	地址 PC[11..2]处存储的指令

然后选择合适的样例对电路进行仿真检验。



PC 初始值为 0，此时 Jump、Branch、Zero 都为 0，在时钟上升沿到来后，PC 的值变为 4，取出的指令为存储器中地址 0x001 处存储的指令 0x00000101。



在上面的基础上，此时 Jump 的值被置为 1，Imm 的值为 8，在时钟上升沿到来后，新的 PC 值变为 PC+8=12，此时取出的值为存储器中地址 0x003 处存储的指令 0x00011111。

通过上述的仿真检验，发现电路的输出符合实际电路中的下地址逻辑功能，这个

结果可以在一定程度上验证所实现电路的正确性。

2.3 实验错误与原因分析

这部分实验得到电路原理图后依据电路原理图进行实现即可，但实现过程中要尤其注意 PC 变化与存储器地址编码的关系等细节，若直接取 PC 的低位，则存储器的取数地址会越过中间的指令而到达 4 条指令之后，其余的在这部分实验过程中没有出现什么特别的错误。

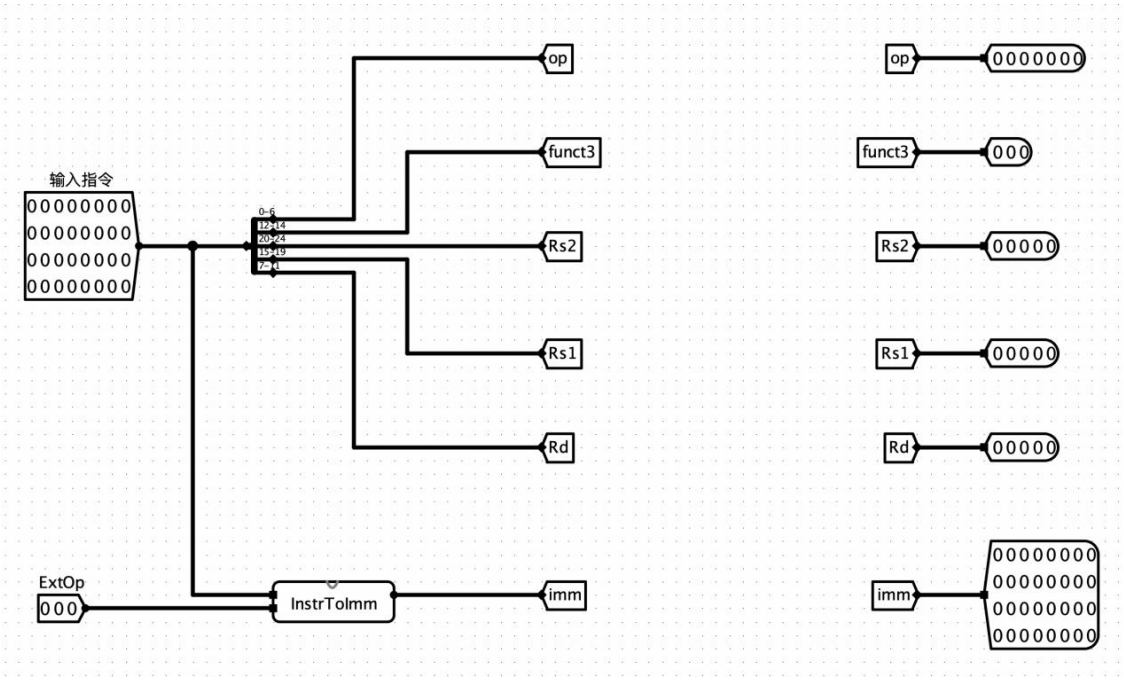
3. RISC-V 指令解析

3.1 实验操作流程

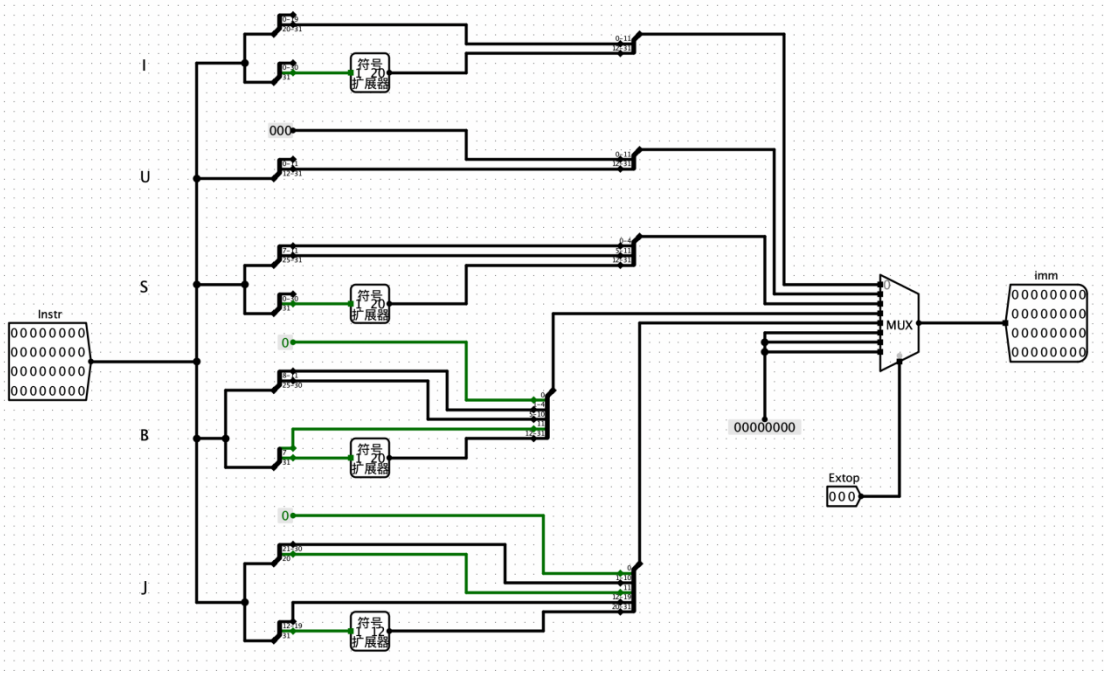
要构建一个 RISC-V 指令的解析电路，首先要了解 RISC-V 不同类型指令的编码格式，具体如下：

	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

依据上图中 RISC-V 的编码格式，将立即数的提取和其余控制信号的提取进行分别处理，指令解析电路具体的电路原理图如下：



可以看到在上图中利用分线器将指令中不同位对应的控制信号提取出来，下面的 ExtOp 输入端对应着不同的指令类型，是由控制器电路对上面得到的控制信号进行解析后得出的，将该信号与指令输入子电路 InstrToImm 来获得指令中的立即数，子电路 InstrToImm 实现的电路原理图如下：



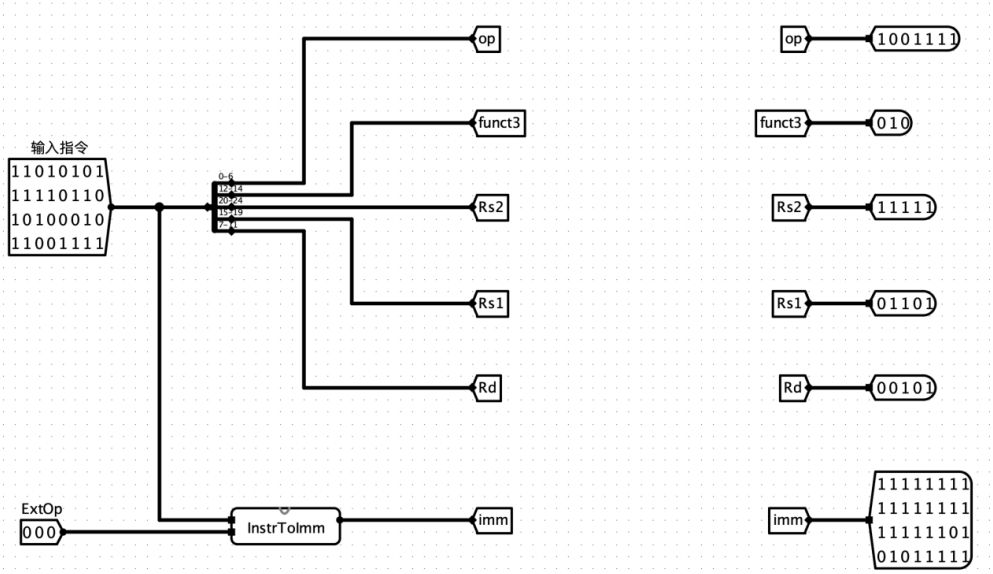
在电路实现完成后查看实验结果，对电路进行进一步的观察与检验。

3.2 实验结果

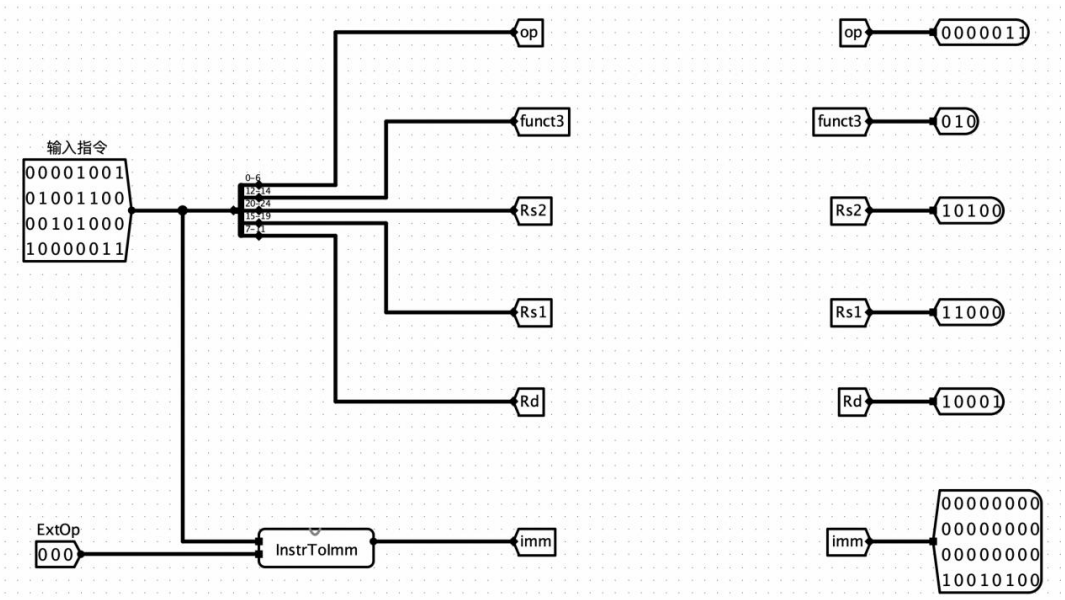
下面列出的是 RISC-V 指令解析器对应的指令解析格式：

	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

然后选择合适的样例对 RISC-V 指令解析器进行仿真检验。



上图的指令输入情况下，解析出的 op=1001111，funct3=010，Rs1=01101，Rs2=11111，Rd=00101，imm=0xfffffd5f。



上图的指令输入情况下，解析出的 op=0000011，funct3=010，Rs1=11000，Rs2=10100，Rd=10001，imm=0x00000094。

经过上面的检验，任意挑选的样例进行仿真检验所计算得出的结果都符合预期，可以在一定程度上验证所实现的 RISC-V 指令解析电路的正确性。

3.3 实验错误与原因分析

该部分电路根据已有的实验手册中给出的指令格式图可以得到不同指令格式下指令的构成方式，再通过具体电路对相应的位进行提取和处理即可得到目标电路，实验进行的过程中没有出现什么特别的错误。

4. 控制器

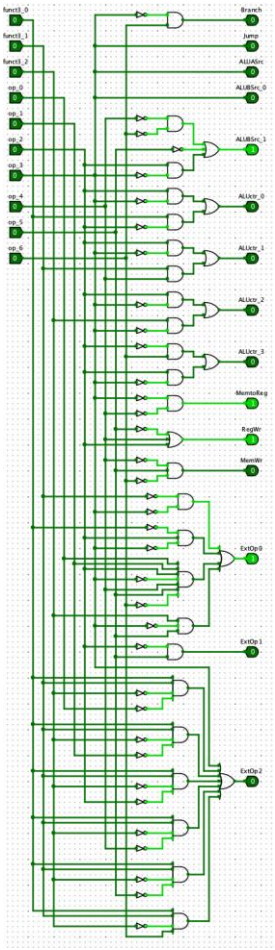
4.1 实验操作流程

要实现一个 RISC-V 的控制器电路，根据指令生成对应的控制信号，其指令控制信号对应值表如下所示：

funct3	000	010	011	110	无关	010	010	000	无关
op	0110011	0110011	0110011	0010011	0110111	0000011	0100011	1100011	1101111
控制信号	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

有了对应的转换表，可以发现输入输出都有较多位，自行通过组合逻辑的运算化简来生成最终的逻辑表达式会极为繁琐，即便如此最终的逻辑表达式也是较为复杂的。由于这一部分电路已经有了明确的输入输出对应表，因此考虑直接使用 Logisim 的组合电路分析功能来填充真值表，进而自动生成最终的组合逻辑电路是一种较为直接便捷的方法。

生成的电路原理图如下：



可以发现，即使是 op 中一些输入为定值，即在这部分电路中与输出无关的位也参与了组合逻辑，这是组合电路分析的过程中自动利用冗余项对最终的逻辑表达式进行了化简，若是人工化简工作量无疑是巨大的，这就是本部分实验采用自动分析生成的原因和优势所在。

在电路生成完成后查看实验结果，对电路进行进一步的观察与检验。

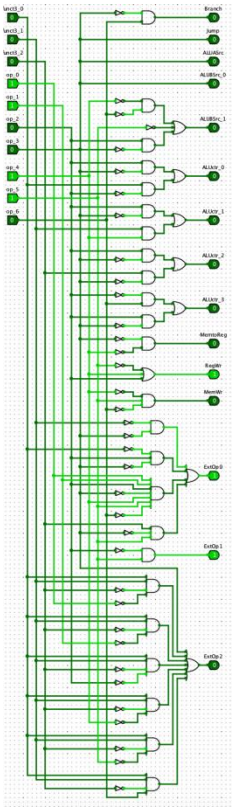
4.2 实验结果

下面列出所实现的 RISC-V 控制器电路的输入输出对应表：

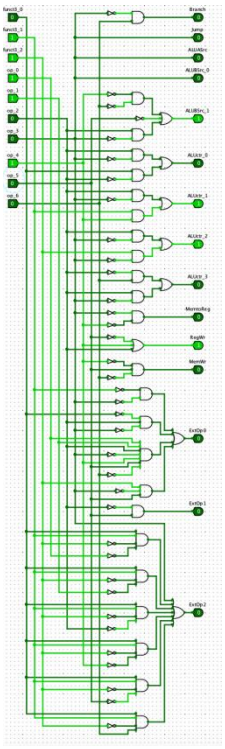
funct3	000	010	011	110	无关	010	010	000	无关
op	0110011	0110011	0110011	0010011	0110111	0000011	0100011	1100011	1101111
控制信号	add	slt	sltu	ori	lui	lw	sw	beq	jal
Branch	0	0	0	0	0	0	0	1	0
Jump	0	0	0	0	0	0	0	0	1
ALUASrc	0	0	0	0	×	0	0	0	1
ALUBSrc<1:0>	00	00	00	10	10	10	10	00	01

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

然后挑取合适的输入样例对所实现的电路进行仿真检验，通过将上表中有效情况下的 funct3 和 op 进行输入后观察是否可以得到预期的控制信号值即可。



当 funct3=000，op=0110011 时，ExtOp 的值无要求，其余的控制信号中除 RegWr=1 外全部为 0。



当 funct3=110，op=0010011 时，有 ALUctr=0110，ALUBSrc=10，ExtOp=000，Branch=Jump=ALUASrc=MemtoReg=MemWr=0。

经过上述仿真检验，任意挑选样例进行仿真检验所计算得出的结果都符合预期，可以在一定程度上验证所实现电路的正确性。

4.3 实验错误与原因分析

这一阶段的实验采用 Logisim 组合电路分析功能自动生成，主要的易错点在于写真值表时由于输入输出项较多因此比较容易出错，在细心保证真值表输入正确的基础上本阶段实验没有遇到什么错误。

5. 思考题

1.RAM 存储空间如何扩展？如何实现读写一个字节或一个字的数据？

在地址编码方式不变的基础上将 RAM 中的地址编码位数进行扩展后即可实现 RAM 存储空间的扩展。

要实现直接读写一个字节或一个字的数据，应当改变 RAM 元件的编码方式，若想一次读写一个字节，则应当改变 logisim 中 RAM 元件的数据位宽为 8，则元件变为按字节编址，每次读出或写入一个字节的数据；同样若要一次读写一个字，则将数据位宽改为字长，每次就会读出或写入一个字的数据。

2.读取指令实现过程中，除了使用加法器实现地址计算外，还可以使用什么方法来实现？

a.可以通过减法器进行运算，将原先的相对地址位移量取负数的补码即可；

b.可以通过将加数通过分线器分成单独的一位值，然后利用基础的与门模拟串行进位加法运算得到结果；

c.可以根据先行进位逻辑和一位多输入与或门实现地址计算。

6. 实验总结

本次实验对 RISC-V 指令相关的功能组件进行了实现，对下指令逻辑、指令译码等功能组件的运行机制和实现原理有了更清晰直观的认识。同时，这次实验中实现的功能电路是实现一个单周期 CPU 的基础，会在之后的实验中作为实现整体通路的子电路使用。