

实验 5：单周期处理器的控制器设计实验

一、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、理解指令类型与指令格式之间的关系，掌握取指部件、指令解析和立即数扩展器的设计方法。
- 3、理解每条目标指令的功能和数据通路，掌握单周期处理器的控制器设计方法。

二、实验环境

Logisim-ITA V2.16.1.0。

三、实验内容

- 1、利用 Logisim 中的 RAM 组件进行数据读写操作实验。Logisim 中 RAM 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（如图 1 所示）；否则，使用同一数据端口连接数据总线。注意：当设置数据位宽为 32 位时，采用按字编址方式（32 位），而不是采用按字节编址方式。

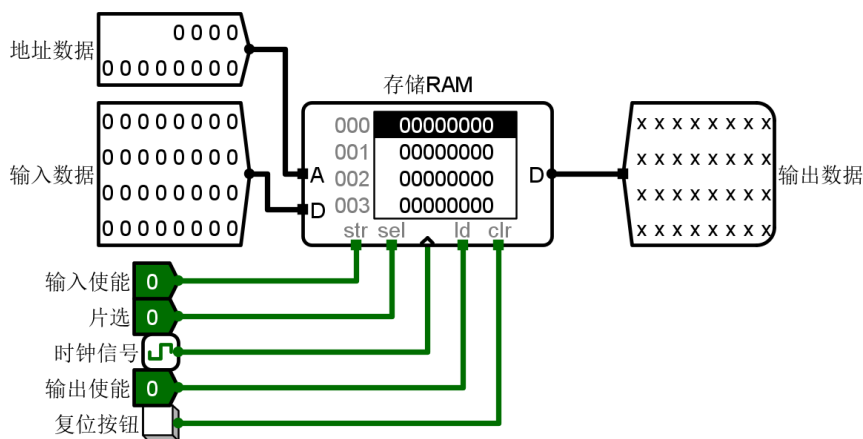
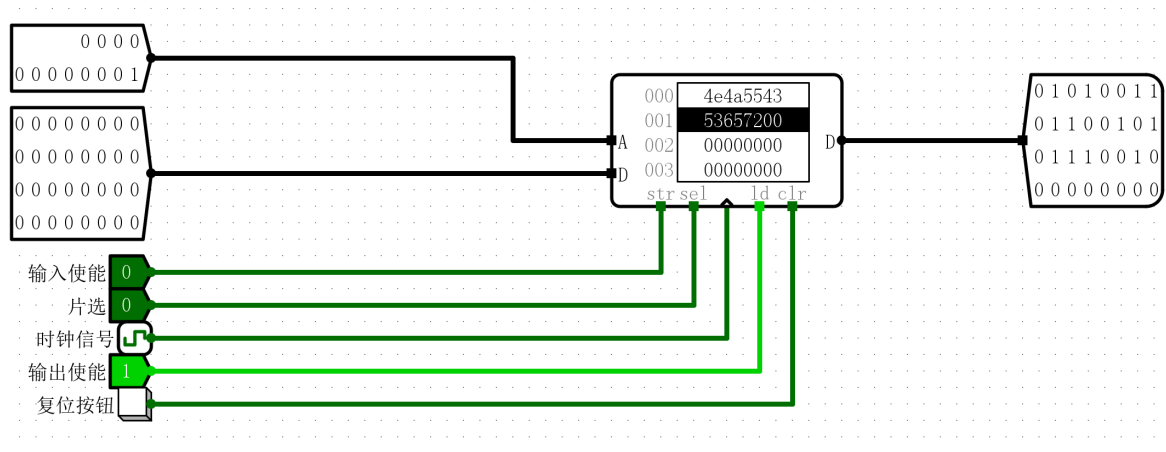
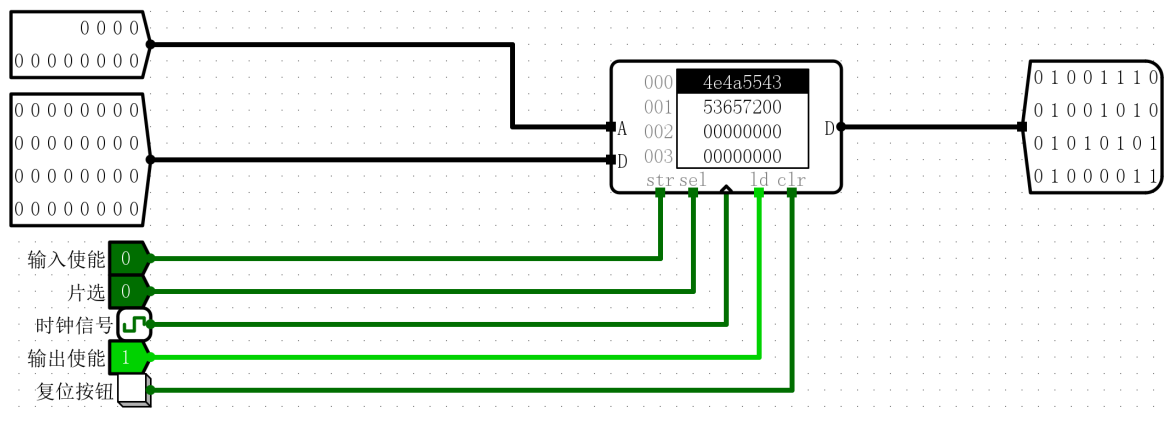


图 1 采用分离加载和存储模式的 32 位数据读取实验图

实验要求 RAM 组件的地址位宽设置为 12 位，数据接口模式设置为分离的加载和存储引脚模式。实验过程与验证步骤如下：

- (1) 设置数据位宽为 32 位，即可访问空间大小为 16KB；连接必要的输入输出信号并选择合适的控制信号；从 0 地址处开始顺序写入以下两个 32 位的十六进制数据：0x4E4A5543、0x53657200；然后再读出所存储的数据。



- (2) 设置数据位宽为 8 位，即可访问空间大小为 4KB；将输出数据端口连接到如图 2 所示的文本终端 TTY；从 0 地址开始顺序写入以下八个字节的十六进制数据：4E4A554353657200；然后按字节为单位读出并输出到文本终端 TTY，观察显示的内容。

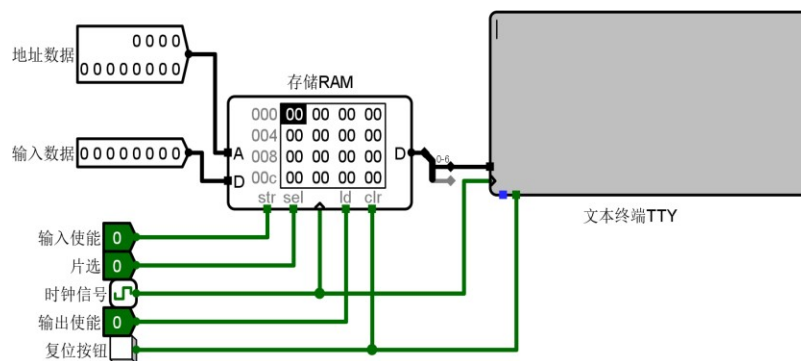
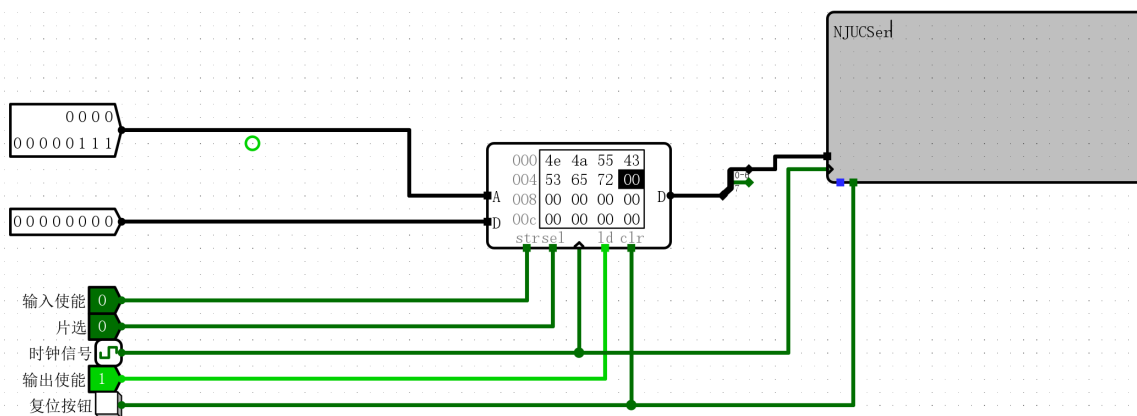


图 2 采用分离加载和存储模式的 8 位数据读取实验图





- (3) Logisim 中 RAM 和 ROM 组件的数据输入还可以采用 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，则弹出菜单框（如图 3 所示），选中“编辑存储内容”，则打开 Logisim 十六进制编辑器（如图 4 所示），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，可点击保存按钮，把输入的数据保存到数据镜像文件(image)中。可在鼠标右键菜单中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。要求：分别把步骤（1）和（2）的输入数据保存到数据镜像文件中，观察数据镜像文件格式，说明设置不同数据位宽对文件格式的影响。



图 3 存储器组件鼠标右键菜单

图 4 Logisim 十六进制编辑器

 1.image	2021/6/15 20:24	IMAGE 文件	1 KB
 2.image	2021/6/15 20:25	IMAGE 文件	1 KB

v2.0 raw
4e4a5543 53657200

32位时，文件的格式为此，由于是8位16进制等同于32进制

v2.0 raw
4e 4a 55 43 53 65 72 00

8位时，文件格式为此，2位16进制数

2、取指令部件设计实验。已知 RISC-V 指令格式如图 5 所示。

	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	

图 5 RISC-V 指令编码格式

根据如图 6 和图 7 所示的单周期处理器取指令部件原理图，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 中的 ROM 组件实现，要求指令存储器容量为 16KB（按字编址，即数据位宽 32 位，地址位宽 12 位，即 A[11:0]），指令字长为 32 位。

提示：当 Logisim 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 架构中的 4 个单元。因而存取指令存储器时，32 位指令地址 PC[31:0] 中，PC[13:2]=A[11:0]，其余的位（高 18 位和最低两位）皆无关。

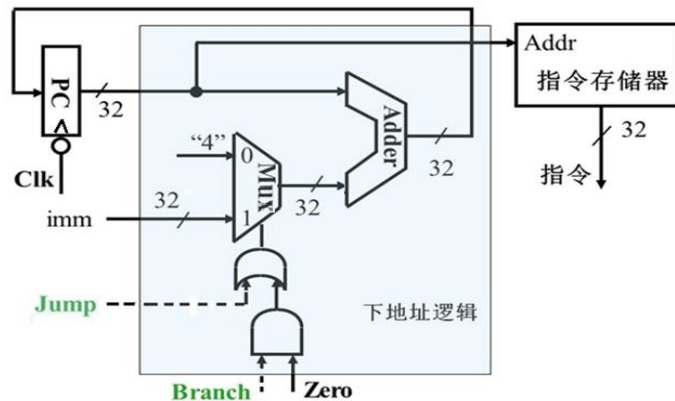


图 6 取指令部件原理图

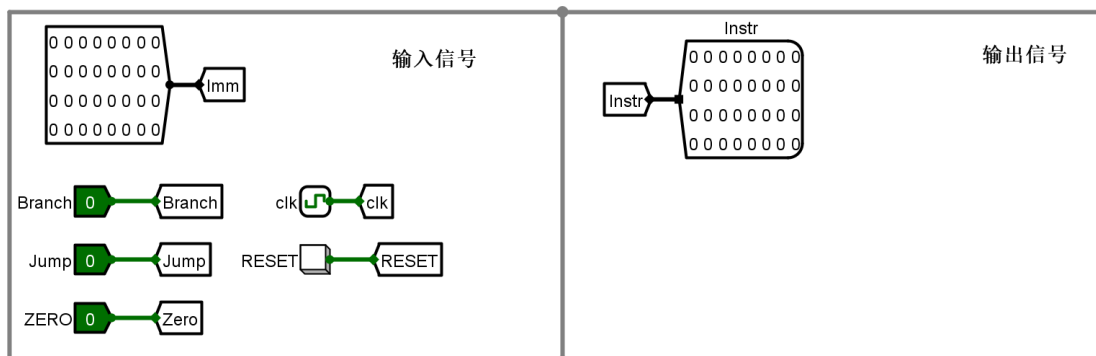
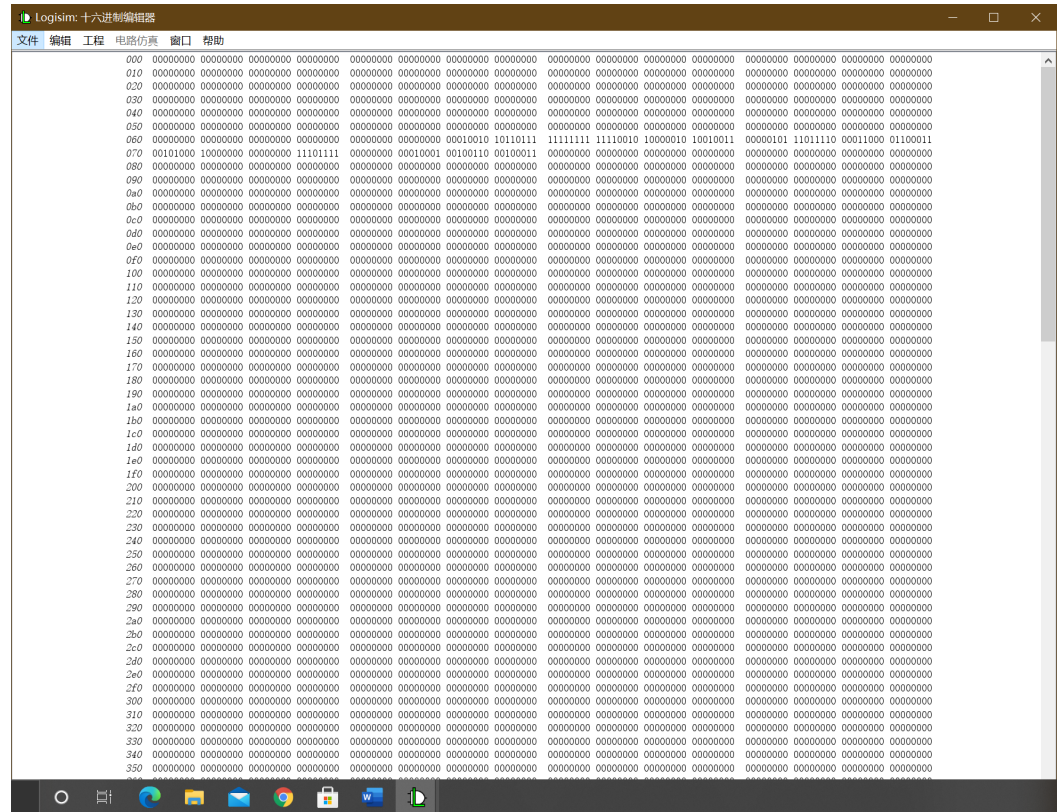


图 7 取指令部件 IFU 的引脚示意图

实验过程和验证步骤如下。

（1）从指令存储器的地址第 100 单元开始，依次写入以下 5 条指令：

0000 0000 0000 0000 0001 00101 0110111 （对应汇编指令 lui x5, 1）
 1111 1111 1111 00101 000 00101 0010011 （对应汇编指令 addi x5, x5, -1）
 0000010 11101 11100 001 10000 1100011 （对应汇编指令 bne x28,x29,label1）
 0010 1000 1000 0000 0000 00001 1101111 （对应汇编指令 jal ra, printf）
 0000000 00001 00010 010 01100 0100011 （对应汇编指令 sw ra,12(sp)）



- (2) 分别读出第(1)步写入的 5 条指令，并将读出的指令分解出 opcode、rd、funct3、rs1、rs2 和 funct7 字段（如图 8 所示）。
- (3) 设计如图 9 所示的立即数扩展器，对指令中的立即数字段进行扩展生成 32 位立即数。

指令解析

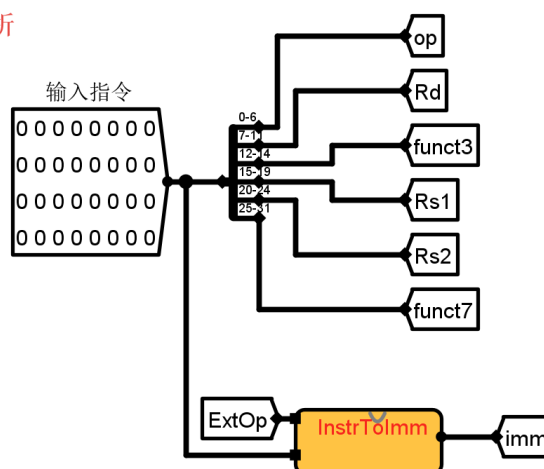


图 8 指令各字段分解原理图

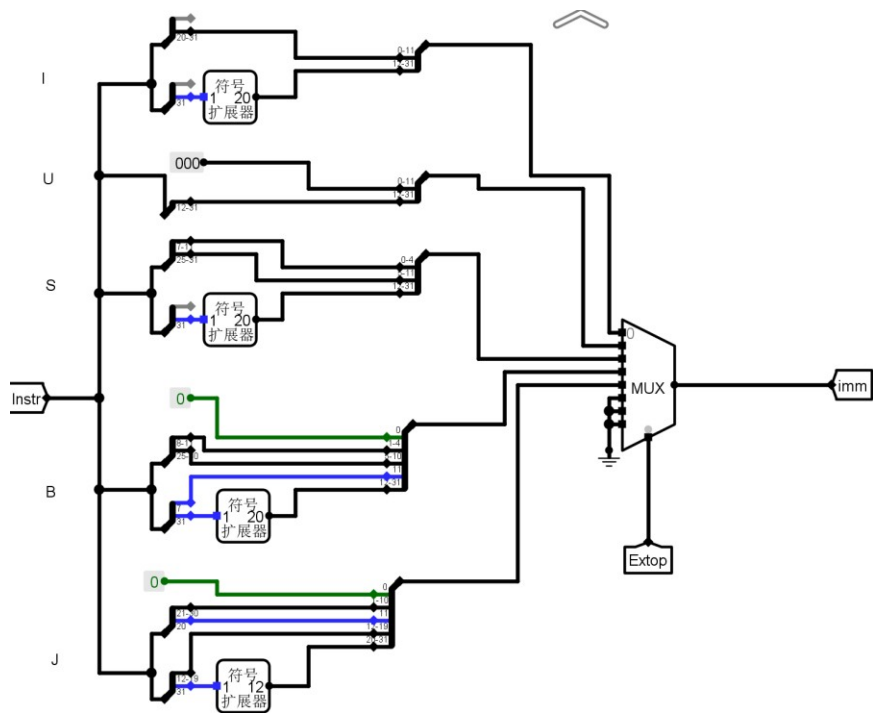
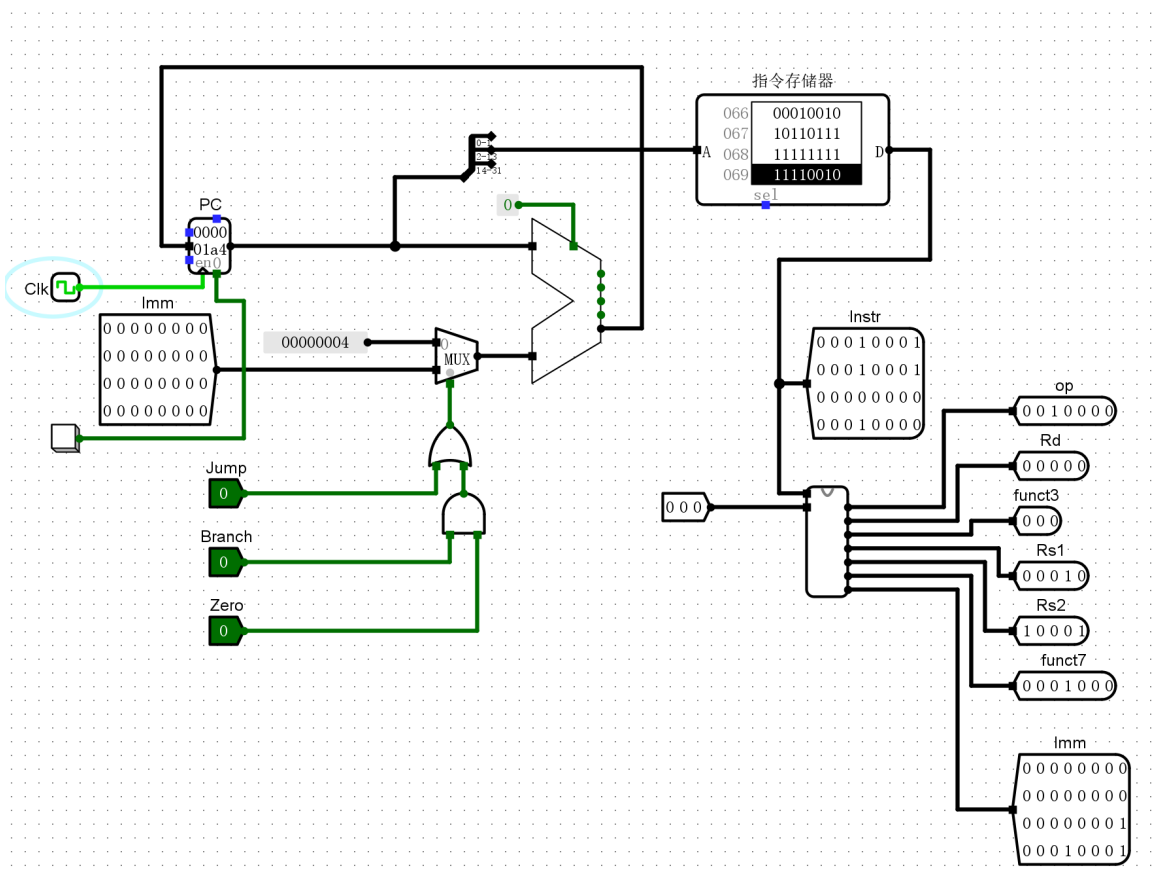


图9 立即数扩展原理图



3、图 10 给出了实现表 1 中 9 条指令的单周期数据通路，根据表 1 所示的 RISC-V 指令与控制信号之间的关系表，设计实现该单周期处理器的控制电路，并验证电路的正确性。

表 1 RISC-V 指令与控制信号关系

func3 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

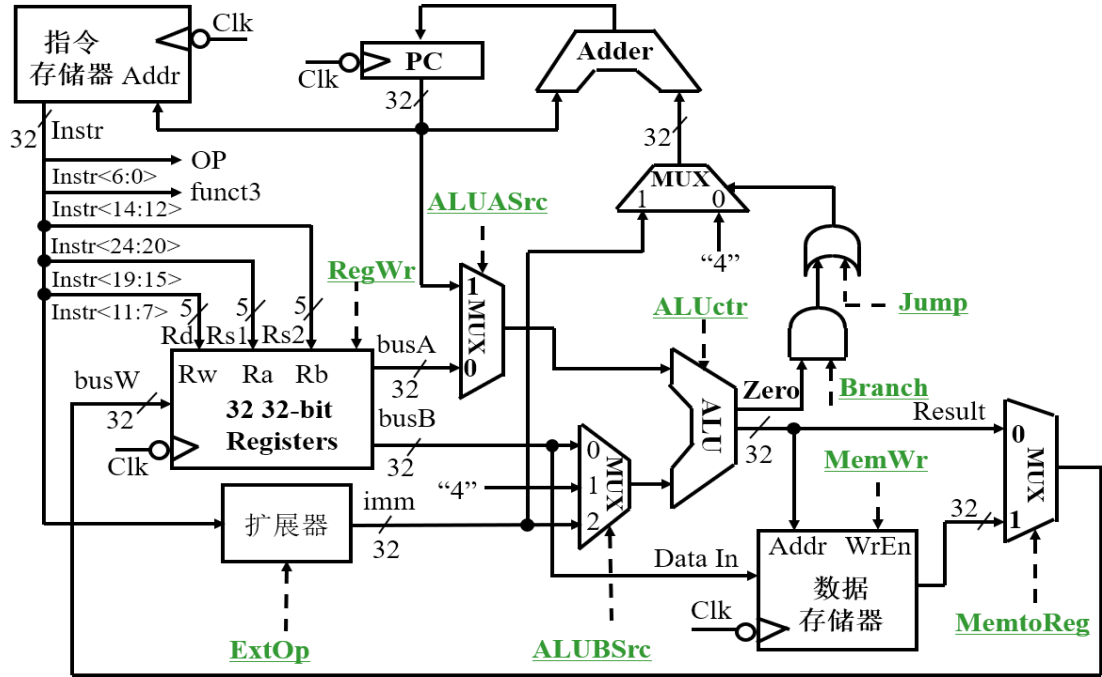


图 10 RISC-V 单周期数据路通

四、思考题

- 1、图 10 所示的数据通路是否可以执行 `auipc` 指令？增加 `auipc` 指令后，相应的控制器电路需要进行哪些修改？

可以，`ALUASrc`为1，`imm`经过0扩展，`ALUctr`采用加并且`MemtoReg`为0后存到寄存器组中

- 2、如果不采用如图 10 所示的统一扩展器进行 5 种立即数扩展，而是分别用 5 个扩展器进行立即数扩展，则图 10 所示数据通路应该如何修改？

扩展器处增加多路选择器，将`ExtOp`改为3位后接到MUX处，MUX选择段位3位，输入端后三位不设置