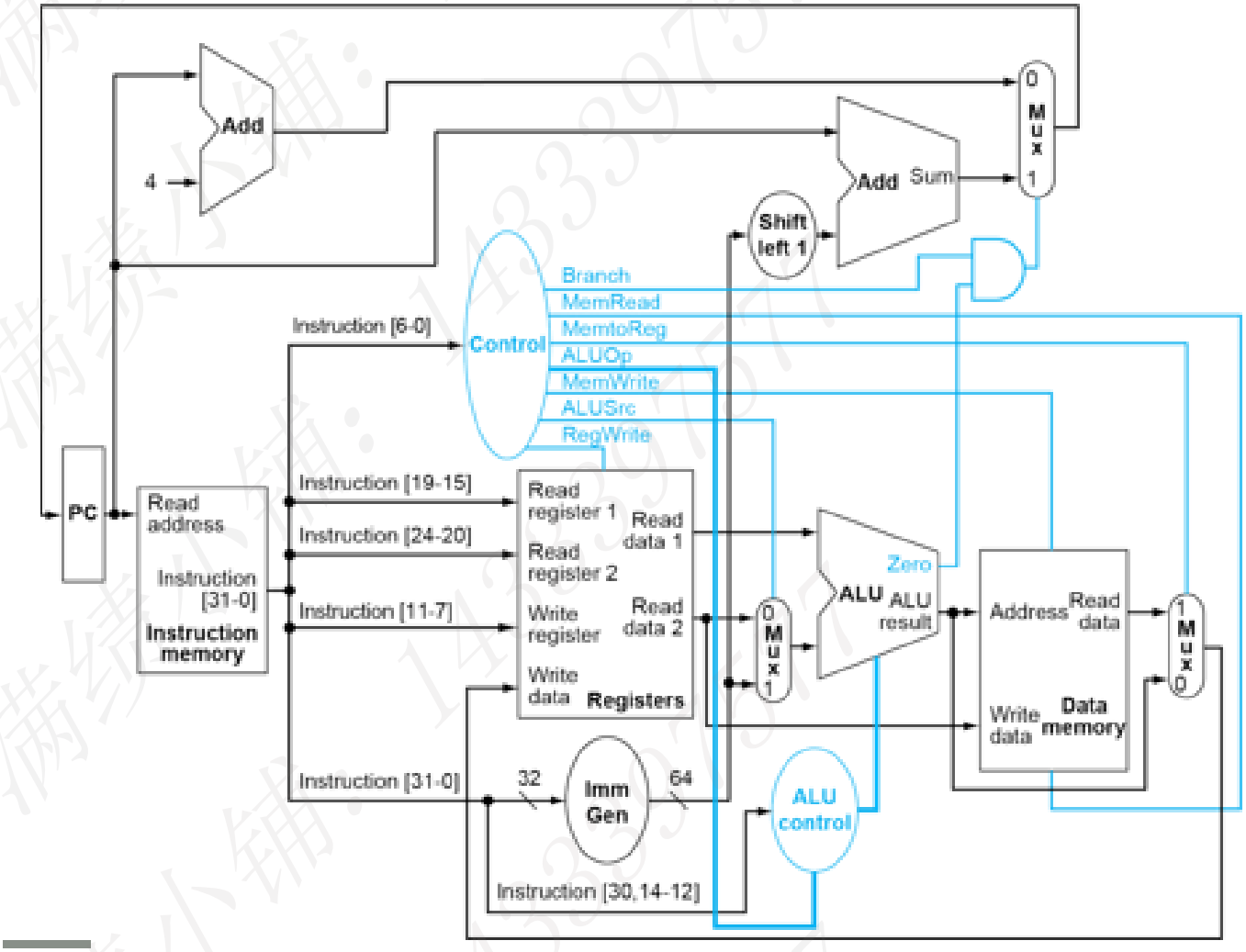
一、已知单周期处理器结构图如下：



1. 在下面的表格中列出各指令的主控制信号状态：

参考：

Ins	RegWrite	ALUSrc	MemWrite	ALUOp（2 bits）	MemtoReg	MemRead	Branch
Add	1	0	0	1 0	0	0	0
sub	1	0	0	1 0	0	0	0
and	1	0	0	1 0	0	0	0
or	1	0	0	1 0	0	0	0
nor	1	0	0	1 0	0	0	0
slt	1	0	0	1 0	0	0	0
Ld	1	1	0	0 0	1	1	0
sd	0	1	1	0 0	x	0	0
beq	0	0	0	0 1	x	0	1
Addi	1	1	0	0 0	0	0	0

2. 在下面的表格中列出相关部件在各条指令的数据通路中的作用（**请先看后面的说明**）：

参考：综合考虑，ImmGen 和 BAAdder 应属于分支指令的数据通路。

Ins	+4Adder	IM	RegFile	ImmGen	BAAdder	ALU	DM
Add	有用功	有用功	有用功	无用功	无用功	有用功	不做功
sub	有用功	有用功	有用功	无用功	无用功	有用功	不做功
and	有用功	有用功	有用功	无用功	无用功	有用功	不做功
or	有用功	有用功	有用功	无用功	无用功	有用功	不做功
nor	有用功	有用功	有用功	无用功	无用功	有用功	不做功
slt	有用功	有用功	有用功	无用功	无用功	有用功	不做功
Ld	有用功	有用功	有用功	有用功	无用功	有用功	有用功
sd	有用功	有用功	有用功	有用功	无用功	有用功	有用功
Beq（条件不成立）	有用功	有用功	有用功	无用功	无用功	有用功	不做功
Beq（条件成立）	无用功	有用功	有用功	有用功	有用功	有用功	不做功
Addi	有用功	有用功	有用功	有用功	无用功	有用功	不做功

说明：+4Adder 表示左上方的 PC+4 加法器，IM 表示指令存储器，RegFile 表示寄存器堆，ImmGen 表示常数生成器，BAAdder 表示右上方的分支目标地址加法器，DM 表示数据存储器。

类似地，也可以列表说明剩下的 PC、移位寄存器以及 3 个多路选择器在每条指令执行过程中的作用。

所有做有用功的部件构成此指令的**数据通路**，或**关键路径**。

二、前述的单周期处理器结构图做适当的修改或增加一些功能部件（连带着可能需要调整或增加相应的控制信号）就可以执行更多的指令。

说明为了达到以下目的，需要修改结构图吗？如何修改？

1. 支持 lui 指令；

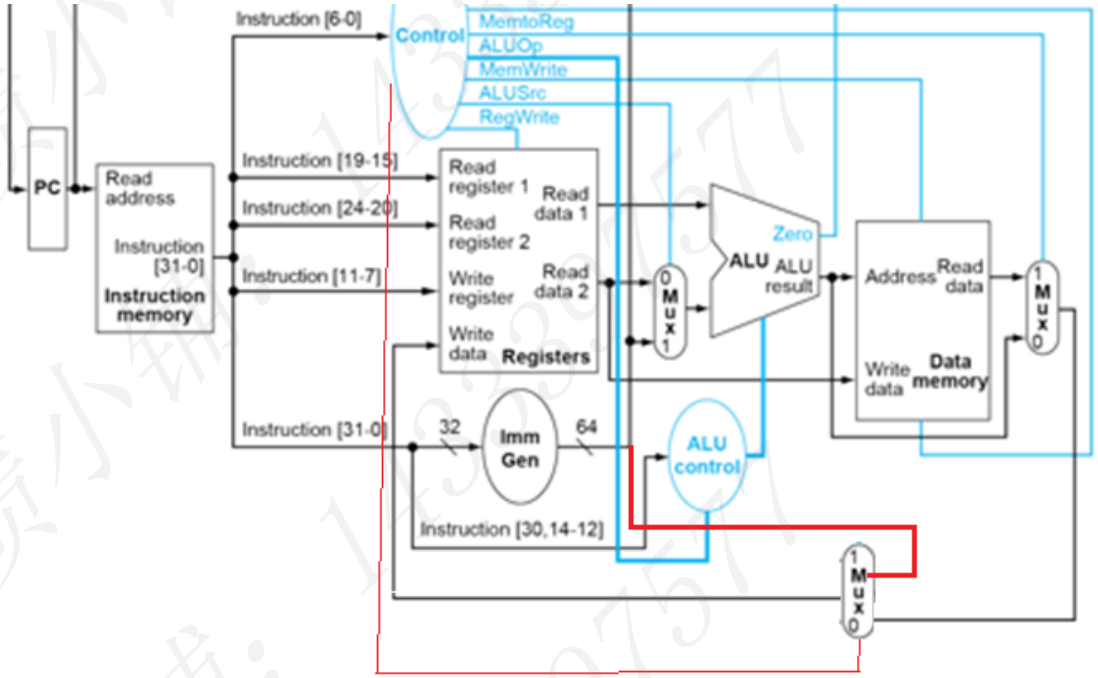
参考：指令 lui 的汇编格式和机器格式如下

立即数加载指令：lui rd, imm

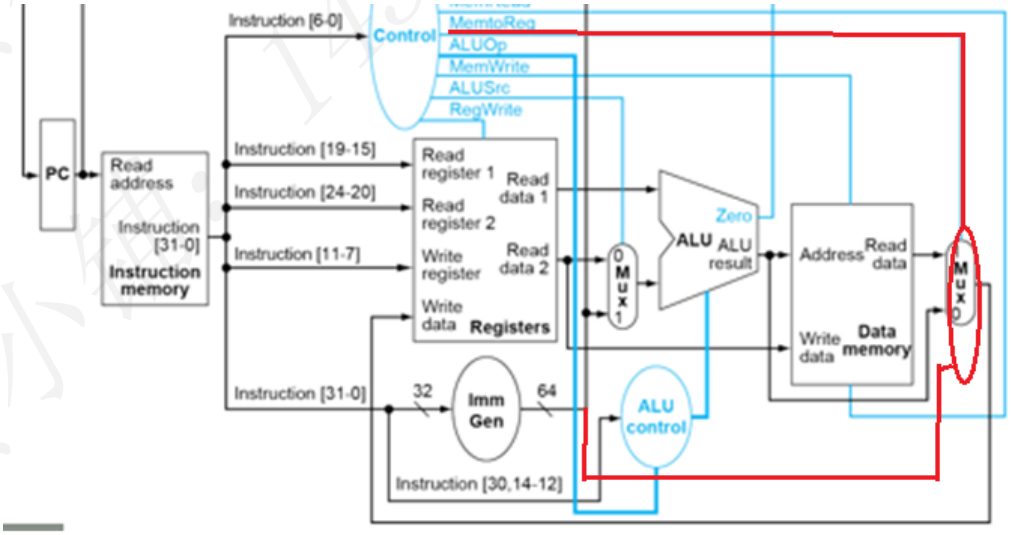
功能：将其中的 20 位常数前面补 32 个 0，后面补 12 个 0，得到 64 位常数；然后将得到的 64 位常数写入目的寄存器。

生成常数可以利用常数生成器，无需增加部件。

为此：为写目的寄存器时的数据源增加一个 2：1 选择器，同时主控制单元为其增加一个 2：1 选择控制信号，如下图所示：

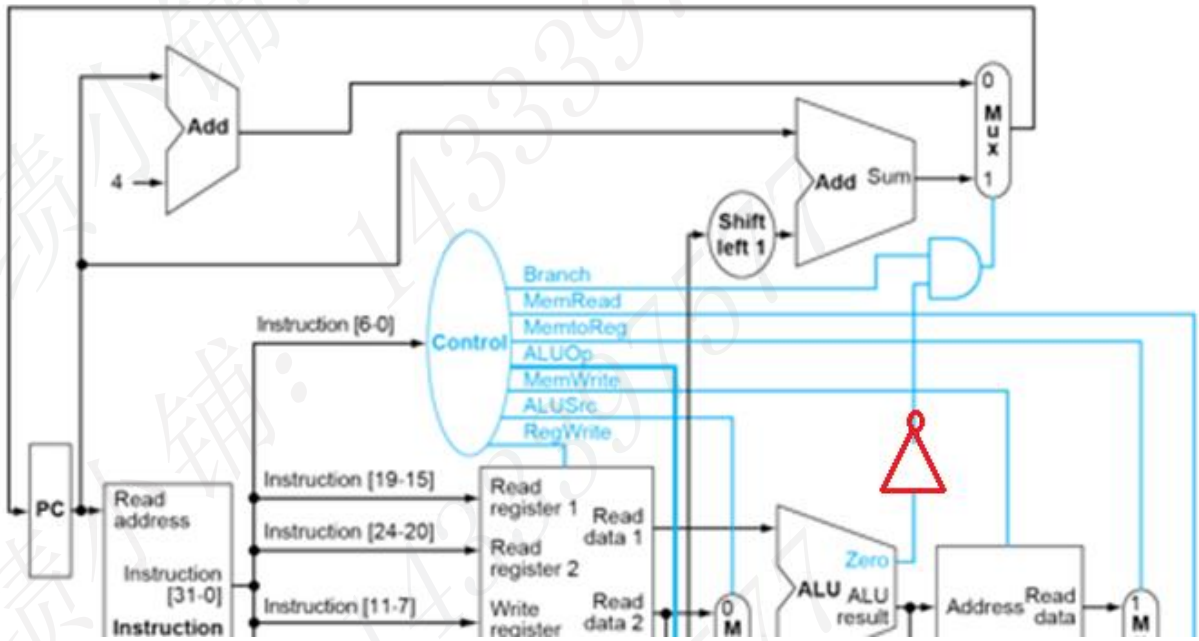


或者：用 3：1 选择器取代原来的 2：1 选择器，主控制单元用 2 位选择控制信号取代原来的 1 位选择控制信号，如下图所示：



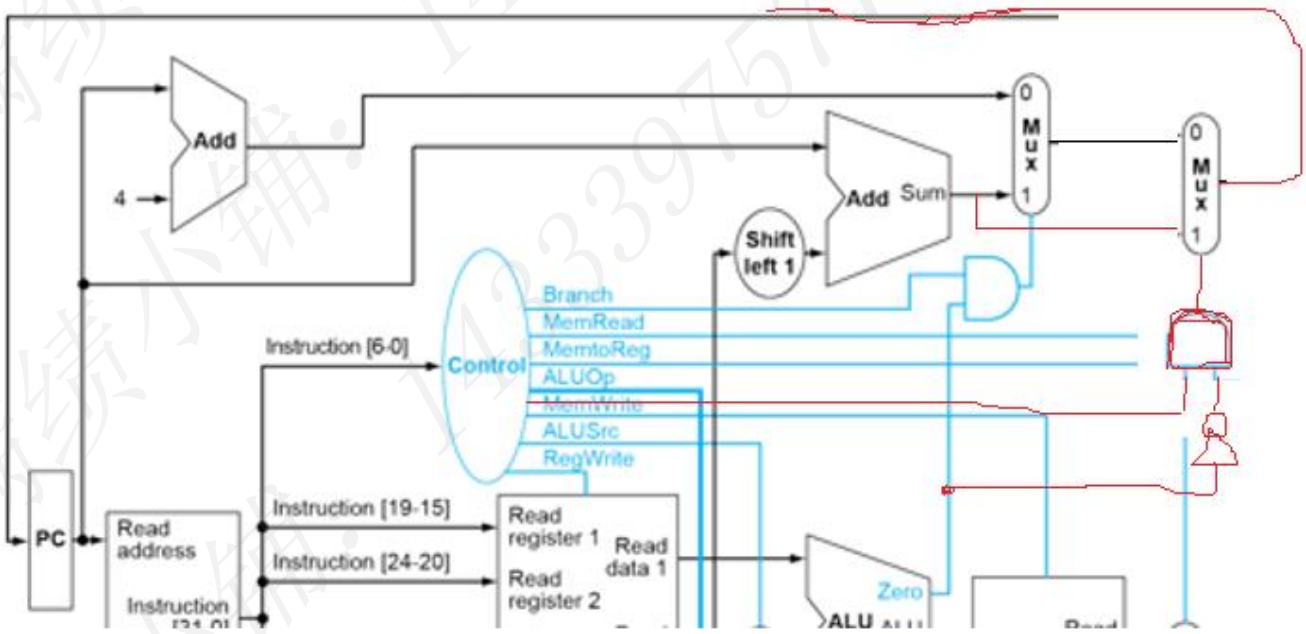
2. 用 bne 指令取代 beq 指令；

参考：在 ALU 的 zero 输出端增加一个反相器（非门），如下图所示：

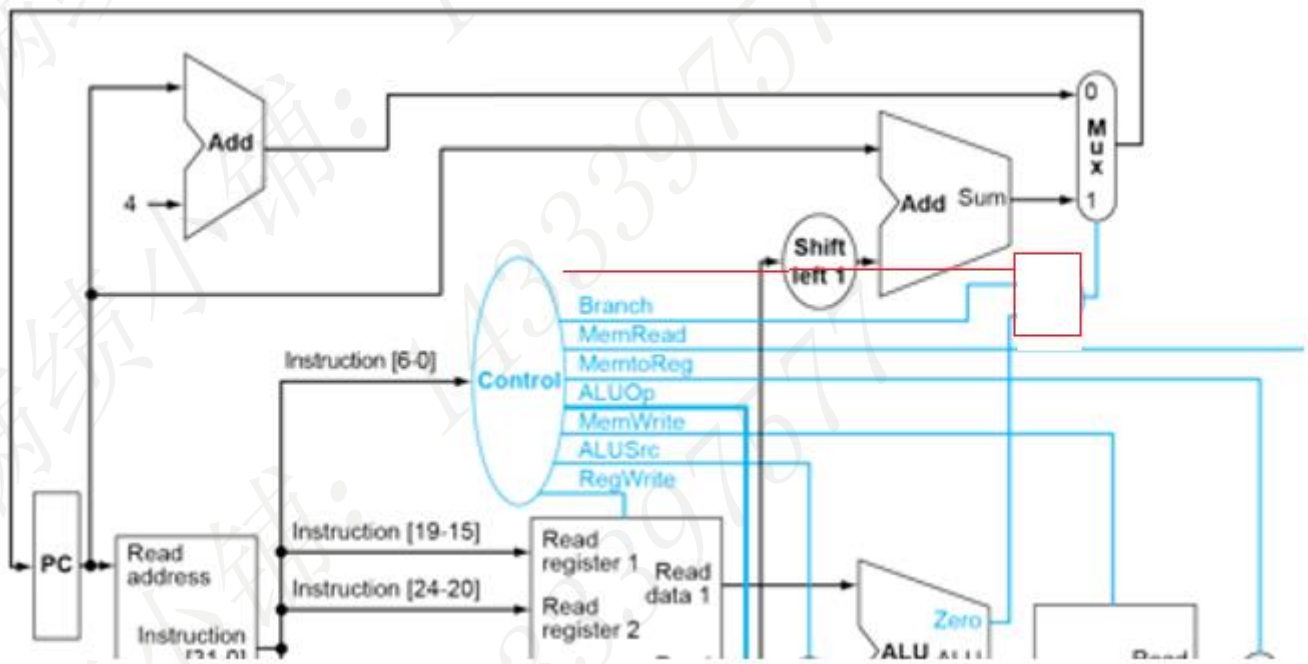


3. 同时支持 beq 指令和 bne 指令；

参考：**方案 1**：为 bne 增加一个 2：1 选择器、一个与门、一个反相器，同时主控制单元为 bne 增加一个控制信号，如下图所示：



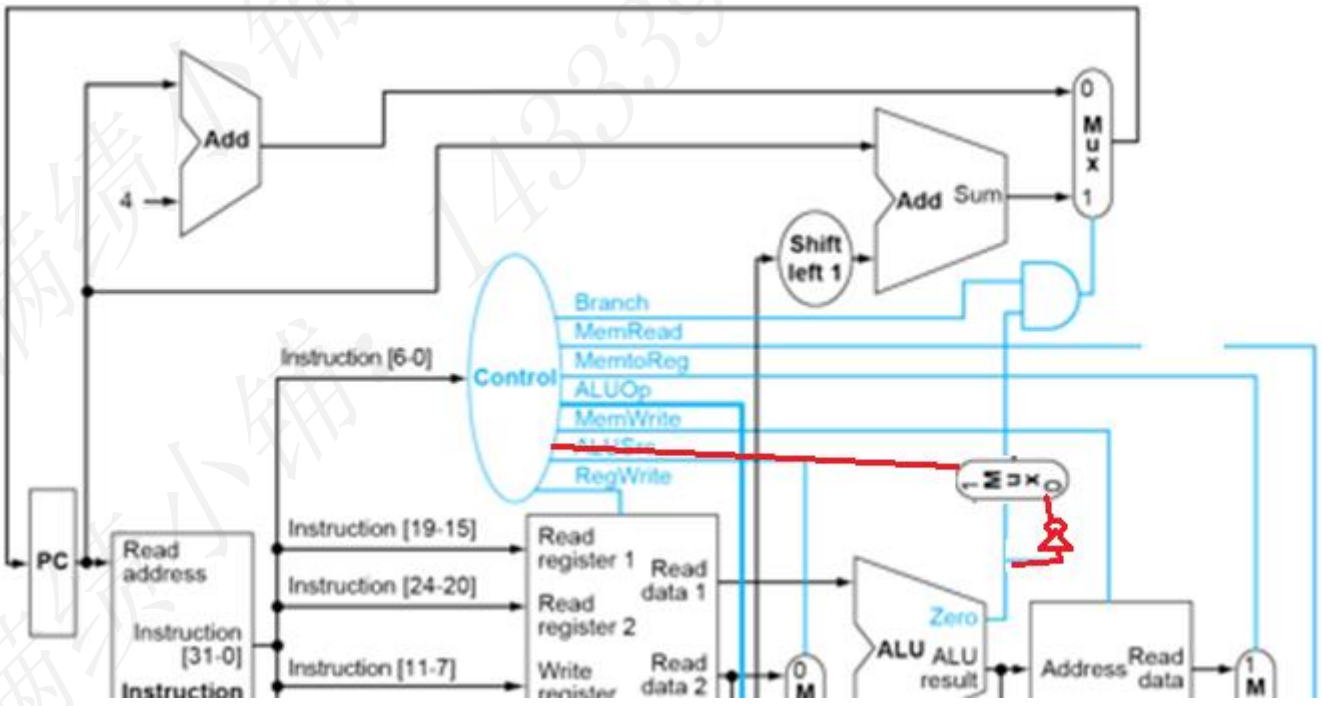
方案 2：主控制单元增加一个控制信号（1 位），同时用新的选择控制逻辑取代原来的与门，如下图所示：



新的选择控制逻辑的真值表：

Branch（原来的）	新增的	Zero	输出	说明
0	0	x	0	非分支指令
1	0	0	0	Beq 指令且条件不成立
1	0	1	1	Beq 指令且条件成立
0	1	0	1	Bne 指令且条件成立
0	1	1	0	Bne 指令且条件不成立

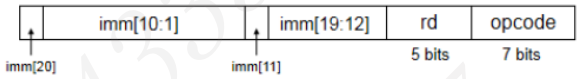
方案 3：原 branch 区分是否分支指令，新增的区分是 beq 还是 bne



4. 支持 jal 指令；

参考：指令 jalr 的汇编格式和机器格式如下

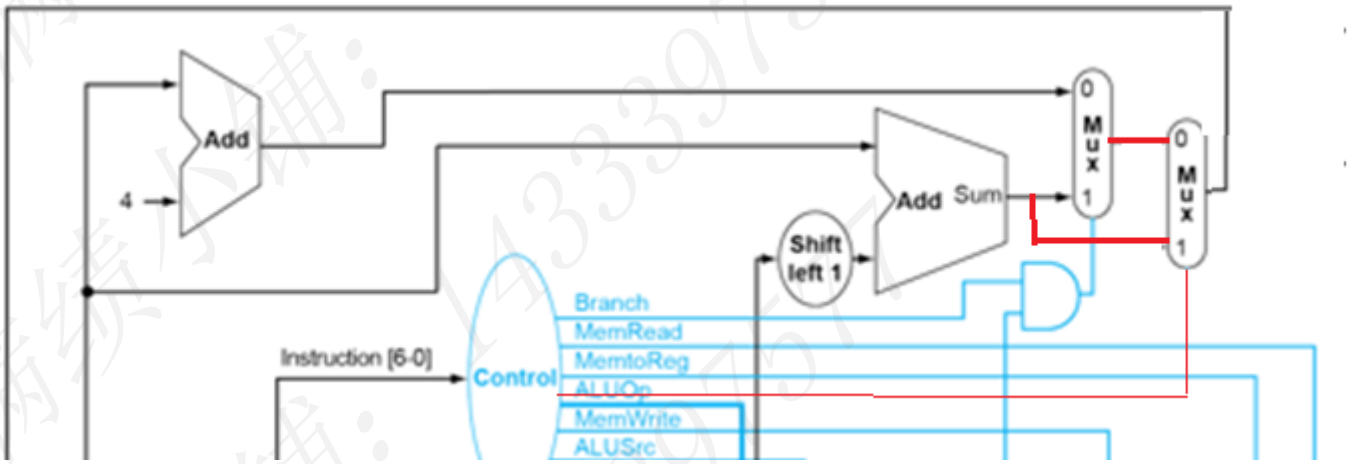
转移连接指令：jal rd, label



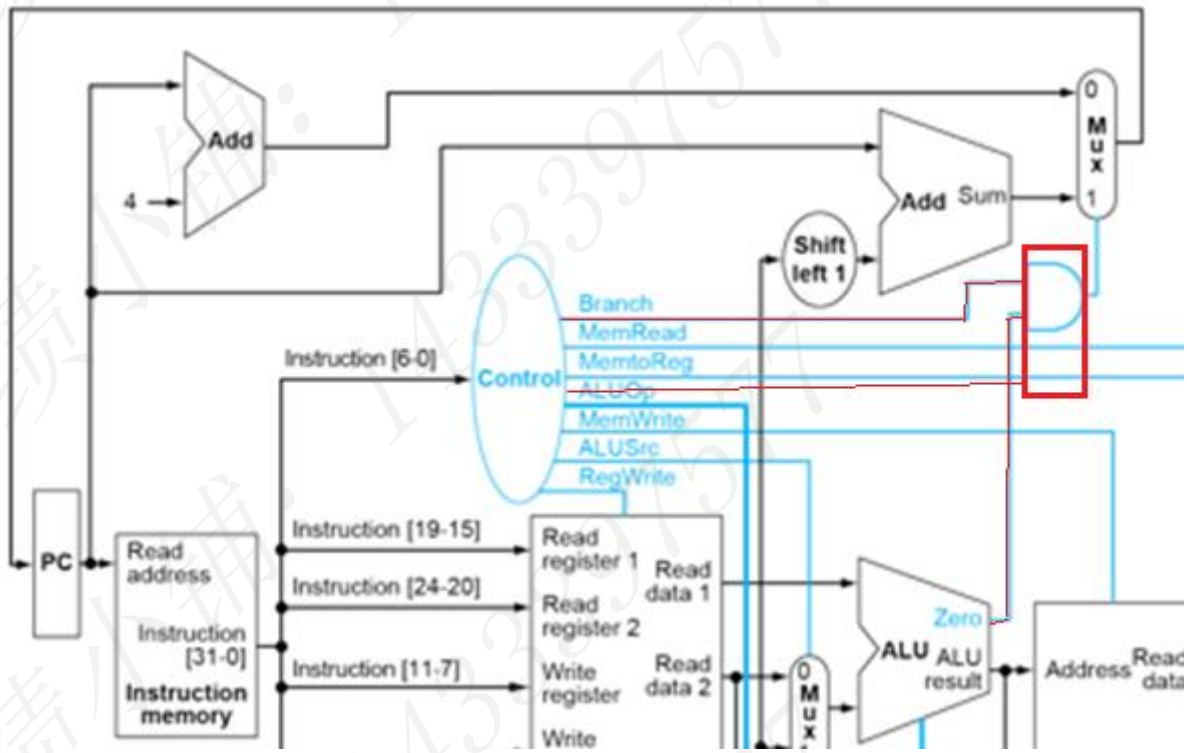
功能：将其中的 12 位常数（偏移量）经过有符号扩展得到 64 位常数；扩展后的 64 位偏移量左移 1 位（乘 2）后与当前地址（PC 提供）相加得到目标地址；用计算出的目标地址去修改 PC（无条件转移），同时将 PC+4（返回地址）写入目的寄存器。

计算目标地址的过程跟 Branch 指令一样，无需增加部件。

为此：**1**）为写 PC 时的地址源增加一个 2：1 选择器，同时主控制单元为其增加一个 2：1 选择控制信号，如下图所示：



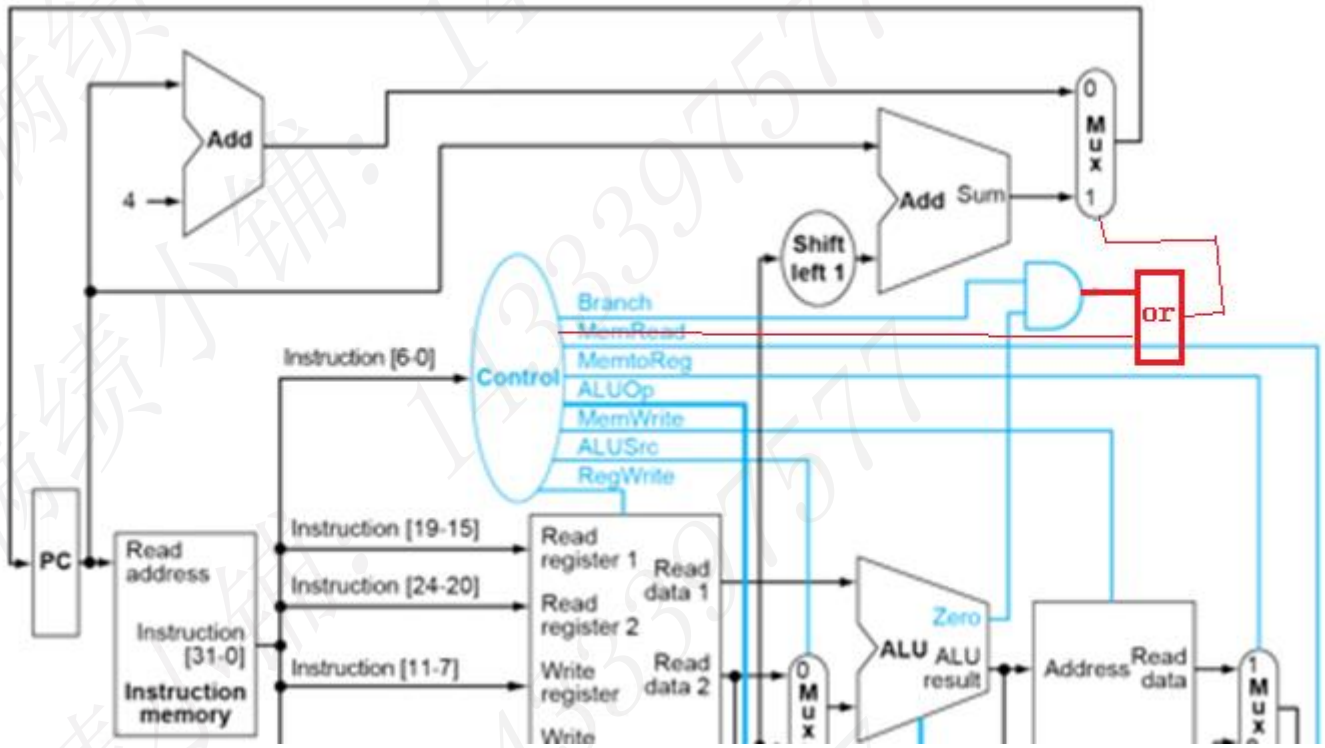
或者：主控制单元增加一个控制信号（1 位），同时用新的选择控制逻辑取代原来的与门，如下图所示：



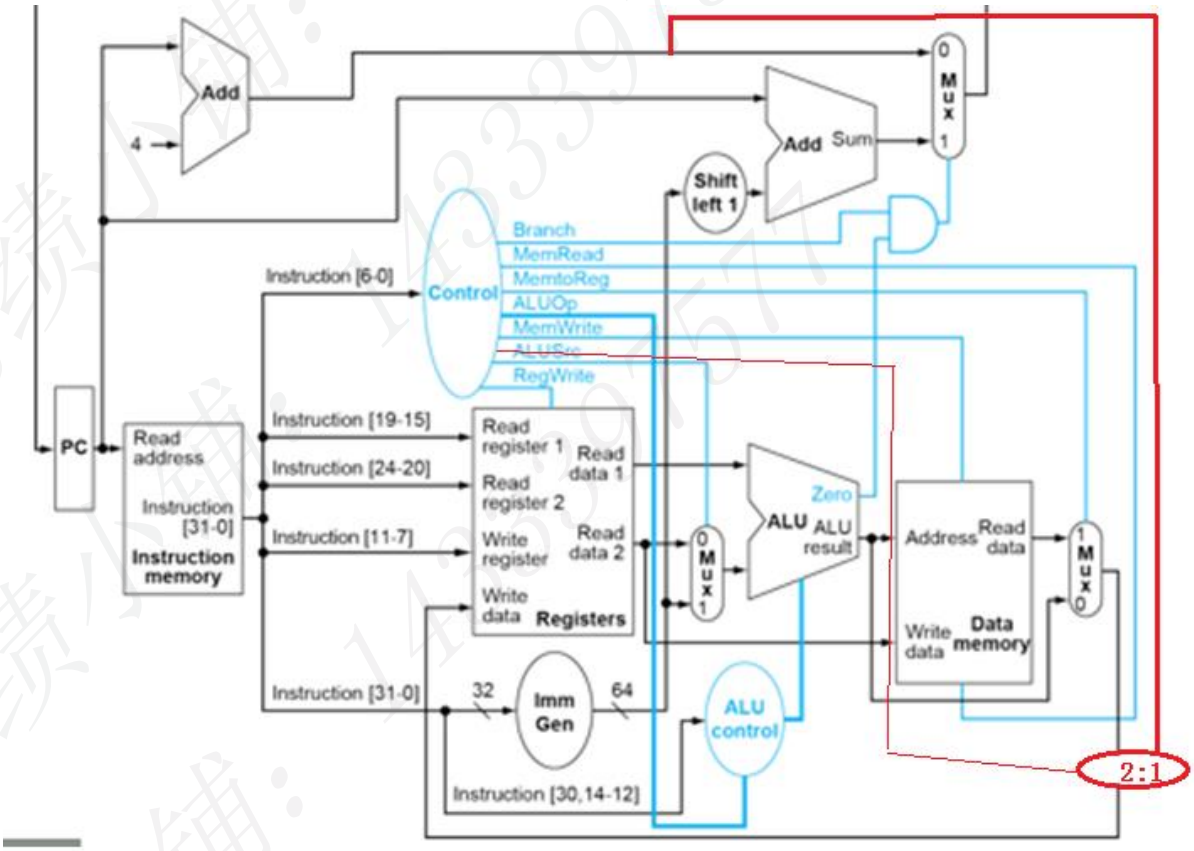
新的选择控制逻辑的真值表：

Branch（原有）	新增	Zero	输出	说明
0	0	x	0	非 beq、非 jal
1	0	0	0	Beq 指令
1	0	1	1	Beq 指令
0	1	x	1	Jal 指令

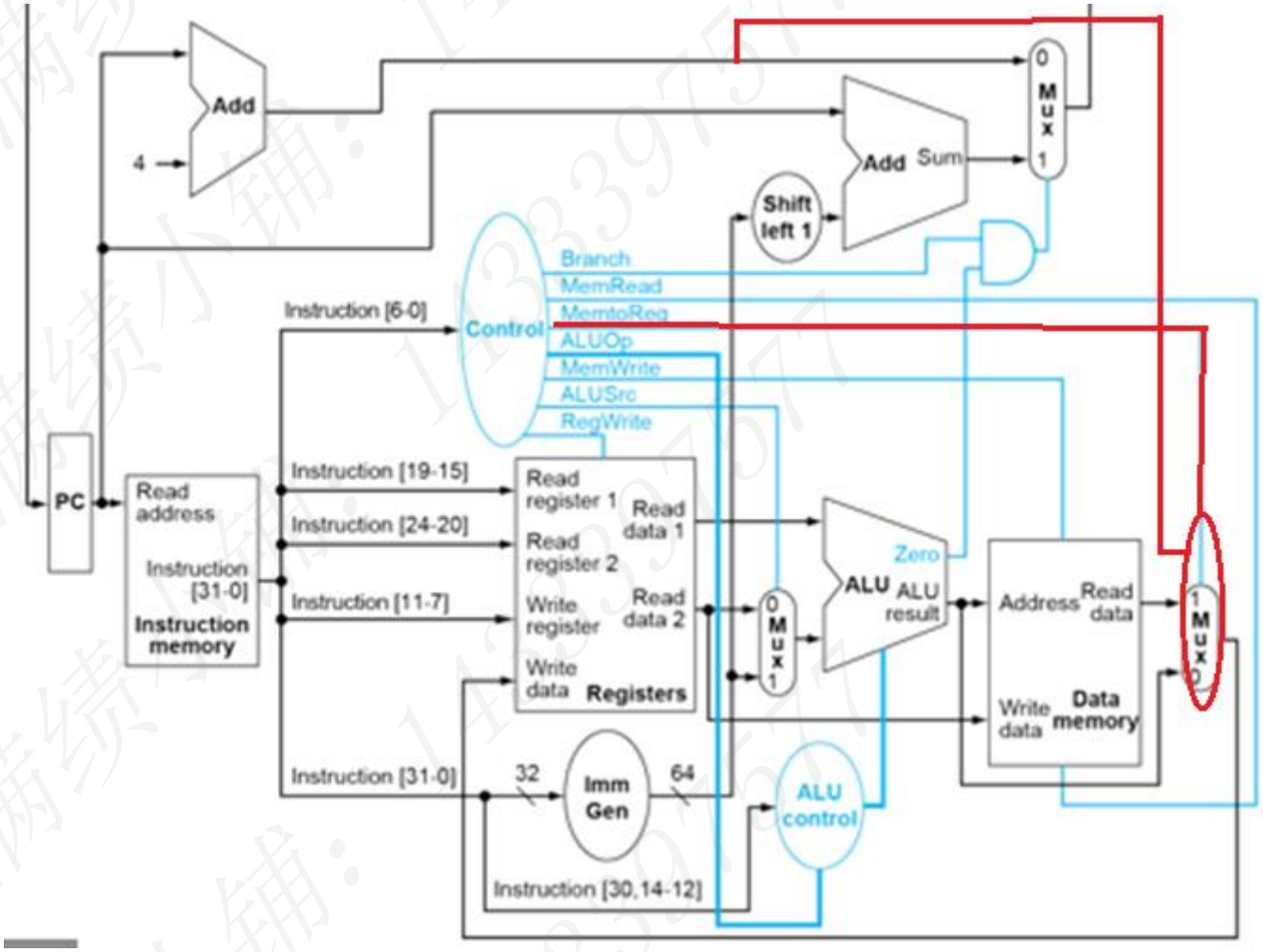
或者：在与门（branch 和 zero）的输出端增加一个或门，同时主控制单元为其增加一个控制信号。或门的输入：一个是与门的输出，另一个就是主控制单元为其增加的控制信号。或门的输出再送给原来的 2：1 选择器。如下图所示：



2）为写目的寄存器时的数据源增加一个 2：1 选择器，同时主控制单元为其增加一个 2：1 选择控制信号，如下图所示：



或者：用 3：1 选择器取代原来的 2：1 选择器，主控制单元用 2 位选择控制信号取代原来的 1 位选择控制信号，如下图所示：



5. 支持 jalr 指令。

参考：指令 jalr 的汇编格式和机器格式如下

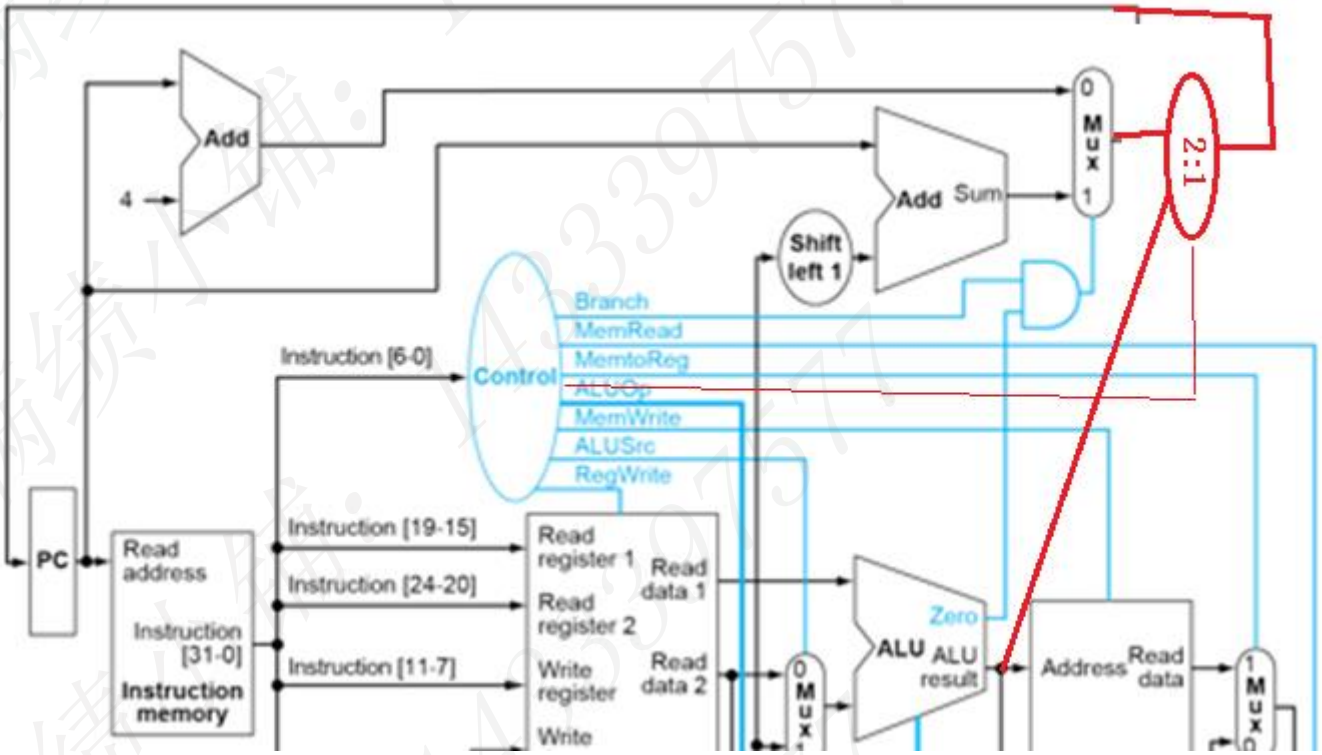
寄存器转移连接指令：jalr rd, imm(rs1)

immediate	rs1	funct3	rd	opcode
12 bits	5 bits	3 bits	5 bits	7 bits

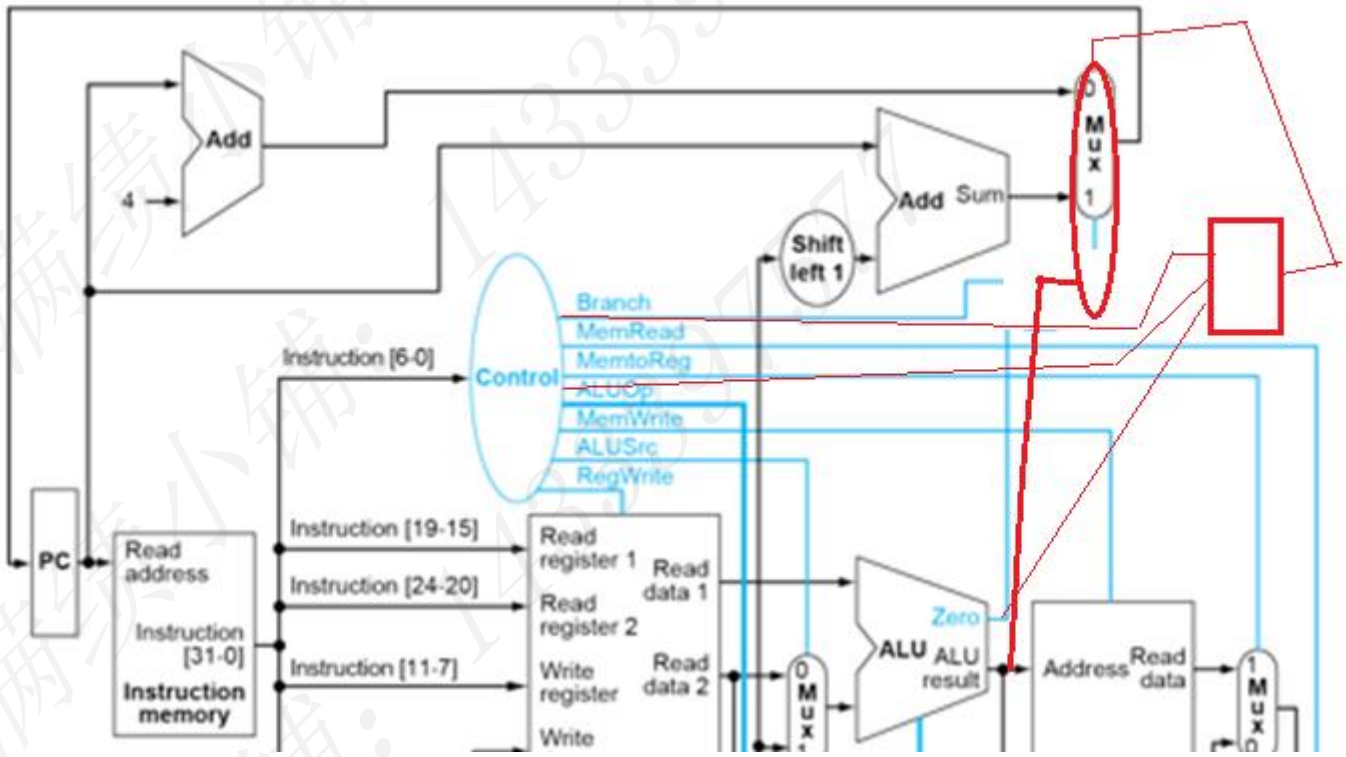
功能：将其中的 12 位常数（偏移量）经过有符号扩展得到 64 位常数；扩展后的 64 位偏移量与源寄存器 1 的内容相加得到目标地址；用计算出的目标地址去修改 PC（无条件转移），同时将 PC+4（返回地址）写入目的寄存器。

计算目标地址的过程跟 Load/Store 指令计算内存地址的过程一样，无需增加部件。

为此：1）为写 PC 时的地址源增加一个 2：1 选择器，同时主控制单元为其增加一个 2：1 选择控制信号，如下图所示：



或者：用 3：1 选择器取代原来的 2：1 选择器，主控制单元新增 1 位控制信号，同时用新的选择控制逻辑取代原来的与门，如下图所示：



新的选择控制逻辑的真值表：

Branch（原有）	新增	zero	输出	说明
0	0	x	0	非 beq、非 jalr
1	0	0	0	Beq 指令
1	0	1	1	Beq 指令
0	1	x	2	Jalr 指令

2）为写目的寄存器时的数据源增加一个 2：1 选择器，同时主控制单元为其增加一个 2：1 选择控制信号，参看 jal。

或者：用 3：1 选择器取代原来的 2：1 选择器，主控制单元用 2 位选择控制信号取代原来的 1 位选择控制信号，参看 jal。

三、设计一条新指令 mysd，其汇编、机器格式以及功能同 sd，但是采用相对基址变址寻址。汇编格式：Mysd rs2,offset(rs1)，内存地址=rs1 的内容（基址）+ rs2 的内容（变址）+offset（偏移量）。请说明为支持 mysd，前述的单周期处理器结构图应做哪些修改？

参考：方案 1：利用已有的 ALU 计算 **rs1 的内容（基址）+offset（偏移量）**，然后：

1）增加一个专用加法器计算 **ALU 的 Result（rs1 的内容（基址）+offset（偏移量））+ rs2 的内容（变址）**；

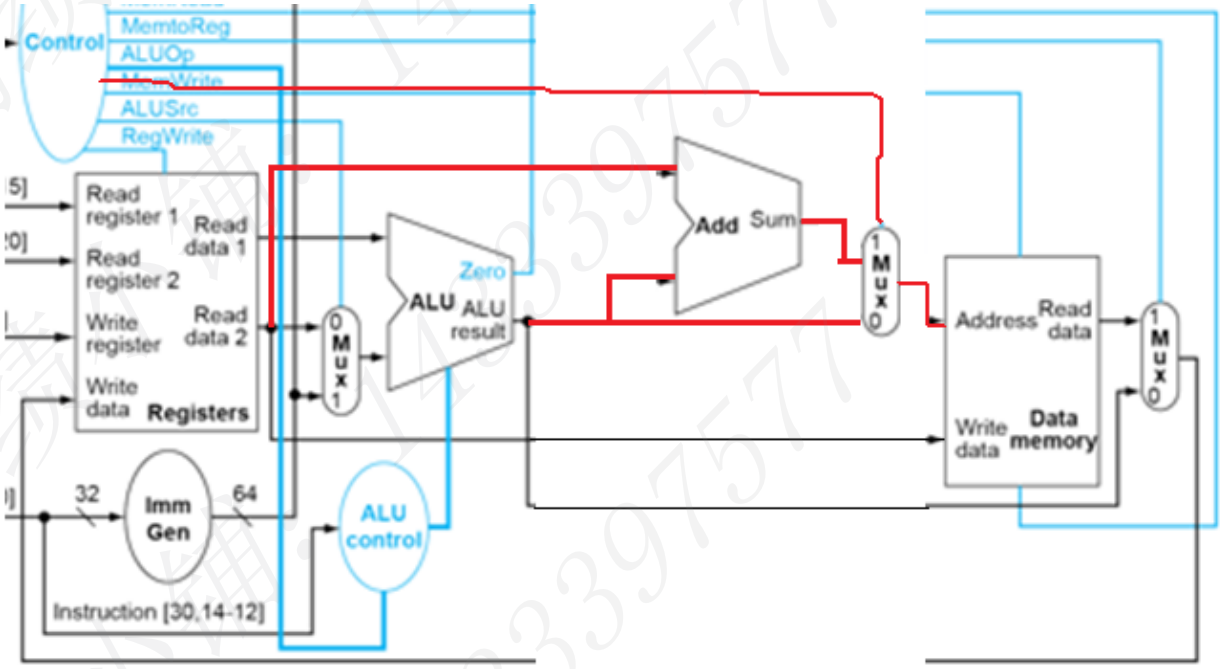
2）数据存储器的地址输入端增加一个 2：1 选择器，同时主控制单元为其增加一个选择控制信号（1 位）

方案 2：利用已有的 ALU 计算 **rs1 的内容（基址）+ rs2 的内容（变址）**，然后：

1）增加一个专用加法器计算 **ALU 的 Result（rs1 的内容（基址）+ rs2 的内容（变址））+ offset（偏移量）**；

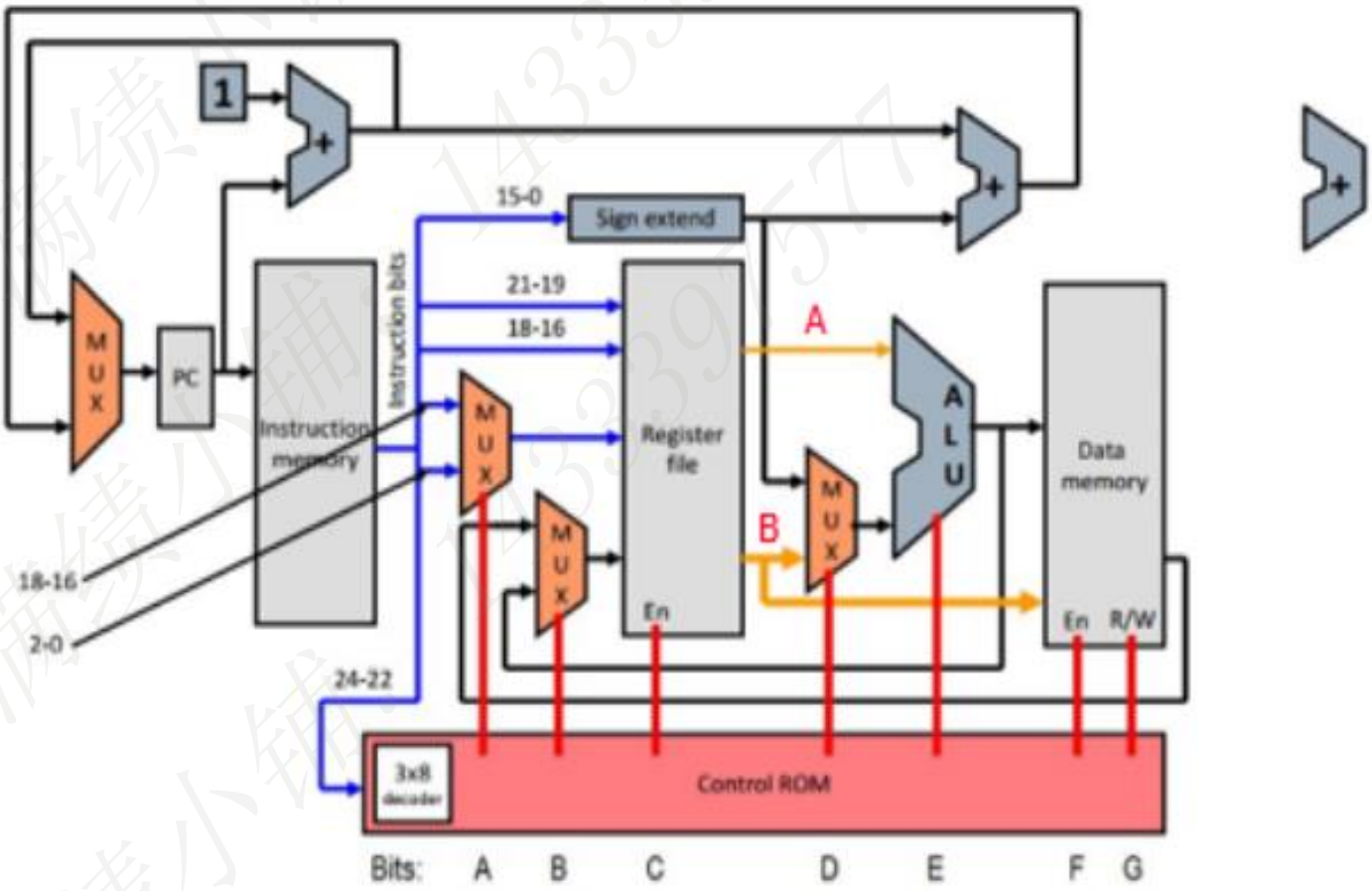
2）数据存储器的地址输入端增加一个 2：1 选择器，同时主控制单元为其增加一个选择控制信号（1 位）

两种方案的差别只在内存地址的计算。方案 1 修改后的结构图如下所示：



说明：以上做法是把新增的专用加法器安排在 ALU 的输出端，也可以安排在 ALU 的输入端，所以有多种具体方案。

四、一个单周期处理器如下图所示



控制器产生的控制信号有：A、B、C、D、E、F、G。A、B 和 D 用来控制 MUX，为

“0”选择最上边的输入信号；E=0 时，ALU 做加法运算；C=1 时，RF 正常工作；F=1，Data Memory 正常工作，G=0，读操作，G=1，写操作。

1. 修改上图的数据通路以支持指令“add2”，add2 实现的功能：

regB = regA + regB + offset

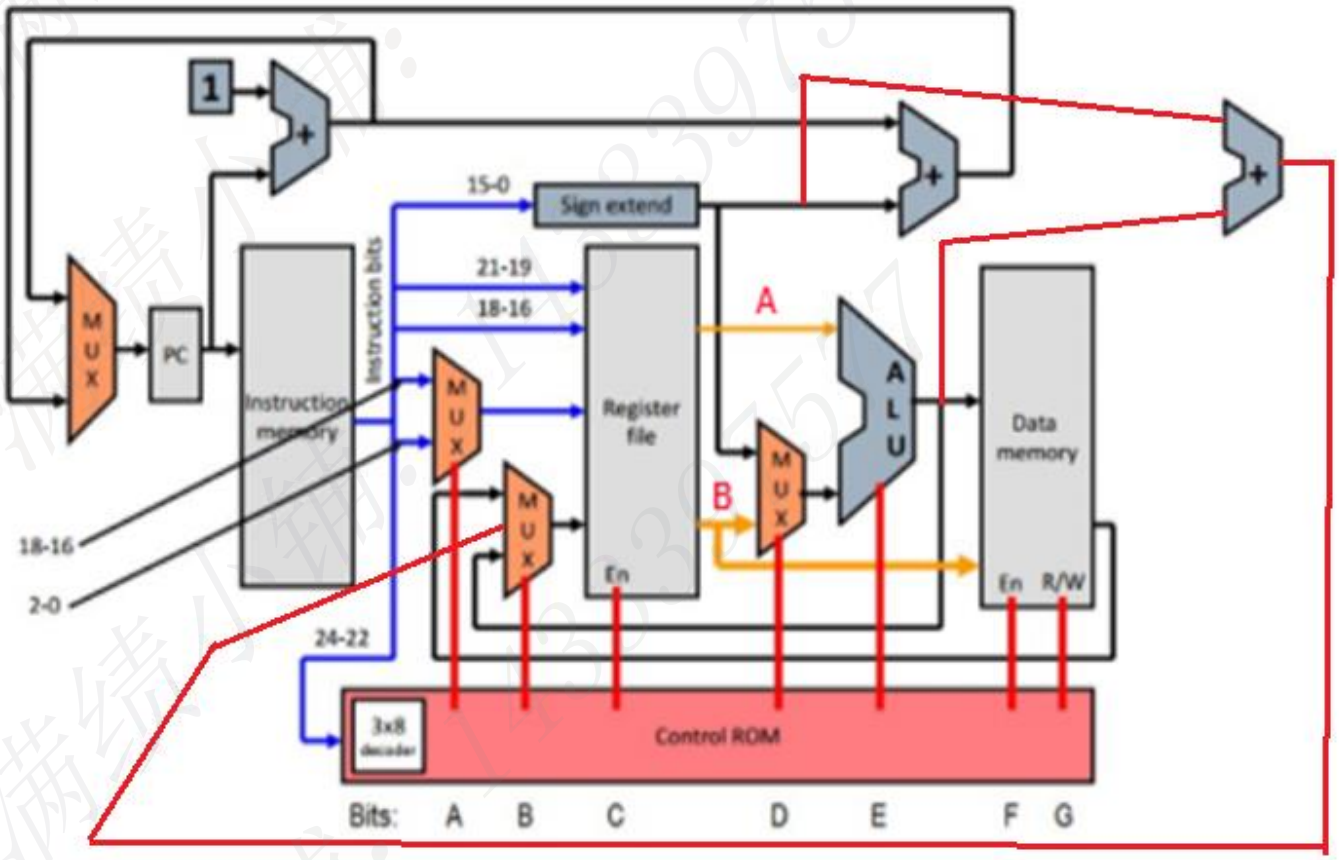
必须满足以下条件：（a）使用额外的加法器（右上角）；

（b）将其中一个 2:1 MUX 扩展成 3:1 MUX；

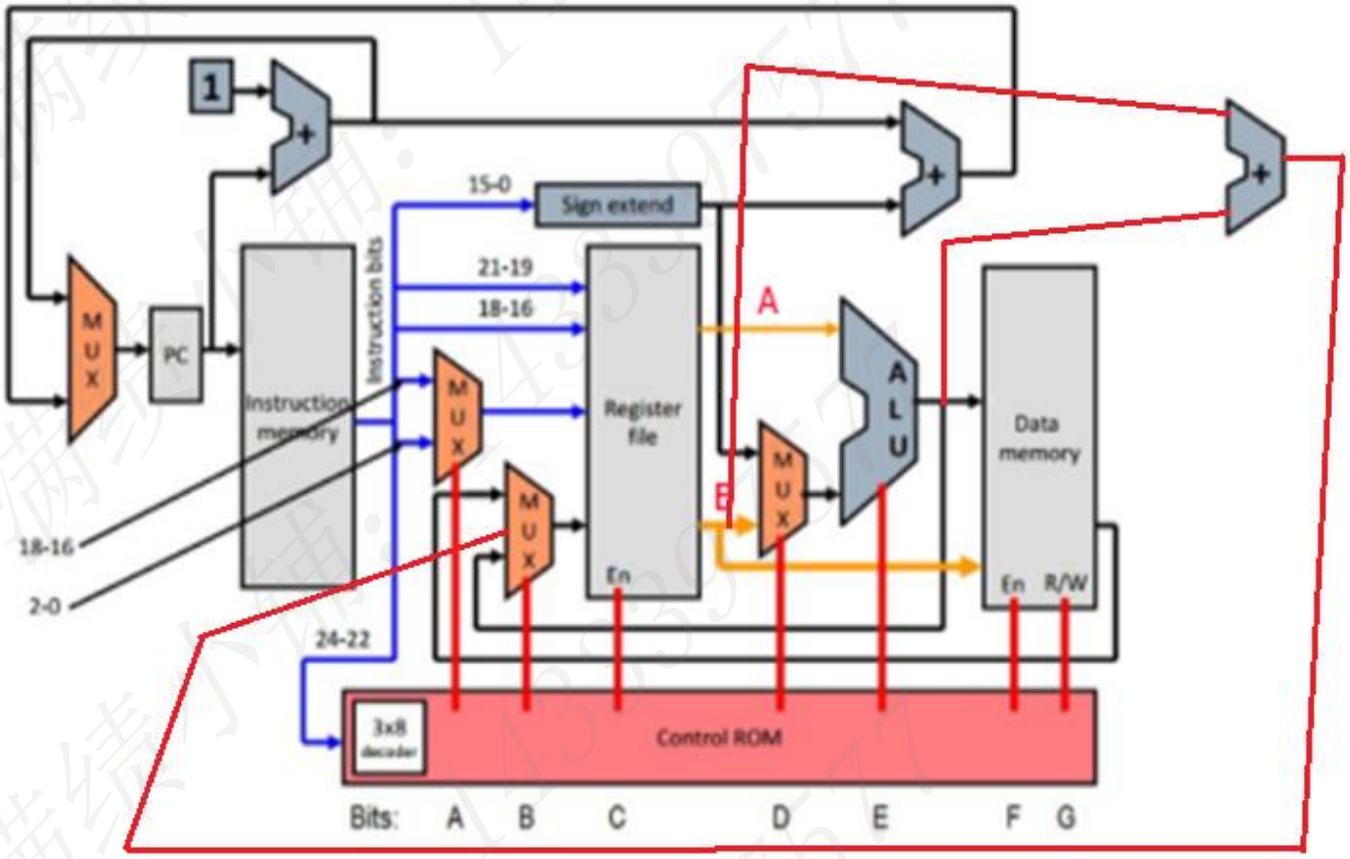
请用文字或者画图方式给出解决方案。

参考：

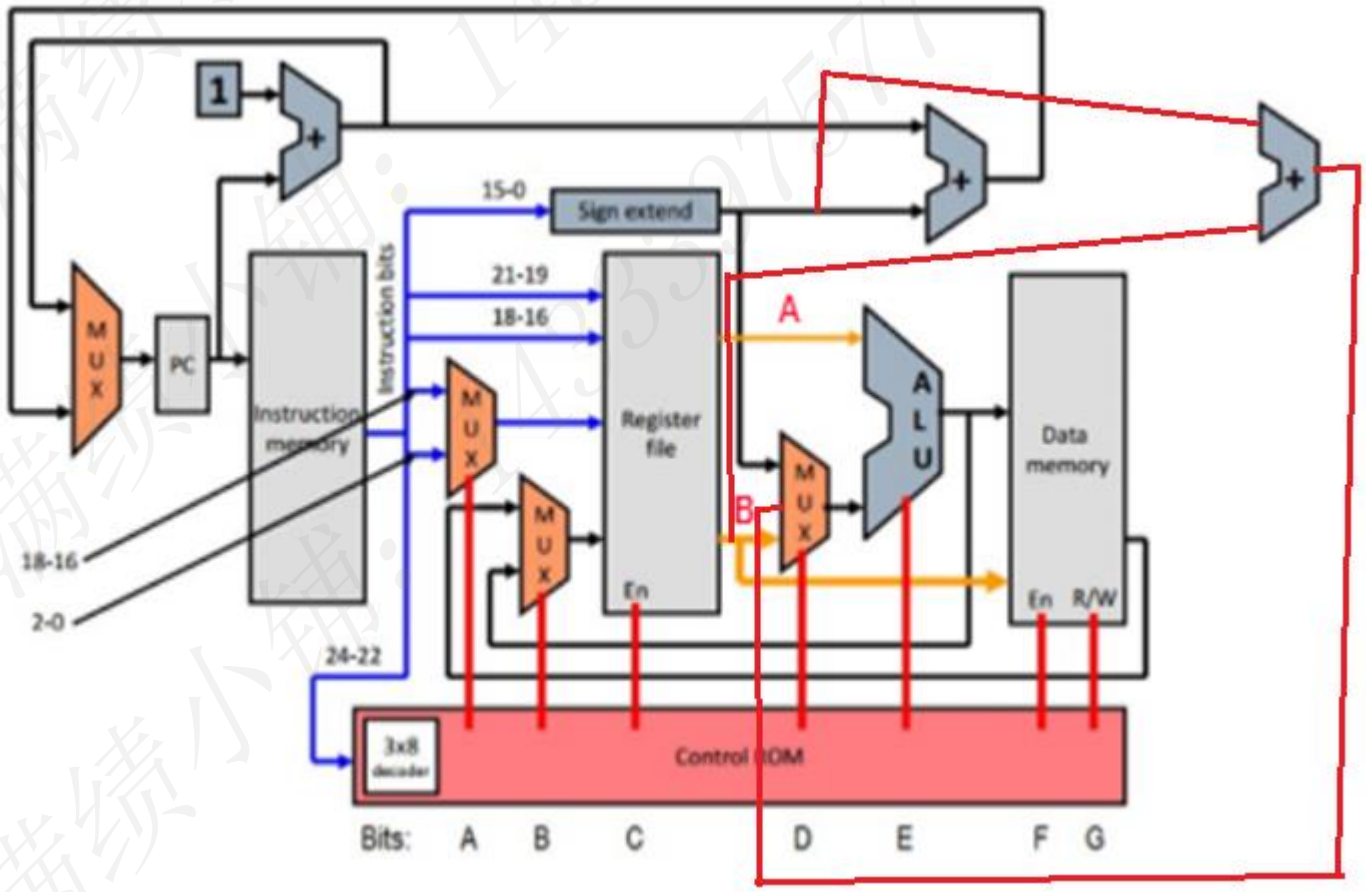
方案 1：regA 和 regB 经 ALU 相加，ALU 结果送 adder 和 offset 相加，adder 结果送 MUXB（改为 3 选 1），如下图所示：



方案 2：regA 和 offset 经 ALU 相加，ALU 结果送 adder 和 regB 相加，adder 结果送 MUXB（改为 3 选 1），如下图所示：



方案 3：regB 和 offset 经 adder 相加，adder 结果送到 MUX D（改为 3 选 1）和 regA 经 ALU 相加，如下图所示：



2. 在下表中填写实现指令 add2 的控制信号，注意 MUX 的新控制信号。

A	B	C	D	E	F	G

参考：

控制信号	A	B	C	D	E	F	G
方案 1	0	1x	1	1	0	0	x
方案 2	0	1x	1	0	0	0	x
方案 3	0	1	1	1x	0	0	X

说明：假设 3：1 选择器的选择控制信号为 1x 时选择新增加的第 3 路输入！