第三章 微处理器——原型MIPS微处理器

华中科技大学
电子信息与通信学院
School of Electronic Information and Communications

学习目标

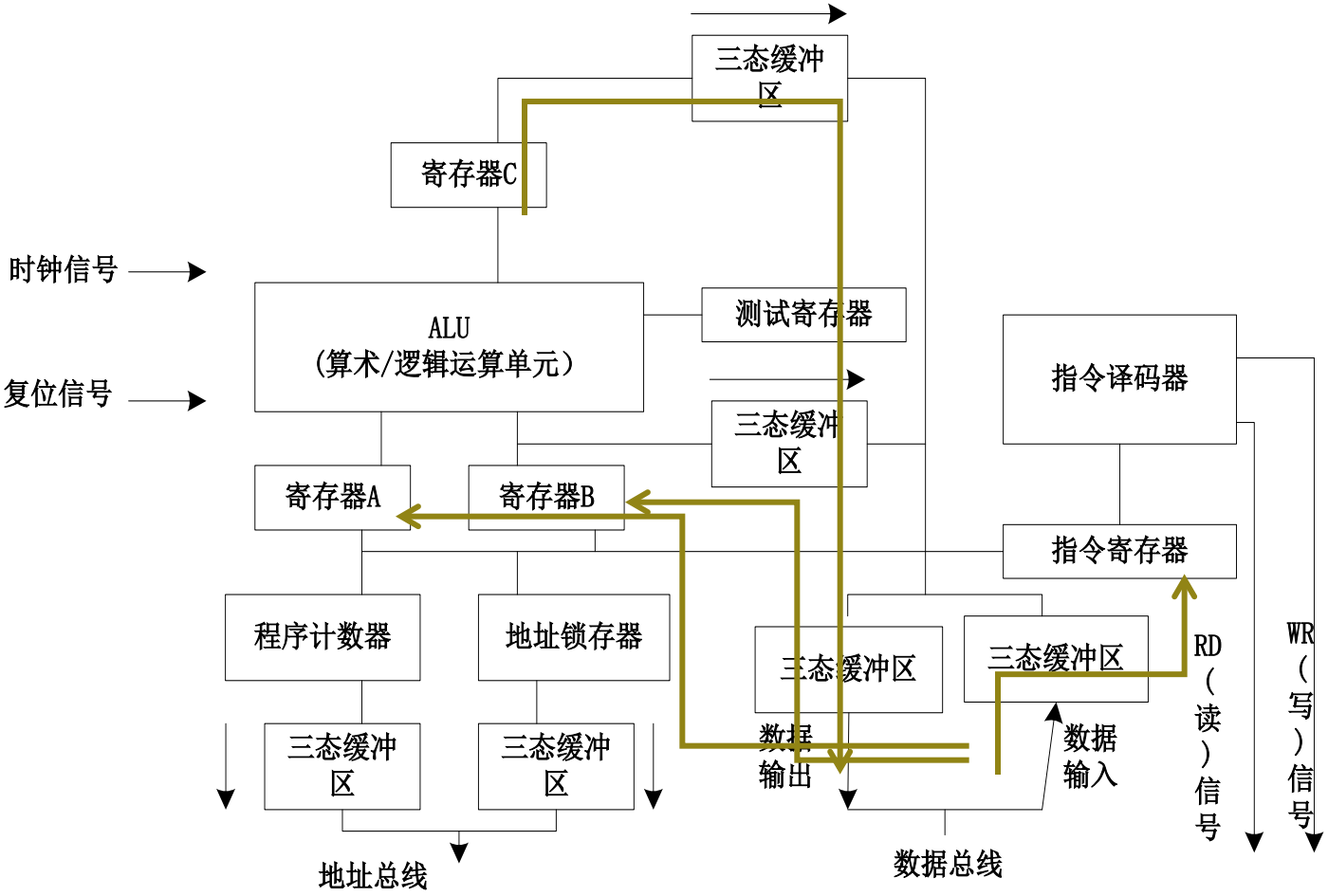
- 了解微处理器的基本构成
- 能设计执行简单MIPS指令集的微处理器

3.1微处理器基本构成

- 微处理器一般执行三种基本工作：
  - 通过使用ALU（算术/逻辑单元），微处理器执行数学计算。例如：加法、减法、乘法和除法等。
  - 微处理器可以将数据从一个内存位置移动到另一个位置。
  - 微处理器可以做出决定，并根据这些决定跳转到一组新指令。

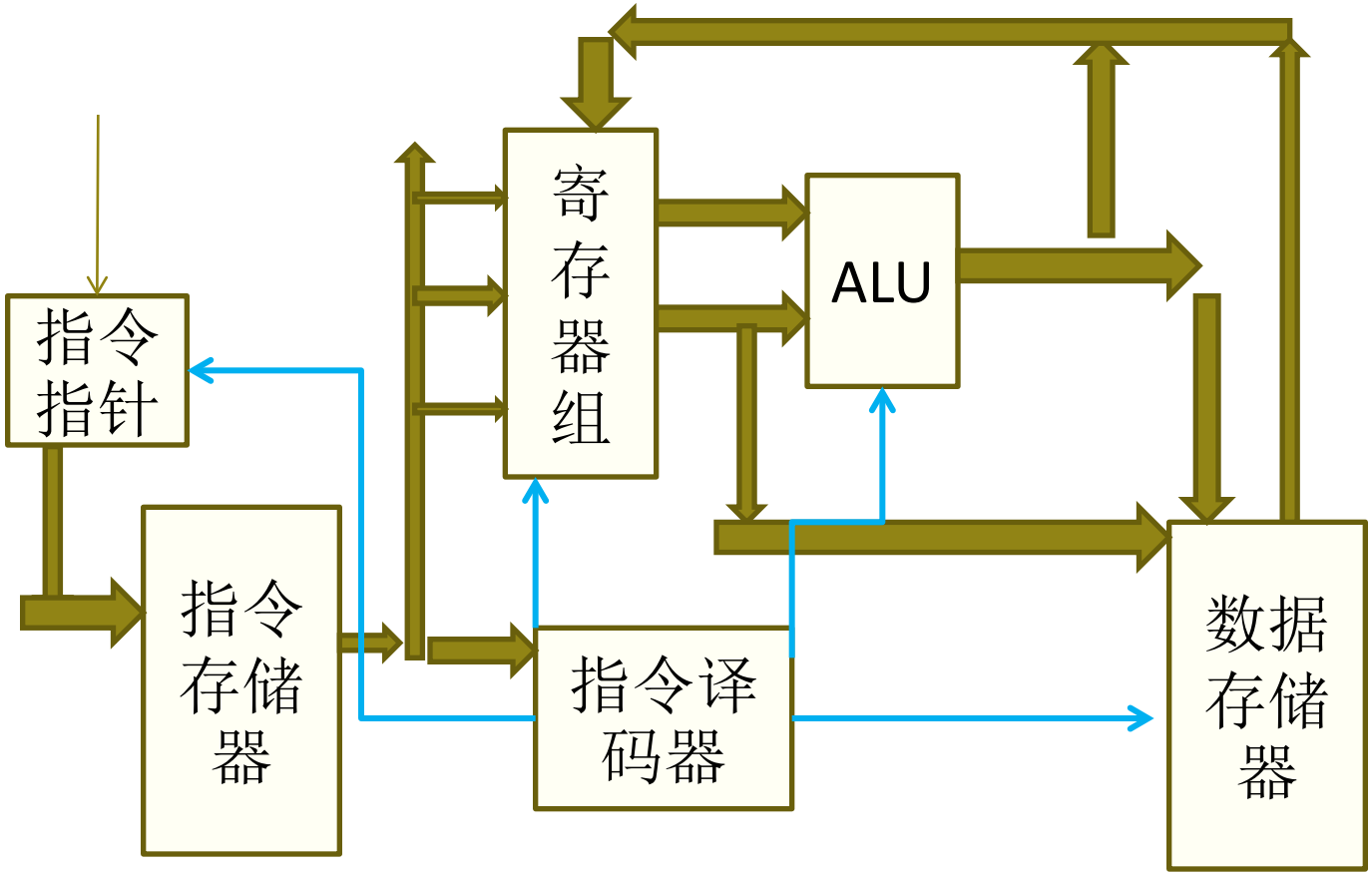
- 微处理器与外部组件的基本接口包括：
  - 一组地址总线（总线宽度可以8位、16位或32位），用于向内存发送地址。
  - 一组数据总线（总线宽度可以是8位、16位或32位），能够将数据发送到内存或从内存取得数据。
  - 一条RD（读）和WR（写）控制信号，告诉内存它是希望将数据写入某个地址位置还是从某个地址位置获得内容。
  - 时钟信号，将时钟脉冲序列发送到处理器，控制微处理器进行工作。
  - 复位信号，用于将程序计数器重置为零（或者其他内容）并重新开始执行。

简单微处理器的基本构成



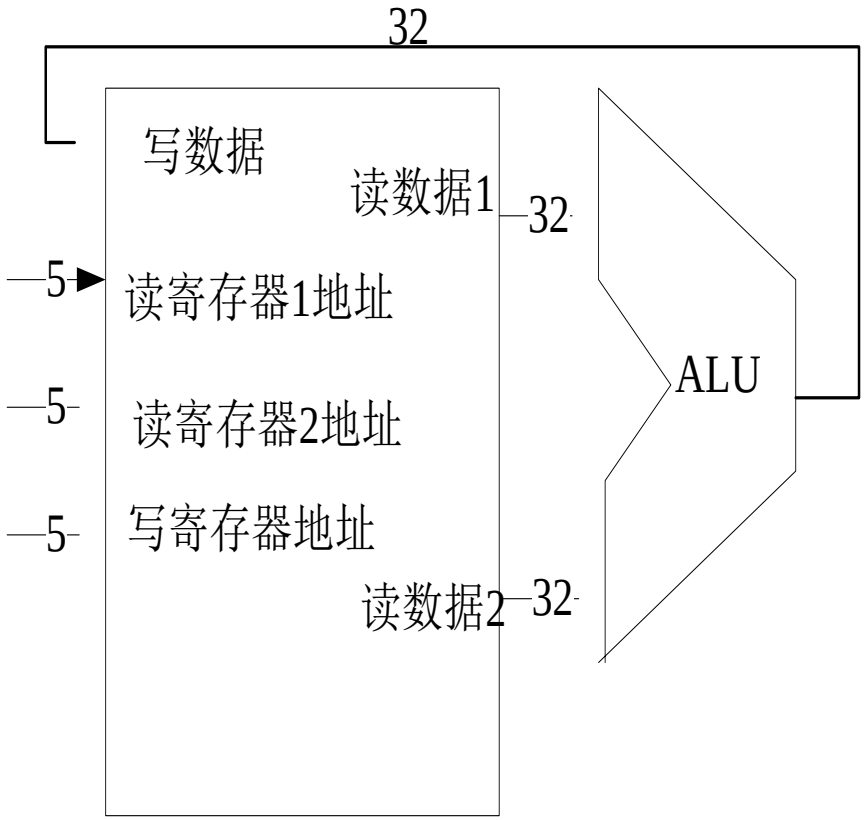
3.2简单MIPS指令集微处理器基本构成

- MIPS微处理器支持以下指令:
  - 内存操作如lw，sw指令
  - 算术逻辑运算如add，sub，and，or，slt指令
  - 程序控制如beq，j指令

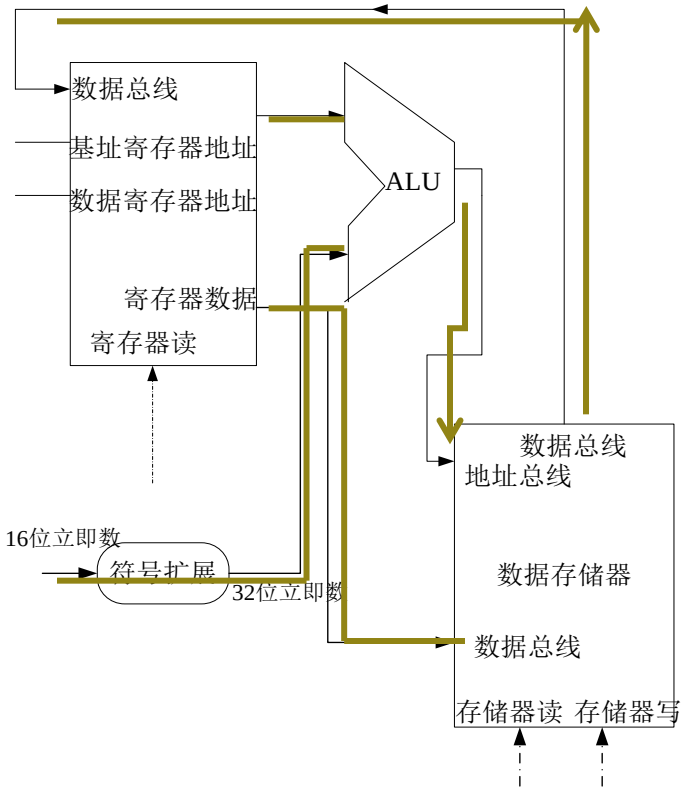

寄存器组
ALU
指令指针
指令存储器
指令译码器
数据存储器

3.3数据通路实现原理

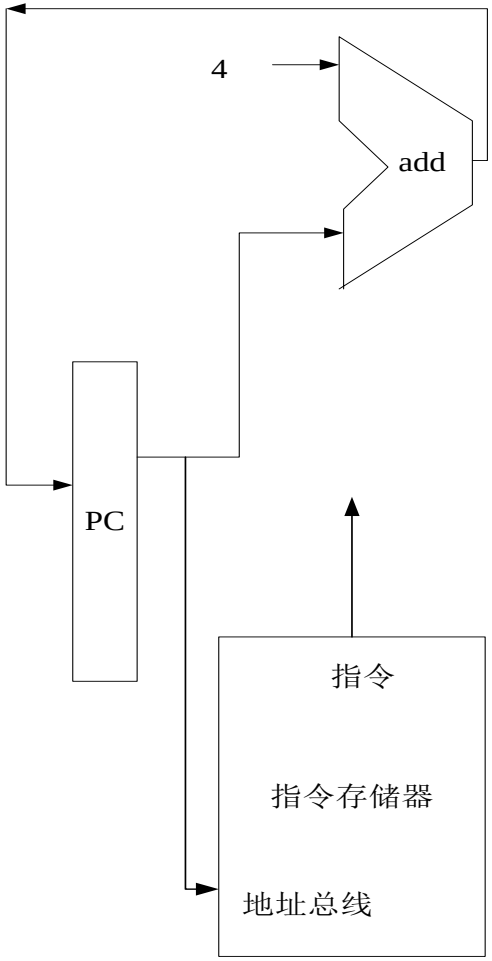
R型指令实现部件



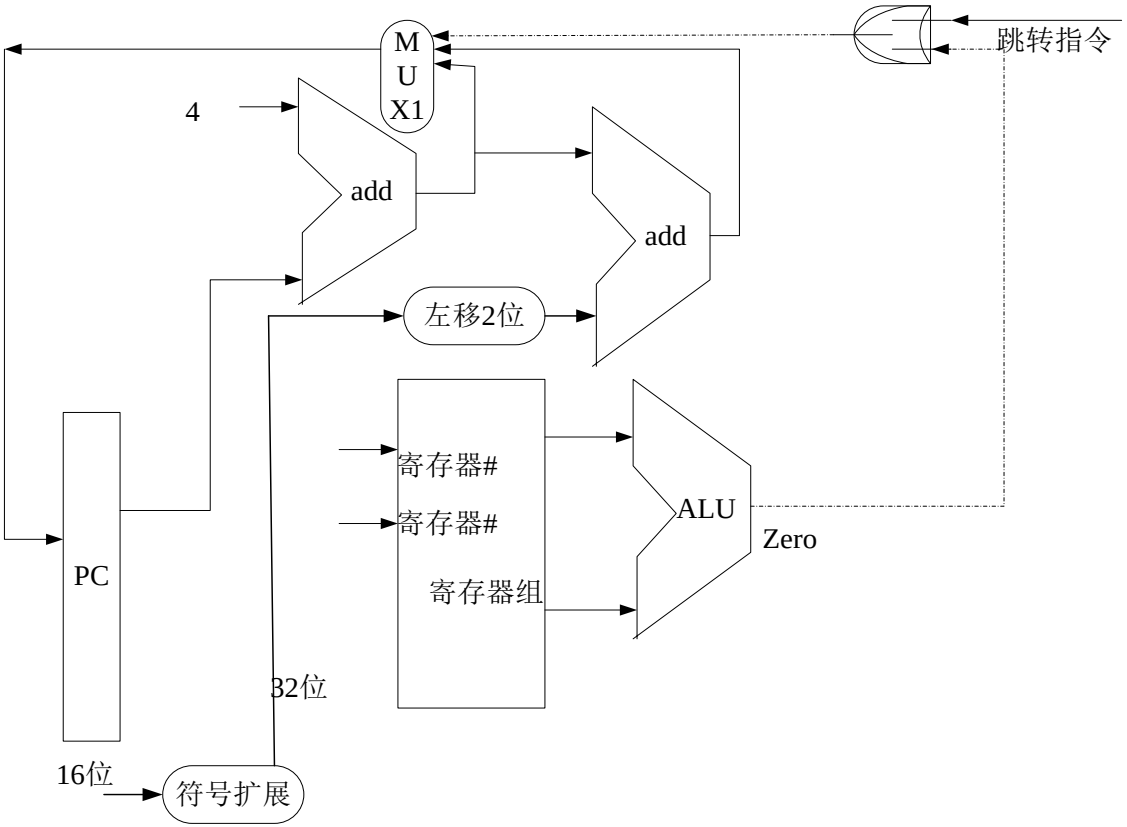
存储器数据存取部件


数据总线
基址寄存器地址
数据寄存器地址
寄存器数据
寄存器读
ALU
数据总线
地址总线
16位立即数
符号扩展
32位立即数
数据存储器
数据总线
存储器读
存储器写

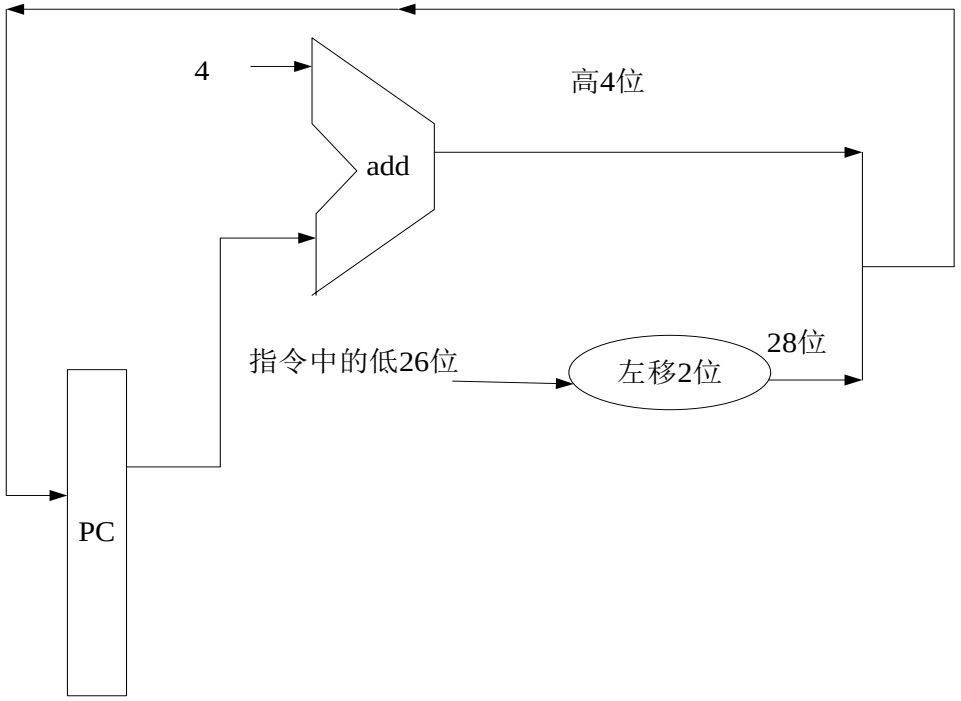
- 指令获取部件



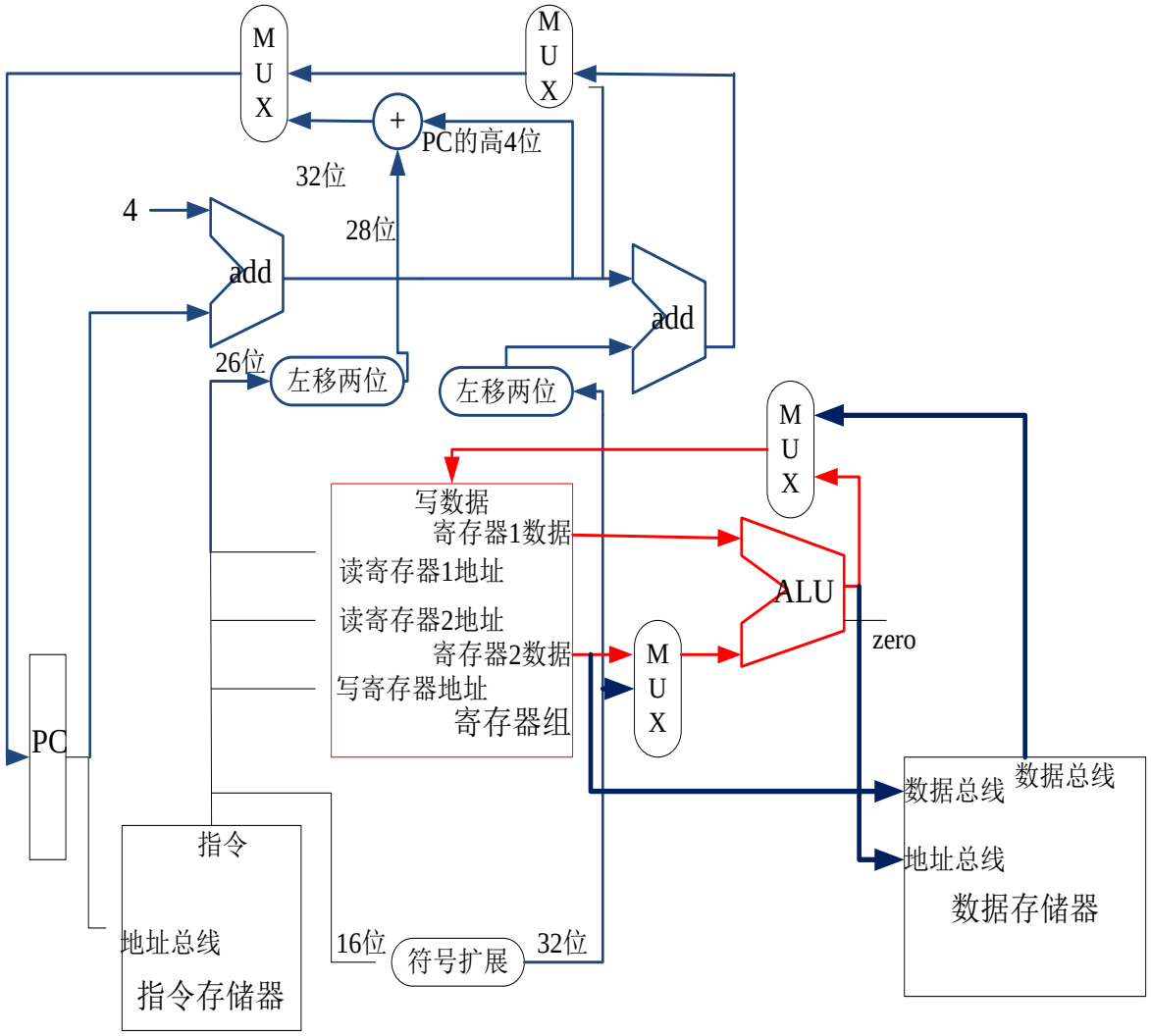
条件跳转控制



无条件伪直接寻址部件



完整的数据通路


M
U
X
M
U
X
+
PC的高4位
32位
4
28位
add
add
26位
左移两位
左移两位
M
U
X
写数据
寄存器1数据
读寄存器1地址
读寄存器2地址
寄存器2数据
写寄存器地址
寄存器组
ALU
zero
M
U
X
PC
指令
地址总线
指令存储器
16位
符号扩展
32位
数据总线
数据总线
地址总线
数据存储器

3.4控制器实现_ALU控制信号需求分析

- 微处理器需支持add、sub、and、or、slt运算指令需要利用ALU单元实现运算
- 数据存储指令sw、lw需要通过ALU单元计算存储器地址(加)
- 条件跳转指令beq需要ALU来比较两个寄存器是否相等(减)
- 包含的操作为加，减，与，或，小于设置等5种不同的操作

- 实际的MIPS处理器中， ALU单元利用4根输入控制线的译码决定执行何种操作

输出

输入控制信号编码	操作类型
0000	与
0001	或
0010	加
0110	减
0111	小于设置

输入？

R

I

功能码

操作码

两级译码

- 操作码产生中间译码（ALU操作类型码）
- 中间译码结果与功能码进一步译码产生ALU控制信号

指令	6位操作码中间译码	指令功能	6位功能码	ALU的运算	ALU的控制信号
LW	00	取字	XXXXXX	加	0010
SW	00	存字	XXXXXX	加	0010
BEQ	01	相等跳转	XXXXXX	减	0110
R型指令	10	加	100000	加	0010
R型指令	10	减	100010	减	0110
R型指令	10	与	100100	与	0000
R型指令	10	或	100101	或	0001
R型指令	10	小于设置	101010	小于设置	0111

4位ALU单元控制信号的真值表

- 本指令集6位功能码的前2位完全相同，可以不参与译码

操作码		功能码						ALU控制信号
位1	位0	位5	位4	位3	位2	位1	位0	
0	0	X	X	X	X	X	X	0010
0	1	X	X	X	X	X	X	0110
1	0	X	X	0	0	0	0	0010
1	0	X	X	0	0	1	0	0110
1	0	X	X	0	1	0	0	0000
1	0	X	X	0	1	0	1	0001
1	0	X	X	1	0	1	0	0111

主控制器

写寄存器地址

R型指令

op	rs	rt	rd	shamt	funct
6位	5位	5位	5位	5位	6位

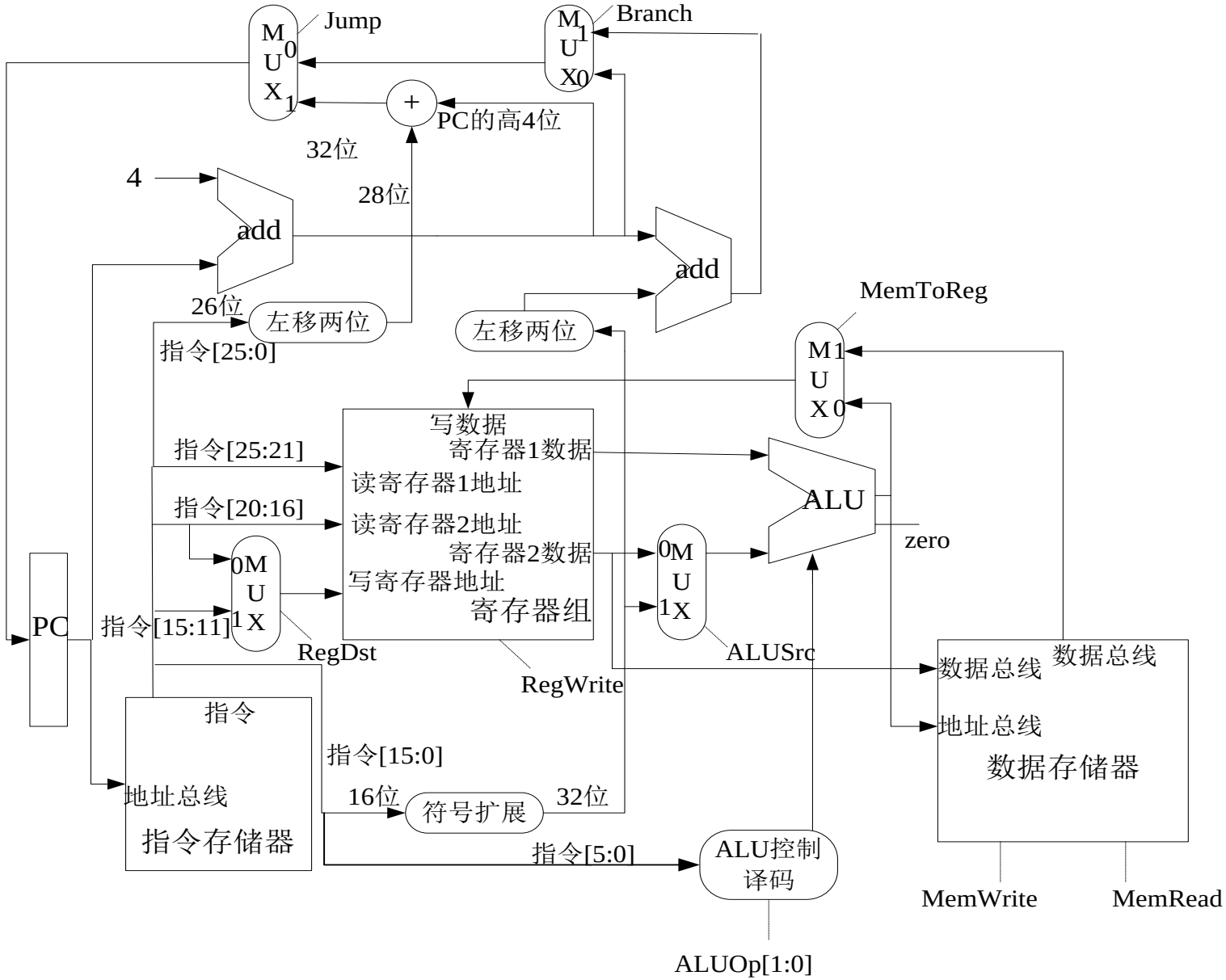
I型指令

op	rs	rt	constant address
6位	5位	5位	16位

J型指令

op	constant address
6位	26位

- 加上写寄存器地址复用器以及ALU控制译码部件的数据通路



控制信号定义

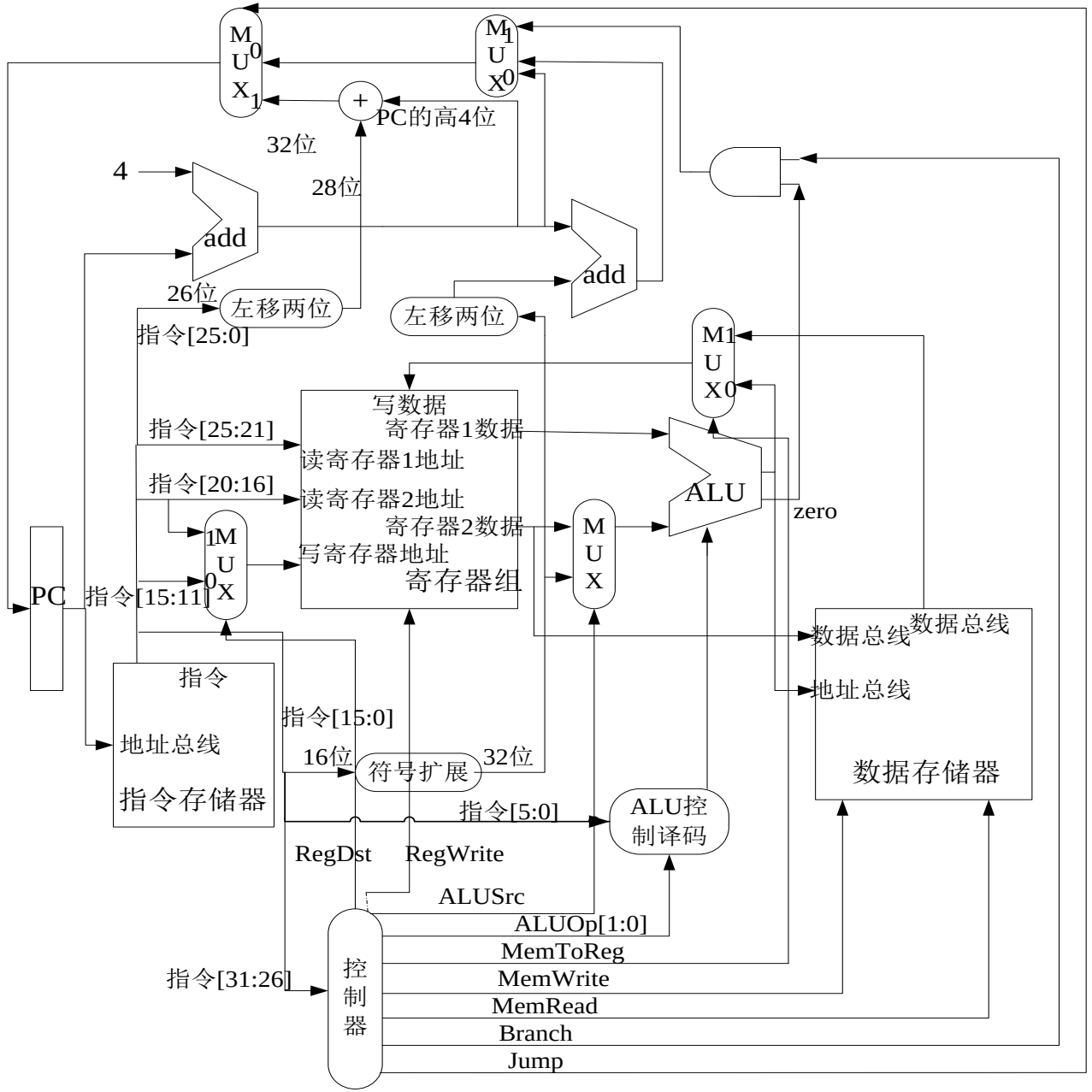
控制信号名称	置1	清0
RegDst	表示写寄存器的地址来自指令[15:11]	来自指令[20:16]
Jump	表示PC的值来自伪直接寻址	来自另一个复用器
Branch	表示下一级复用器的输入来PC相对寻址加法器	来自PC+4
MemToReg	表示写寄存器数据来自存储器数据总线	来自ALU结果
ALUSrc	表示ALU的第二个数据源来自指令[15:0]	来自读寄存器2
RegWrite	将写寄存器数据存入写寄存器地址中	无操作
MemWrite	将写数据总线上的数据写入内存地址单元	无操作
MemRead	将内存单元的内容输出到读数据总线上	无操作

控制信号与指令之间的关系

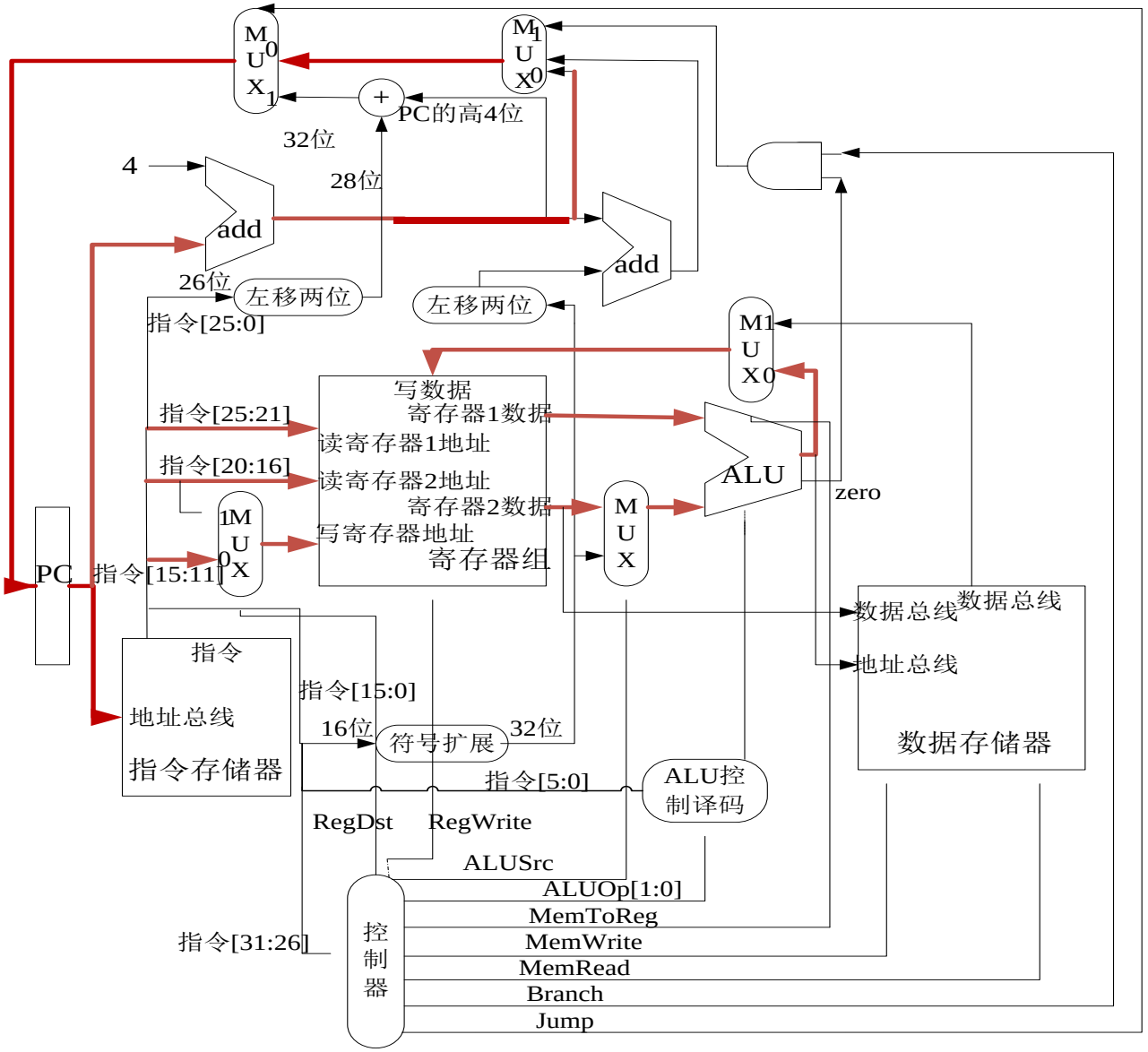
指令	RegDst	Jump	Branch	MemToReg	ALUSrc	RegWrite	MemWrite	MemRead	ALUOp1	ALUOp0
R型	1	0	0	0	0	1	0	0	1	0
lw	0	0	0	1	1	1	0	1	0	0
sw	X	0	0	X	1	0	1	0	0	0
beq	X	0	1	X	0	0	0	0	0	1
j	X	1	0	X	X	0	0	0	X	X

指令操作码与控制信号逻辑关系真值表

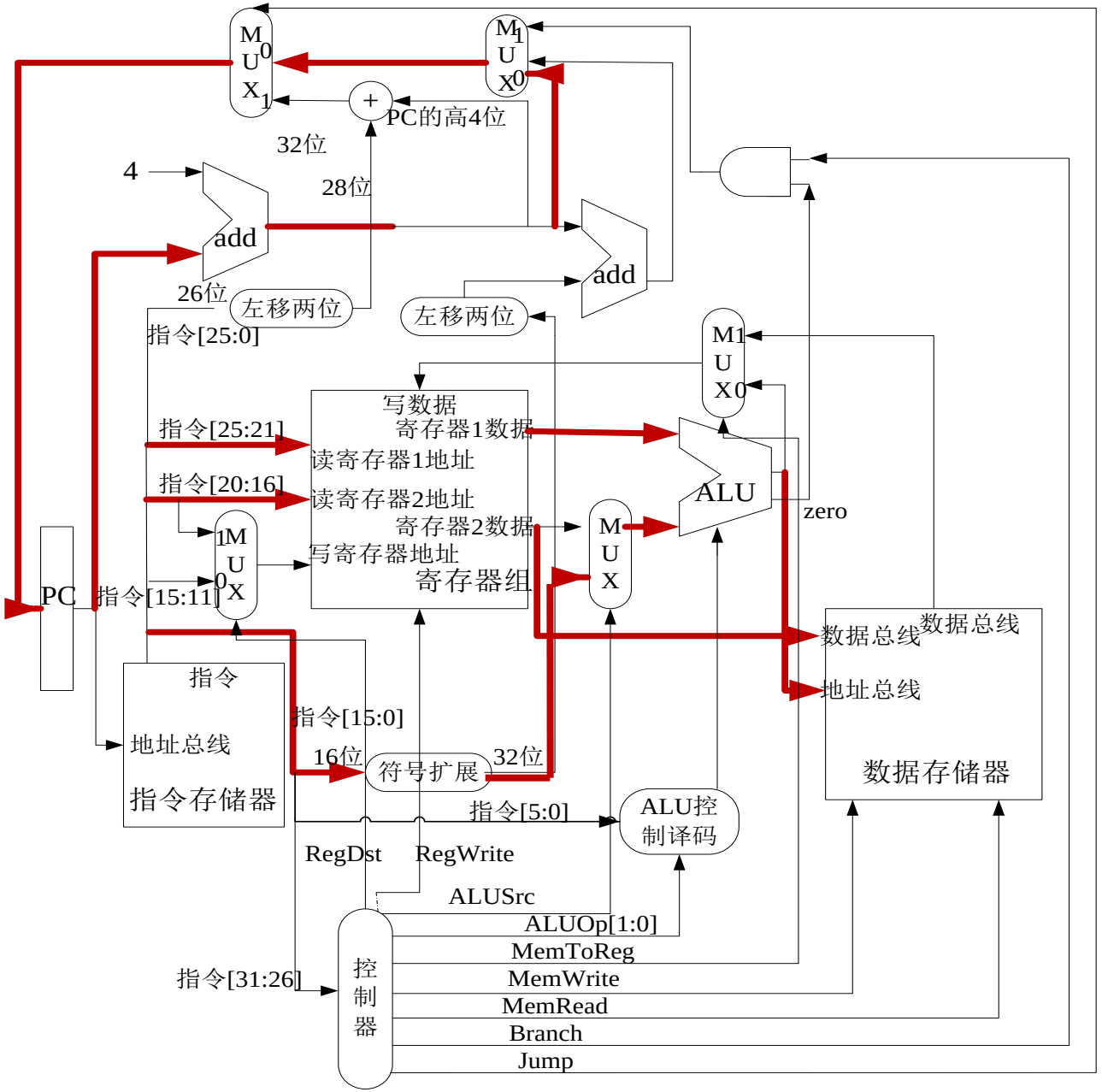
输入输出	信号名称	R型	lw	sw	beq	j
输入	op5	0	1	1	0	0
	op4	0	0	0	0	0
	op3	0	0	1	0	0
	op2	0	0	0	1	0
	op1	0	1	1	0	1
	op0	0	1	1	0	0
输出	ALUop1	1	0	0	0	X
	ALUop0	0	0	0	1	X
	RegDst	1	0	X	X	X
	ALUSrc	0	1	1	0	X
	RegWrite	1	1	0	0	0
	MemWrite	0	0	1	0	0
	MemRead	0	1	0	0	0
	Branch	0	0	0	1	0
	Jump	0	0	0	0	1
	MemToReg	0	1	X	X	X


M
U
X
0
1
+
PC的高4位
32位
4
add
28位
26位
左移两位
指令[25:0]
左移两位
add
M1
U
X0
写数据
寄存器1数据
读寄存器1地址
指令[25:21]
指令[20:16]
读寄存器2地址
寄存器2数据
写寄存器地址
寄存器组
ALU
zero
M
U
X
PC
指令[15:11]
数据总线
数据总线
地址总线
指令
地址总线
指令存储器
指令[15:0]
16位
符号扩展
32位
数据存储器
指令[5:0]
ALU控
制译码
RegDst
RegWrite
ALUSrc
ALUOp[1:0]
MemToReg
MemWrite
MemRead
Branch
Jump
指令[31:26]
控
制
器

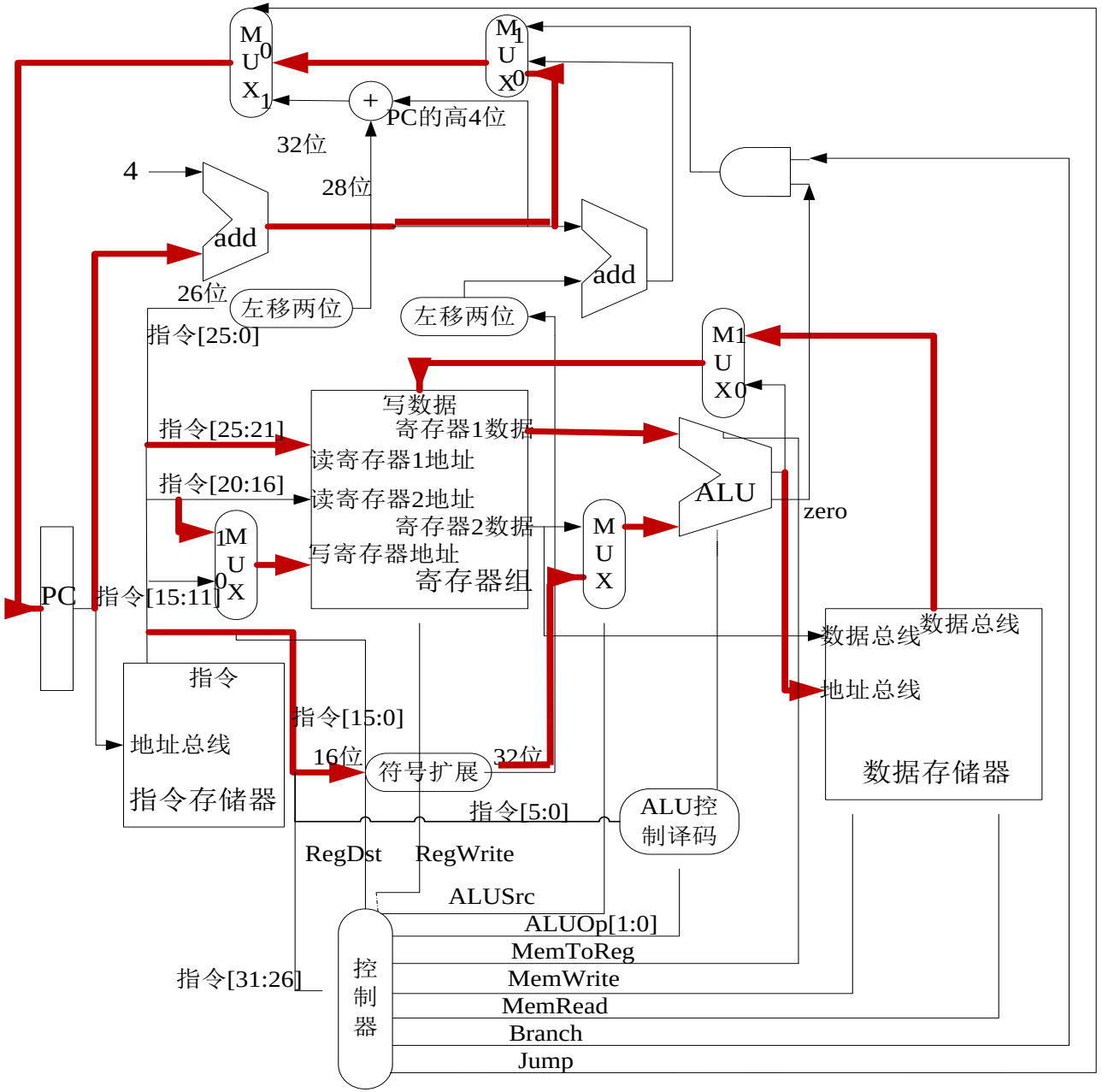
add $t1,$t2,$t3执行过程



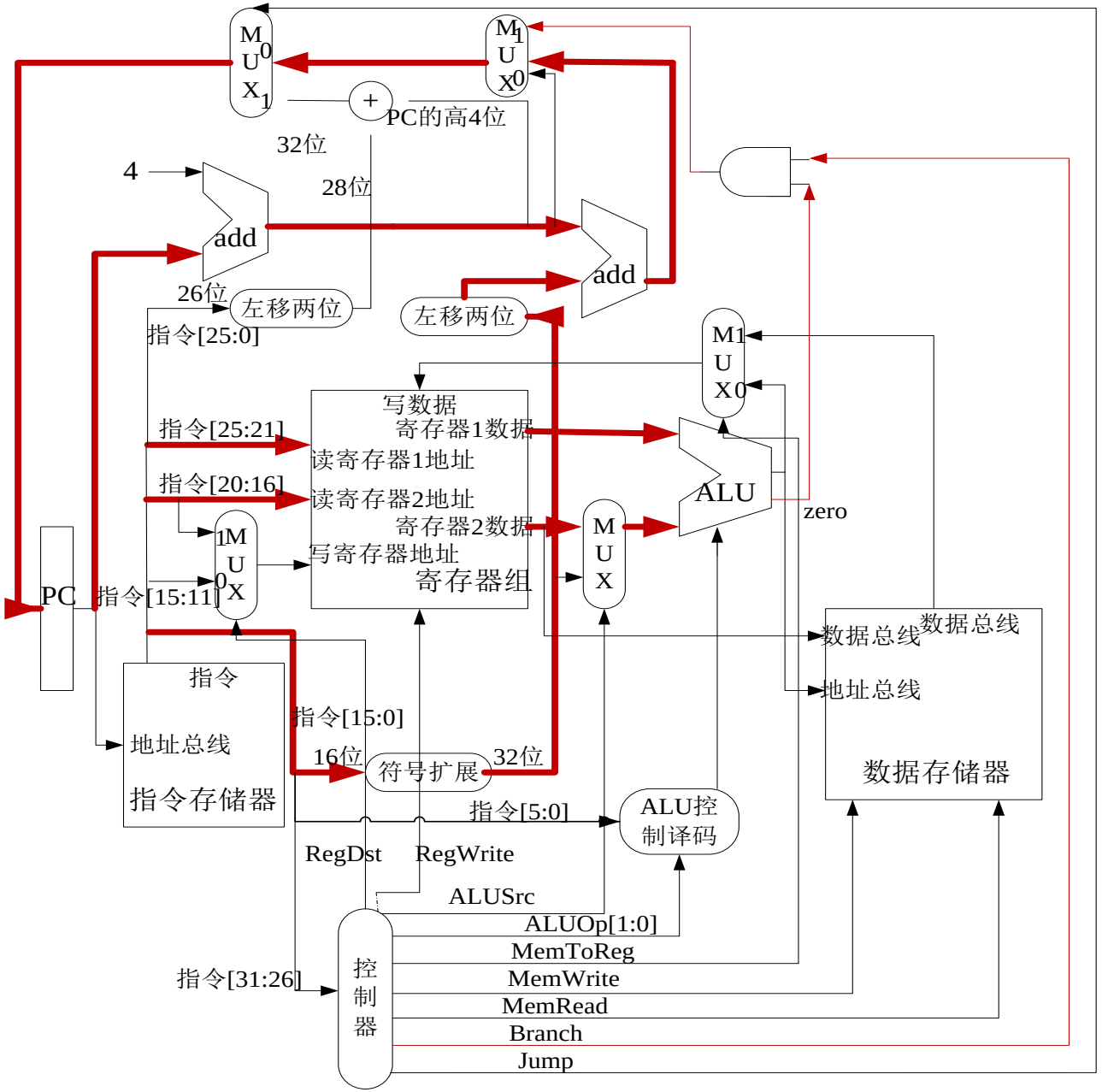
sw $t1,8($t2)


M
U
X
0
1
+
PC的高4位
32位
4
28位
add
add
26位
左移两位
左移两位
指令[25:0]
M1
U
X0
写数据
寄存器1数据
指令[25:21]
读寄存器1地址
指令[20:16]
读寄存器2地址
寄存器2数据
写寄存器地址
寄存器组
ALU
zero
M
U
X
1M
U
0X
PC
指令[15:11]
指令
地址总线
指令存储器
指令[15:0]
16位
符号扩展
32位
数据总线
数据总线
地址总线
数据存储器
指令[5:0]
ALU控
制译码
RegDst
RegWrite
ALUSrc
ALUOp[1:0]
MemToReg
MemWrite
MemRead
Branch
Jump
指令[31:26]
控
制
器

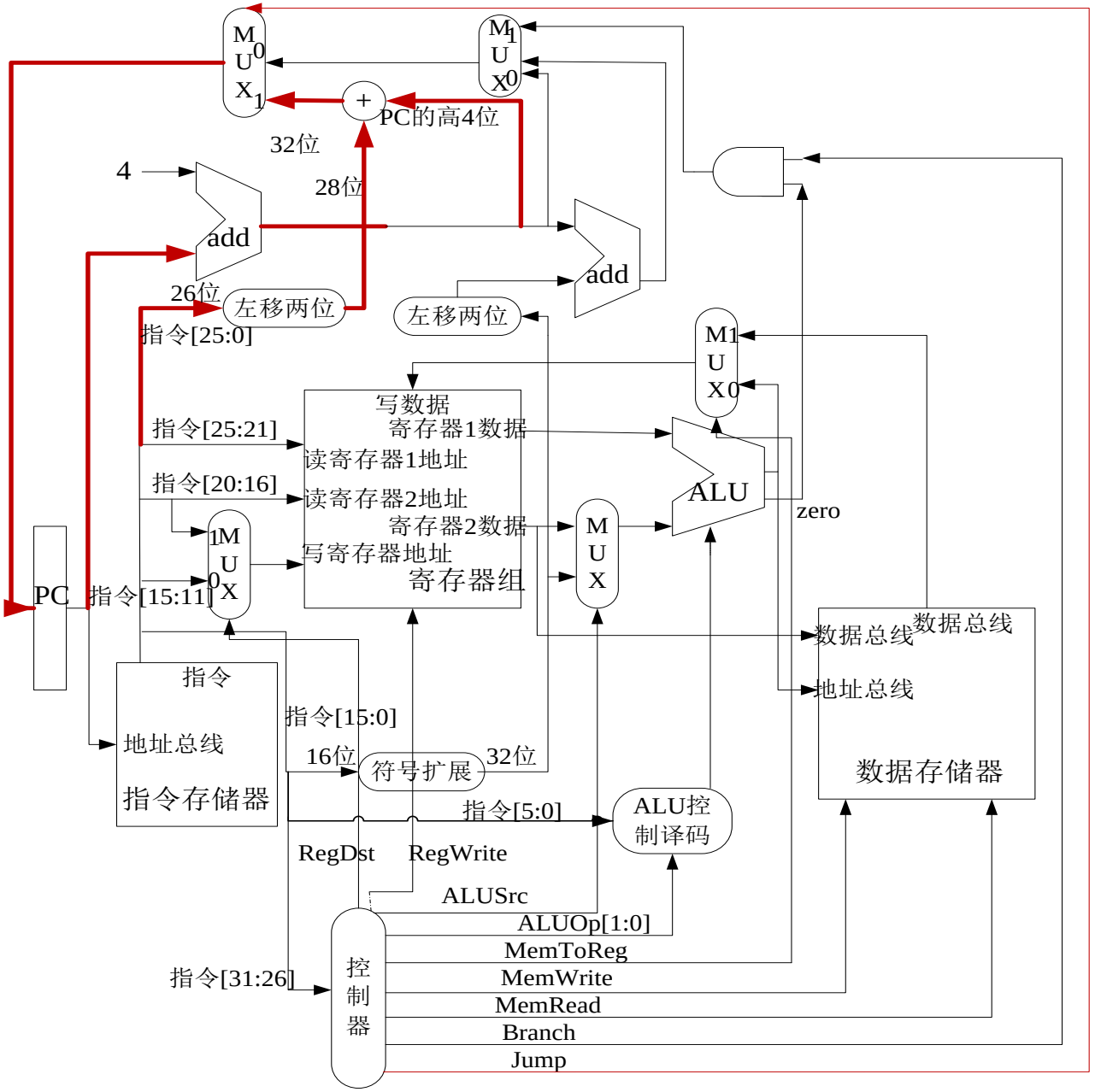
lw $t1,8($t2)


M
U
X
0
1
M1
U
X0
+
PC的高4位
32位
28位
4
add
add
26位
左移两位
左移两位
指令[25:0]
M1
U
X0
写数据
寄存器1数据
读寄存器1地址
指令[25:21]
指令[20:16]
读寄存器2地址
寄存器2数据
写寄存器地址
寄存器组
ALU
zero
1M
U
0X
M
U
X
PC
指令[15:11]
数据总线
数据总线
地址总线
指令
地址总线
指令存储器
指令[15:0]
16位
符号扩展
32位
数据存储器
指令[5:0]
ALU控
制译码
RegDst
RegWrite
ALUSrc
ALUOp[1:0]
MemToReg
MemWrite
MemRead
Branch
Jump
指令[31:26]
控
制
器

beq $t1,$t2,L1


M
U
X
0
1
M1
U
X0
+
PC的高4位
32位
28位
4
add
add
26位
左移两位
左移两位
指令[25:0]
M1
U
X0
写数据
寄存器1数据
读寄存器1地址
指令[25:21]
指令[20:16]
读寄存器2地址
寄存器2数据
写寄存器地址
寄存器组
ALU
zero
1M
U
0X
M
U
X
PC
指令[15:11]
数据总线
数据总线
地址总线
指令
地址总线
指令存储器
指令[15:0]
16位
符号扩展
32位
数据存储器
指令[5:0]
ALU控
制译码
RegDst
RegWrite
ALUSrc
ALUOp[1:0]
MemToReg
MemWrite
MemRead
Branch
Jump
指令[31:26]
控
制
器

j L1


M
U
X
0
1
+
PC的高4位
32位
4
28位
add
add
26位
左移两位
左移两位
指令[25:0]
M1
U
X0
写数据
寄存器1数据
读寄存器1地址
读寄存器2地址
寄存器2数据
写寄存器地址
寄存器组
指令[25:21]
指令[20:16]
指令[15:11]
PC
ALU
zero
M
U
X
数据总线
地址总线
数据存储器
指令
地址总线
指令存储器
指令[15:0]
16位
符号扩展
32位
指令[5:0]
ALU控
制译码
RegDst
RegWrite
ALUSrc
ALUOp[1:0]
MemToReg
MemWrite
MemRead
Branch
Jump
指令[31:26]
控
制
器

作业

- 采用Verilog语言实现支持基本指令集控制器的译码设计
  - 输入信号为：opcode[5:0]，func[5:0]
  - 输出信号为：ALUctr[3:0],以及5个mux的控制信号，寄存器和存储器写控制信号
  - 采用两级译码（两个模块）
    - 模块1根据opcode[5:0]->ALUop[1:0]
    - 模块2根据ALUop[1:0],func[5:0]-> ALUctr[3:0], 5个mux的控制信号以及寄存器和存储器写控制信号