第三章 微处理器

学习目标

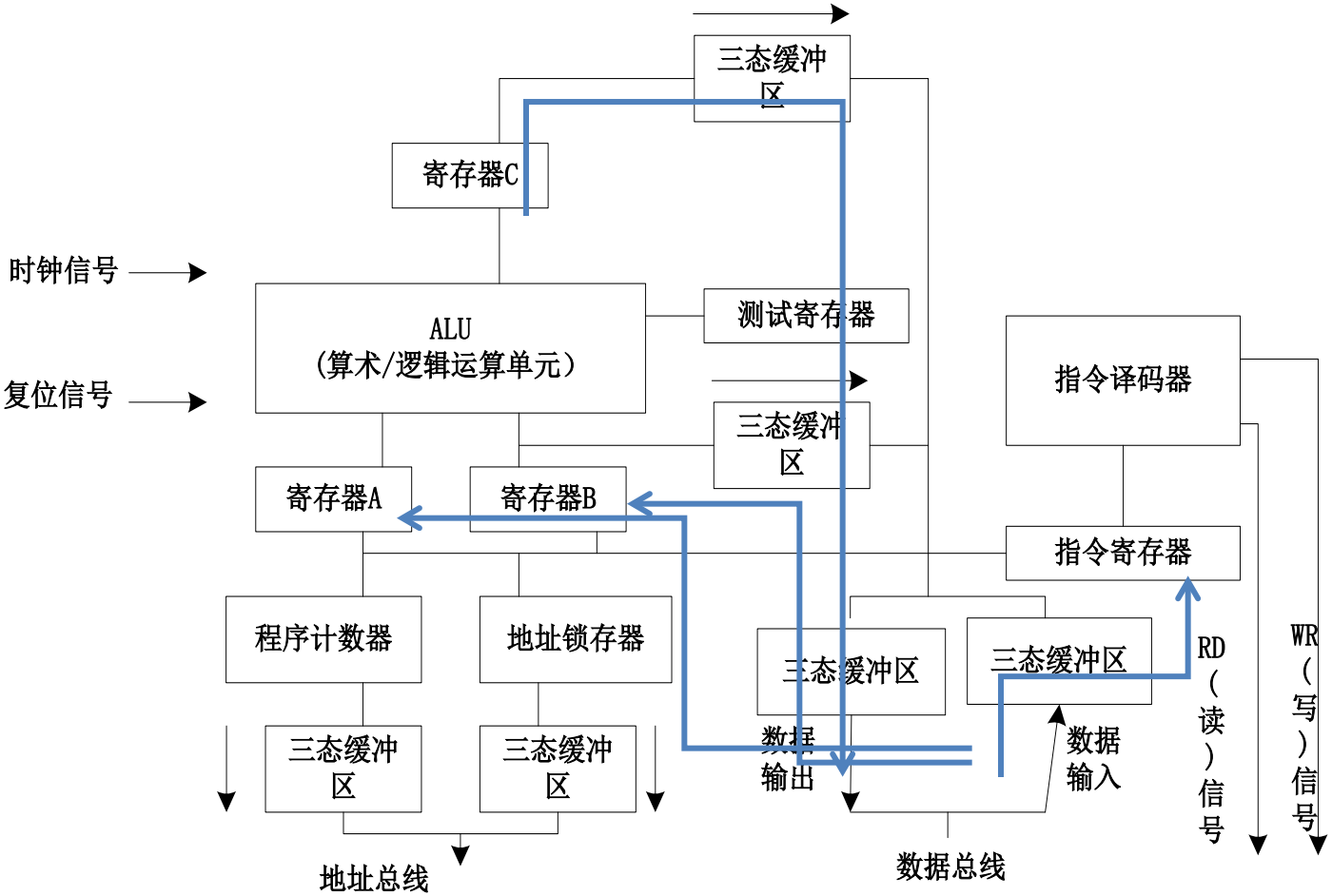
- 了解微处理器的基本构成
- 能设计执行简单MIPS指令集的微处理器
- 了解现代微处理器的流水线技术和超标量技术原理
- 理解微处理器异常处理机制
- 掌握MicroBlaze微处理器的应用

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。现代微处理器包含完整的浮点处理器，它可以对很大的浮点数执行非常复杂的浮点运算。
  - 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。

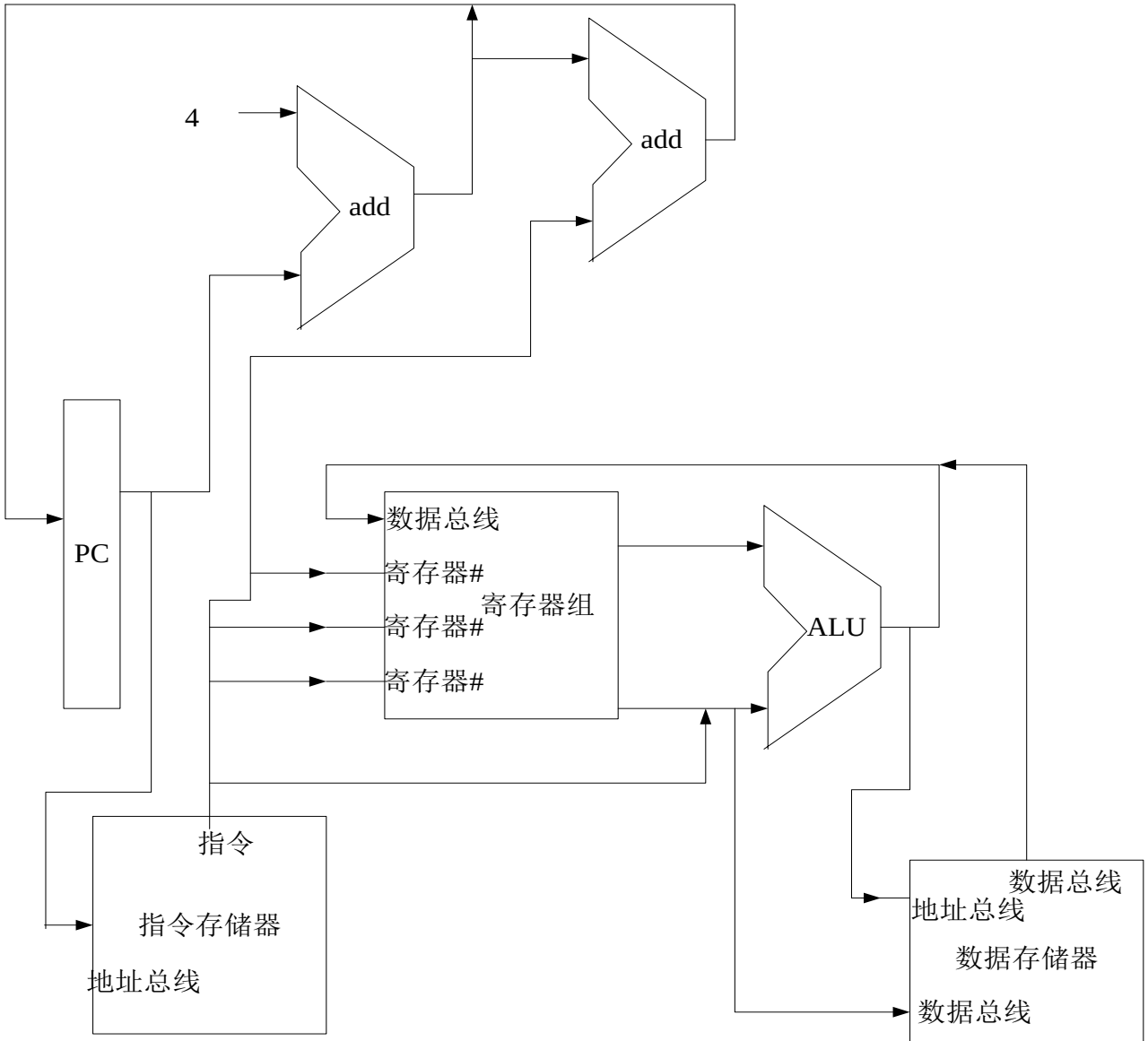
简单微处理器的基本构成



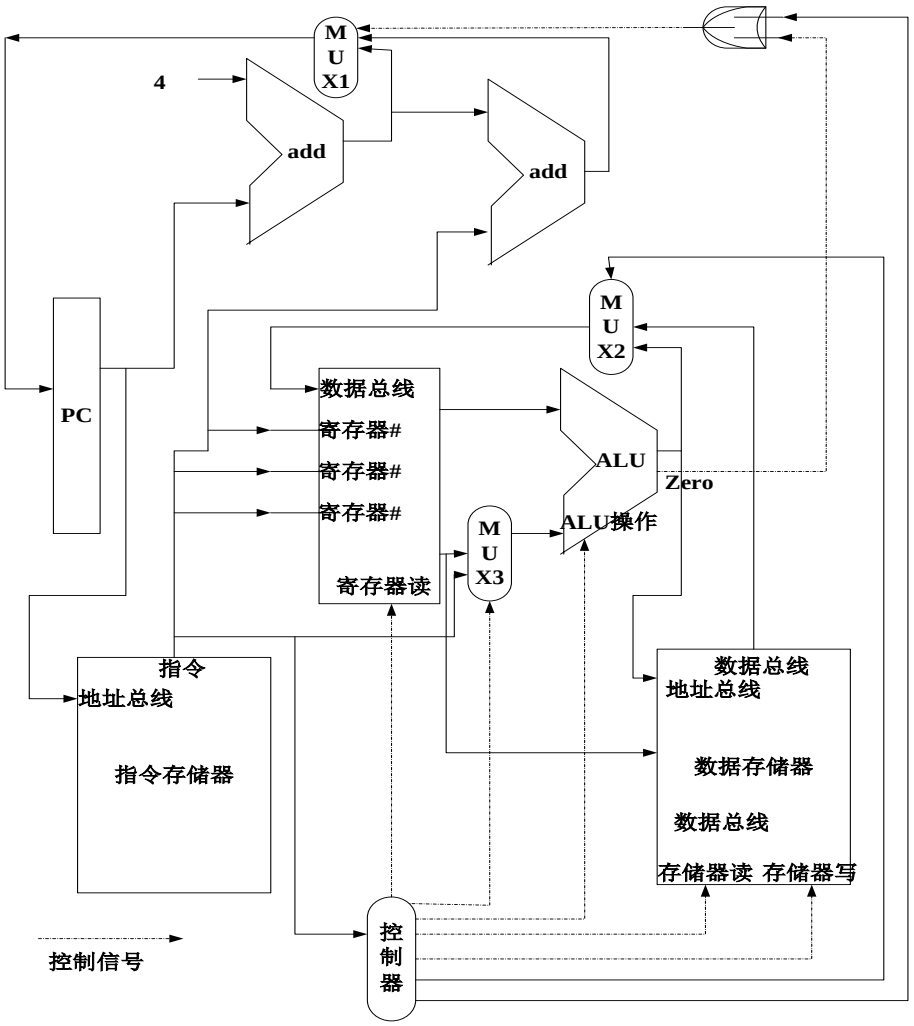
3.2简单MIPS指令集微处理器基本构成

- 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

简单指令集MIPS微处理器数据通路简化结构

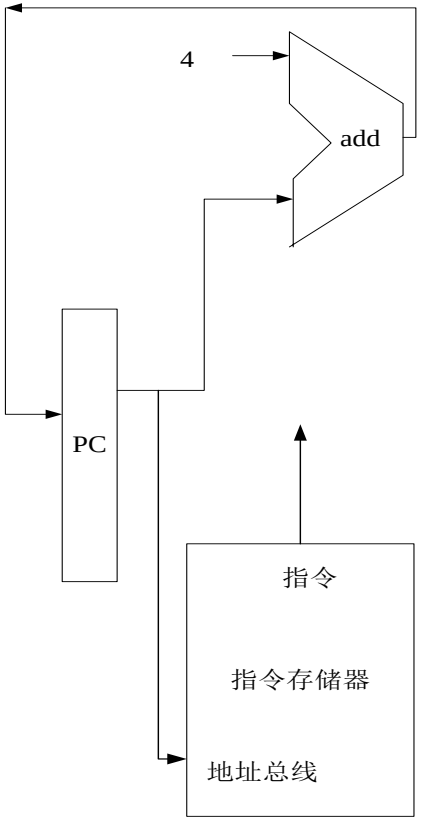


简单指令集MIPS处理器的逻辑实现

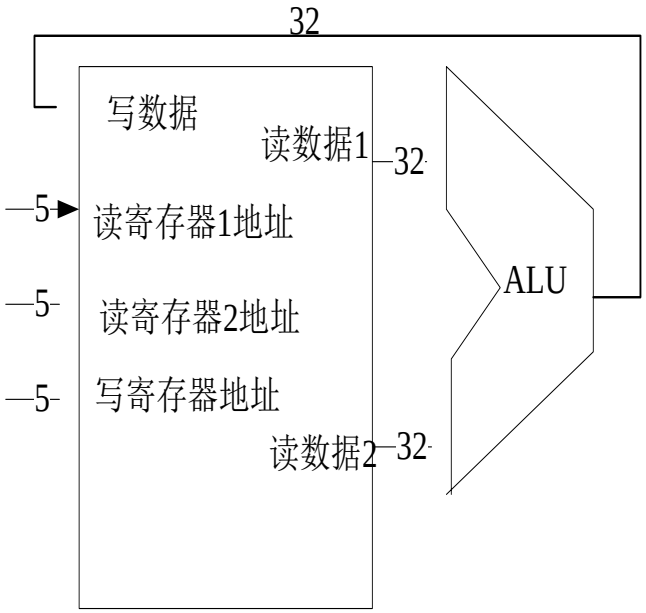


3.3数据通路实现原理

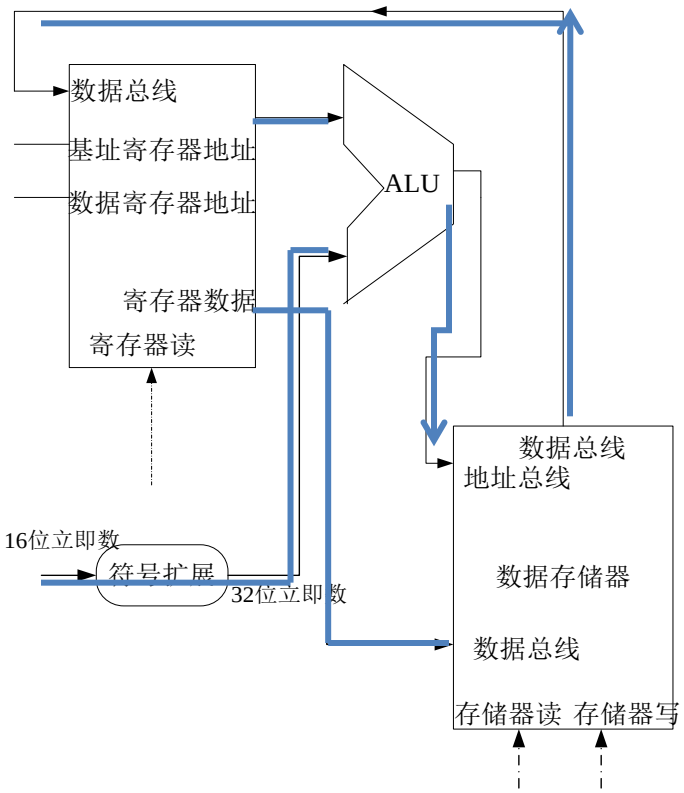
- 指令获取部件



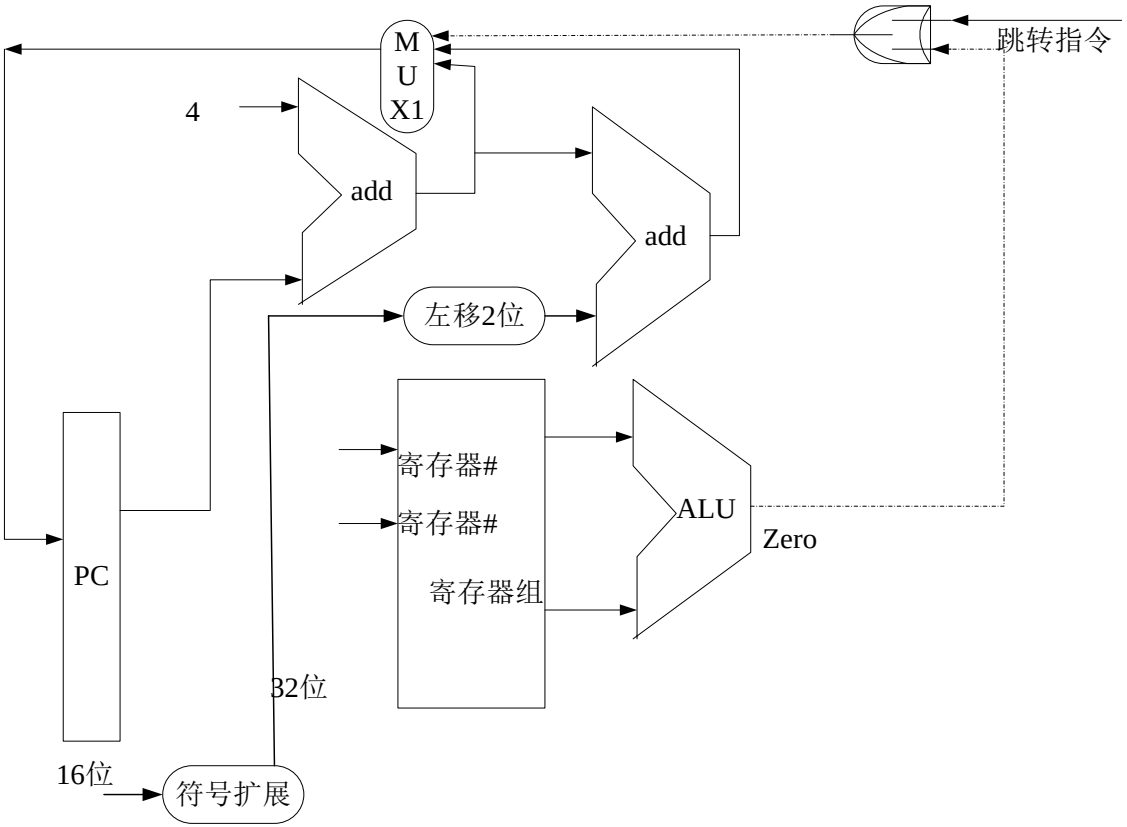
R型指令实现部件



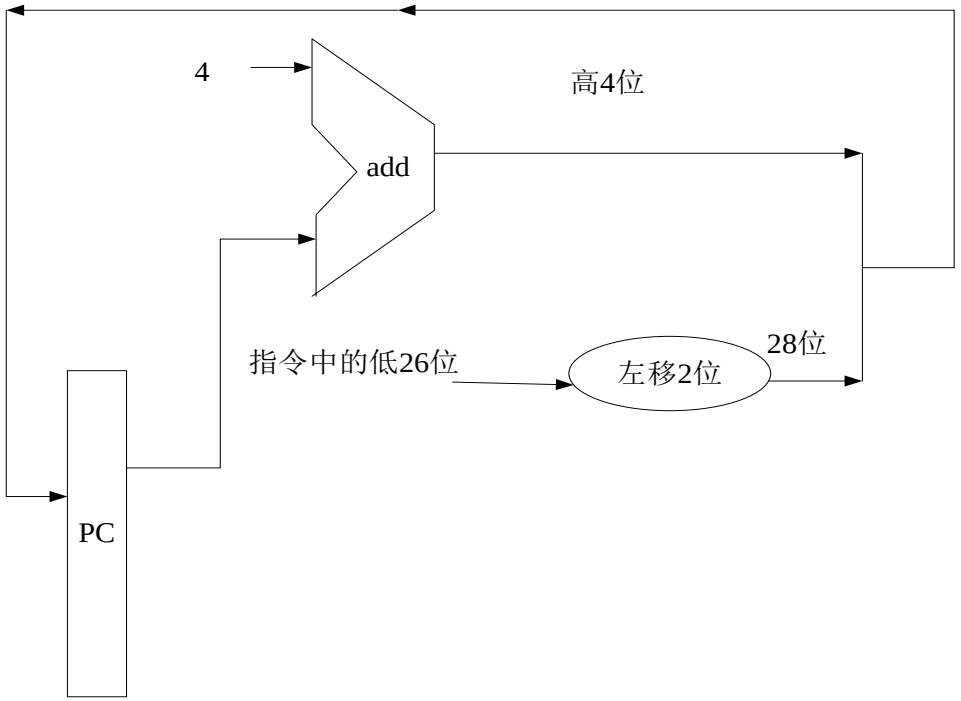
存储器数据存取部件


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

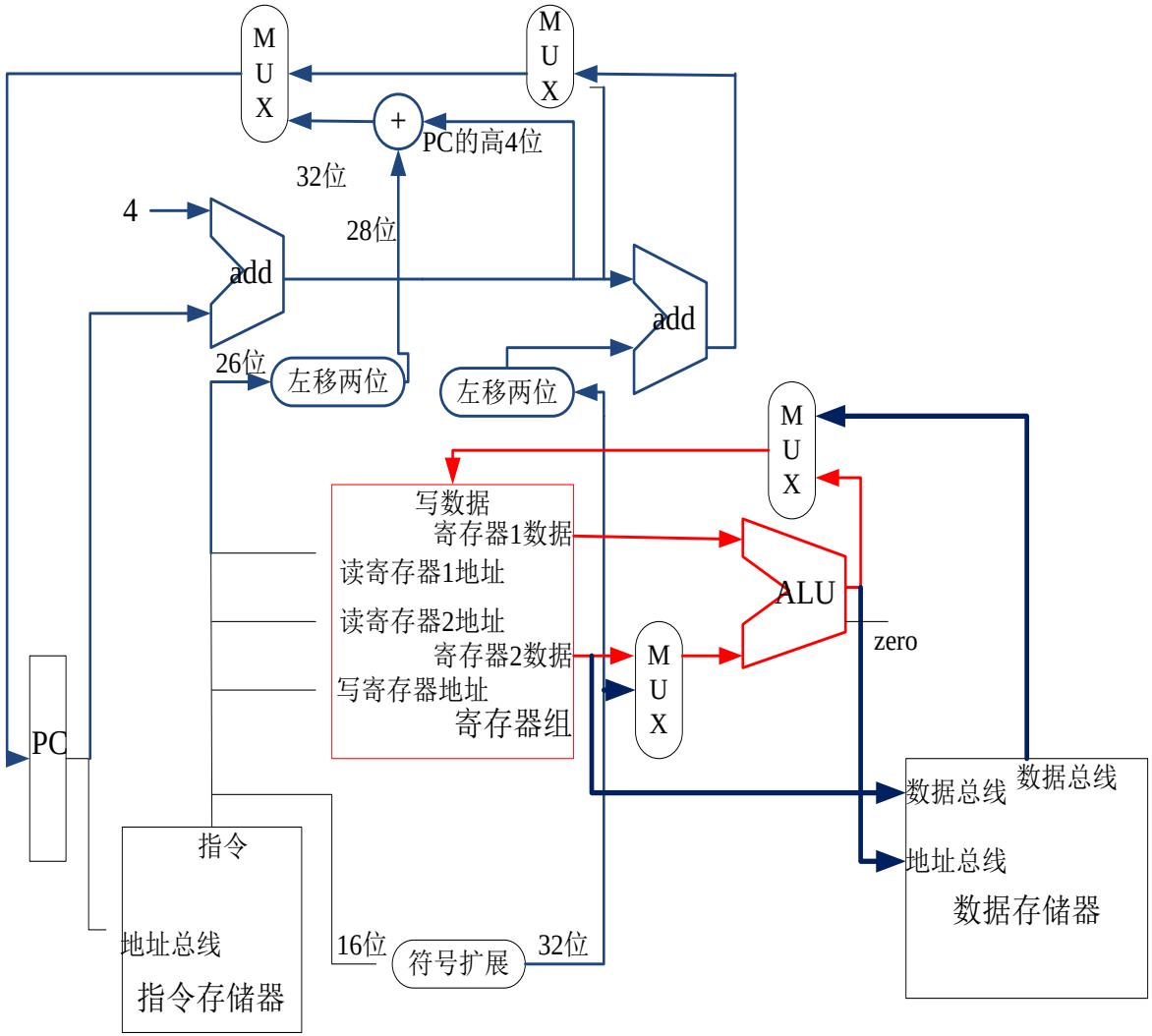
条件跳转控制



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

- 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

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

6位操作码，R型指令，进一步采用6位功能码来表示R型指令的具体操作

sw，lw以及beq指令没有功能码，只有操作码

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	2位操作码	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

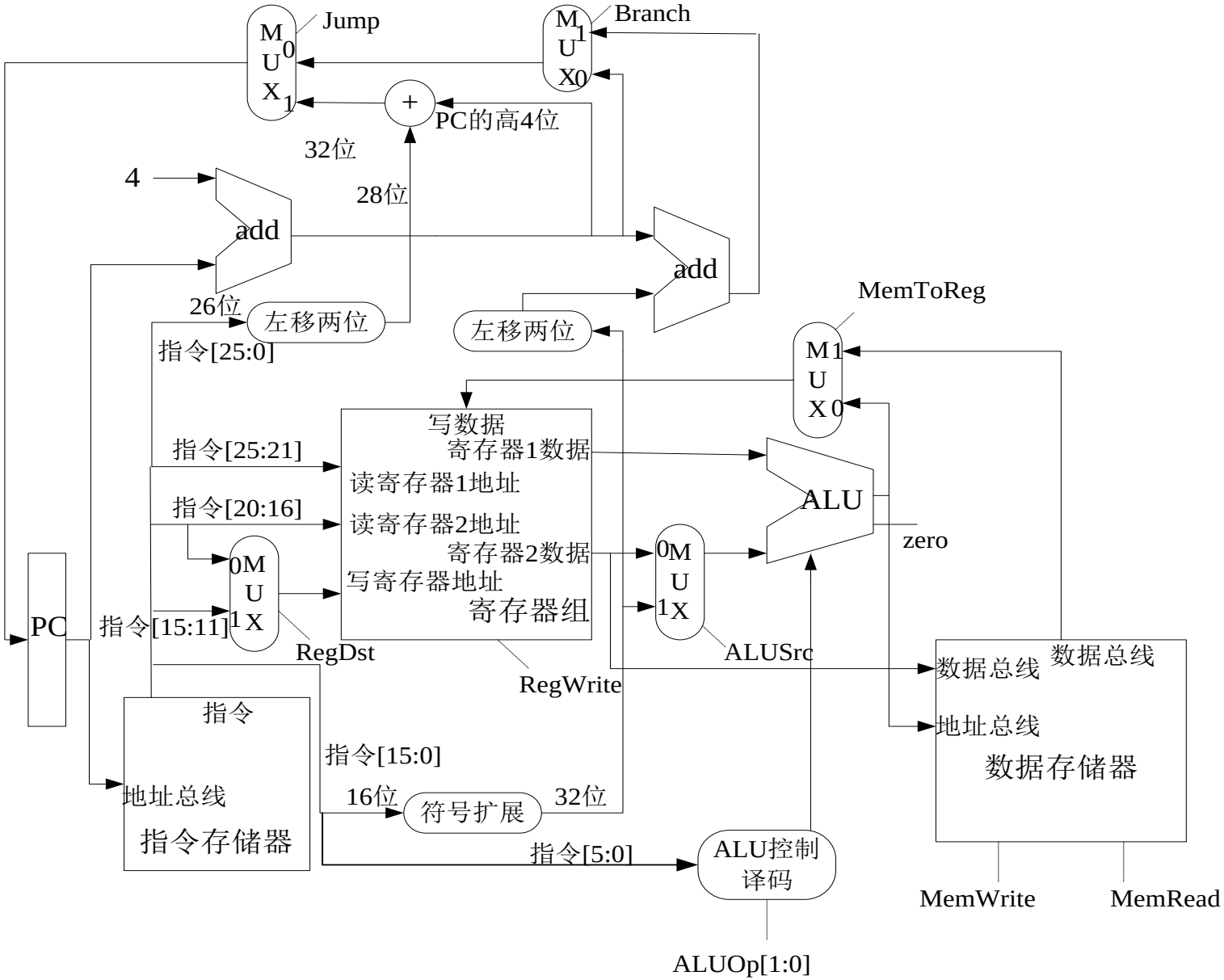
I型指令

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

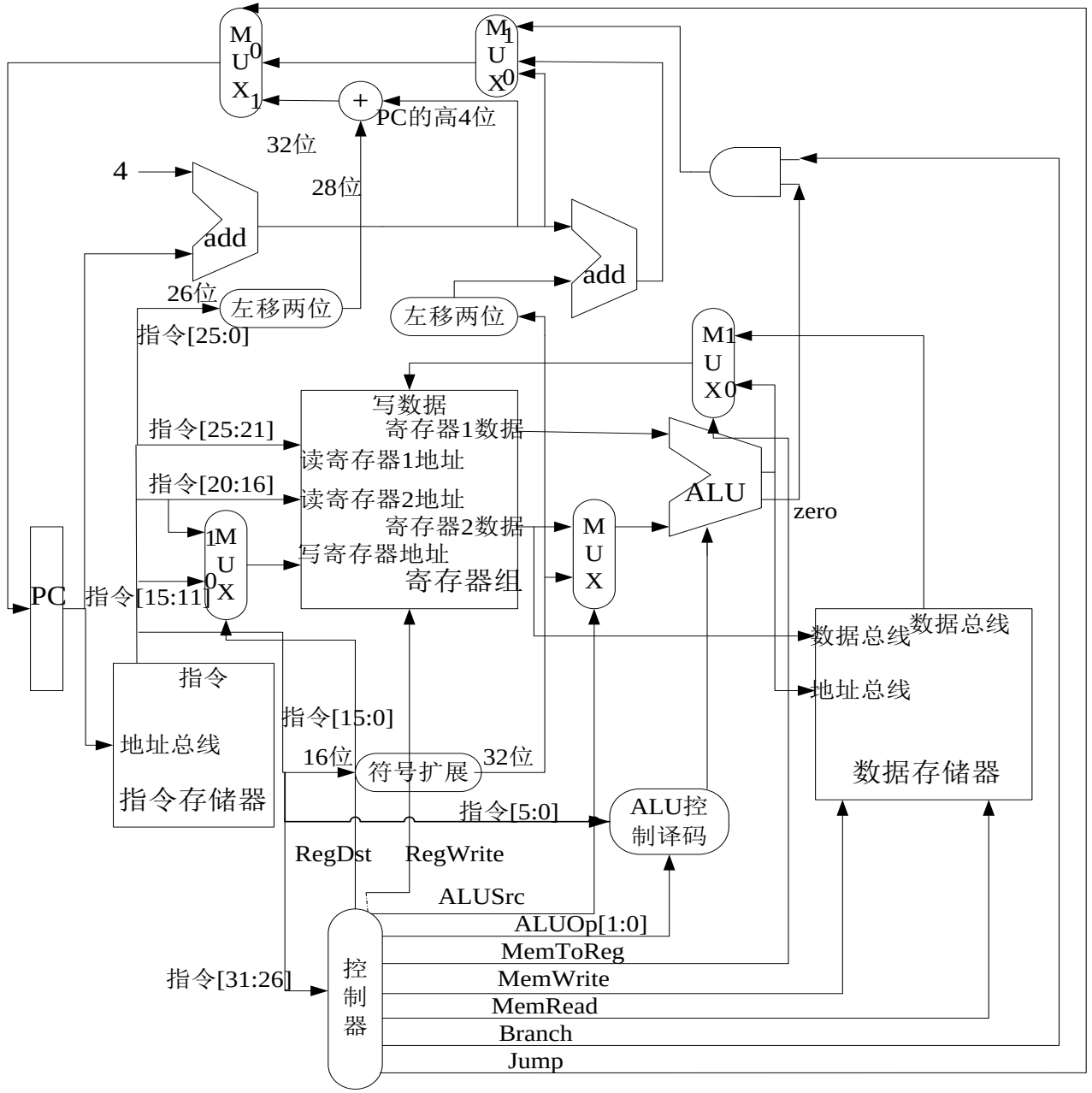
控制信号名称	置1	清0
RegDst	表示写寄存器的地址来自指令[15:11]	来自指令[20:16]
Jump	表示PC的值来自伪直接寻址	来自另一个复用器
Branch	表示下一级复用器的输入来PC相对寻址加法器	来自PC+4
MemToReg	表示写寄存器数据来自存储器数据总线	来自ALU结果
ALUSrc	表示ALU的第二个数据源来自指令[15:0]	来自读寄存器2
RegWrite	将写寄存器数据存入写寄存器地址中	无操作
MemWrite	将写数据总线上的数据写入内存地址单元	无操作
MemRead	将内存单元的内容输出到读数据总线上	无操作

控制信号与指令之间的关系

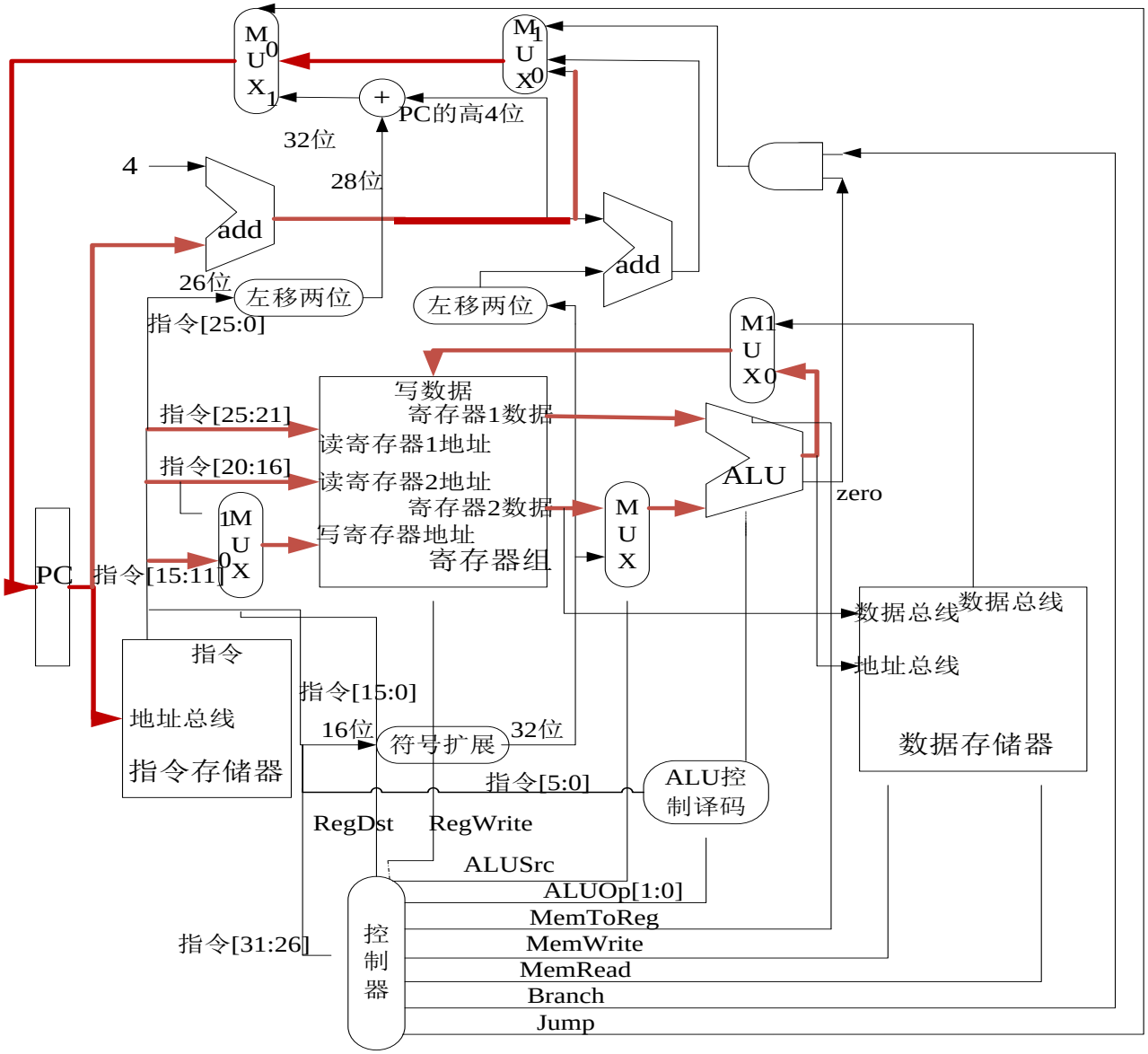
指令	RegDst	Jump	Branch	MemToReg	ALUSrc	RegWrite	MemWrite	MemRead	ALUOp1	ALUOp0
R型	1	0	0	0	0	1	0	0	1	0
lw	0	0	0	1	1	1	0	1	0	0
sw	X	0	0	X	1	0	1	0	0	0
beq	X	0	1	X	0	0	0	0	0	1
j	X	1	0	X	X	0	0	0	X	X

指令操作码与控制信号逻辑关系真值表

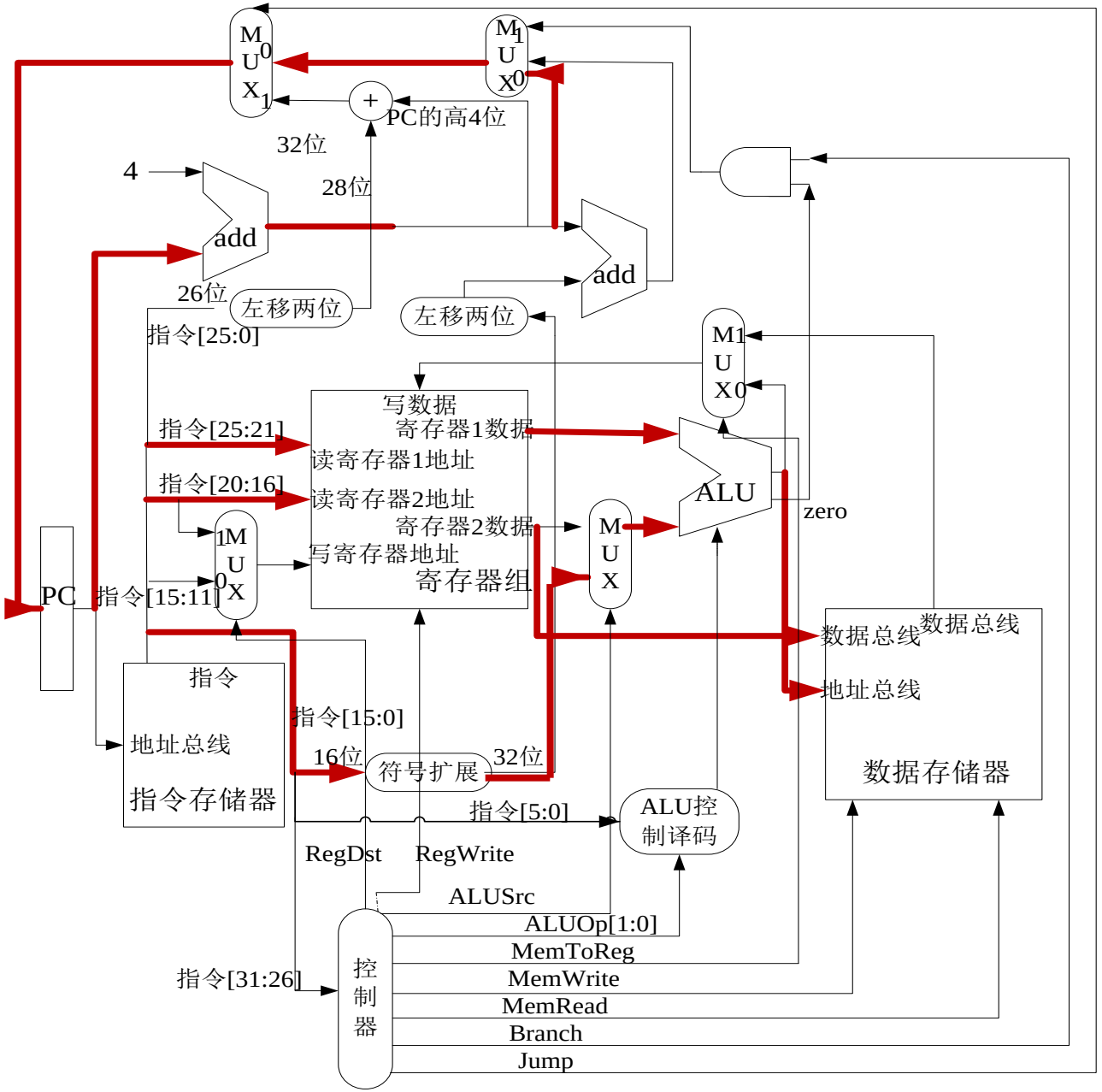
输入输出	信号名称	R型	lw	sw	beq	j
输入	op5	0	1	1	0	0
	op4	0	0	0	0	0
	op3	0	0	1	0	0
	op2	0	0	0	1	0
	op1	0	1	1	0	1
	op0	0	1	1	0	0
输出	ALUop1	1	0	0	0	X
	ALUop0	0	0	0	1	X
	RegDst	1	0	X	X	X
	ALUSrc	0	1	1	0	X
	RegWrite	1	1	0	0	0
	MemWrite	0	0	1	0	0
	MemRead	0	1	0	0	0
	Branch	0	0	0	1	0
	Jump	0	0	0	0	1
	MemToReg	0	1	X	X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


M
U
X
0
1
M
U
X
1
0
+
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]
M
U
X
1
0
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
M
U
X
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

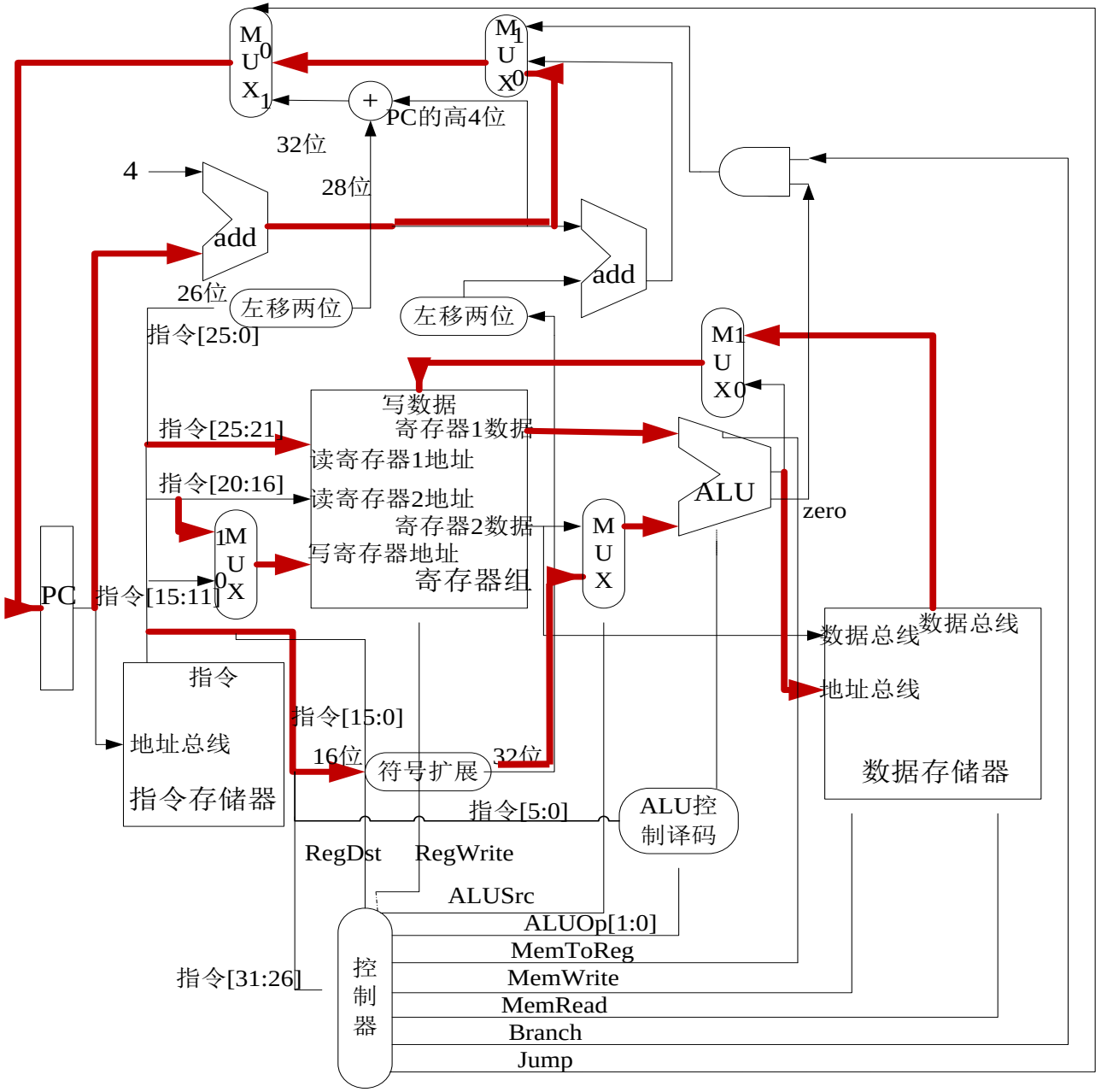
add $t1,$t2,$t3执行过程



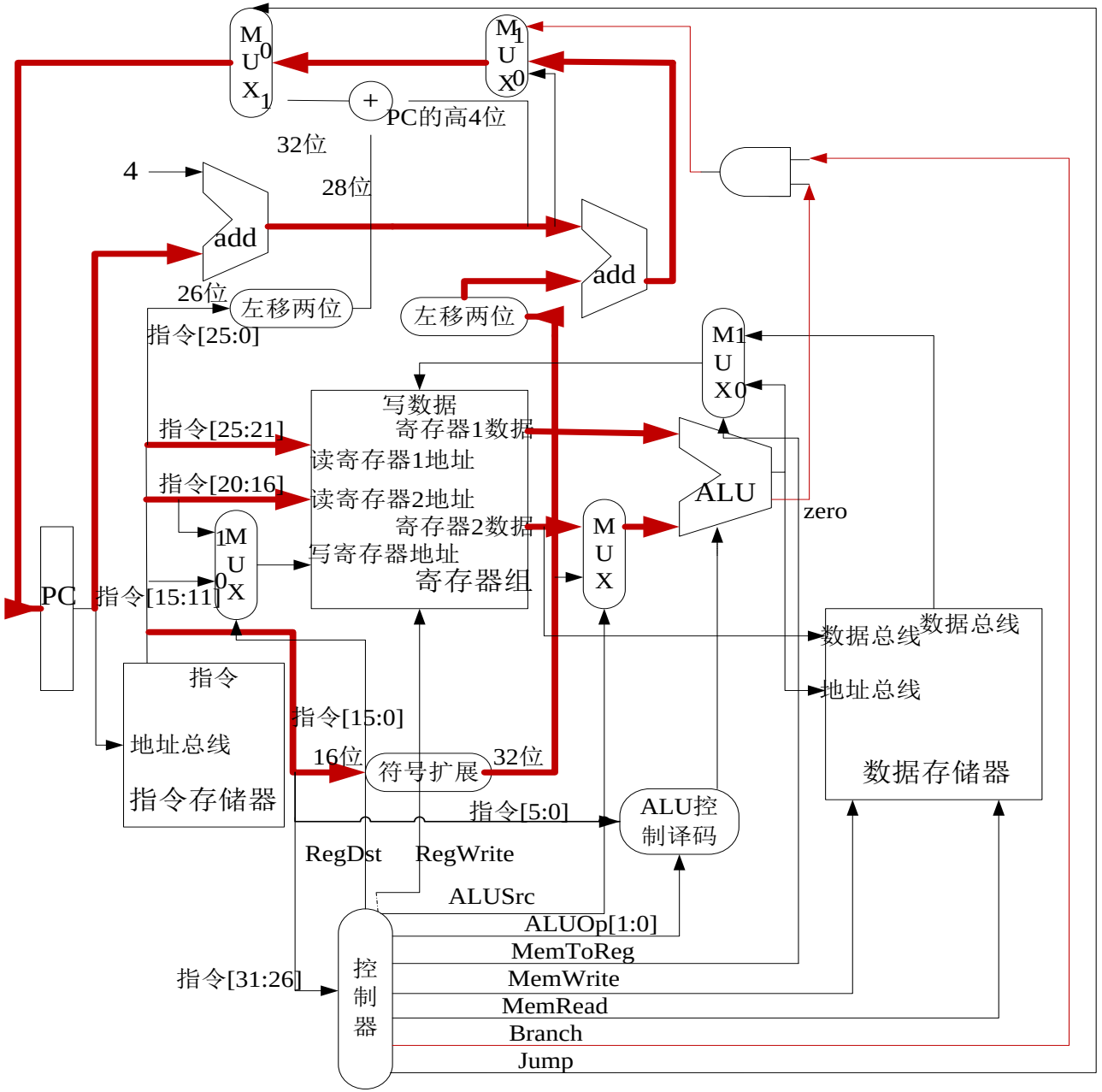
sw $t1,8($t2)

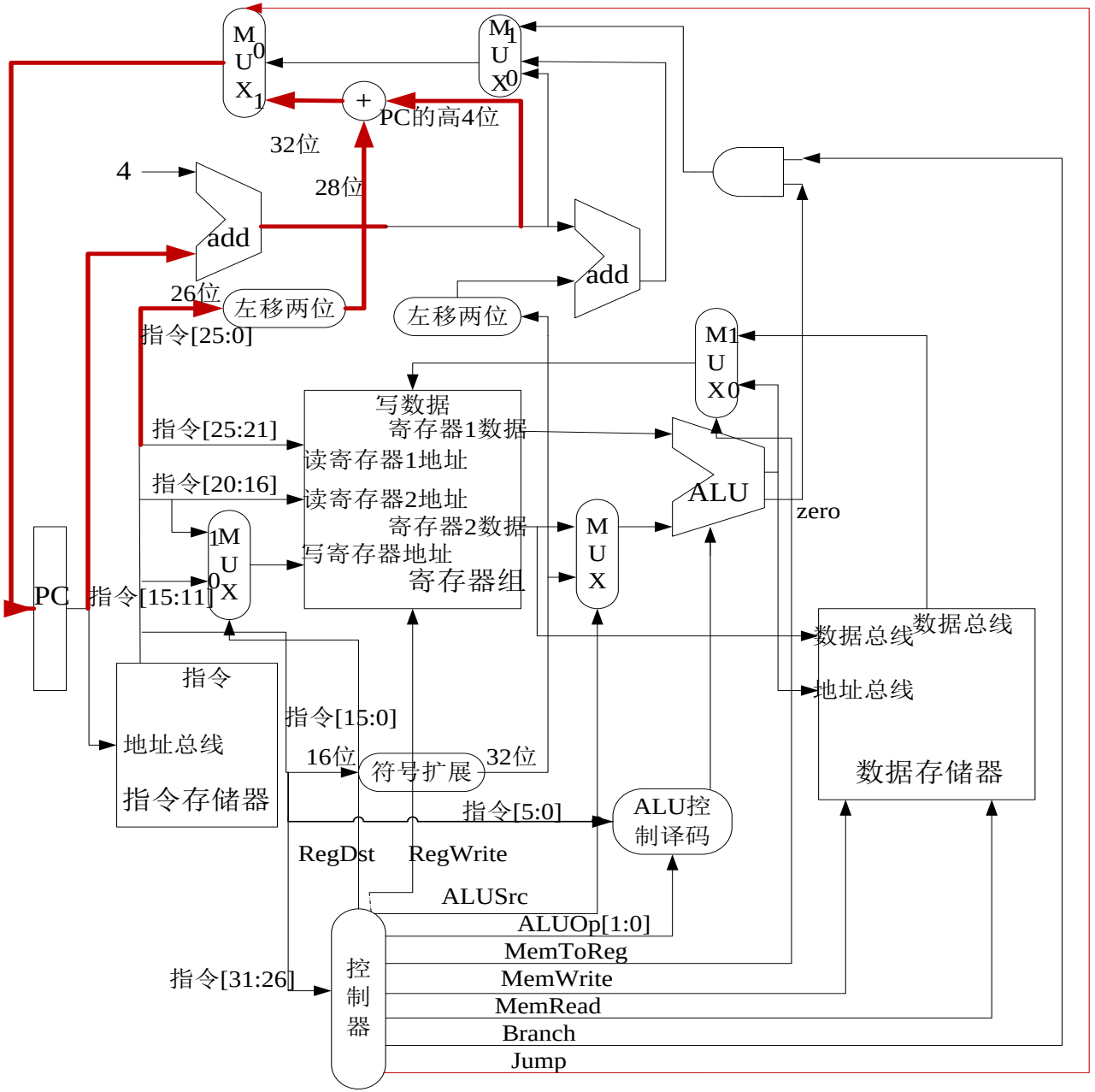

M
U
X
0
1
+
PC的高4位
32位
4
28位
add
add
26位
左移两位
左移两位
指令[25:0]
M1
U
X0
写数据
寄存器1数据
读寄存器1地址
读寄存器2地址
寄存器2数据
写寄存器地址
寄存器组
指令[25:21]
指令[20:16]
ALU
zero
M
U
X
PC
指令[15:11]
数据总线
数据总线
地址总线
数据存储器
指令
地址总线
指令存储器
指令[15:0]
16位
符号扩展
32位
指令[5:0]
ALU控
制译码
RegDst
RegWrite
ALUSrc
ALUOp[1:0]
MemToReg
MemWrite
MemRead
Branch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

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]
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
M1
U
X0
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

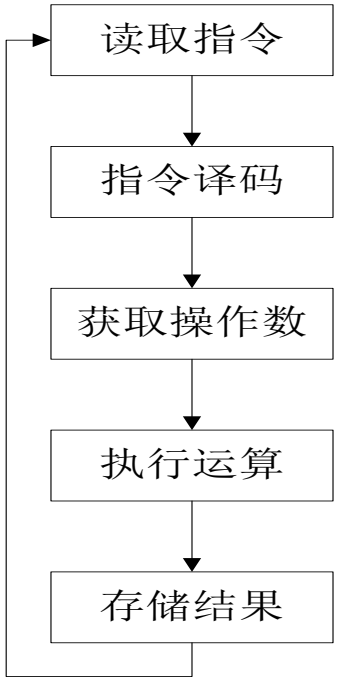

M
U
X
0
1
M1
U
X0
+
PC的高4位
32位
28位
4
add
add
26位
左移两位
左移两位
指令[25:0]
M1
U
X0
写数据
寄存器1数据
读寄存器1地址
读寄存器2地址
寄存器2数据
写寄存器地址
寄存器组
指令[25:21]
指令[20:16]
1M
U
0X
M
U
X
ALU
zero
PC
指令[15:11]
数据总线
数据总线
地址总线
数据存储器
指令
地址总线
指令存储器
指令[15:0]
16位
符号扩展
32位
指令[5:0]
ALU控
制译码
RegDst
RegWrite
ALUSrc
ALUOp[1:0]
MemToReg
MemWrite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

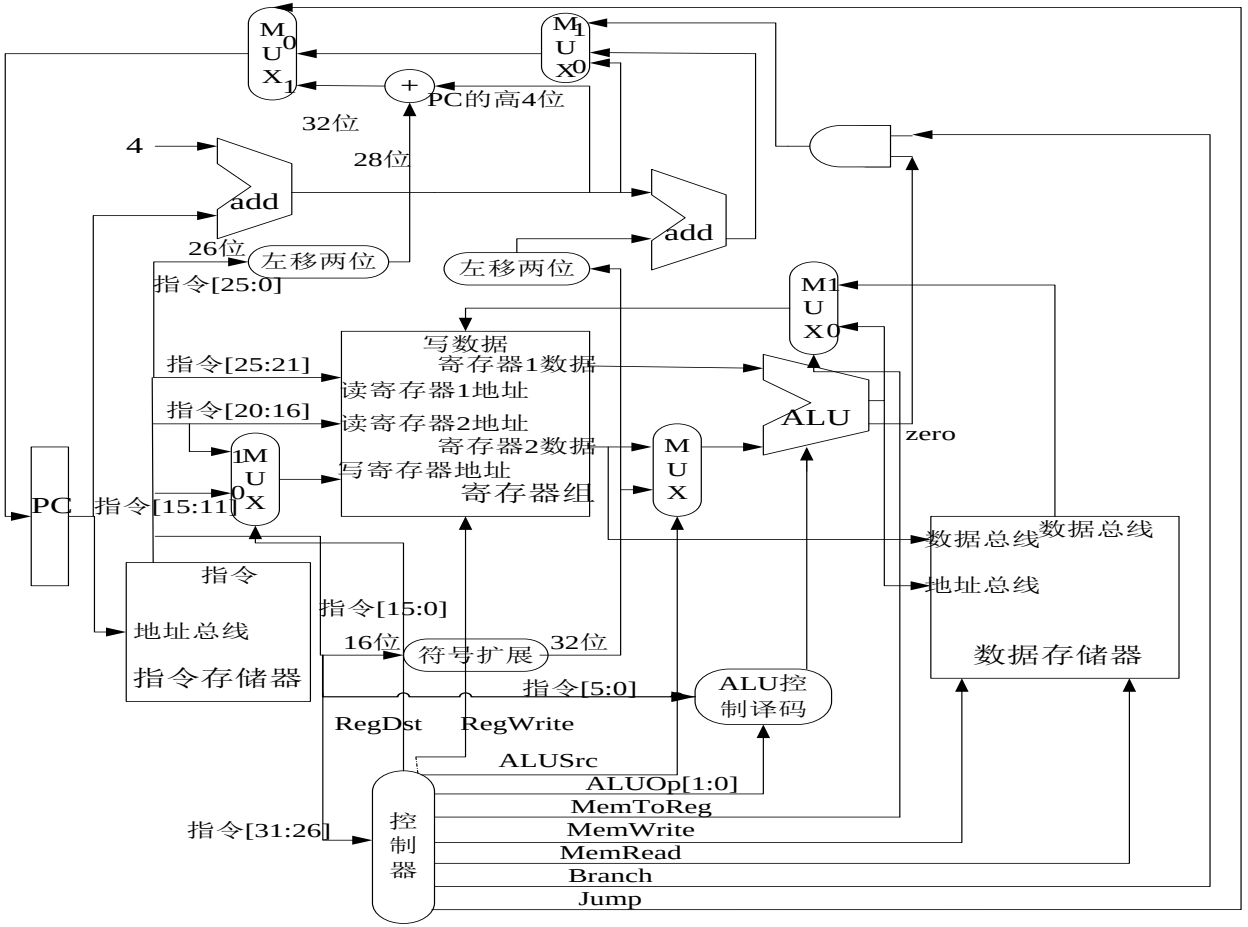

M
U
X
0
1
M
U
X
1
0
+
PC的高4位
32位
4
28位
add
add
26位
左移两位
左移两位
指令[25:0]
M
U
X
1
0
写数据
寄存器1数据
读寄存器1地址
读寄存器2地址
寄存器2数据
写寄存器地址
寄存器组
指令[25:21]
指令[20:16]
指令[15:11]
M
U
X
M
U
X
ALU
zero
PC
指令
地址总线
指令存储器
指令[15:0]
16位
符号扩展
32位
指令[5:0]
ALU控
制译码
数据总线
数据总线
地址总线
数据存储器
RegDst
RegWrite
ALUSrc
ALUOp[1:0]
MemToReg
MemWrite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

3.5 现代微处理器新技术

- 流水线技术
- 超标量技术
- 异常处理机制

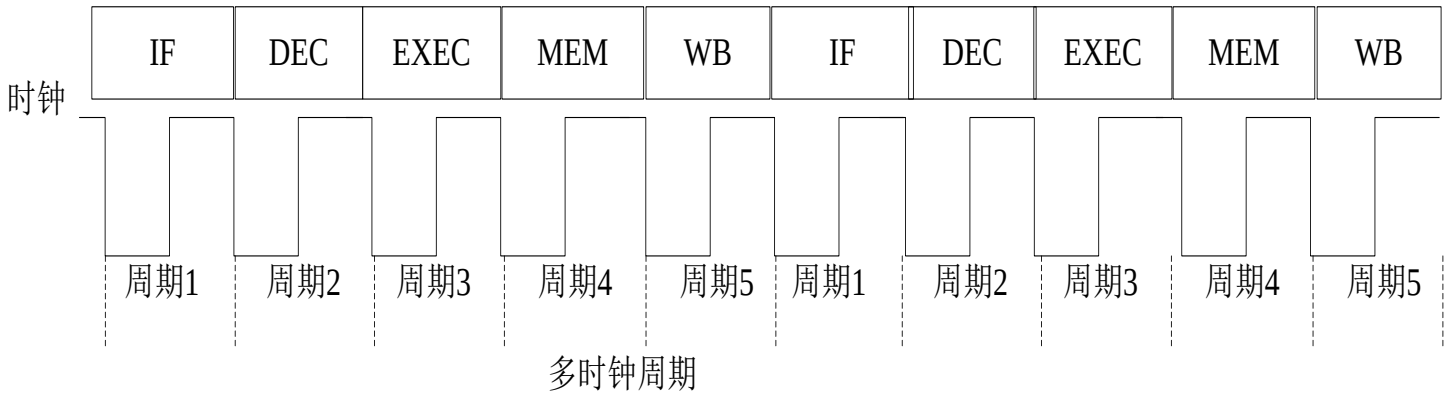
流水线技术


读取指令
指令译码
获取操作数
执行运算
存储结果


M U X
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]
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]
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
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

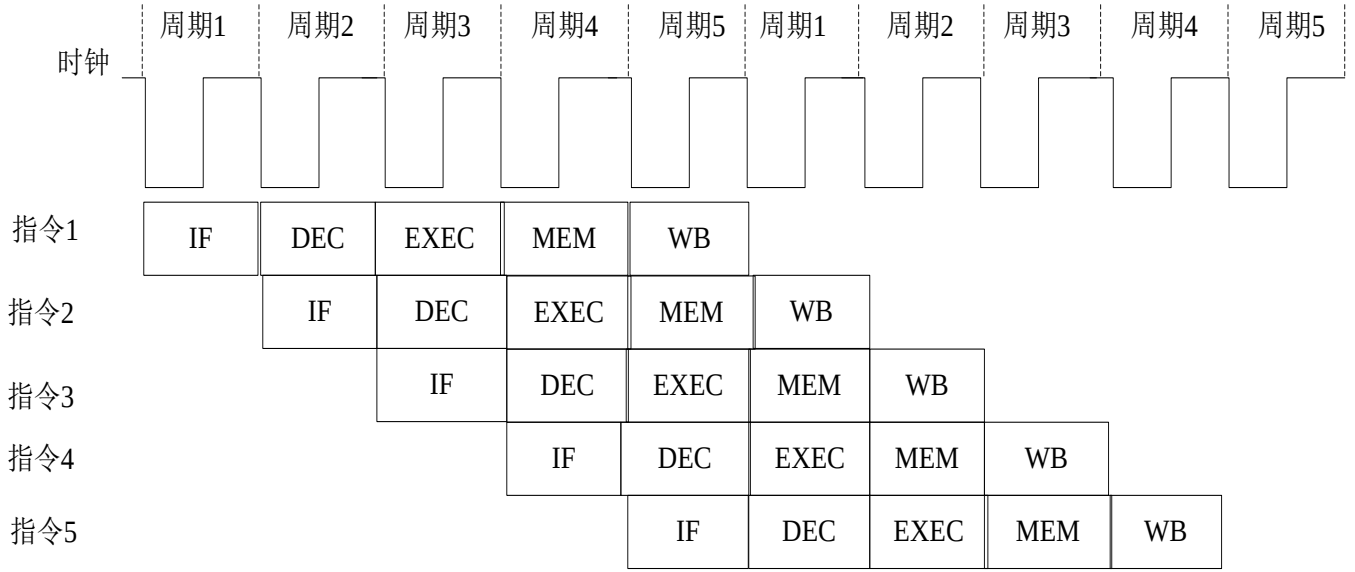
流水线技术

- 流水线的基本原理是把一个重复的过程分解为若干个子过程，前一个子过程为下一个子过程创造执行条件，每一个过程可以与其它子过程同时进行。



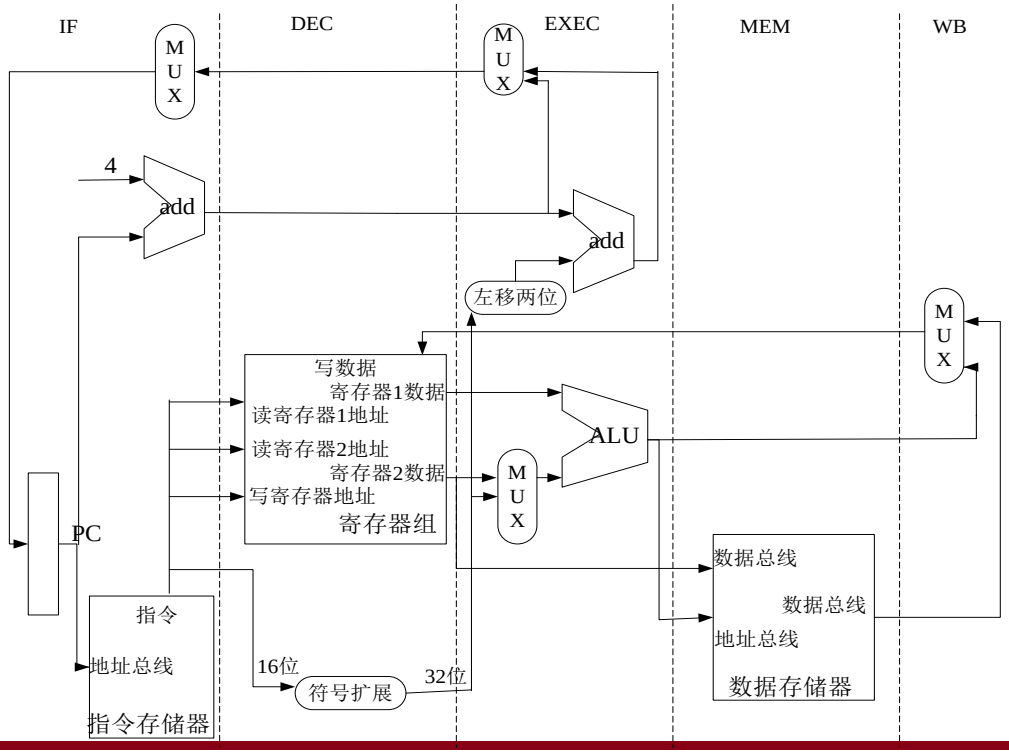
流水线技术

- 指令执行的重叠性(使一条流水线畅通)和同时性(多条流水线同时工作)就称为指令级并行。



流水线技术

- 数据通路的5级流水线划分
- 每两个流水线部件之间必须增加器件来实现流水线不同部件之间的接口。



超标量技术

- 微处理器集成多个ALU、多个译码器和多条流水线，以并行处理的方式来提高性能
- 假设处理器有一个整数部件和一个浮点部件，处理器至多能发出两条指令，一条是整数类型的指令，包括整数算术逻辑运算，存储器访问操作和转移指令；另一条必须是浮点类型的指令

异常处理机制

- 处理异常事件的机制

异常种类	来源	MIPS处理器命名
I/O设备	外部	中断
用户程序唤醒操作系统	内部	异常
计算结果溢出	内部	异常
未定义的指令（非法指令）	内部	异常
硬件出错	两者	异常或中断

异常处理系统需具备的功能

- 微处理器要能够实现异常处理需要完成以下几方面的功能：
  - 记录异常发生的原因
  - 记录程序断点处的指令在存储器中的地址
  - 记录不同种类的异常处理程序在内存中的地址
  - 建立异常种类与异常处理程序地址之间的对应关系。

异常事件识别机制

- 状态位法（MIPS）：
  - 在微处理器中利用一个寄存器对每种异常事件确定一个标志位，当有异常事件发生时，寄存器中对应的位置1，一个32位的寄存器可以表示32种不同类型的异常事件
- 向量法(Intel）：
  - 对不同类型的异常事件进行编码，这个编码叫中断类型码或异常类型码。

断点保存和返回

- 寄存器法（嵌入式）
  - 在微处理器中设计一个寄存器EPC，当微处理器出现异常时，就将PC的值保存到EPC中。异常处理完之后，再把EPC的值赋给PC，这样就可以实现中断的返回
- 栈（PC）
  - 微处理器直接将PC的值压入栈中，异常处理完之后，再从栈顶把值弹出来赋给PC

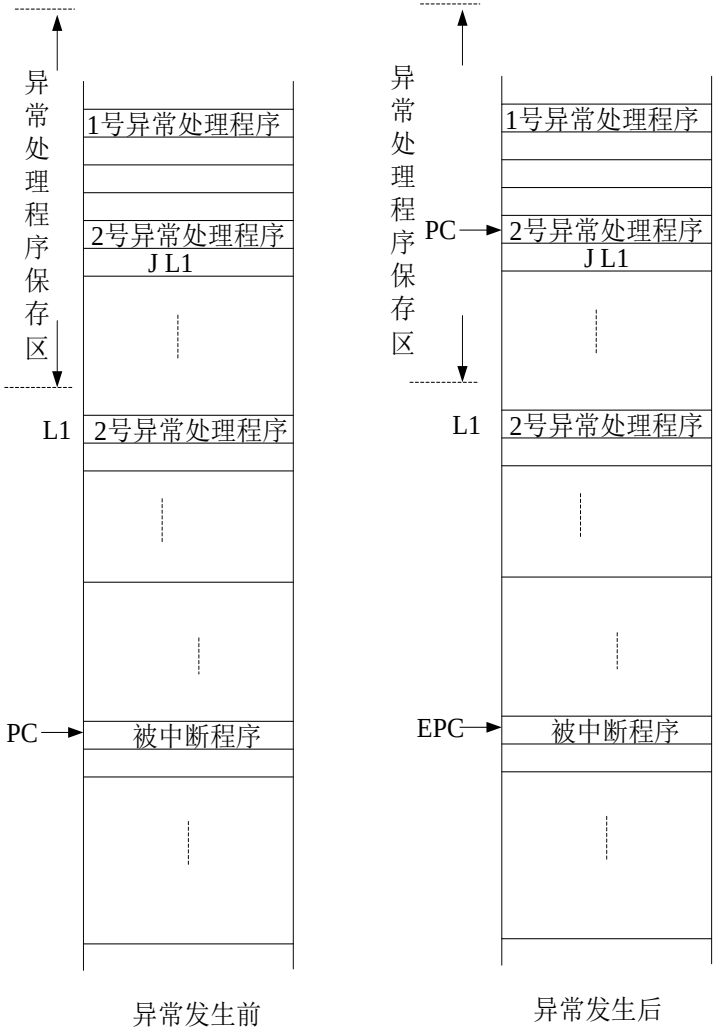
异常处理程序进入方式

- 1）专门的内存区域保存异常处理程序
  - 在这块内存区域中为每个异常处理程序分配固定长度的空间如32个字节或8条指令长度的空间，而且针对每个异常事件其异常处理程序的存放地址是固定的。
- 2）仅提供一个异常处理程序存放地址
  - 发生任何异常事件都首先转移到该地址执行总的异常处理，并在总异常处理程序中分析异常事件的原因，然后再根据异常的原因通过子程序调用的方式去执行相应的异常处理。

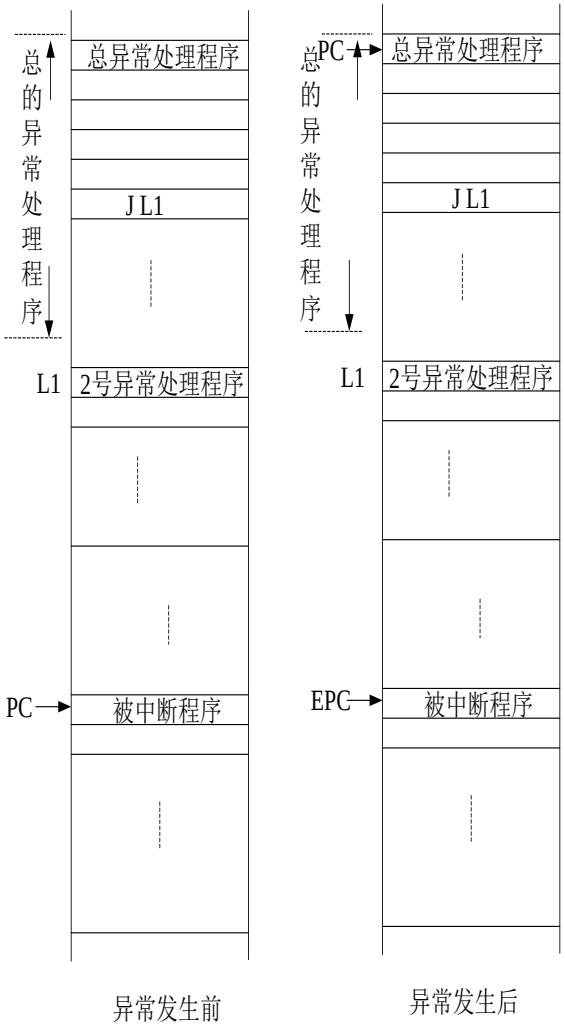
异常处理程序进入方式

- 3）分配一块专门的内存区域保存异常处理程序的入口地址
  - 异常处理程序的入口地址叫中断向量
  - 保存异常处理程序的入口地址的内存区域叫做中断向量表
  - 异常处理程序可以存放在内存中的任意位置，只需要把该异常处理程序的入口地址保存到中断向量表中正确的地址中，当异常发生时，微处理器就可以通过中断向量表查找到中断服务程序的入口地址。

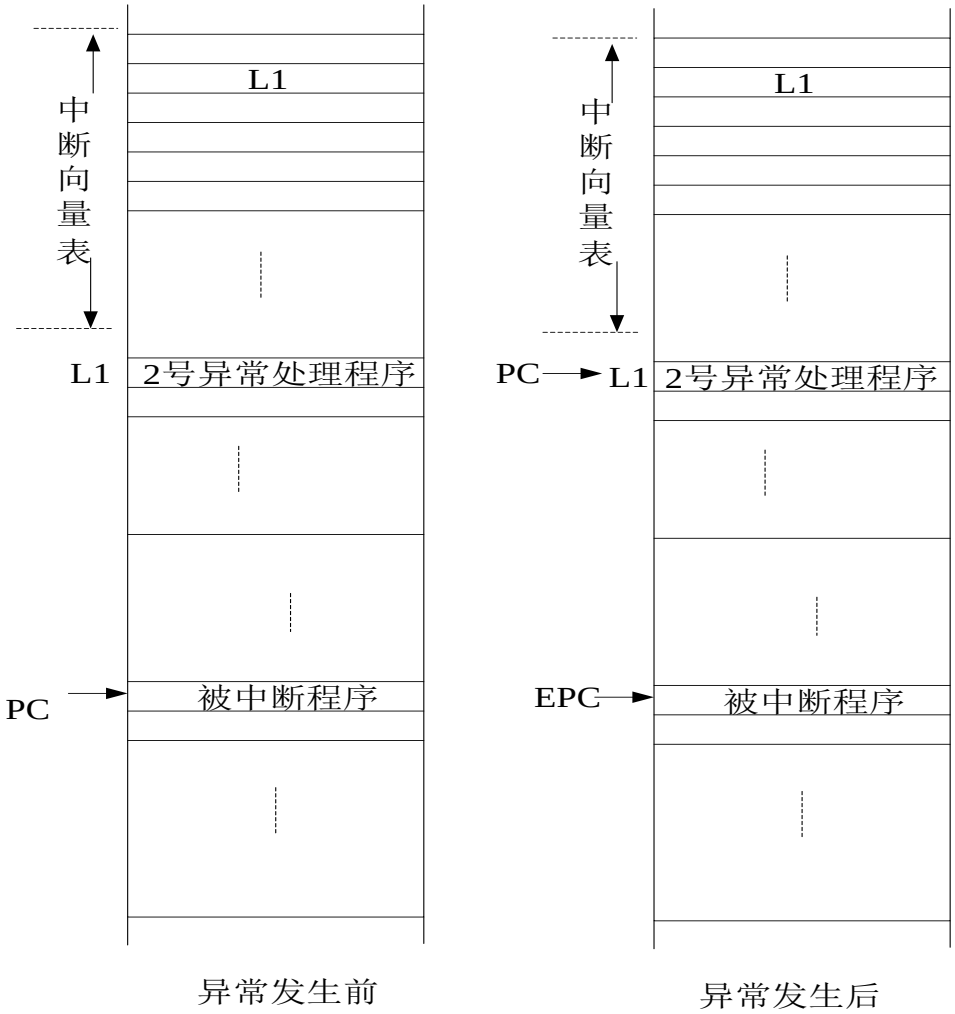
异常处理程序进入方式1）



异常处理程序进入方式2）

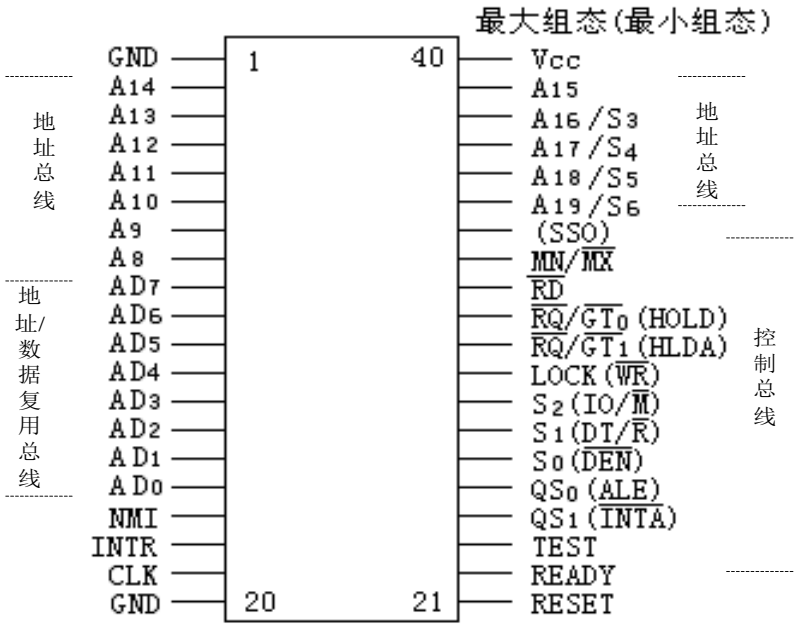


异常处理程序进入方式3）



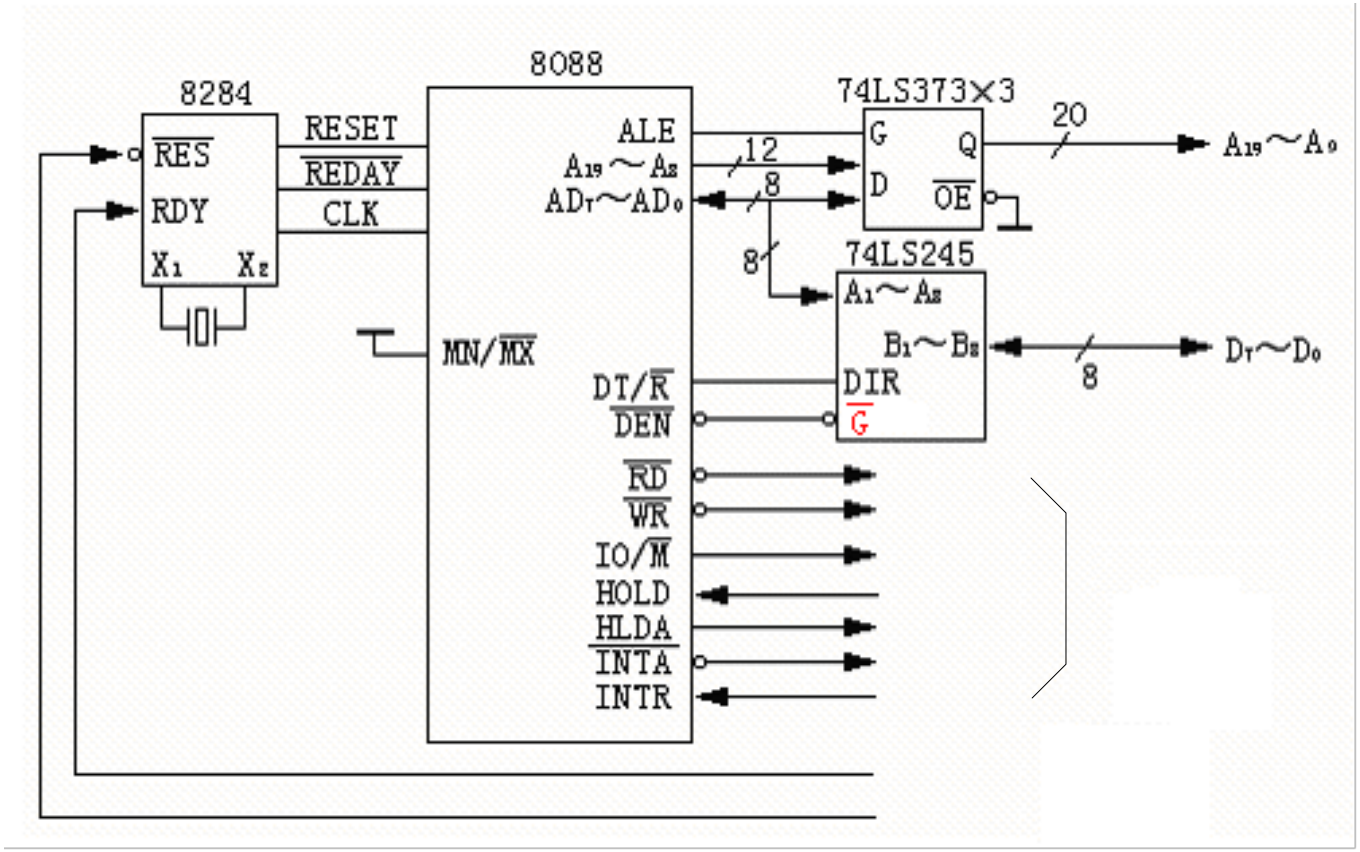
微处理器外部接口

- 微处理器为实现与计算机系统内的其他部件之间的信息交互必须提供地址、数据和控制总线

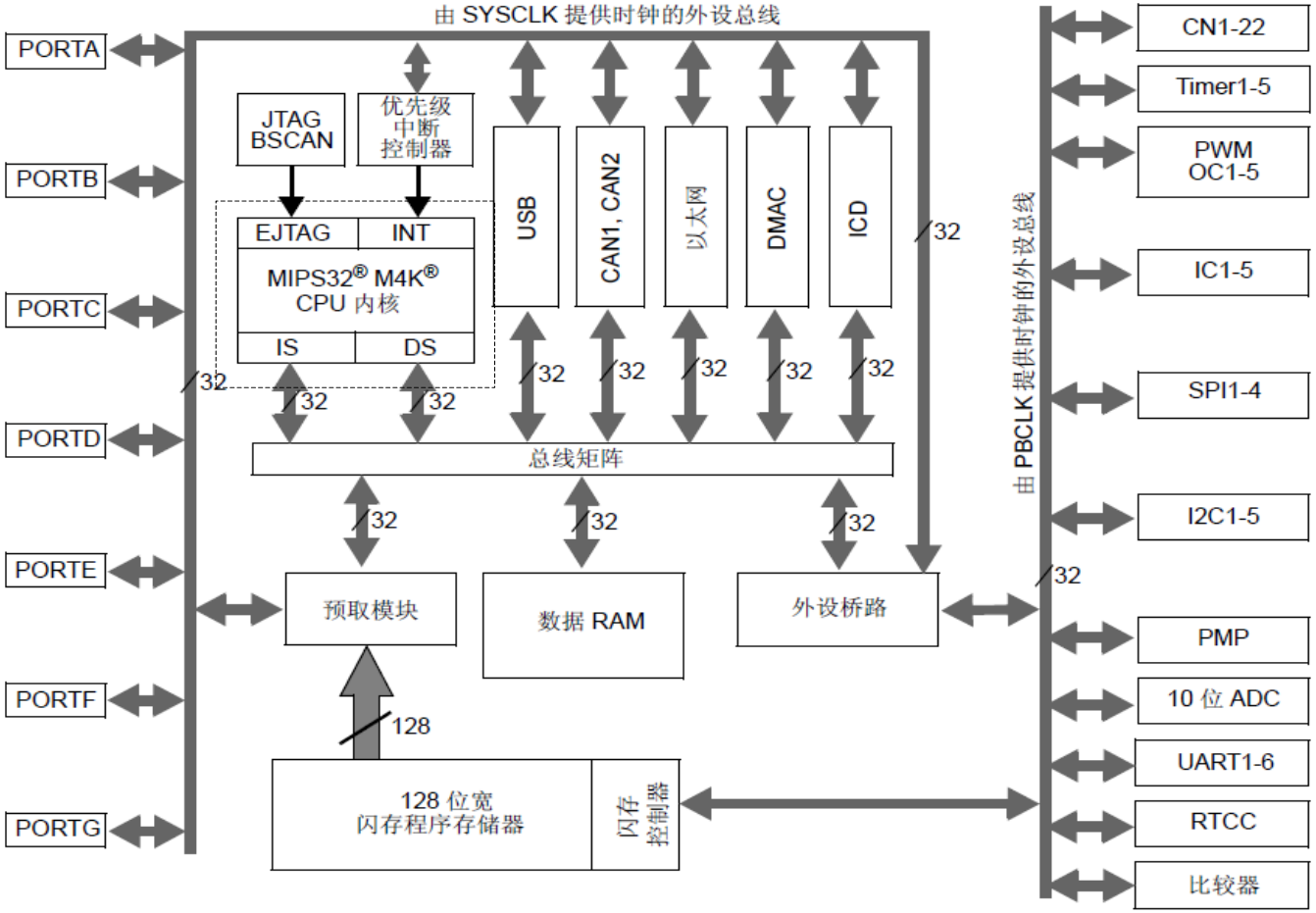


8088微处理器外部接口

8088微处理器最小组态系统总线接口电路

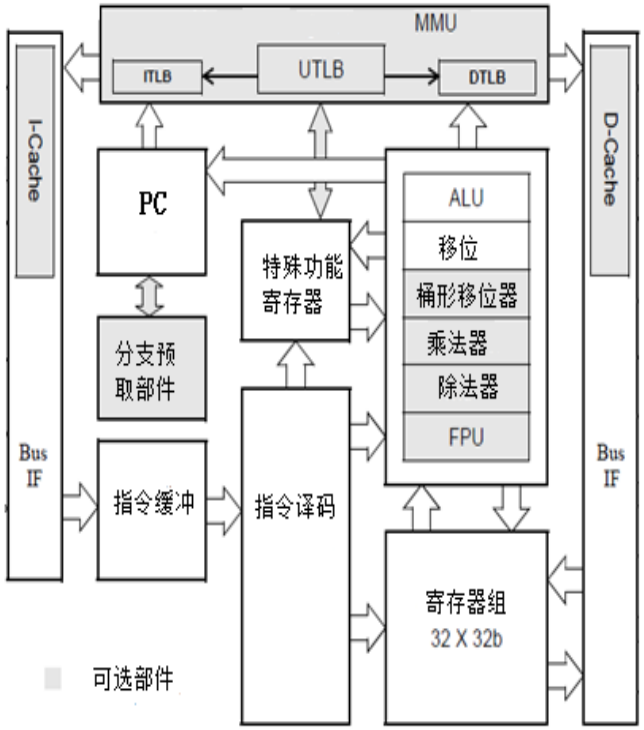


嵌入式芯片PIC32MX5XX/6XX/7XX系列框图



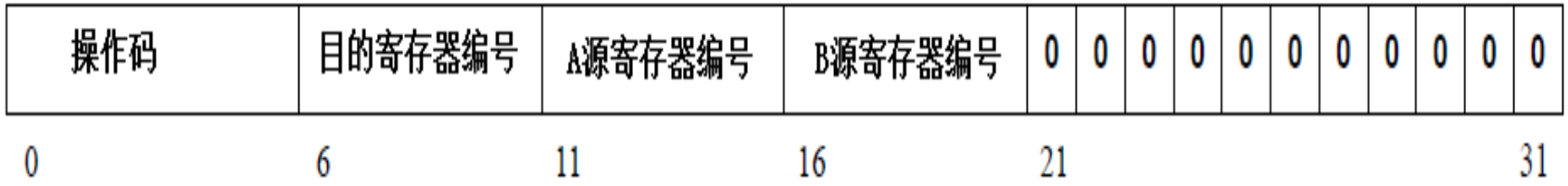
MicroBlaze微处理器简介

- 可以配置为支持大字节序或小字节序,
  - 当采用PLB外部总线时为大字节序,
  - 采用AXI4外部总线时为小字节序。
  - 支持三级或五级流水线,
  - 具有可选的指令和数据cache,
  - 支持虚拟内存管理。
  - 支持通过PLB、LMB、AXI总线与外围接口或部件相连。



- 具有32个32位的通用寄存器，使用规则与MIPS微处理器的通用寄存器使用规则相同，命名为R0~R31。
- R14，R15，R16，R17又用做异常返回地址寄存器，
- 还具有18个32位的特殊功能寄存器，包括PC，MSR等

MicroBlaze指令架构



作业

- 7
  - 12周周三前完成数据通路和控制器各个独立模块的仿真， 汇编程序到COE文件制作
  - 12周周三提交这部分的实验报告
- QQ群：计算机原理与接口技术 308772815
  - 答疑
  - 部分视频课件
  - 课程讨论
  - 加入验证信息：班级学号姓名