第3章 微处理器

CHAPTER 3

本章主要讲述微处理器的基本构成，并以 MIPS 微处理器为例，介绍一个基于给定简单指令集的类 MIPS 微处理器数据通路及控制器的设计，同时介绍现代微处理器流水线技术、超标量技术、多核技术和异常处理机制，最后介绍微处理器的外部接口及软核微处理器——MicroBlaze 微处理器及其构成的嵌入式计算机硬件最小系统。

学完本章内容，要求掌握以下知识：

- 微处理器的基本构成；
- 单周期简单指令集 MIPS 微处理器设计；
- 流水线技术、超标量技术、多核技术基本原理；
- 微处理器异常处理机制；
- MicroBlaze 微处理器结构；
- 嵌入式计算机硬件最小系统构成。

3.1 微处理器基本结构

微处理器是计算机系统的核心部件，完成计算机的运算和控制功能。随着计算机技术的不断发展，微处理器的设计也越来越复杂。

微处理器执行一组机器指令，这组指令可向微处理器告知应执行哪些操作。微处理器通常执行三种基本工作：

(1) 使用 ALU(算术逻辑单元)，执行算术、逻辑运算，如加、减、乘、除和逻辑运算。

(2) 将数据从一个存储位置移动到另一个位置。

(3) 做出决定，并根据这些决定跳转到一组新指令。

微处理器能够执行许多非常复杂的工作，但是所有工作都属于上述三种基本操作范畴。

图 3-1 显示了一个能够执行上述三种基本操作的简单微处理器基本结构，包括：

(1) 算术逻辑运算单元 ALU，实现各种算术、逻辑运算。

(2) 指令译码器，实现指令译码，并根据译码结果控制 CPU 完成相关操作。

(3) 寄存器，包括通用寄存器、指令寄存器、程序计数器等。通用寄存器用来暂存参与 ALU 单元运算的数据和中间结果，指令寄存器用来暂存从存储器读入的指令，程序计数器用来指示下一条要执行的指令在存储器中的存放地址。

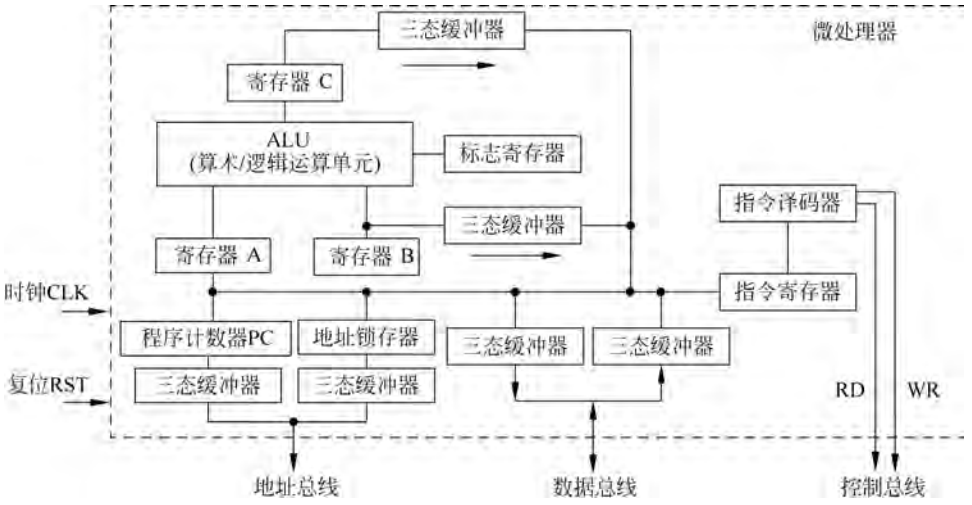


图 3-1 简单微处理器基本结构

微处理器与计算机中其他组件之间的基本接口包括：

(1) 一组地址总线，用于向存储器发送地址。

(2) 一组数据总线，用于将数据发送到存储器或从存储器取得数据。

(3) 一条 RD(读)和 WR(写)控制信号，告诉存储器希望将数据写入某个地址还是从某个地址获得数据。

(4) 时钟信号，将时钟脉冲序列发送到微处理器，控制微处理器进行工作。

(5) 复位信号，用于将程序计数器、通用寄存器重置为某个值并重新开始执行。

3.2 单周期简单指令集 MIPS 微处理器设计

假定单周期简单指令集 MIPS 微处理器支持的指令集如下。

(1) 算术、逻辑运算指令：add、sub、and、or、slt；

(2) 数据传送指令：lw、sw；

(3) 程序控制指令：beq、j。


实践视频

3.2.1 简单指令集 MIPS 微处理器数据通路

数据通路是指指令执行过程中实现指令获取以及数据处理的电路模块和传输路径。

本节首先分析简单指令集不同类型指令的执行部件构成，然后再将各个部件综合形成简单指令集 MIPS 微处理器的完整数据通路。

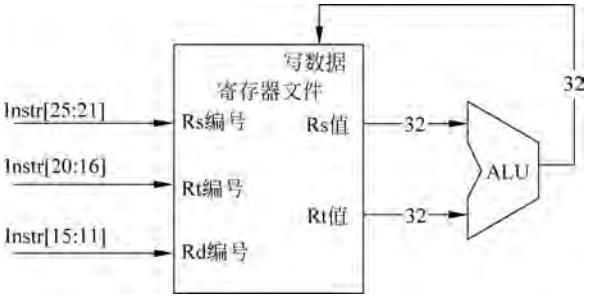
1. R 型指令执行部件

简单指令集 R 型指令包括 add、sub、and、or、slt。为便于阅读，这里将指令的机器码采用 Instr 表示，以区分数据。由此得到 R 型指令的编码如图 3-2 所示。

Instr[31..26]	Instr[25..21]	Instr[20..16]	Instr[15..11]	Instr[10..6]	Instr[5..0]
Op[5..0]	Rs	Rt	Rd	Shamt	Funct[5..0]

图 3-2 R 型机器指令编码

R 型指令执行时，首先根据指令字段 Instr[25..21]、Instr[20..16]获取 $Rs、$Rt 的值，然后根据字段 Instr[31..26]、Instr[5..0]的译码结果执行算术、逻辑运算，最后再将结果保存到字段 Instr[15..11]指示的 $Rd 中。由此可知，R 型指令执行时，寄存器文件需同时提供：三组寄存器编号，包括 $Rs 编号、$Rt 编号、$Rd 编号；两组输出数据线，包括 $Rs 的值、$Rt 的值；一组输入数据线，即写入 $Rd 的值。R 型指令执行算术、逻辑运算，由算术逻辑单元(ALU)完成，且数据来源分别为 $Rs 和 $Rt 的值；运算结果输出到寄存器文件写入 $Rd 的数据线。由于 MIPS 微处理器所有寄存器都为 32 位，因此所有输入/输出数据线都为 32 位；寄存器编号都为 5 位，因此寄存器编号输入信号都为 5 位，由此得到 R 型指令执行部件结构如图 3-3 所示。




实践视频

图 3-3　R 型指令执行部件结构

2. 数据传送指令执行部件


实践视频

简单指令集中数据传输指令包括 lw、sw 指令，都为 I 型指令，I 型指令编码如表 2-3 所示，采用 Instr 改写之后指令编码如图 3-4 所示。

数据传输指令中存储器地址由 RF[$Rs]与指令中立即数(Imm)之和构成，因此需要 ALU 运算单元计算存储单元地址。由于 ALU 单元仅支持 32 位数据运算，指令中立即数为 16 位，需先符号扩展为 32 位，再参与运算。数据在数据存储器与寄存器文件之间双向传输，因此数据传送指令执行部件包含寄存器文件、ALU、数据存储器及符号扩展部件。它们之间的连接关系如图 3-5 所示。

Instr[31..26]	Instr[25..21]	Instr[20..16]	Instr[15..0]
Op[5..0]	Rs	Rt	Imm

图 3-4　数据传输机器指令编码

当执行 lw 指令时，首先从 $Rs 中获取基地址，并通过 ALU 运算单元形成存储单元地址，再由存储器输出读数据到寄存器文件的写数据，寄存器文件在寄存器写信号的控制下将数据保存到 $Rt 中；当执行 sw 指令时，同样首先从 $Rs 中获取基地址，并通过 ALU 运算单元形成存储单元地址，同时将 $Rt 中的数据输出到存储器写数据，再在存储器写控制信号的作用下将数据写入数据存储器中。

3. 顺序程序指令获取部件

微处理器采用特殊寄存器——程序计数器 PC 保存下一条指令地址。MIPS 微处理器所有指令等长，都为 4 字节，且存储器以字节为单位编址，因此程序顺序执行时，PC 的值自动

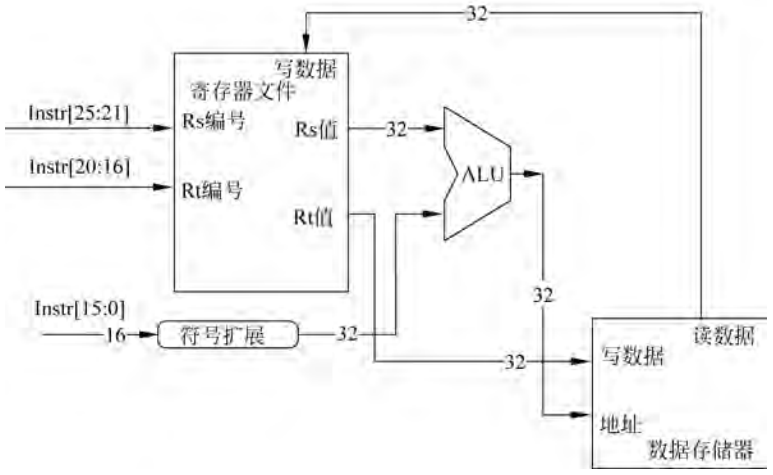


图 3-5 数据传输指令执行部件连接关系

加 4 指向下一条指令。由此可知，MIPS 微处理器顺序程序指令获取部件构成如图 3-6 所示。

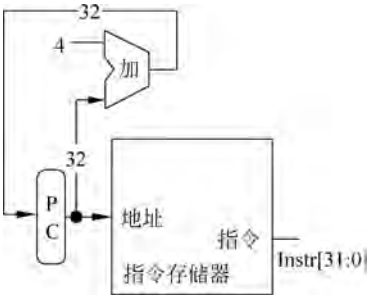


图 3-6 MIPS 微处理器顺序执行程序指令获取部件构成

4. 条件跳转指令执行部件

简单指令集中的条件跳转指令 beq 为 I 型指令，I 型指令编码如表 2-3 所示，采用 Instr 改写之后指令编码如图 3-7 所示。

Instr[31..26]	Instr[25..21]	Instr[20..16]	Instr[15..0]
Op[5..0]	Rs	Rt	Imm

图 3-7 beq 编码格式

条件跳转指令首先比较 $Rs、$Rt 的值，然后根据比较结果 Zero 决定是否跳转。因此该指令需要比较两个寄存器的值，同时还需要计算目标跳转地址，这是两种不同的运算。为简化电路设计，设计时采用 ALU 单元比较两个寄存器的值，另外再采用一个加法器专门计算跳转目标地址。MIPS 条件跳转控制指令跳转目标地址由下一条指令 PC 的值与指令中立即数(Imm)的和构成。由于立即数在指令中仅 16 位，因此需通过符号扩展部件扩展为 32 位。指令编码中存储的立即数为跳过的指令条数(每条指令 4 字节)，因此立即数(Imm)符号扩展之后需要进一步左移 2 位形成跳转距离，即跳转距离的形成需要用到符号扩展以及移位部件。根据以上分析可知，条件跳转指令执行部件构成如图 3-8 所示。当执行 beq

指令且条件成立(Zero=1)时,与门输出 1 从而控制复用器将跳转目标地址输出到 PC 寄存器;否则将 PC 加 4 输出到 PC 寄存器。

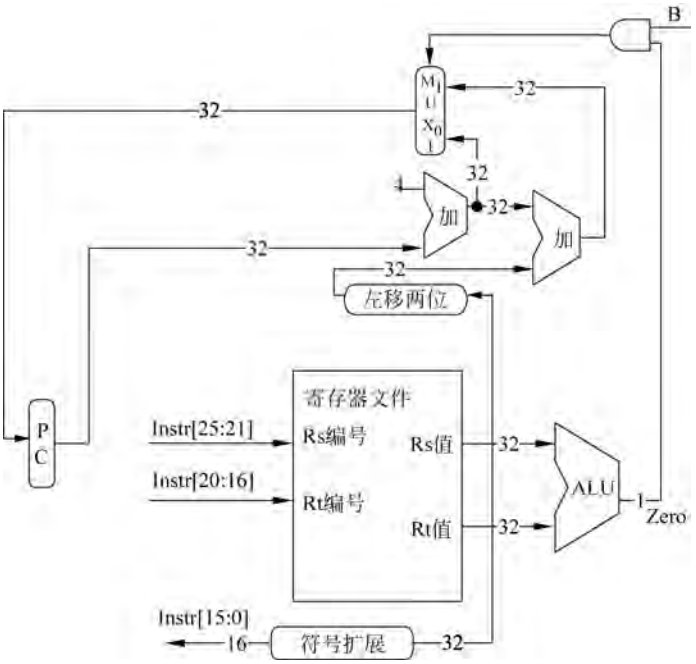


图 3-8 条件跳转指令执行部件

5. 无条件伪直接跳转指令执行部件

简单指令集中的无条件伪直接跳转 j 指令为 J 型指令,指令编码如表 2-6 所示,采用 Instr 改写之后指令编码如图 3-9 所示。

Instr[31:26]	Instr[25:0]
Op[5..0]	Imm

图 3-9 伪直接跳转 j 指令编码

该指令中除了 6 位操作码之外,其余位为伪直接跳转地址立即数(Imm)。实际跳转目标地址由 PC 的高 4 位及指令低 26 位立即数(Imm)左移 2 位后合并而成。由此得到无条件伪直接跳转指令执行部件构成如图 3-10 所示。

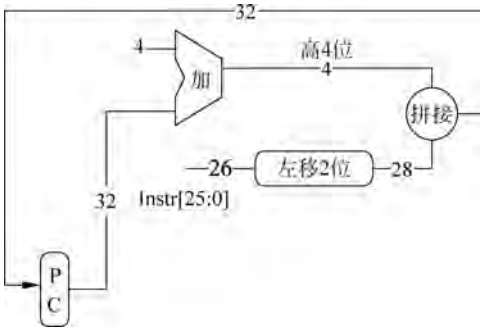


图 3-10 无条件伪直接跳转指令执行部件

6. 完整数据通路

为简化电路，各指令执行部件的公共部分复用，寄存器 PC 的值有 3 种来源：伪直接跳转目标地址、条件跳转目标地址及顺序执行地址。由于条件跳转控制指令执行部件可实现条件跳转目标地址及顺序执行地址的二选一，因此只需再增加一个复用器，通过两级二选一即可实现三选一复用。寄存器文件写入寄存器编码有两种来源，即 R 型指令中的 $Rd 及 I 型指令中的 $Rt，它们在指令中处于不同的域，因此需通过一个二选一复用器实现控制。寄存器文件写入寄存器数据也有两种来源，即 R 型指令 ALU 的运算结果及 I 型指令数据存储器的读数据，此处需通过一个二选一复用器实现控制。ALU 运算单元的第二个数据来源也有两种情况，即 R 型指令和 beq 指令的 RF[$Rt]及数据传送指令中的 Imm 符号扩展结果，此处也需通过一个二选一复用器实现控制。

复用之后简单指令集 MIPS 微处理器完整数据通路如图 3-11 所示。该数据通路统一采用二选一复用器实现多路信号复用。复用器 MUX1 复用伪直接跳转目标地址和其他目标地址作为下一条指令地址写入 PC；复用器 MUX2 复用分支跳转目标地址和顺序执行时的下一条指令地址作为 MUX1 的其他目标地址；复用器 MUX3 复用 ALU 运算结果和数据存储器的读数据作为寄存器文件写数据；复用器 MUX4 复用 $Rt 的值和指令中立即数符号扩展结果作为 ALU 的第二个源；复用器 MUX5 复用指令中的 $Rt 编号和 $Rd 编号作为写寄存器编号。

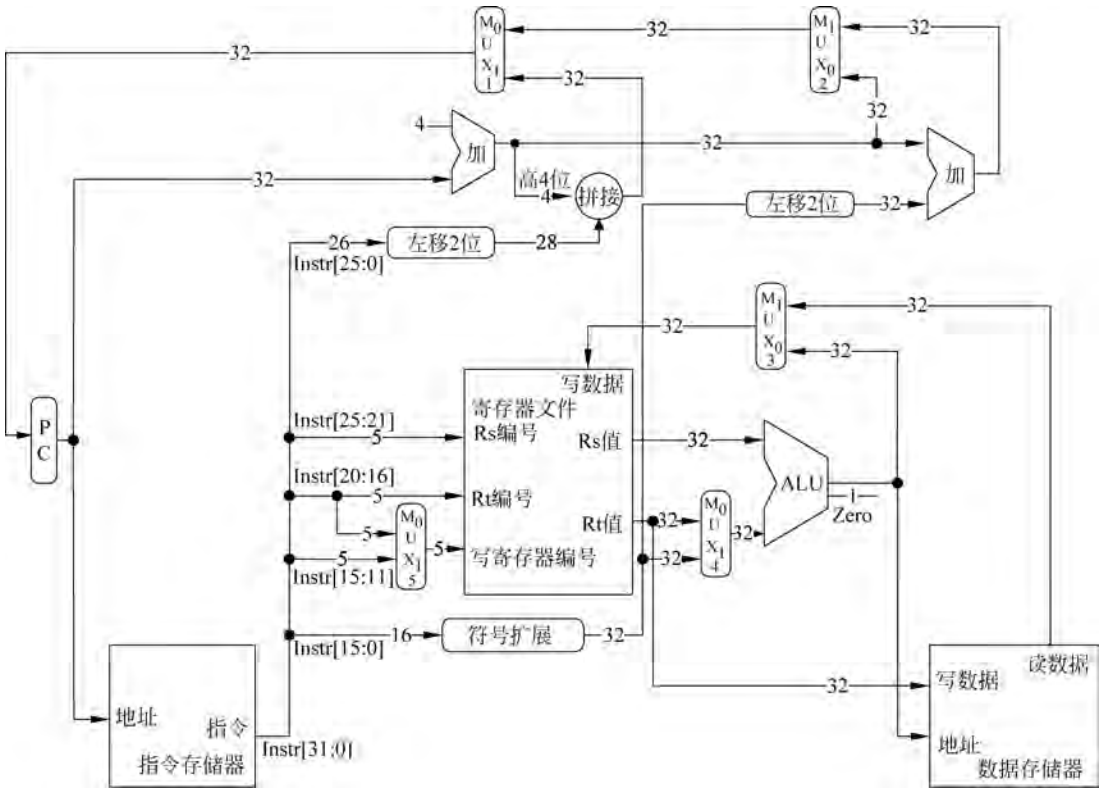


图 3-11 简单指令集 MIPS 微处理器完整数据通路

3.2.2 简单指令集 MIPS 微处理器控制器


实践视频

由第 2 章 MIPS 指令编码可知，简单指令集中指令功能决定因素有两种：①仅操作码 Op(I 型、J 型指令)；②操作码 Op 及功能码 Funct 的联合(R 型指令)。不同指令的不同功能需通过控制图 3-11 所示数据通路各个部分协调动作实现。简单指令集中 R 型指令功能的不同主要体现在 ALU 执行不同的运算；其他指令功能的不同主要体现在复用器通道选择信号、寄存器文件写信号、数据存储器写信号取值不同。由此可知，控制器仅需产生 ALU、复用器、寄存器文件、数据存储器等的控制信号。

1. ALU 控制信号

首先分析 ALU 单元执行的运算类型。简单指令集 MIPS 微处理器仅支持 add、sub、and、or、slt 等 R 型运算指令；数据传输指令 sw、lw 通过 ALU 单元执行加运算计算存储器地址；条件跳转控制指令 beq 利用 ALU 单元执行减运算比较两个寄存器的值是否相等。所有这些指令包含的运算有加、减、与、或、小于设置 5 种。5 种不同运算理论上仅需采用 3 位二进制编码即可表示，为方便指令集扩展，本书直接采用 MIPS 微处理器 ALU 控制信号编码。

MIPS 微处理器 ALU 单元具有 4 位控制信号(ALUCtr[3:0])，ALU 单元通过对 4 位控制信号译码决定执行何种操作，对应加、减、与、或、小于设置 5 种运算的 ALU 控制信号编码如表 3-1 所示。

表 3-1 对应加、减、与、或、小于设置 5 种运算的 ALU 控制信号编码

ALUCtr[3:0]	0000	0001	0010	0110	0111
操作类型	与	或	加	减	小于设置

MIPS 微处理器 R 型指令 6 位操作码全部为 0，由 6 位功能码指示 R 型指令的具体功能。简单指令集 MIPS 微处理器支持的 R 型指令全部为运算指令，若仅考虑 R 型指令，可以直接对 R 型指令 6 位功能码译码产生 ALUCtr[3:0]。但是 sw、lw 以及 beq 指令也需使用 ALU 执行运算且没有功能码，因此还需根据操作码区分 ALU 的运算类型。

对指令的操作码译码，可将简单指令集需利用 ALU 单元执行运算的指令分为 3 类：R 型(add、sub、and、or、slt)、I 型加法(lw、sw)、I 型减法(beq)。因此 2 位编码(ALUOp[1:0])即可表示这些不同指令的类型。I 型指令可直接根据 ALUOp[1:0]产生 ALUCtr[3:0]；R 型指令需根据 ALUOp[1:0]以及 6 位功能码 Funct[5:0]产生 ALUCtr[3:0]。由此可知，根据指令操作码以及功能码形成 ALUCtr[3:0]信号的译码电路可采用两级译码电路，如图 3-12 所示。

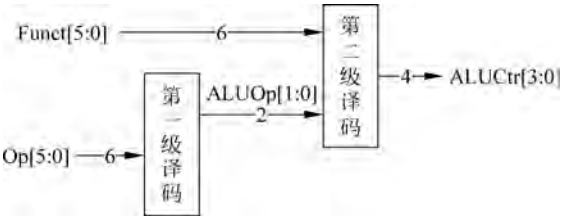


图 3-12 形成 ALUCtr[3:0]信号的两级译码电路

若定义I型加法的ALUOp[1:0]为$(00)_2$、I型减法的ALUOp[1:0]为$(01)_2$、R型指令的ALUOp[1:0]为$(10)_2$，可得到指令操作码、功能码与ALUOp[1:0]、ALUCtr[3:0]之间的关系如表3-2所示。

表3-2 指令操作码、功能码与ALUOp[1:0]、ALUCtr[3:0]之间关系

Op[5:0]	ALUOp[1:0]	指令功能	Funct[5:0]	ALU运算类型	ALUCtr[3:0]
100011(lw)	00	取字	xxxxxx	加	0010
101011(sw)	00	存字	xxxxxx	加	0010
000100(beq)	01	相等跳转	xxxxxx	减	0110
000000(R型 add)	10	加	100000	加	0010
000000(R型 sub)	10	减	100010	减	0110
000000(R型 and)	10	与	100100	与	0000
000000(R型 or)	10	或	100101	或	0001
000000(R型 slt)	10	小于设置	101010	小于设置	0111

由表3-2可以发现，R型指令6位功能码的前2位完全相同，因此在译码时可以不予考虑，这样就得到如表3-3、表3-4所示针对图3-12中第一级译码电路和第二级译码电路的真值表。

表3-3 ALUOp译码电路真值表

输入(Op[5:0])	100011	101011	000100	000000
输出(ALUOp[1:0])	00	00	01	10

表3-4 ALUCtr译码电路真值表

输入	ALUOp[1:0]	00	01	10	10	10	10	10
	Funct[5:0]	xxxxxx	xxxxxx	xx0000	xx0010	xx0100	xx0101	xx1010
输出	ALUCtr[3:0]	0010	0110	0010	0110	0000	0001	0111

2. 主控制器

MIPS微处理器主控制器除了产生ALU控制信号之外，还需根据指令操作码Op产生各个复用器的通道选择信号以及寄存器文件、数据存储器写信号。

图3-11所示数据通路上有5个复用器，所有复用器都需1位通道选择控制信号，因此需5个复用器通道选择信号，再加上存储器写、寄存器写控制信号以及ALUOp[1:0]共9个控制信号。各控制信号在数据通路上所处位置及名称定义如图3-13所示。

图3-13中复用器结构框图如图3-14所示，复用器功能如表3-5所示。

若图3-13中寄存器文件及数据存储器写控制信号(RegWr、MemWr)都为高电平有效，则图3-13中各个控制信号的含义如表3-6所示。

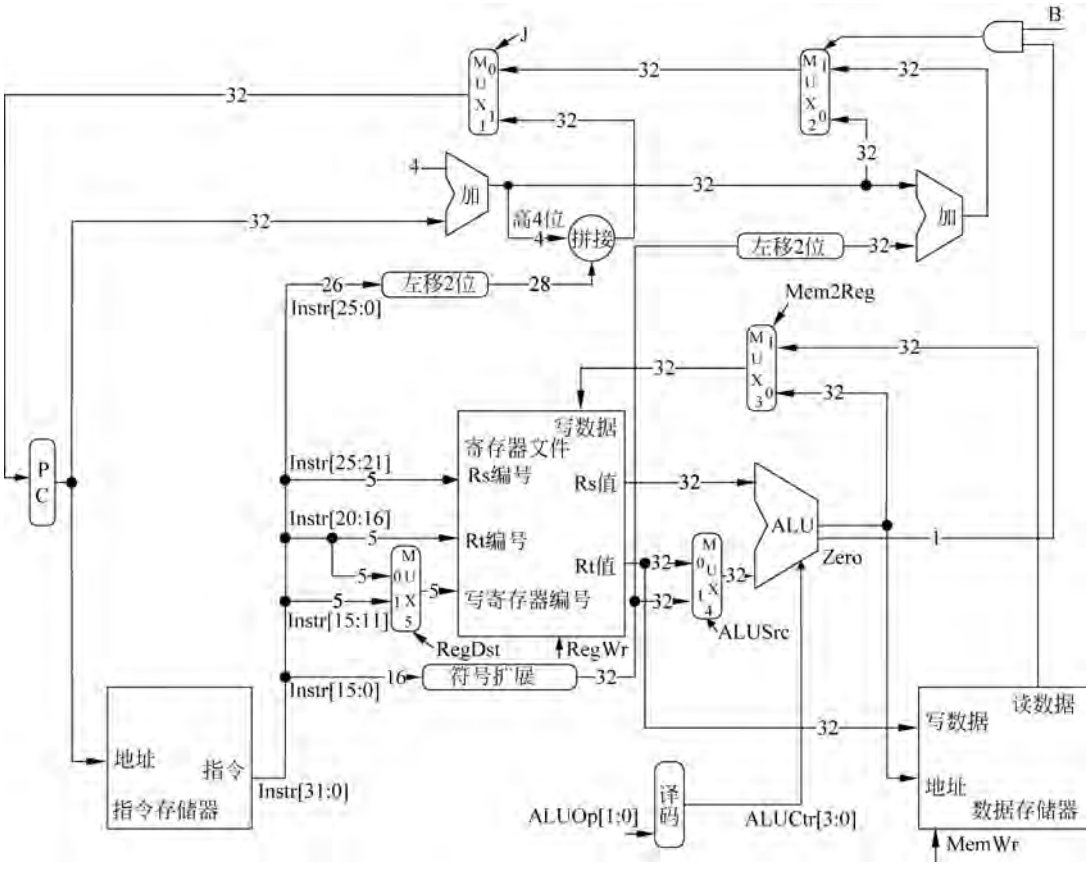


图 3-13　各控制信号在数据通路上所处位置及名称定义

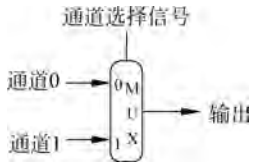


图 3-14　复用器结构框图

表 3-5　复用器功能

通道选择信号	输出数据来源	通道选择信号	输出数据来源
0	通道 0	1	通道 1

表 3-6　控制信号含义

信号名称	取 值 含 义	
	1	0
RegDst	写寄存器编号来自 Instr[15:11]($Rd)	写寄存器编号来自 Instr[20:16]($Rt)
J	PC 来自伪直接跳转地址	PC 来自复用器 MUX2 的输出
B	条件跳转指令	非条件跳转指令

续表

信号名称	取值含义	
	1	0
Mem2Reg	寄存器文件写数据来自数据存储器读数据	寄存器文件写数据来自 ALU 运算结果
ALUSrc	ALU 第二个数据源自 Instr[15:0](Imm)的 32 位符号数据扩展	ALU 的第二个数据源自 RF[$Rt]
RegWr	寄存器文件写数据写入写寄存器编号的寄存器中	无意义
MemWr	数据存储器写数据写入地址对应的存储单元中	无意义

简单指令集 R 型指令执行时，首先从指令存储器中取出指令，并且将 RF[$Rs]、RF[$Rt]输出到 ALU，经过 ALU 运算之后，将结果保存到$Rd。同时，PC 修改为 PC+4 的值，以便顺序执行下一条指令。R 型指令执行时数据通路有效传输路径如图 3-15 中深黑色线路所示，MUX1～MUX5 选择的通道分别为通道 0、通道 0、通道 0、通道 0、通道 1，由此可知简单指令集 R 型指令执行时各个控制信号取值如表 3-7 所示。

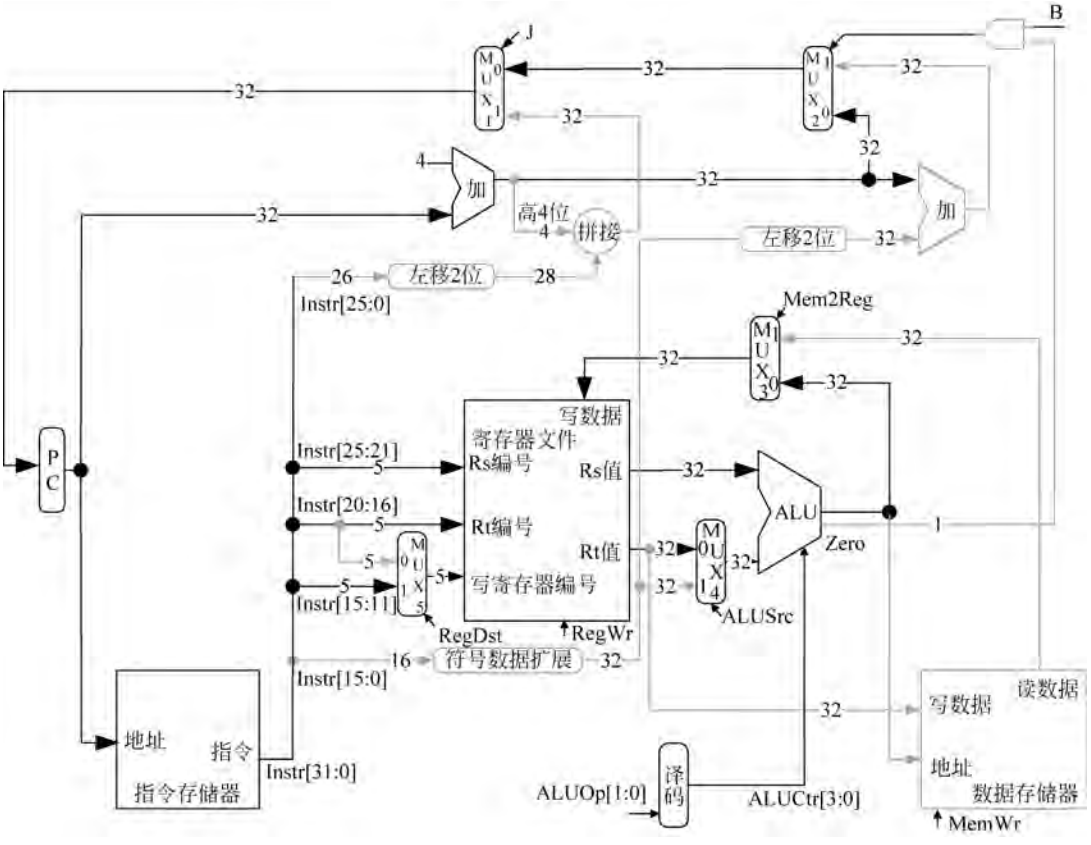


图 3-15 R 型指令执行时的有效数据通路

表 3-7　R 型指令执行时各控制信号取值

指令	RegDst	J	B	Mem2Reg	ALUSrc	RegWr	MemWr	ALUOp[1:0]
R 型	1	0	0	0	0	1	0	10

简单指令集装载指令(lw)执行时,首先从指令存储器中取出指令,并且将 RF[$Rs]、Instr[15:0](Imm)符号扩展后输出到 ALU,经过 ALU 加运算将结果输出到数据存储器地址输入端,最后将数据存储器输出的读数据保存到 $Rt。同时,PC 修改为 PC+4 的值,以便顺序执行下一条指令。lw 指令执行时数据通路有效传输路径如图 3-16 中深黑色线路所示,MUX1～MUX5 选择的通道分别为通道 0、通道 0、通道 1、通道 1、通道 0,由此可知装载指令(lw)执行时各个控制信号取值如表 3-8 所示。

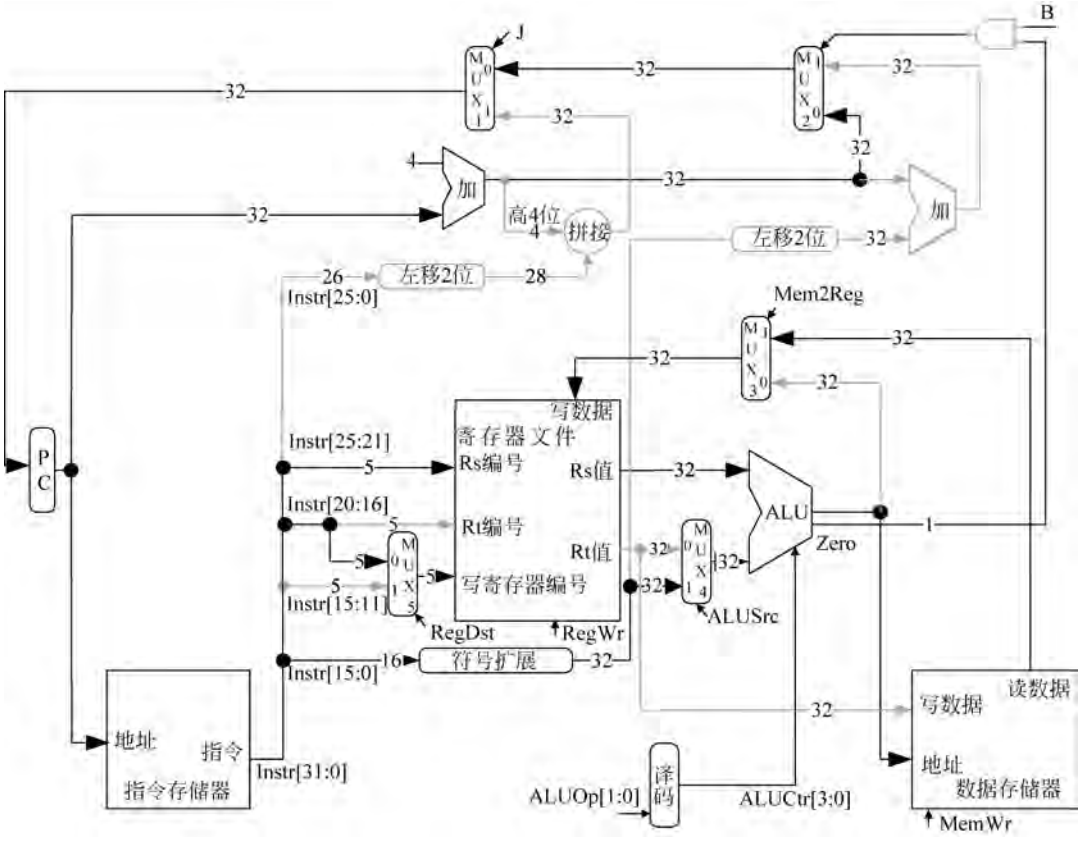


图 3-16　lw 指令执行时的有效数据通路

表 3-8　lw 指令执行时各控制信号取值

指令	RegDst	J	B	Mem2Reg	ALUSrc	RegWr	MemWr	ALUOp[1:0]
lw	0	0	0	1	1	1	0	00

简单指令集存储指令(sw)执行时,首先从指令存储器中取出指令,并且将 RF[$Rs]、Instr[15:0](Imm)符号扩展后输出到 ALU,经过 ALU 加运算将结果输出到数据存储器的

地址输入端，RF[$Rt]输出到数据存储器的写数据，最后 RF[$Rt]保存到数据存储器相应存储单元。同时，PC 修改为 PC+4 的值，以便顺序执行下一条指令。sw 指令执行时数据通路有效传输路径如图 3-17 中深黑色线路所示，MUX1～MUX5 选择的通道分别为通道 0、通道 0、任意、通道 1、任意，由此可知存储指令(sw)执行时各个控制信号取值如表 3-9 所示。

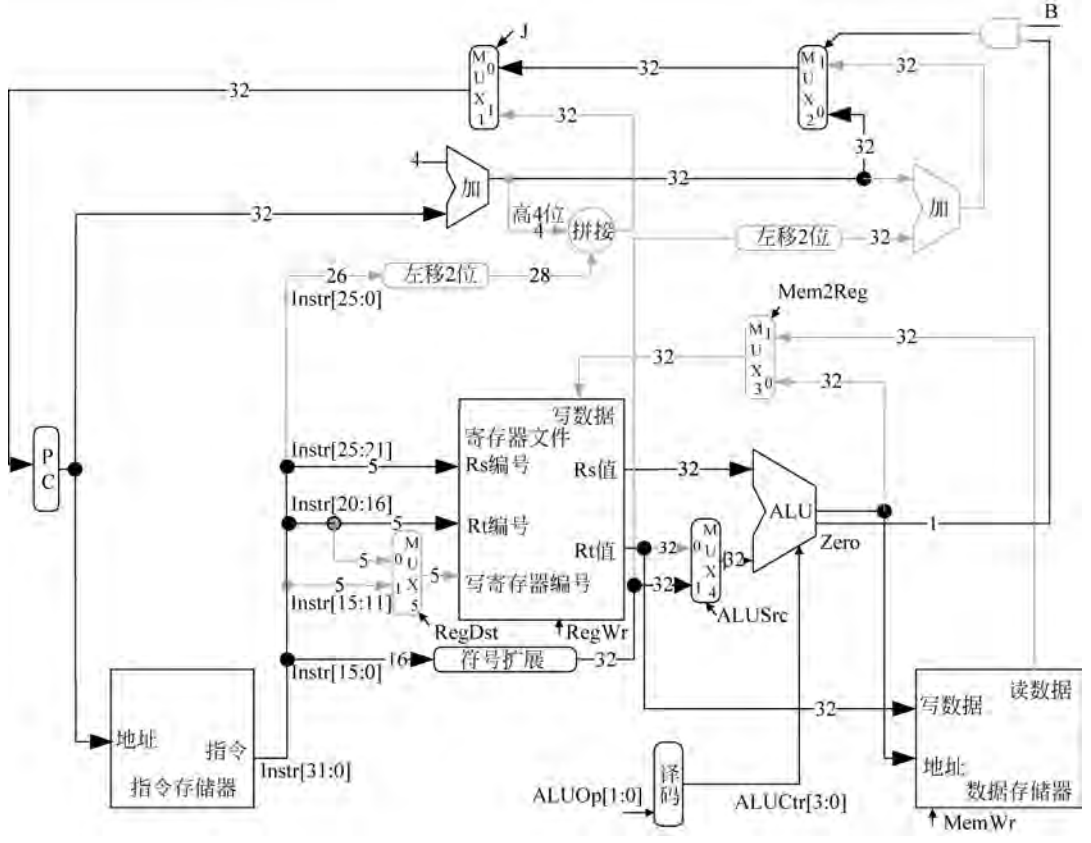


图 3-17　sw 指令执行时的有效数据通路

表 3-9　sw 指令执行时各控制信号取值

指令	RegDst	J	B	Mem2Reg	ALUSrc	RegWr	MemWr	ALUOp[1:0]
sw	x	0	0	x	1	0	1	00

简单指令集条件跳转控制指令(beq)执行时，首先从指令存储器中取出指令，并且将 Rs、Rt 数据输出到 ALU，经过 ALU 减运算之后，判断结果是否为零，并且将 Zero 标志输出以判断条件是否成立。当条件成立时，PC 修改为 PC+4 与指令中立即数(Imm)符号扩展并左移 2 位的和，跳转到目标指令；否则 PC 修改为 PC+4 的值，顺序执行下一条指令。数据通路有效传输路径如图 3-18 中深黑色线路所示，MUX1～MUX5 选择的通道分别为通道 0、通道 0 或 1(由 Zero 决定)、任意、通道 0、任意，由此可知 beq 指令执行时各个控制信号的取值如表 3-10 所示。

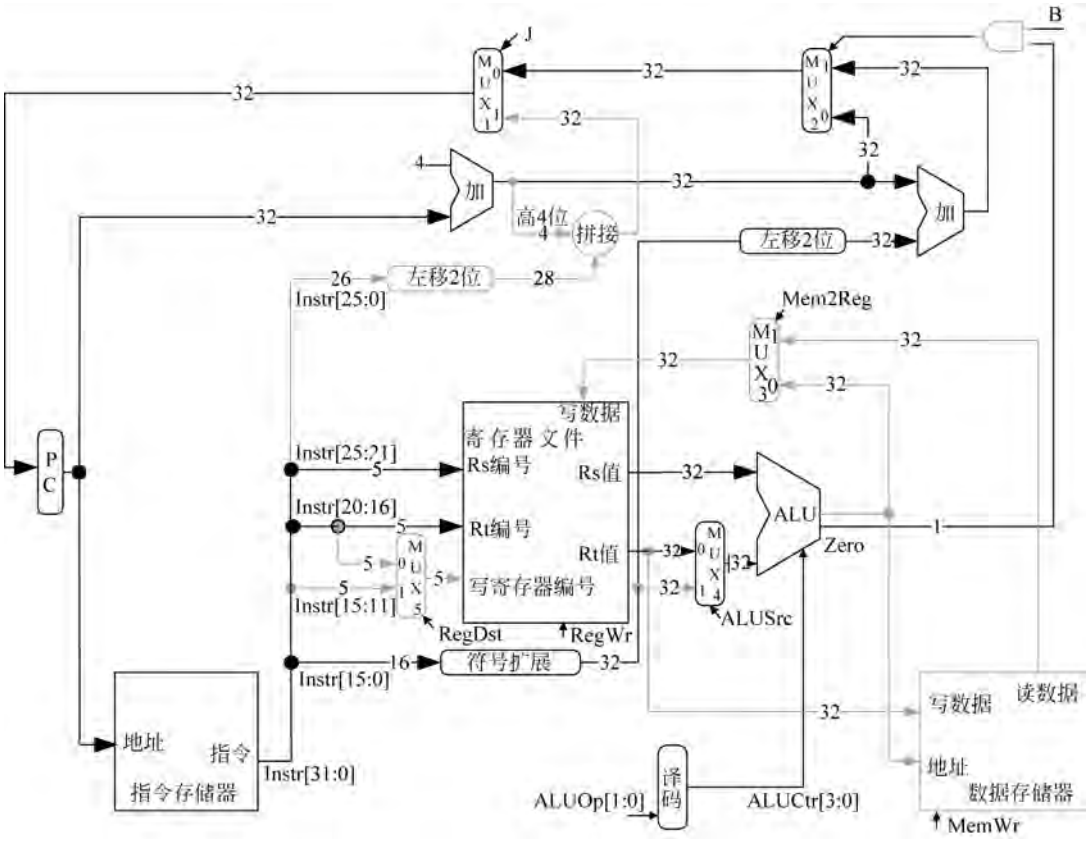


图 3-18　beq 指令执行时的有效数据通路

表 3-10　beq 指令执行时各控制信号取值

指令	RegDst	J	B	Mem2Reg	ALUSrc	RegWr	MemWr	ALUOp[1:0]
beq	x	0	1	x	0	0	0	01

简单指令集伪直接跳转控制指令(j)执行时,从指令存储器中取出指令,直接将 PC 修改为 PC+4 的高 4 位与 Instr[25:0](Imm)左移 2 位合并之后的值,从而跳转到目标地址。j 指令执行时数据通路有效传输路径如图 3-19 中深黑色线路所示,MUX1～MUX5 中仅 MUX1 有意义,且选择的通道为通道 1,其余复用器都无意义,由此可知 j 指令执行时各个控制信号的值如表 3-11 所示。

根据以上分析,得到简单指令集各指令执行时各个控制信号的值如表 3-12 所示。

根据 MIPS 汇编指令编码可知,表 3-12 中各控制信号产生的依据为指令 Op 字段(Instr[31:26])。若在数据通路中加入一个控制器,对指令 Op 字段译码产生这些控制信号,就形成了如图 3-20 所示简单 MIPS 微处理器。指令 Op 字段为主控制器的输入,各个控制信号为主控制器的输出,主控制器功能如表 3-13 所示。

若微处理器各个部件在同一时钟作用下工作,并在一个时钟周期内执行一条指令,就构成了单周期简单指令集 MIPS 微处理器,其中 PC 寄存器、指令存储器、寄存器文件、数据存储器都可采用同一时钟控制。

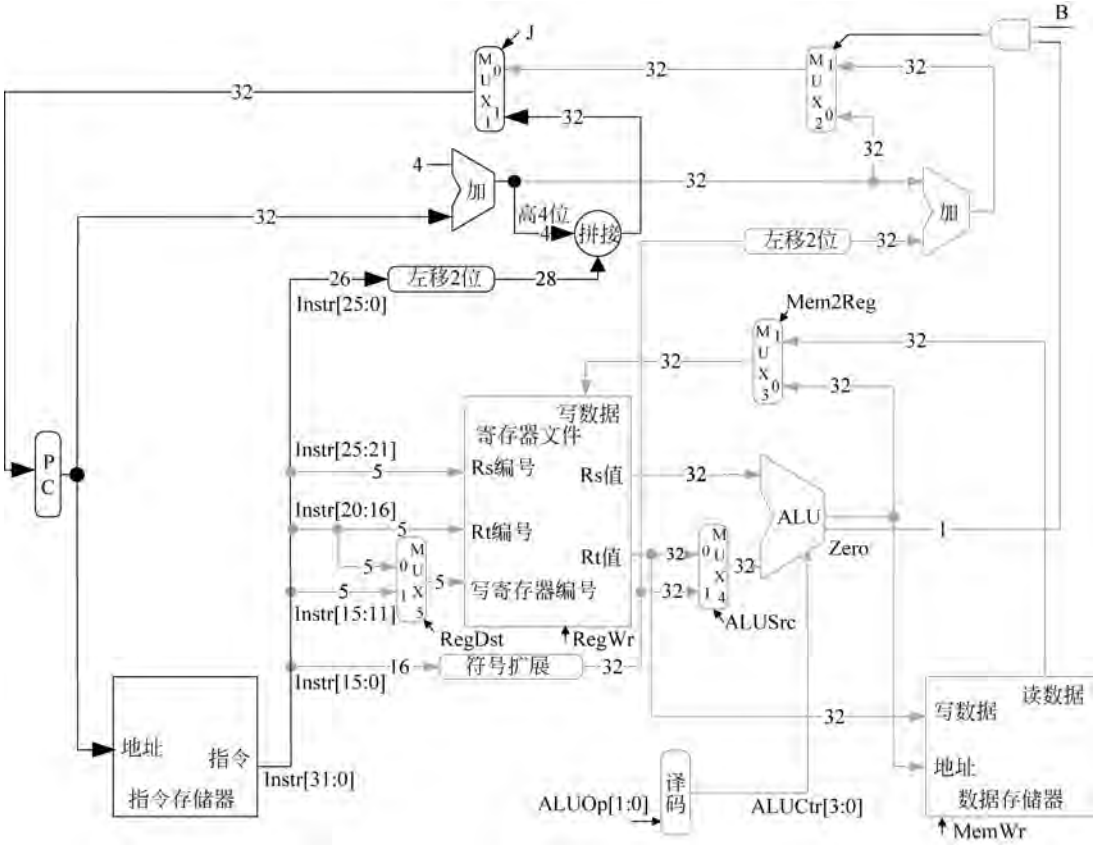


图 3-19 j 指令执行时的有效数据通路

表 3-11 j 指令执行时各控制信号取值

指令	RegDst	J	B	Mem2Reg	ALUSrc	RegWr	MemWr	ALUOp[1:0]
j	x	1	x	x	x	0	0	xx

表 3-12 简单指令集中各指令执行时各个控制信号的值

指令	RegDst	J	B	Mem2Reg	ALUSrc	RegWr	MemWr	ALUOp[1:0]
R 型	1	0	0	0	0	1	0	10
lw	0	0	0	1	1	1	0	00
sw	x	0	0	x	1	0	1	00
beq	x	0	1	x	0	0	0	01
j	x	1	x	x	x	0	0	xx

表 3-13　主控制器功能

输　入	输　出							
Op[5:0](Instr[31:26])	RegDst	J	B	Mem2Reg	ALUSrc	RegWr	MemWr	ALUOp[1:0]
000000(R 型)	1	0	0	0	0	1	0	10
100011(lw)	0	0	0	1	1	1	0	00
101011(sw)	x	0	0	x	1	0	1	00
000100(beq)	x	0	1	x	0	0	0	01
000010(j)	x	1	x	x	x	0	0	xx

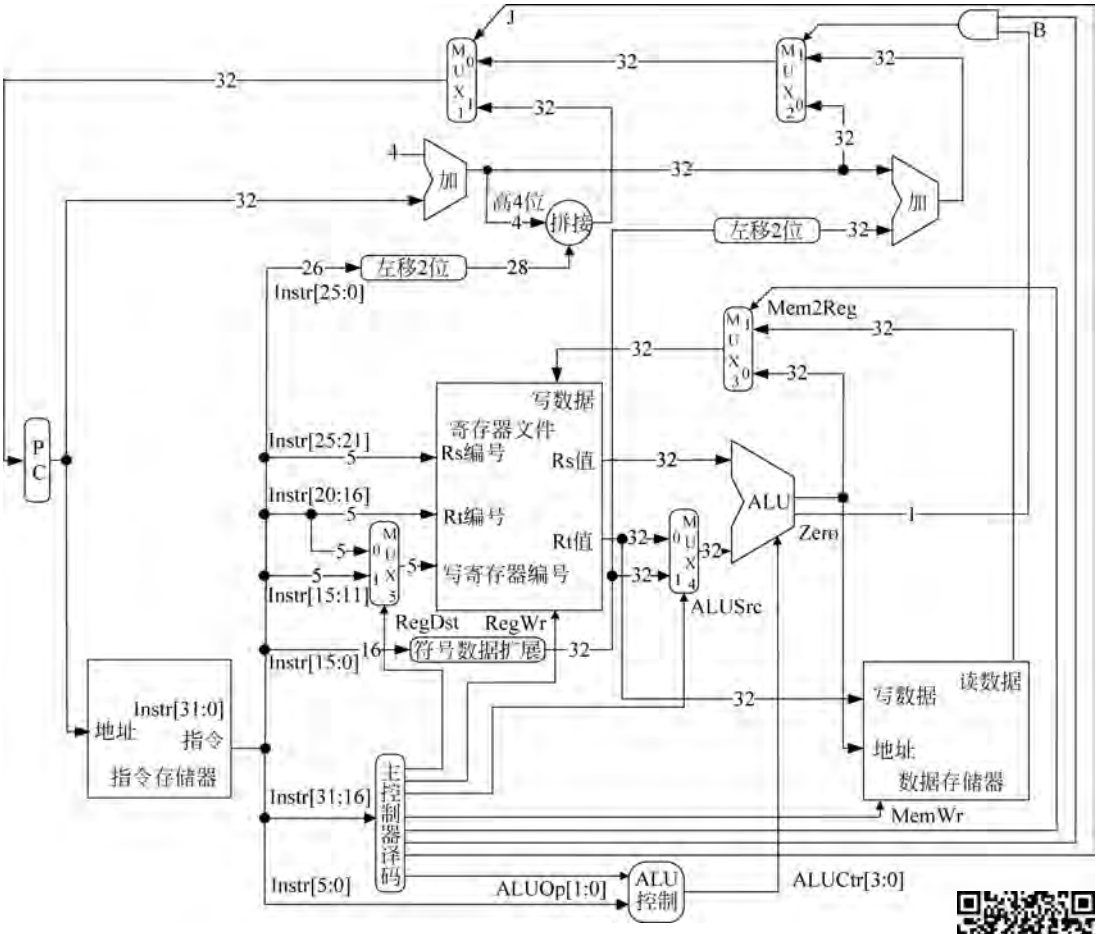


图 3-20　加上控制器的简单指令集 MIPS 微处理器完整框图

3.2.3　简单指令集 MIPS 微处理器典型指令执行过程

实践视频

本节结合图 3-20 所示简单指令集 MIPS 微处理器完整框图解释典型指令的执行过程。

实践视频

简单 MIPS 微处理器执行指令前，要求指令预先存储在指令存储器中。假定指令存储器中从地址 0 开始预先存储了如图 3-21 所示汇编指令序列的机器码，且所有通用寄存器初始值与其编号一致，寄存器 PC 的初始值为 0x0，

数据存储器各个存储单元的初始值都为 0。

```
L1: add $t1, $t2, $t3
    sw $t1,2($t2)
    beq $t1, $t2,L1
    j L1
```

图 3-21　指令存储器中预存储机器码对应的汇编指令序列

根据以上设定，MIPS 微处理器上电前的初始状态如图 3-22 所示。上电之后，由于 PC 的初始值为 0，首先取出第一条指令 add $t1，$t2，$t3 执行，然后顺序取出第二条指令 sw $t1，2($t2)执行，再次顺序取出第三条指令 beq $t1，$t2，L1 执行，该指令为条件跳转指令，因此根据执行时的条件确定是否跳转，若跳转则回到第一条指令；若不跳转则继续取出第四条指令 j L1 执行，该指令为无条件跳转，因此无论如何都跳回到第一条指令，依此周而复始。

下面根据简单 MIPS 微处理器的电路框图结构，逐条分析各指令执行时微处理器内部模块及连接线的状态。

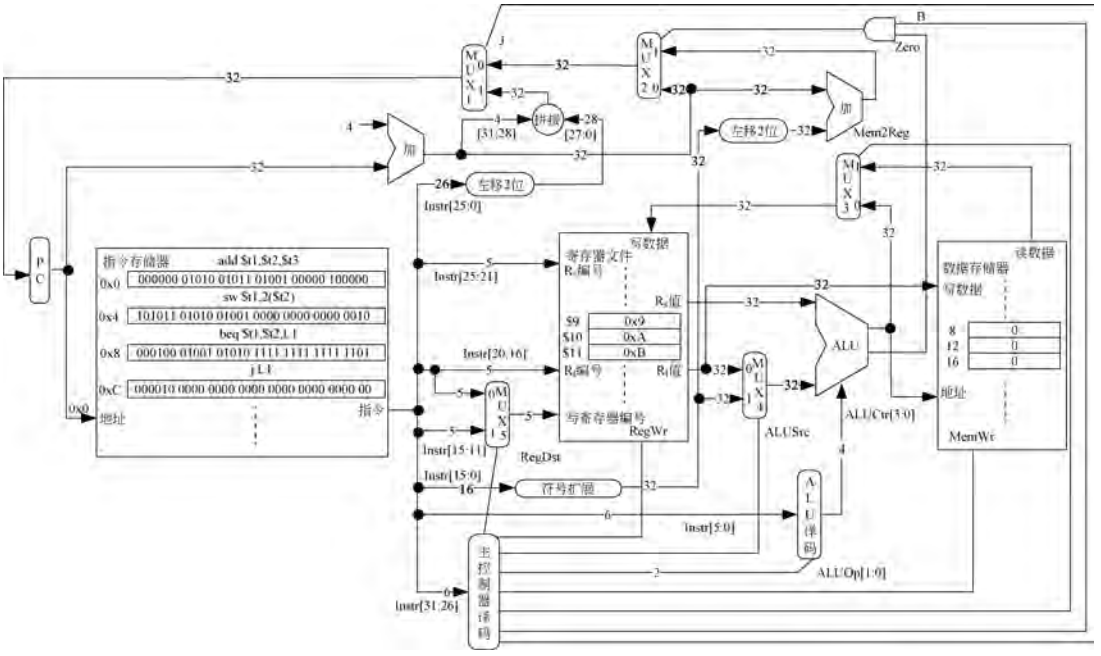


图 3-22　MIPS 微处理器上电前的初始状态

1. R 型指令 add $t1，$t2，$t3

执行过程为：

(1) 根据 PC 的初始值 0，从指令存储器中获得指令机器码，并将 PC 加 4。

(2) 根据指令机器码，寄存器文件输出 RF[$t2($10)]和 RF[$t3($11)]，同时主控制器译码 Instr[31:26]产生相应控制信号。此时 ALUSrc 为 0，RF[$t3]进入 ALU 参与运算。

(3) R 型指令主控制器译码后 $ALUOp_1$ 为 1，$ALUOp_0$ 为 0，ALU 译码单元进一步对指令功能码译码，控制 ALU 单元执行加法运算。此时 Mem2Reg 为 0，且 RegWr 为 1，ALU 运算结果(0x15)保存到 $t1($9)。

add $t1,$t2,$t3 执行时,MIPS 微处理器内部各模块以及连接线状态如图 3-23 所示。下一时钟到来后,PC 为 4,因此顺序执行下一条指令“sw $t1,2($t2)”。

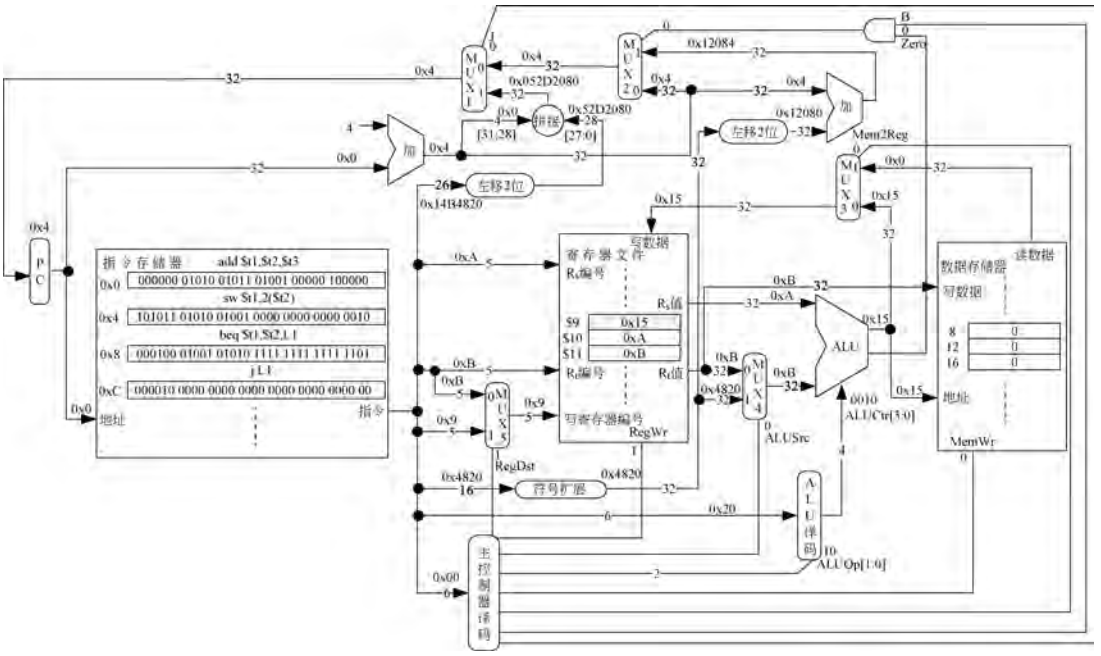


图 3-23 add $t1,$t2,$t3 执行时 MIPS 微处理器内部各模块及连接线状态

2. 数据传输指令 sw $t1,2($t2)


实践视频

执行过程为:

(1) 根据 PC 的值 4,从指令存储器中获取指令机器码,并将 PC 加 4。

(2) 根据指令机器码,寄存器文件输出 RF[$t2]、RF[$t1],同时主控制器译码 Instr[31:26]产生相应控制信号。此时 ALUSrc 为 1,Instr[15:0]符号扩展之后进入 ALU 单元参与运算。

(3) 由于 $ALUOp_1$ 为 0,$ALUOp_0$ 为 0,因此 ALU 单元执行加法运算,ALU 单元计算结果送入数据存储器地址。同时,MemWr 为 1,因此 RF[$t1]写入到由 ALU 单元运算结果(12)所指示的数据存储器地址中。

sw $t1,2($t2)执行时,MIPS 微处理器内部各模块以及连接线状态如图 3-24 所示。下一时钟到来时,PC 为 8,因此顺序执行下一条指令“beq $t1,$t2,L1”。

3. 条件跳转控制指令 beq $t1,$t2,L1

执行过程为:

(1) 根据 PC 的值 8,从指令存储器中获取指令机器码,并将 PC 加 4,同时也将 Instr[15:0]符号扩展并左移 2 位之后与 PC+4 的值相加得到跳转目标地址。

(2) 根据指令机器码,寄存器文件输出 RF[$t1]、RF[$t2],同时主控制器译码 Instr[31:26]产生相应控制信号。此时 ALUSrc 为 0,RF[$t2]送入到 ALU 单元参与运算。

(3) 由于 $ALUOp_1$ 为 0,$ALUOp_0$ 为 1,因此 ALU 单元执行减法运算,并判断结果是否为 0。若为 0 则使 Zero 输出 1,否则输出 0。

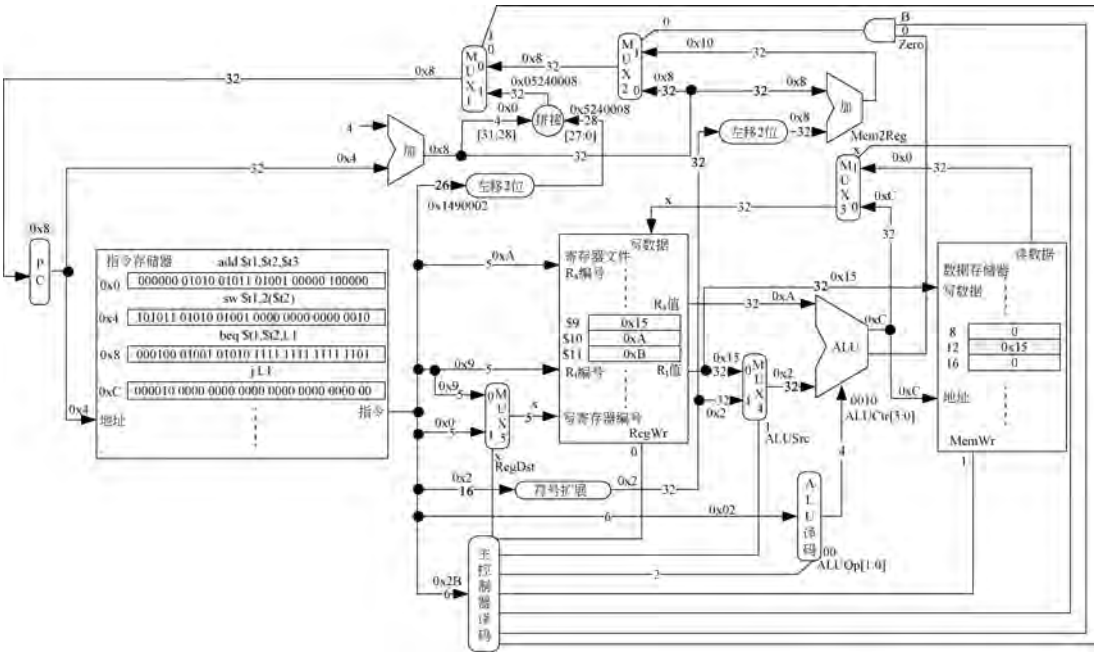


图 3-24　sw $t1,2($t2)执行时 MIPS 微处理器内部各模块及连接线状态

(4) 若 Zero 为 1,则经过与门之后控制复用器(MUX2)选择跳转目标地址赋给 PC,从而实现跳转；若 Zero 为 0,则选择 PC+4 的值赋给 PC,从而顺序执行程序。

beq $t1,$t2,L1 执行时,MIPS 微处理器内部各模块以及连接线状态如图 3-25 所示。下一时钟到来后,PC 为 0xc,因此顺序执行下一条指令“j L1”。

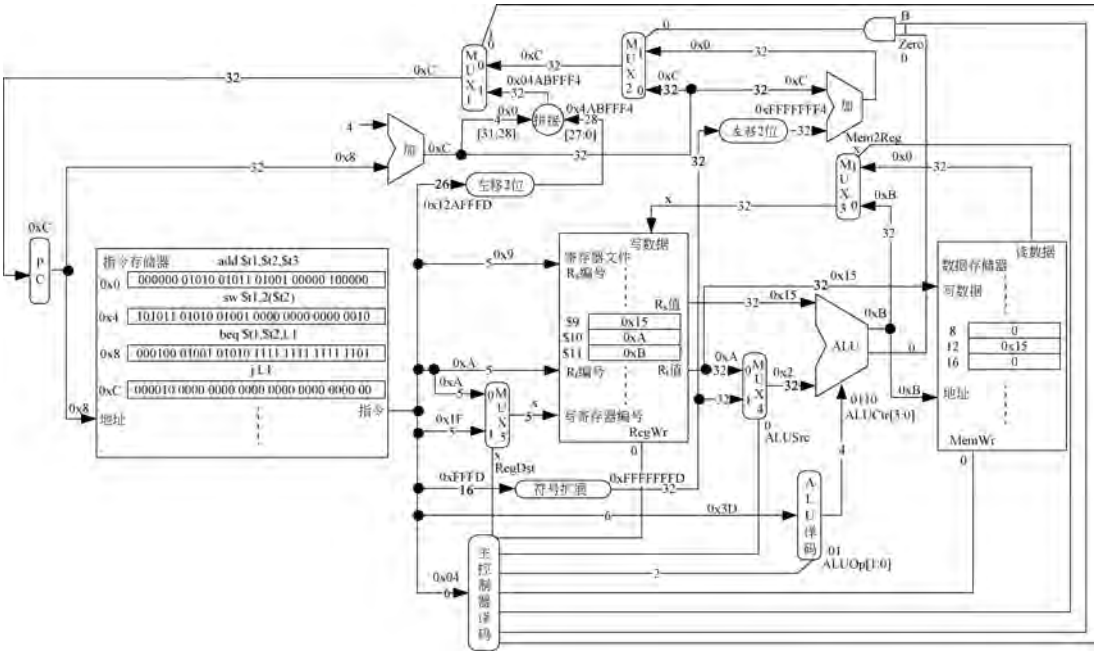


图 3-25　beq $t1,$t2,L1 执行时 MIPS 微处理器内部各模块以及连接线状态

4. J 型指令 j L1


实践视频

执行过程为：

(1) 根据 PC 的值 0xc，从指令存储器中获取指令机器码，并将 PC 加 4，同时也将 Instr[25:0]左移 2 位之后再与 PC+4 的高 4 位合并，形成伪直接跳转目标地址 0x0。

(2) 根据指令机器码，主控制器译码指令操作码使得 J 为 1，此时将伪直接跳转目标地址赋给 PC，即直接跳转到 L1 处。

j L1 执行时，MIPS 微处理器内部各模块以及连接线状态如图 3-26 所示。下一时钟到来后，从头开始执行指令。

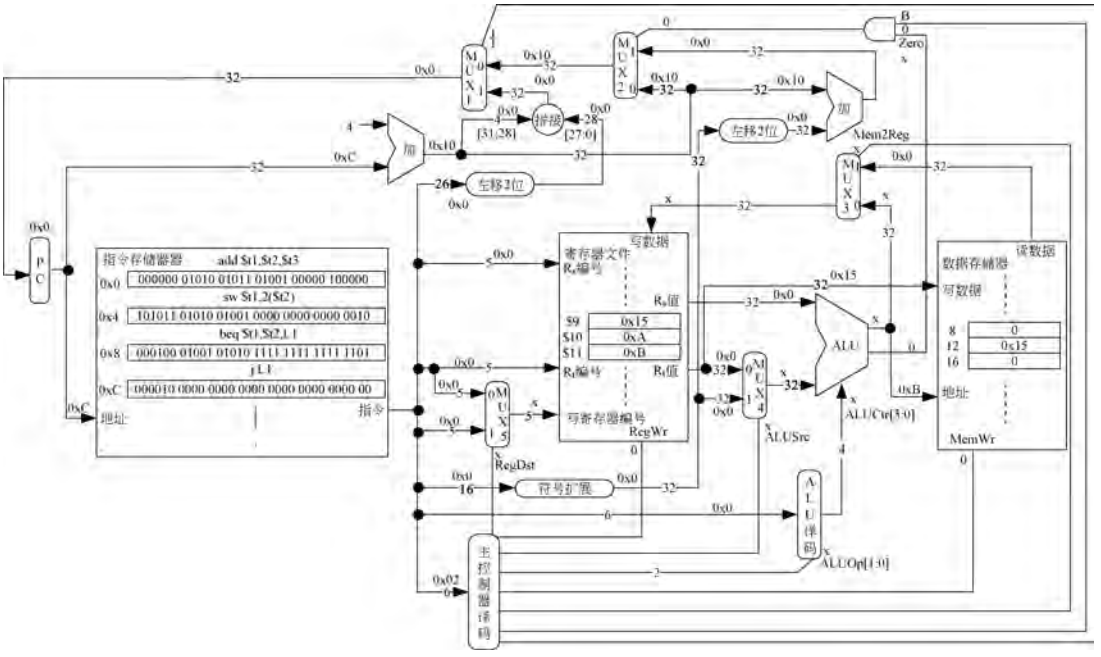


图 3-26 j L1 执行时 MIPS 微处理器内部各模块以及连接线状态

必须注意：不管执行什么指令，数据通路上不受控制的部件一直处于工作状态，如各个地址加法器。指令执行时，有些部件的输出结果没有意义。

至此，阐述了简单指令集 MIPS 微处理器设计的基本原理。现代微处理器比简单 MIPS 微处理器要复杂得多。简单 MIPS 微处理器，指令一条一条地执行，且任何指令都在一个时钟周期内完成。虽然不同指令执行时所需有效数据通路不同，简单 MIPS 微处理器将耗时最长指令的执行时间作为微处理器的最短时钟周期，这种设计方案降低了微处理器的执行效率。

3.3 微处理器新技术

3.3.1 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子过程，前一子过程为下一子过程

创造执行条件，每一个子过程可以与其他子过程同时进行。简而言之，就是“功能分解，空间上顺序依次进行，时间上重叠并行”。流水线微处理器就是把一条指令的操作分为若干级，每一级在一个时钟周期完成，微处理器在每个周期发出一条指令。这样，多条指令就可以由微处理器并行执行，每条指令处在不同级。

下面以 MIPS 微处理器为例，简要阐述微处理器流水线技术的实现原理。

从 MIPS 微处理器执行指令的过程可知，指令执行通常可以分为以下 5 个部分：

(1) 从指令存储器中取指令(取指令)；

(2) 从寄存器中读取数据的同时，对指令进行译码(指令译码)；

(3) 执行指令对应的操作或计算数据存储器地址(运算)；

(4) 从数据存储器中存、取数据(存储器操作)；

(5) 将结果写入寄存器中(回写结果)。

微处理器指令的执行通常都由这 5 个部分构成，分别把这 5 个部分命名为取指令(IF)、指令译码(DEC)、运算(EXEC)、存储器操作(MEM)、回写结果(WB)。如果将微处理器时钟周期缩短，每个部分利用一个时钟周期完成，那么微处理器执行指令的过程可描述为如图 3-27 所示。

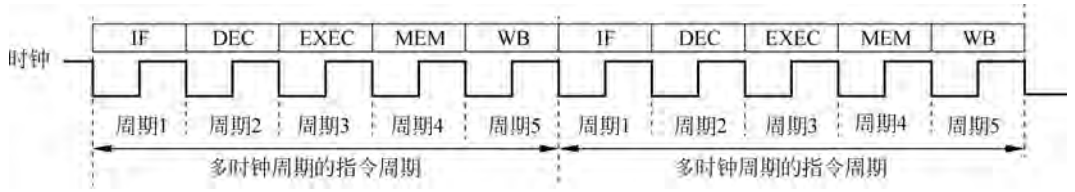


图 3-27 多时钟周期微处理器指令的执行过程

如果微处理器把这 5 个部分设计为相对独立的部件，那么它们可以同时执行对应的工作。即在同一条指令中顺序执行，在不同的指令中并行执行，这就是流水线的基本原理，如图 3-28 所示。流水线具有多少个不同的执行部件，则称这条流水线具有多少级。

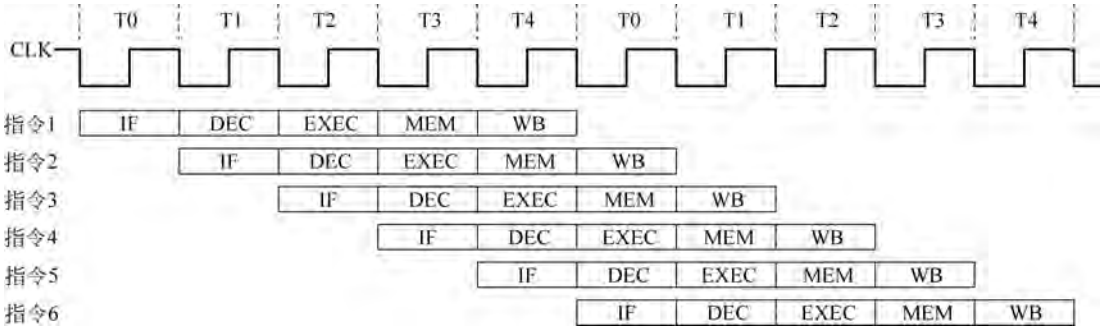


图 3-28 指令流水线执行过程

对比图 3-27 和图 3-28 可以看出，不采用流水线的微处理器在 10 个时钟周期内仅完成 2 条指令，而采用流水线则可以在 10 个时钟周期内完成 6 条指令。

如果流水线的每个部件执行时间相同，那么在流水线结构中，每条指令执行的时间间隔可以表示为

$$\text{指令执行的时间间隔} = \frac{\text{每条指令占有的时钟周期个数}}{\text{流水线级数}}$$

简单 MIPS 微处理器数据通路若按 5 级流水线重新组织，结构变为如图 3-29 所示。

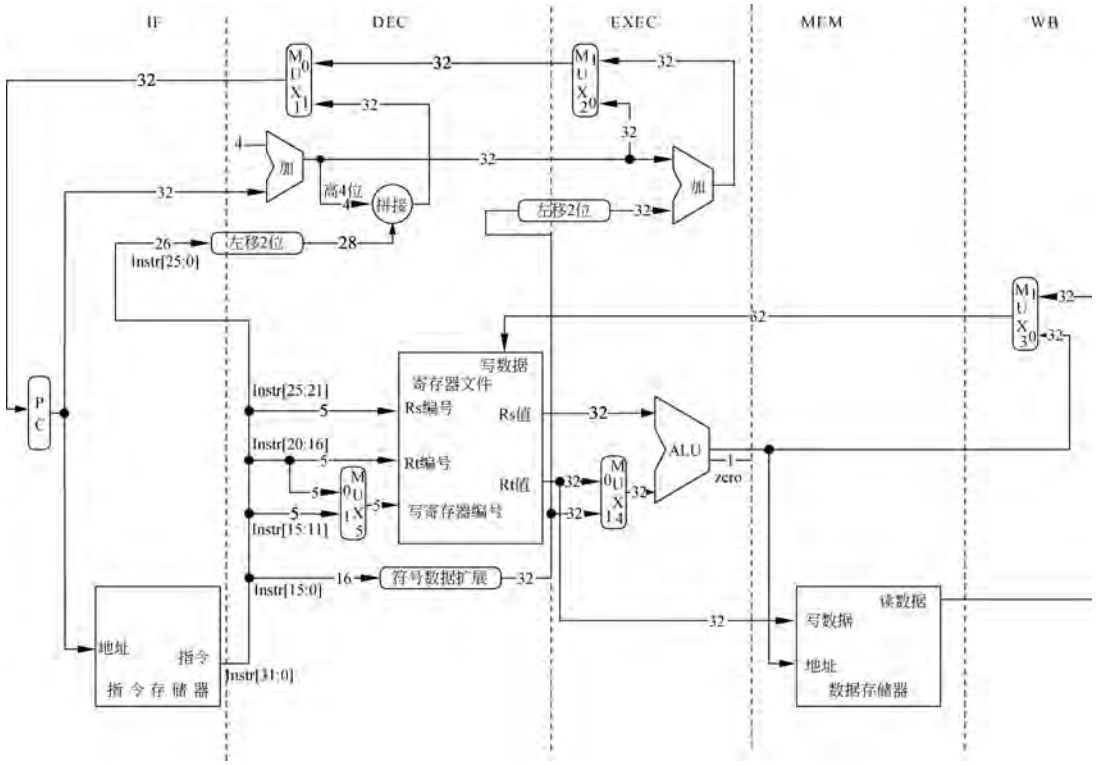


图 3-29　MIPS 微处理器数据通路的 5 级流水线划分

由指令执行过程可知，指令的机器码由流水线的 IF 部件取出来、DEC 部件译码、EXEC 部件计算。因此当 IF 部件取下一条指令时，必须保存上一次取得的指令以供后面的部件使用，也就是说在每两个流水线部件之间必须增加器件实现流水线不同部件之间的接口。虽然在流水线处理器中增加少量的器件就较大幅度地提高了微处理器的执行效率，但是它要处理以下问题。

(1) 结构相关：由于硬件资源不充足而导致流水线不畅通。

(2) 数据相关：由于指令的源操作数依赖其他指令的计算结果而导致读数据不正确。

(3) 转移相关：由于流水线操作导致在转移发生之前，若干条转移指令的后续指令已被取到流水线处理器中。

由于这些问题的存在，流水线并不能从根本上解决微处理器的执行效率问题。

3.3.2　超标量技术

超标量技术是指微处理器内部集成多个 ALU、多个译码器和多条流水线，以并行处理的方式提高性能。采用超标量技术可以提高指令级并行度。这种指令执行的重叠性(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。

流水线处理器在每个时钟周期最多发出 1 条指令,而超标量微处理器可以在每个时钟周期发出 1～8 条指令。同时发出的指令必须是不相关的并且不能出现资源冲突。假设微处理器有一个整数部件和一个浮点部件,微处理器至多能发出 2 条指令:一条是整数类型的指令,包括整数运算、存储器访问和转移指令;另一条必须是浮点类型的指令。如果微处理器执行部件足够,则资源冲突可能性会减小。这样的微处理器指令的调度策略是一个需要考虑的问题。为实现指令的调度,通常微处理器中会设计一个指令缓冲区,用来存放等待调度的指令。

3.3.3 多核处理器

多核处理器是一个芯片上集成两个或多个完整的计算引擎(逻辑 CPU 核),片内总线上的多个逻辑处理器核由总线控制器提供所有总线控制信号和命令信号。每个微处理器核实质上都是一个相对简单的单线程微处理器或者比较简单的多线程微处理器,这样多个微处理器就可以并行地执行程序代码,因而具有了较高的线程级并行性。通过在多个执行内核之间划分任务,多核处理器可在特定的时钟周期内执行更多任务。

由于多核处理器采用了相对简单的微处理器作为处理器核心,多核处理器具有高主频、设计和验证周期短、控制逻辑简单、扩展性好、易于实现、功耗低、通信延迟低等优点。此外,多核处理器还能充分利用不同应用的指令级并行和线程级并行,具有较高线程级并行性的应用如商业应用等可以很好地利用这种结构来提高性能。单芯片多处理器已经成为处理器体系结构发展的一种必然趋势。

3.4 微处理器异常处理机制

计算机除了正常执行程序中的指令之外,往往还有一些特殊的情况需要处理,如计算结果产生了溢出、碰到了非正确的机器指令以及外部设备需要进行数据输入、输出等。微处理器往往不能预测这些事件发生的时间,它们不受程序控制。计算机系统把这些事件称为异常(exception)或中断(interrupt)。MIPS 微处理器中的中断特指微处理器外部事件,而 Intel 微处理器将内部异常事件和外部异常事件统称为中断。计算机系统常见异常事件如表 3-14 所示。

表 3-14 计算机系统常见异常事件

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

微处理器必须提供某种机制处理这类事件,即需具备异常处理机制。异常处理是指计算机在正常执行程序的过程中,由于种种原因,CPU 暂时停止当前程序的执行,而转去处理临时发生的异常事件,并当异常事件处理完毕后,再返回去继续执行暂停程序的过程。也就是说,程序正常执行过程中,意外插入另外一段程序处理异常事件。

微处理器完成异常处理需实现以下功能:

(1) 记录异常事件的类型。

(2) 记录被暂停程序暂停处指令在存储器中的地址,即断点。只有这样,异常事件处理完之后才能继续执行被暂停的程序。

(3) 记录不同异常事件处理程序入口在存储器中的地址。

(4) 建立异常事件与异常处理程序入口地址之间的对应关系。只有建立了这种对应关系,微处理器才能在遇到某个特定异常事件时执行相对应的异常处理程序。

3.4.1 异常事件识别

计算机系统中异常事件种类较多,且不同异常事件处理方式不同。因此微处理器为处理异常事件,首先需要知道异常事件的类型。计算机系统识别异常事件有两种方式:①状态位法,微处理器利用一个特殊功能寄存器对每种异常事件设立一个标志位,当有异常事件发生时,该寄存器中相应的位被置 1,因此一个 32 位的寄存器可以表示 32 种不同类型的异常事件;②向量法,不同异常事件具有一个唯一的编码,这个编码称为**中断类型码或异常类型码**。若产生了异常事件,则微处理器可以接收到该异常事件的异常类型码,解码就可以获知异常事件的类型。一个 8 位的寄存器可以表示 256 种不同类型的异常事件。

3.4.2 断点保存和返回

异常事件发生后,CPU 转去处理临时发生的异常事件,处理完毕后,再返回去继续执行暂停的程序。要能够继续执行暂停的程序,需要保存断点。只有这样,才能处理完异常事件之后恢复到被暂停程序的断点处继续执行。

计算机保存断点的方式有两种:①微处理器设置一个特殊寄存器 EPC,当出现异常事件时,将 PC 的值保存到 EPC 中。异常事件处理完之后,再把 EPC 的值赋给 PC,这样就实现了断点的保存和返回。但是如果计算机正在处理异常事件还没有返回,就不能处理再次发生的异常事件,如同仅采用 $ra 保存主程序的返回地址一样。因此为实现异常事件处理的嵌套,微处理器在进入异常处理程序之前需要首先将 EPC 压入栈,然后再保存 PC 到 EPC。②当出现异常事件时,微处理器直接将 PC 压入栈,异常事件处理结束之后,再从栈顶恢复 PC,同样可以实现断点保存和返回,且不用担心异常事件处理的嵌套问题。这种方式断点保存和返回都需要访问栈。

3.4.3 异常处理程序进入方式

不同异常事件具有不同的异常处理程序,这些异常处理程序都必须在异常事件发生之前保存在存储器中,以便异常事件发生时执行相应的处理。异常处理程序的入口地址,称为

异常向量(中断向量)。异常处理程序保存在存储器的位置与微处理器根据异常事件类型码获取异常处理程序的机制相关。

微处理器获取异常处理程序入口地址有以下几种方式：

(1) 微处理器分配一块专门的存储区域保存异常处理程序。每个异常处理程序在这块存储区域分配固定长度的存储空间(如 8 字节或 2 条指令)，由于这段存储空间存储的指令不能完成异常事件处理，因此通常在这里设置一条跳转指令跳转到真正的异常处理程序。

当异常事件发生时，硬件电路获取异常类型码 N，微处理器根据异常类型码 N 计算出中断向量。如异常处理程序特定存储区域起始地址为某个常数 Imm，那么根据异常事件类型码 N 获取中断向量的计算式为 PC=Imm+N×8，其中 8 为每个异常处理程序分配的固定存储空间大小。

(2) 微处理器仅提供一个异常事件处理程序存放地址，且为某个固定值 Imm，发生异常事件时，都首先转移到该地址执行异常事件处理，该异常处理程序为总异常处理程序。总异常处理程序识别异常事件类型，然后再调用子程序执行异常事件处理。在这种方式下，要求总异常处理程序软件维护一个中断向量表。

当异常事件产生时，微处理器硬件电路只需将 Imm 赋值给 PC，即 PC=Imm，进入总异常处理程序，之后的处理流程由软件实现。这种转入总异常处理程序的硬件电路实现简单，但是由于采用软件查询方式识别异常事件，因此进入真正异常处理程序的效率较低。

(3) 微处理器分配一块专门的存储区域保存异常向量(中断向量)。这块保存中断向量的存储区域称为**中断向量表**。由于每个中断向量都是固定长度的(如 4 字节)，因此存储中断向量的存储空间大小也是固定的。异常处理程序可以存放在存储器的任意位置，只需把中断向量保存到中断向量表中正确的地址。当异常事件发生时，微处理器查找中断向量表直接进入异常处理程序。

当异常事件发生时，硬件电路获取异常类型码 N，微处理器首先根据异常类型码 N 计算出中断向量在中断向量表中的存储地址，然后再获取中断向量。若中断向量表的起始地址为 Imm，当异常事件发生时，硬件电路获取异常类型码 N，计算出中断向量存储地址 Addr=Imm+N×4，然后再将该存储空间中的中断向量赋给 PC，即 PC=mem[Imm+N×4]。这种方式硬件电路较复杂，但是进入真正异常处理程序的速度最快。

假设计算机系统产生了 2 号异常事件，3 种不同方式下微处理器进入异常处理程序的过程如图 3-30 所示。方式 1、2 微处理器内部硬件电路直接修改 PC，然后再由软件跳转到特定异常处理程序；方式 3 微处理器内部硬件电路计算出中断向量在中断向量表中的存储地址，然后再将存储在该地址的中断向量赋给 PC，即由微处理器硬件电路直接控制进入特定异常处理程序。方式 3 微处理器内部硬件电路实现复杂，但软件简单，因此 PC 微处理器(如 Intel x86 系列)采用此方式；而嵌入式微处理器(如 MicroBlaze、PIC32MX 系列)通常将方式 1、2 混合使用，即微处理器将异常事件进行两级分类，即大类和子类，各大类异常处理程序进入方式采用方式 1，每个大类下的子异常处理程序进入方式采用方式 2，即由大类异常处理程序识别各个子异常事件，然后再调用相应的异常处理子程序。

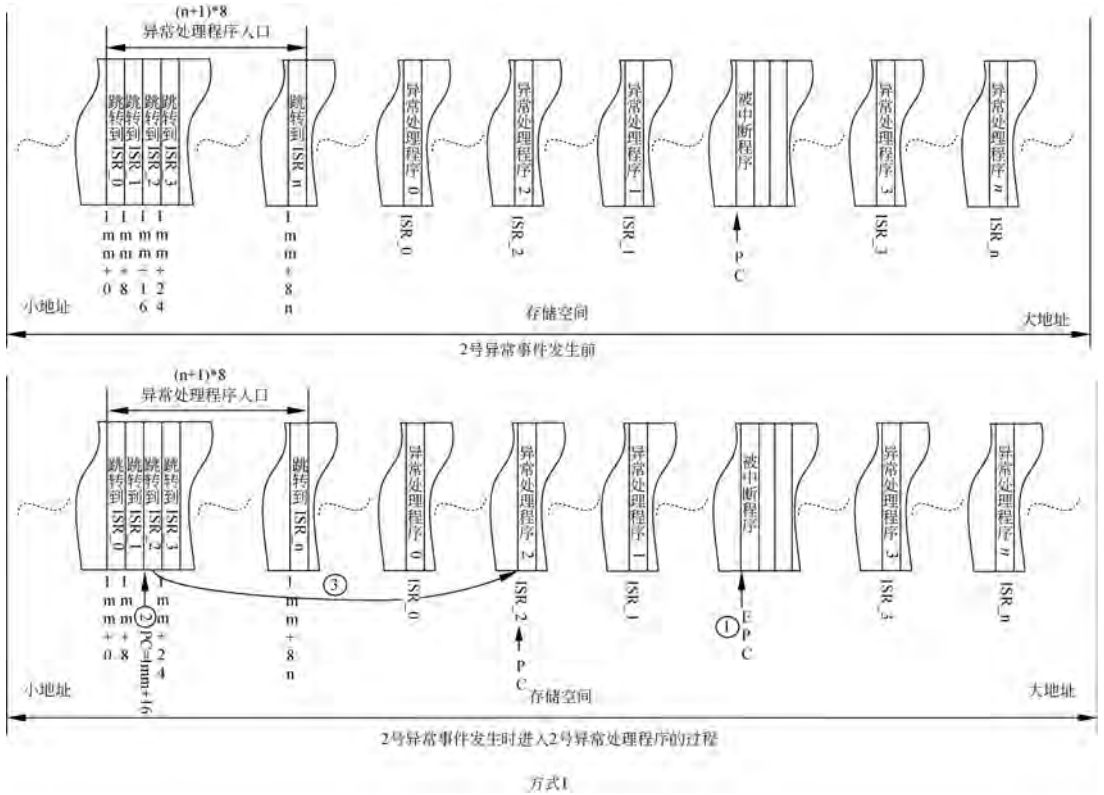


图 3-30　2 号异常事件发生时微处理器 3 种不同方式进入异常处理程序的过程

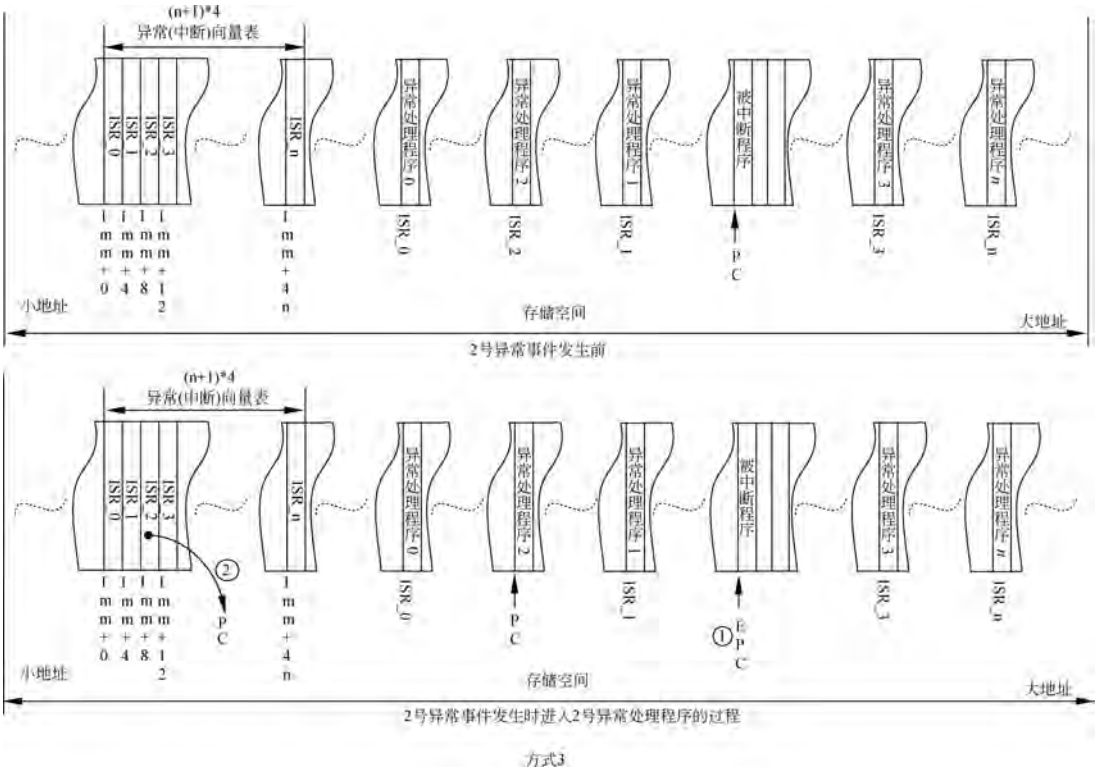


图 3-30　（续）

3.5　微处理器外部接口

3.2 节设计的简单 MIPS 微处理器包含了指令存储器和数据存储器，这样设计只是为了简化微处理器设计。实际上，微处理器内部仅含有少量存储单元，如通用计算机微处理器的片内 cache，嵌入式微处理器的部分指令存储器和数据存储器。也就是说，计算机系统大量的存储单元位于微处理器外部。微处理器与存储器之间的数据交互，必须通过微处理器外部总线。另外，微处理器作为计算核心，处理的数据除了来源于存储器外，还有大量的数据来自计算机外围各种数据采集设备，如鼠标、键盘、麦克风、摄像头等。微处理器也必须通过总线与这些外围设备之间交互数据。即微处理器必须提供总线接口，才能访问外部器件。

3.5.1　Intel x86 微处理器外部接口示例

Intel x86 微处理器的外部接口采用系统总线接口方式，图 3-31 展示了 8088 微处理器的外部引脚。其中，$AD_0 \sim AD_7$ 为 8 位地址、数据复用总线接口，$A_8 \sim A_{19}$ 为高 12 位地址总线，其余引脚除了电源、时钟引脚之外就是控制总线。

图 3-32 是针对 8088 微处理器引脚特点设计的最小组态下系统总线接口电路。ALE 为 CPU 提供的地址锁存信号，当该信号有效时，$AD_0 \sim AD_7$ 及 $A_{16} \sim A_{19}$ 输出地址信号，因此可以利用 3 组锁存器 74xx373 在 ALE 地址锁存使能信号的控制下锁存 CPU 输出的地址

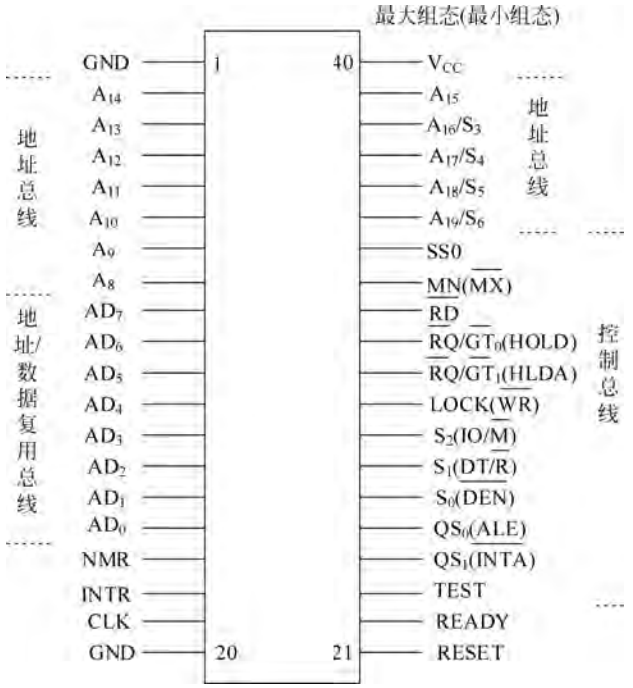


图 3-31　8088 微处理器外部引脚

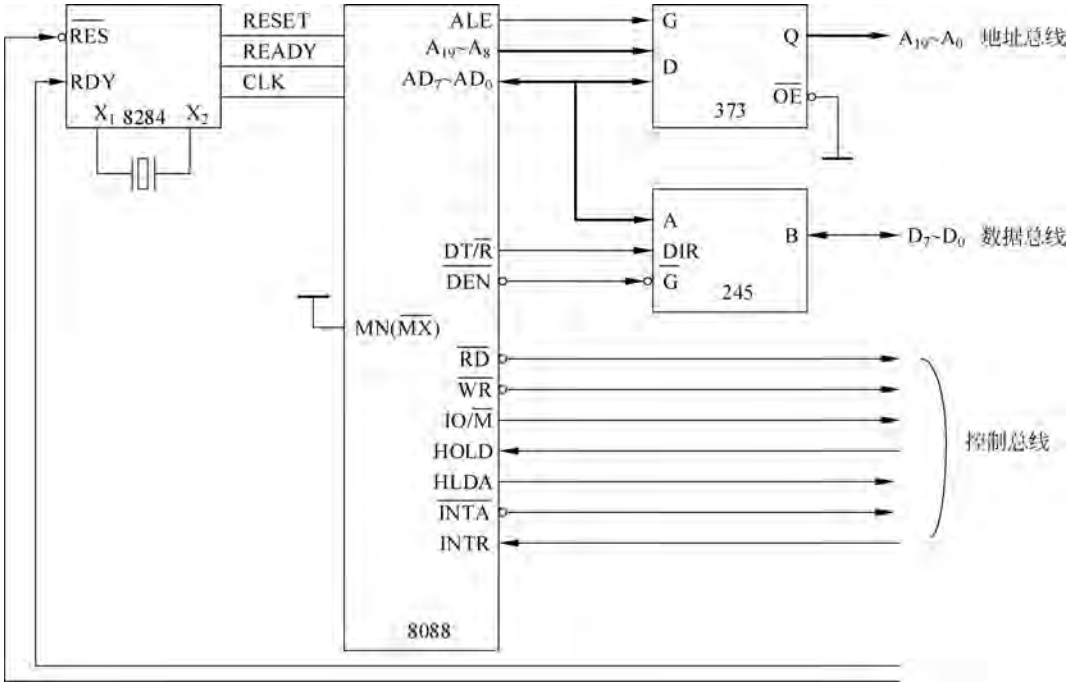


图 3-32　8088 微处理器最小组态系统总线接口电路

信号，从而提供分离的地址总线。DEN 为数据有效信号，当该信号有效时，AD_0～AD_7 传输数据信号，数据的传输方向由 DT/R 引脚指示，可以通过 1 个双向缓冲器 74xx245 在 DEN 以及 DT/R 的控制下分离出数据总线。控制引脚没有复用，直接由 CPU 引脚提供。从图 3-32 可以看出，8088 微处理器工作时，必须外加时钟产生电路 8284 以产生时钟和复位信号。

微型计算机的微处理器外部接口基本上都采用这种总线方式。不同类型的存储芯片或 I/O 设备与微处理器相连时，都可以通过特有的接口电路连接到总线上，一方面为用户设计各种计算机接口电路提供了较大的自由度，另一方面也给用户的接口电路设计增加了难度。

3.5.2　嵌入式微处理器外部接口示例

为降低用户设计接口电路的难度，嵌入式微处理器将大部分常见接口电路都集成到了微控制器内部，因此用户看到的大多数嵌入式微控制器提供的外部接口并非总线方式，而是针对不同输入、输出设备或存储器提供的不同接口。图 3-33 为 32 位 Microchip 嵌入式芯片 PIC32MX5XX/6XX/7XX 系列的内部结构框图。

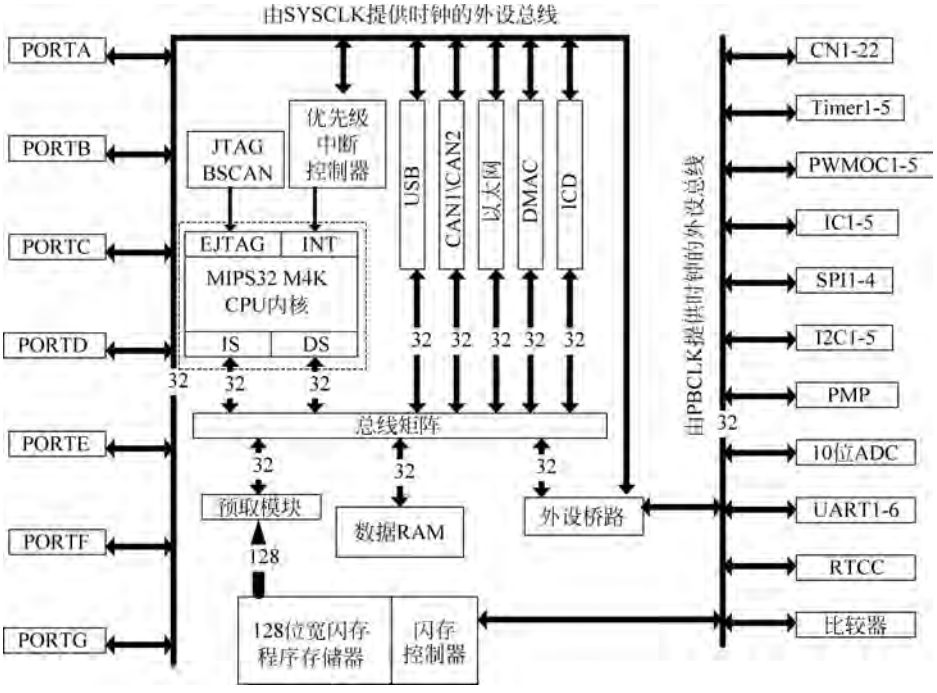


图 3-33　嵌入式芯片 PIC32M×5××/6××/7××系列内部结构框图

从图 3-33 可以看出，这个芯片已经集成了相当多的外部总线或接口，如 USB、CAN、Ethernet、I2C、SPI、UART 等。该芯片已经不再是单纯意义上的微处理器，它是一个微缩的集成计算机硬件系统。微处理器仅是这类嵌入式微控制器的一部分，如图 3-33 虚线框内的 MIPS32 CPU 内核。虽然嵌入式芯片外部引脚结构发生了较大的改变，但是从图 3-33 可以看出，该微控制器的 CPU 仍然是通过统一的总线结构（总线矩阵）与微控制器内各个部件相连。

为便于读者学习计算机工作基本原理，本书中的微处理器指采用统一总线接口方式即三总线方式与外部部件实现数据通信的中央处理器（CPU）。

3.6 MicroBlaze 微处理器简介

MicroBlaze 微处理器是 Xilinx 公司采用硬件描述语言设计的一个软核类 MIPS 指令集微处理器，它支持 32 位数据总线和 32 位地址总线，其基本结构如图 3-34 所示。它包含了简单 MIPS 微处理器的各个部件，并且功能更完善，指令集更复杂。它将指令存储器和数据存储器放在微处理器之外，为实现指令和数据的访问，设计了专门的总线接口部件，由于采用哈佛结构，因此有专门的指令总线接口和数据总线接口。该微处理器可以配置为大字节序或小字节序管理存储器，如采用 PLB 总线为大字节序，采用 AXI 总线为小字节序。支持三级或五级流水线，内部具有可选的指令和数据 cache，同时支持虚拟存储管理。它支持通过 PLB、LMB、AXI 等总线与外围接口或部件相连。

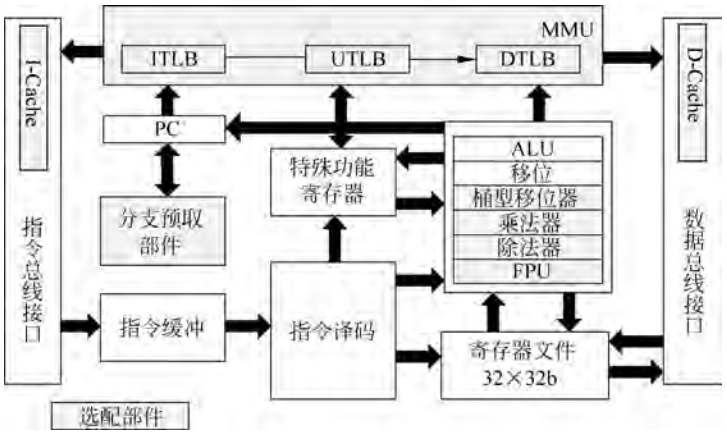


图 3-34 MicroBlaze 微处理器基本结构

3.6.1 指令架构

MicroBlaze 微处理器支持两种指令格式：A 型指令和 B 型指令。A 型指令类似于 MIPS 指令架构中的 R 型指令，A 型指令编码如图 3-35 所示。

图 3-35 A 型指令编码

B 型指令类似于 MIPS 指令架构中的 I 型指令，B 型指令编码如图 3-36 所示。

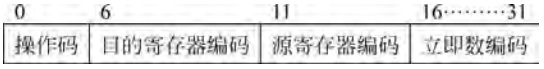


图 3-36 B 型指令编码

MicroBlaze 将 MIPS 指令中的 J 型指令合并到 B 型指令中，此时目的寄存器编码为保存返回地址的寄存器编码，源寄存器编码不再是真正意义的源寄存器编码，各位具体含义如图 3-37 所示。

11	12	13	14	15
D	A	L	0	0

图 3-37 B 型跳转控制指令的源寄存器编码各位含义

其中，D 表示是否延时之后再跳转，1 表示延时数个时钟周期再跳转；0 表示立即跳转。A 表示是否是直接跳转，1 表示指令中的立即数符号扩展后为直接跳转目标地址；0 表示立即数为相对跳转偏移地址，跳转目标地址为 PC+Imm(立即数)。L 表示是否保存返回地址，1 表示保存下一条指令地址到目的寄存器作为返回地址；0 表示不保存返回地址。

3.6.2 寄存器

MicroBlaze 微处理器具有 32 个 32 位通用寄存器，使用规则与 MIPS 微处理器通用寄存器的使用规则基本相同，命名为 R0～R31。其中 R14、R15、R16、R17 又用作异常返回地址寄存器，具体使用规则参考本书中断技术章节相关内容。另外还具有 18 个 32 位特殊功能寄存器，包括 PC、MSR 等。这些特殊功能寄存器根据用户对 MicroBlaze 微处理器的配置决定是否存在。各个特殊功能寄存器的具体含义请读者参考 MicroBlaze 数据手册，本书不再一一介绍。

3.6.3 外部接口

MicroBlaze 微处理器外部接口如图 3-38 所示。它采用哈佛架构，因此具有独立的数据总线接口和指令总线接口。数据总线接口支持高速缓存 AXI 总线主设备接口(M_AXI_DC)、高速缓存 AXI 一致性扩展总线主设备接口(M_ACE_DC)、AXI 外设总线主设备接口(M_AXI_DP)、局部存储器总线接口(DLMB)、16 个 AXI 总线流式主设备接口(M0_AXIS～M15_AXIS)和 16 个 AXI 总线流式从设备接口(S0_AXIS～S15_AXIS)；指令总线接口支持高速缓存 AXI 总线主设备接口(M_AXI_IC)、高速缓存 AXI 一致性扩展总线主设备接口(M_ACE_IC)、AXI 外设总线主设备接口(M_AXI_IP)、局部存储器总线接口(ILMB)。MicroBlaze 微处理器具有调试接口(DEBUG)和跟踪接口(TRACE)，以便系统开发、调试和测试。同时具备多微处理器同步接口(LOCKSTEP)以便多 CPU 协同工作，以及中断接口(INTERRUPT)以实现与外部设备的中断方式通信。

对比图 3-31、图 3-33 以及图 3-38，不难发现 PC 机的 CPU 与嵌入式微处理器的 CPU 外部引脚接口的差别——嵌入式 CPU 需额外提供调试、测试接口以便程序调试、测试。

3.6.4 最小系统


实践视频

MicroBlaze 微处理器支持的总线、接口种类众多，构成计算机硬件最小系统时，并不一定需要使用所有总线、接口。MicroBlaze 微处理器构成的嵌入式计算机硬件最小系统结构框图如图 3-39 所示，包括微处理器(MicroBlaze)、存储器(局部存储器)、总线(AXI 总线)、标准输入/输出接口 UART(Standard Input and Output，STDIO)、调试模块(JTAG BSCAN)以及计算机系统正常工作的时钟、复位模块。

由于嵌入式计算机系统常作为其他设备的嵌入模块，因此通常不具备微型计算机系统

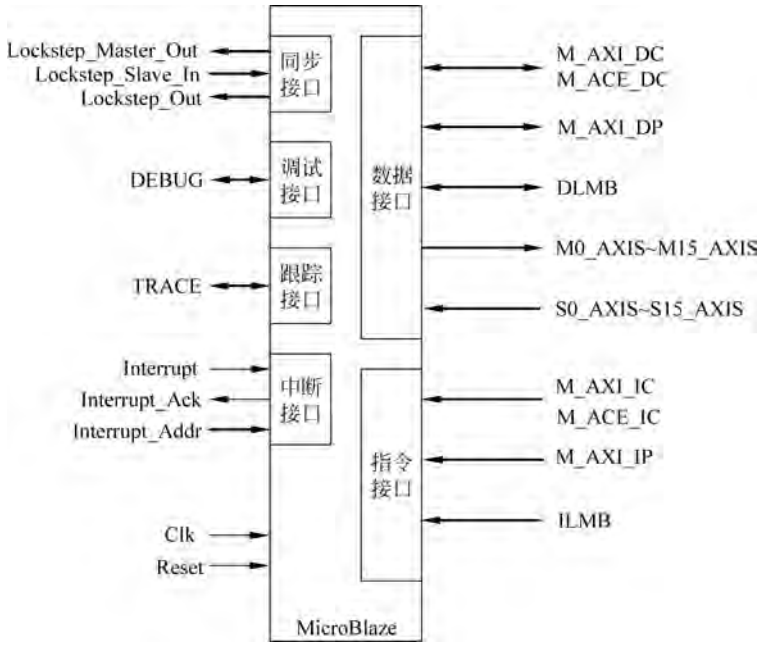


图 3-38　MicroBlaze 微处理器外部接口

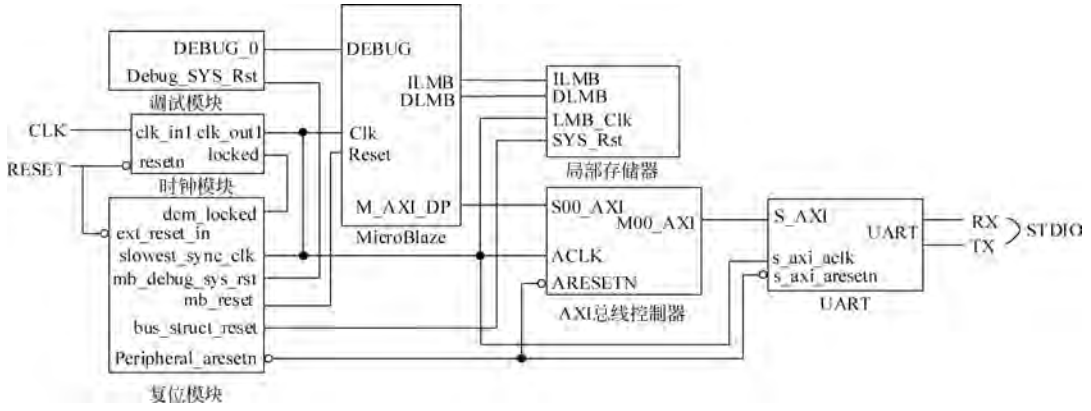


图 3-39　MicroBlaze 微处理器构成的嵌入式最小计算机系统硬件结构框图

所具有的标准 I/O 设备(键盘、鼠标、显示器)。为方便用户调试程序,常将微型计算机的 I/O 设备作为其 I/O 工具。此时由于微型计算机、嵌入式计算机系统分属两个不同的计算机系统,因此必须通过某个接口实现双机通信,嵌入式系统的标准 I/O 通信接口为 UART 串行通信接口。微型计算机系统作为嵌入式计算机系统标准 I/O 设备的连接拓扑如图 3-40 所示。

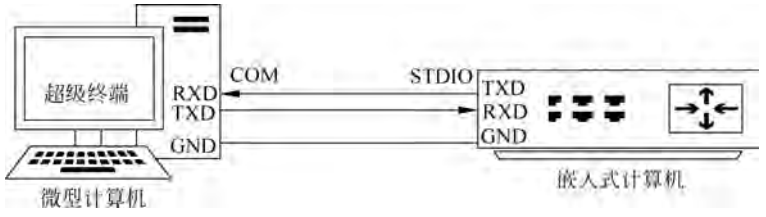


图 3-40　微型计算机系统作为嵌入式计算机系统标准 I/O 设备的连接拓扑

本章小结

微处理器支持的基本操作包括运算、数据传送以及程序控制三类，微处理器通过 ALU、译码器以及寄存器的配合实现以上操作。

简单指令集 MIPS 微处理器数据通路包含寄存器文件、ALU 运算单元、数据存储器、指令存储器、符号数据扩展、移位以及复用器等模块。数据通路上各个部件的控制信号由主控制器以及 ALU 控制器根据指令操作码、功能码译码产生。控制信号产生采用两级译码：第一级根据操作码产生各个模块的写控制信号、复用器的通道选择信号以及 ALU 控制器的编码信号；第二级再根据功能码及 ALU 控制器的编码信号产生 ALU 运算单元的控制信号。

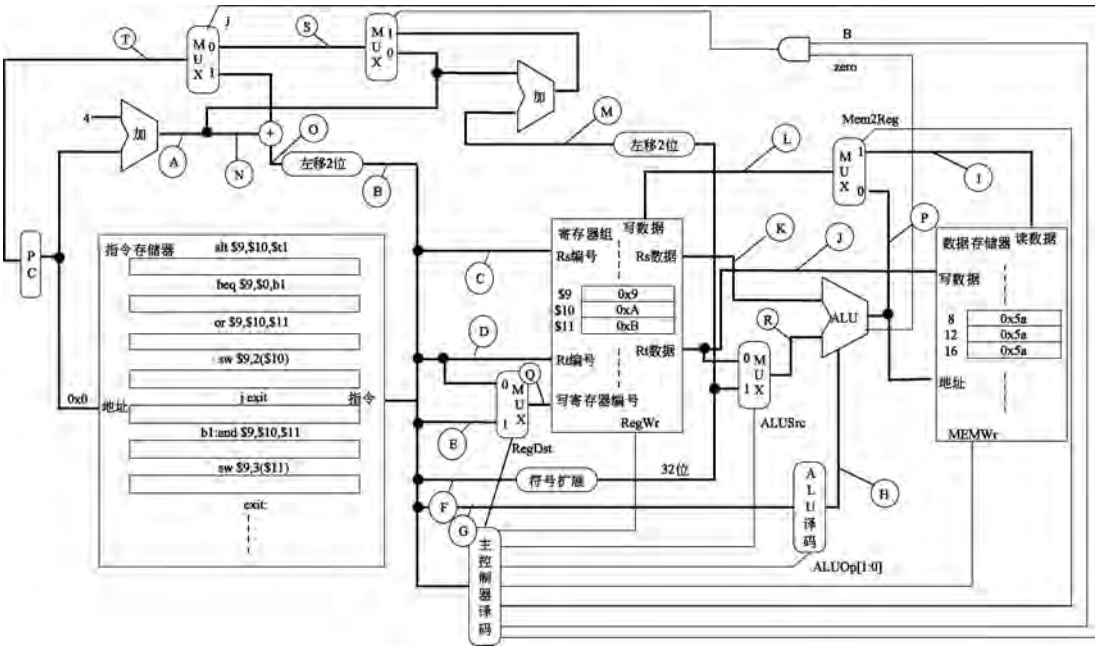
单周期 MIPS 微处理器仅是一个原型微处理器，现代微处理器采用了更复杂的技术，如流水线技术、超标量技术、多核技术等。这些技术可以实现程序的指令级、线程级并行。

微处理器除了在程序的控制下正常执行各类操作之外，还必须支持异常处理。不同微处理采用的异常处理机制不同。无论如何，都必须采取某种方式识别异常事件、保存断点、建立异常事件(中断类型码)与异常处理程序入口地址(中断向量)之间的映射机制。识别异常事件的方法有状态位、中断类型码；保存断点方法有寄存器、栈；进入异常处理程序的方法有分配固定的存储区域保存异常处理程序，为所有异常处理程序提供一个统一的入口，分配固定的存储区域保存异常处理程序入口地址。

微处理器通过总线与计算机系统内其他模块通信，因此微处理器必须具备总线接口。嵌入式微控制器不同于通常意义的微处理器，它集成了微处理器、存储器、接口等。MicroBlaze 微处理器是一个采用硬件描述语言实现的嵌入式类 MIPS 微处理器软核，具有可配置的特点。嵌入式计算机系统通常不支持在系统内编程、调试，因此嵌入式计算机硬件最小系统除了微处理器、存储器之外，还提供调试接口(JTAG BSCAN)以及作为标准 I/O 接口的 UART 接口。

思考与练习

1. 微处理器的三种基本操作分别是什么？
2. 微处理器的基本构成包括哪些部分？这些部分分别起什么作用？
3. 微处理器的数据通路包括哪些部件？分别完成什么功能？
4. 试对比分析图 3-1 与图 3-20 微处理器构成的异同。
5. 支持简单指令集 MIPS 微处理器结构如题图 3-1 所示。当该微处理器初始化时，通用寄存器的值与编码一致，数据存储器以字为最小单位且所有存储单元都初始化为 0x5a，PC 寄存器的初始值为 0，指令存储器从地址 0 开始存储如题图 3-2 所示汇编指令段的机器指令。



题图 3-1　简单指令集 MIPS 微处理器

```
    slt
$9,$10,$11
    beq $9,$0,b1
    or
$9,$10,$11
    sw $9,2($10)
    j exit
    b1:and
$9,$10,$11
    sw $9,3($11)
    exit:
```

题图 3-2　MIPS 汇编程序段

回答以下问题：

(1) 完善题图 3-2 中各汇编指令的机器指令(采用二进制表示)。

汇 编 指 令	Op	Rs	Rt	Rd	Shamt	Funct
slt $9,$10,$11	000000				00000	101010
beq $9,$0,b1	000100					
or $9,$10,$11	000000				00000	100101
sw $9,2($10)	101011					
j exit	000010					
b1:and $9,$10,$11	000000				00000	100100
sw $9,3($11)	101011					

(2) 指出题图 3-1 中各标注信号线的位宽。

信号线编号	信号线位宽	信号线编号	信号线位宽
A		E	
B		F	
N		G	
O		H	
C		M	
D			

(3) 指出题图 3-2 程序段相应指令执行时,题图 3-1 中各标注信号线的值(十六进制表示)。

指　令	A	B	C	D	E	F	G	R	P	L	K	M	S	T
slt $9,$10,$11														
beq $9,$0,b1														
or $9,$10,$11														
sw $9,2($10)														
j exit														

(4) 指出题图 3-2 程序段执行完后,值被修改的寄存器编号及其修改后的值或值被修改的存储单元地址及其修改后的值。

被修改的寄存器编号或存储单元地址	修改后的值

6. 若在简单 MIPS 指令集基础上增加 addiu 指令,图 3-9 所示简单 MIPS 微处理器数据通路构成是否需要修改? 若需要修改,如何修改? 并基于修改后的数据通路完成 ALU 控制器译码和主控制器译码设计,画出各个译码器的逻辑真值表。

7. 若在简单 MIPS 指令集基础上增加 ori 指令,图 3-9 所示简单 MIPS 微处理器数据通路构成是否需要修改? 若需要修改,如何修改? 并基于修改后的数据通路完成 ALU 控制器译码和主控制器译码设计,画出各个译码器的逻辑真值表。

8. 简述流水线技术、超标量技术、多核技术的特点。

9. 微处理器识别异常事件的方法有哪些? 它们各有什么优缺点?

10. 微处理器保存断点的方法有哪些? 它们各有什么优缺点?

11. 微处理器进入异常处理的方法有哪些? 它们各有什么优缺点?

12. 阅读 MicroBlaze 数据手册,说明 MSR 寄存器各位的具体含义。

13. 微控制器与微处理器的区别是什么?

14. 嵌入式计算机硬件最小系统包含哪些模块? 分别实现什么功能?

15. 嵌入式计算机系统常用的标准输入/输出接口是什么接口? 它如何实现嵌入式系统的输入/输出?