

计算机组成原理

第五章 指令系统

5.5 MIPS指令概述

lw \$t0, 0(\$2)

lw \$t1, 4(\$2)


sw \$t1, 0(\$2)

sw \$t0, 4(\$2)

0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111

1

MIPS指令概述

- MIPS (Microprocessor without Intellocked Pipeline Stages)是80年代初期由斯坦福大学Hennessy教授领导的研究小组研制成功; **Million Instructions Per Second**
- 属于精简指令集计算机RISC(Reduced Instruction Set Computer);

复杂指令集计算机CISC(Complex Instruction Set Computer);
- MIPS指令集有MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, MIPS32, 和 MIPS64多个版本;
- 早期主要用于嵌入式系统, 如Windows CE的设备, 路由器, 家用网关和视频游戏机, 现在已经在PC机、服务器中得到广泛应用

- MIPS指令集有以下特点：
 - ◆ 简单的Load/Store结构
 - ◆ 易于流水线CPU设计
 - ◆ 易于编译器开发
 - ◆ MIPS指令的寻址方式非常简单，每条指令的操作也非常简单

2

MIPS指令格式概述

- 只有三种指令格式



- ◆ R_s, R_t 分别为第一、二源操作数； R_d 为目标操作数；



- ◆ 双目、Load/Store: R_s 和立即数是源操作数， R_t 为目标操作数；
- ◆ 条件转移: R_s, R_t 均为源操作数；



- ◆ 26位立即数作为跳转目标地址的部分地址

3

MIPS 寄存器

寄存器名	寄存器编号	用途说明
\$s0	0	保存固定的常数0
\$at	1	汇编器的临时变量
\$v0 ~ \$v1	2 ~ 3	子函数调用返回结果
\$a0 ~ \$a3	4 ~ 7	函数调用参数1 ~ 3
\$t0 ~ \$t7	8 ~ 15	临时变量，函数调用时不需要保存和恢复
\$s0 ~ \$s7	16 ~ 23	函数调用时需要保存和恢复的寄存器变量
\$t8 ~ \$t9	24 ~ 25	临时变量，函数调用时不需要保存和恢复
\$k0 ~ \$k1	26 ~ 27	中断、异常处理程序使用
\$gp	28	全局指针变量(Global Pointer)
\$sp	29	堆栈指针变量(Stack Pointer)
\$fp	30	帧指针变量(Frame Pointer)
\$ra	31	返回地址(Return Address)

◆还有32个32位单精度浮点寄存器 f_0-f_{31}

◆还有2个32位乘、商寄存器 Hi 和Lo；乘法法分别存放64位乘积的高、低 32位；除法时分别存放余数和商。

4

MIPS 寻址方式

■ 在MIPS32指令集中，不单设寻址方式说明字段

◆ R型指令：由op和funct字段共同隐含说明当前的寻址方式；



◆ I型和J型指令：由op字段隐含说明当前指令使用的寻址方式。



4

MIPS 寻址方式

■ 立即数寻址 (Immediate addressing)



addi s1, s2, 10 ($\$s1 \leftarrow \$s2 + E(10)$)

注意：汇编格式和编码格式段的对应关系。

4

MIPS 寻址方式

■ 寄存器直接寻址(Register Addressing)

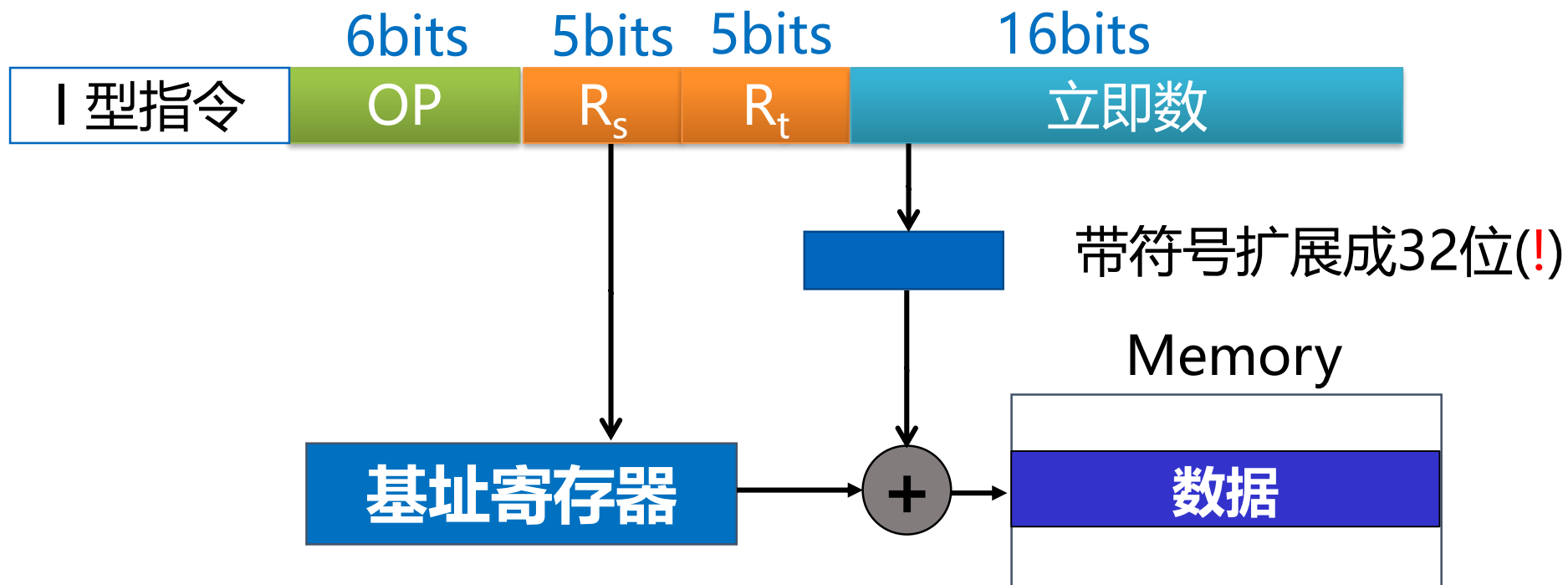


add \$t0,\$s1,\$s2 (\$t0=\$s1+\$s2)

4

MIPS 寻址方式

■ 基址寻址(Basic Addressing)

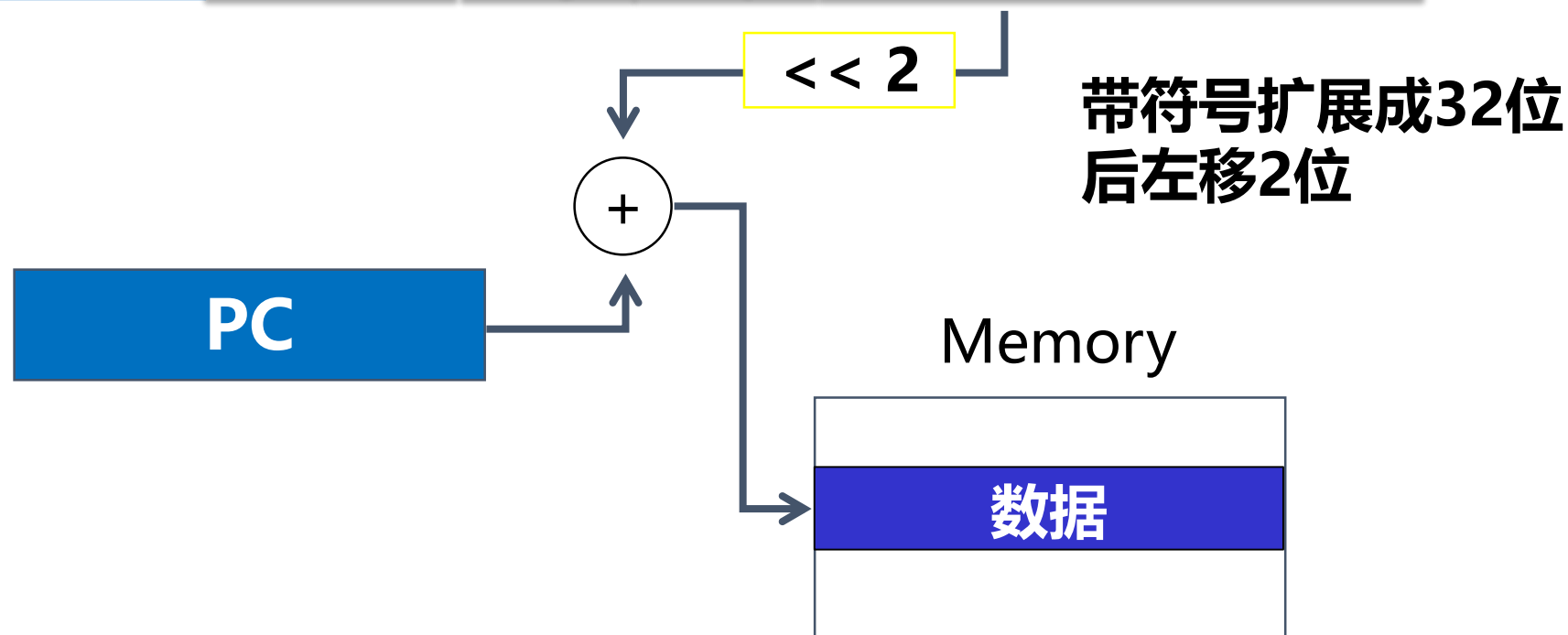


■ 使用基址寻址的指令: lw ,sw, lh, sh, lb, lbu等

LB rt , offset (base)

4 MIPS 寻址方式

■ 相对寻址



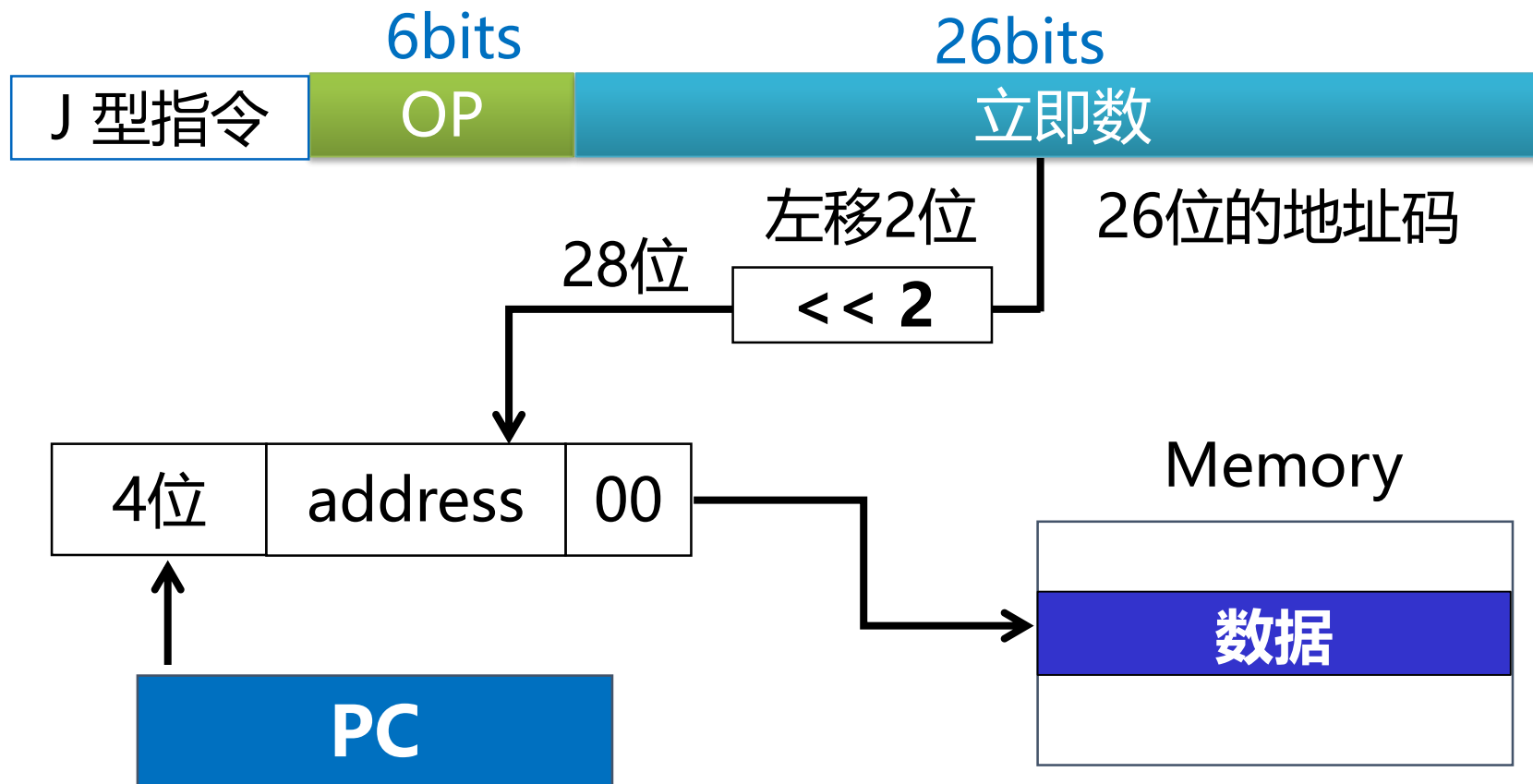
```
if (GRP[rs] == GPR[rt])  
PC = PC + 4 + BranchAddr
```

■ 使用相对寻址的指令: beq, bne

4

MIPS 寻址方式

■ 伪直接寻址(页面寻址)



■ 使用伪直接寻址的指令: j, jal