7.2 MIPS32 指令系统介绍

同济大學
TONGJI UNIVERSITY

目录 | CONTENT

7.2.1 指令格式及类型

- MIPS32架构中的所有指令都是32位，也就是32个0、1编码连在一起表示一条指令，有三种指令格式。如图7.2.1所示。其中op是指令码、func是功能码。

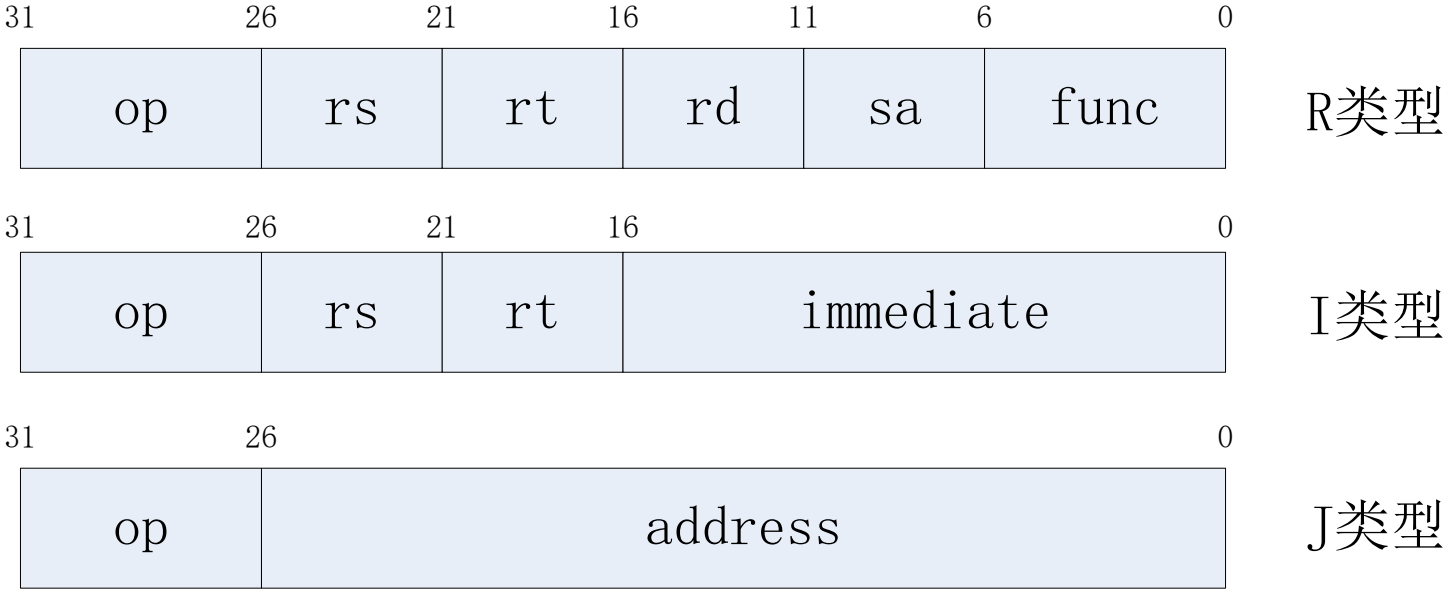


图7.2.1 MIPS32指令类型

① R类型：该类型指令从寄存器堆（register file）中读取两个源操作数，计算结果写回寄存器堆。具体操作由op、func结合指定，rs和rt是源寄存器的编号，rd是目的寄存器的编号，比如：假设目的寄存器是$3，那么对应的rd就是00011（此处是二进制）。MIPS32架构中有32个通用寄存器，使用5位编码就可以全部表示，所以rs、rt、rd的宽度都是5位。sa只有在移位指令中使用，用来指定移位位数。

② I类型：该类型指令使用一个16位的立即数作为一个源操作数。具体操作由op指定，指令的低16位是立即数，运算时要将其扩展至32位，然后作为其中一个源操作数参与运算。

③ J类型：该类型指令使用一个26位的立即数作为跳转的目标地址（target address）。具体操作由op指定，一般是跳转指令，低26位是字地址，用于产生跳转的目标地址。

- 按功能来区分，MIPS32架构中定义的指令可以分为以下几类

① 逻辑操作指令：and、andi、or、ori、xor、xori、nor、lui共8条指令，实现逻辑与、或、异或、或非等运算。

② 移位操作指令：sll、sllv、sra、srav、srl、srlv共6条指令，实现逻辑左移、右移、算术右移等运算。

③ 移动操作指令：movn、movz、mfhi、mthi、mflo、mtlo共6条指令，用于通用寄存器之间的数据移动，以及通用寄存器与HI、LO寄存器之间的数据移动。

④ 算术操作指令： add、addi、addiu、addu、sub、subu、clo、clz、slt、slti、sltiu、sltu、mul、mult、multu、madd、maddu、msub、msubu、div、divu共21条指令，实现了加法、减法、比较、乘法、乘累加及除法等运算。

⑤ 转移指令：jr 、jalr 、j 、jal、b、bal、beq、bgez、bgezal、bgtz、blez、bltz、bltzal、bne共14条指令，其中既有无条件转移，也有条件转移，用于程序跳转执行。

⑥ 加载存储指令：lb、lbu、lh、lhu、ll、lw、lwl、lwr、sb、sc、sh、sw、swl、swr共14条指令，以“l”开始的都是加载指令，以“s”开始的都是存储指令，这些指令用于从存储器中读取数据，或者向存储器中保存数据。

⑦ 协处理器访问指令：mtc0、mfc0共2条指令，用于读取协处理器CP0中某个寄存器的值，或者将数据保存到协处理器CP0中的某个寄存器。

⑧ 异常相关指令：有14条指令，其中有12条自陷指令，包括：teq、tge、tgeu、tlt、tltu、tne、teqi、tgei、tgeiu、tlti、tltiu、tnei，此外还有系统调用指令syscall、异常返回指令eret。

⑨ 其余指令：nop、ssnop、sync、pref共4条指令，其中 nop是空指令，ssnop是一种特殊类型的空指令，sync指令用于保证加载、存储操作的顺序，pref指令用于缓存预取。

7.2.2 指令的寻址

- MIPS32架构的寻址模式有**寄存器寻址**、**立即数寻址**、**寄存器相对寻址和PC相对寻址**四种。实际上，MIPS硬件只支持一种寻址模式，即：寄存器基地址 + 立即数偏移量，且offset必须在-32768～32767之间（16位），任何载入和存储机器指令都可以写成：lw $1，offset（$2） 可以使用任何寄存器作为目的操作数或源操作数。
- MIPS汇编器可以利用合成指令来支持多种寻址方式。MIPS32架构的寻址模式中，寄存器寻址和立即数寻址与其他架构类似，在此不再详述，下面具体介绍下寄存器相对寻址以及PC相对寻址。

寄存器相对寻址

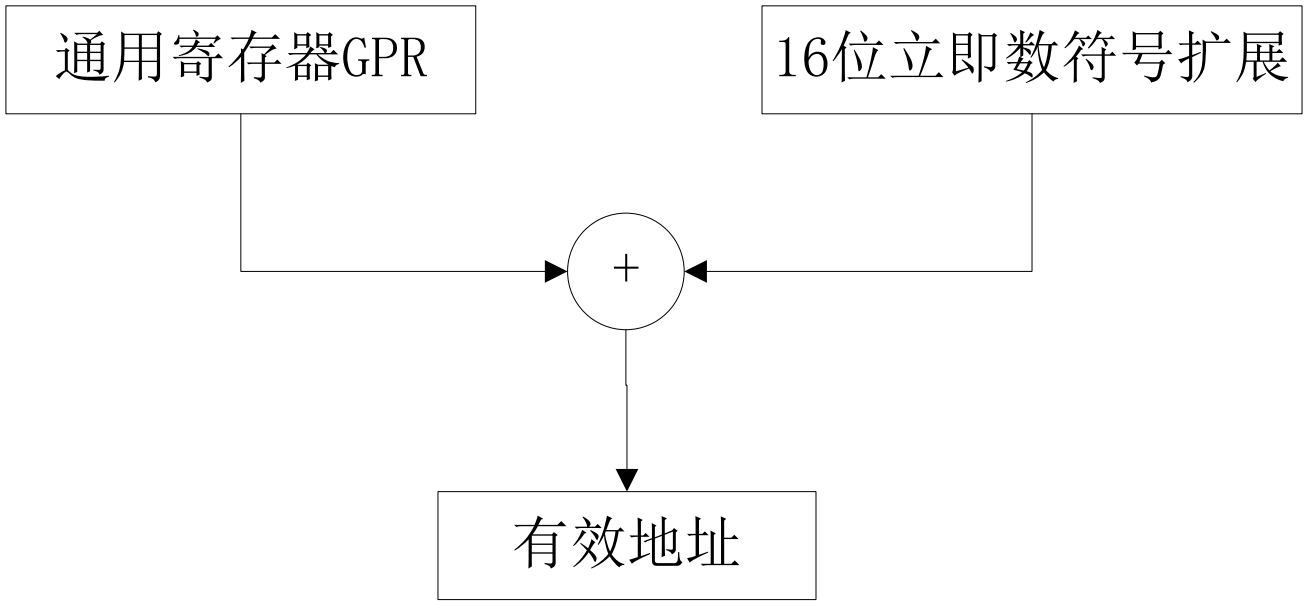


图7.2.2寄存器相对寻址

PC相对寻址

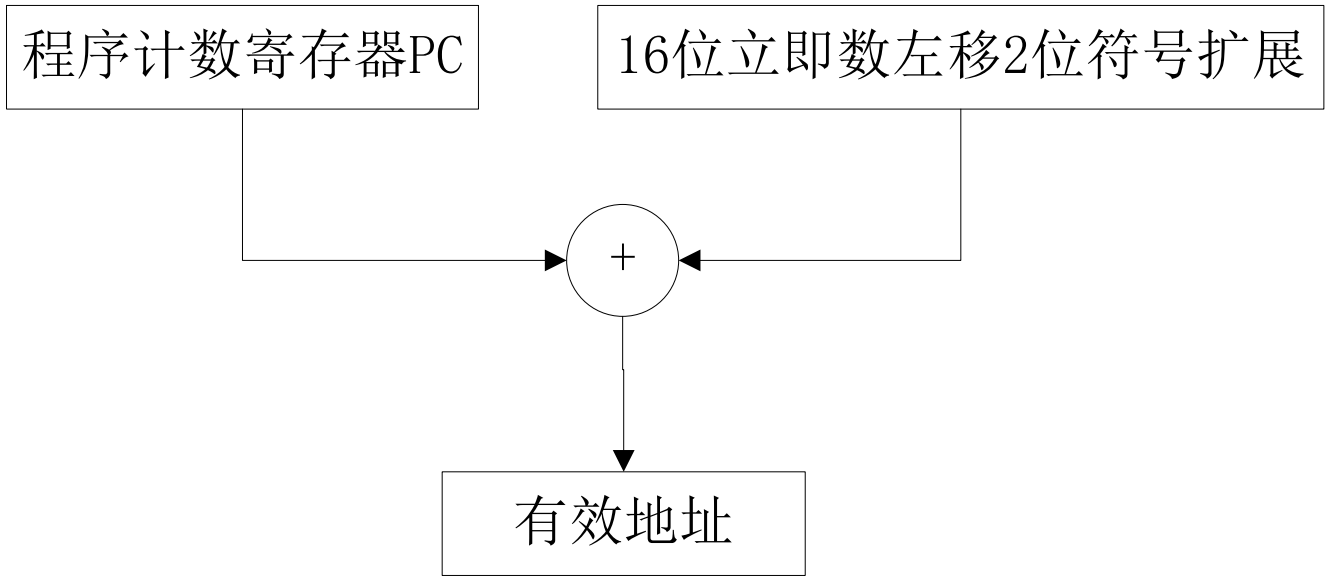


图7.2.3 PC相对寻址

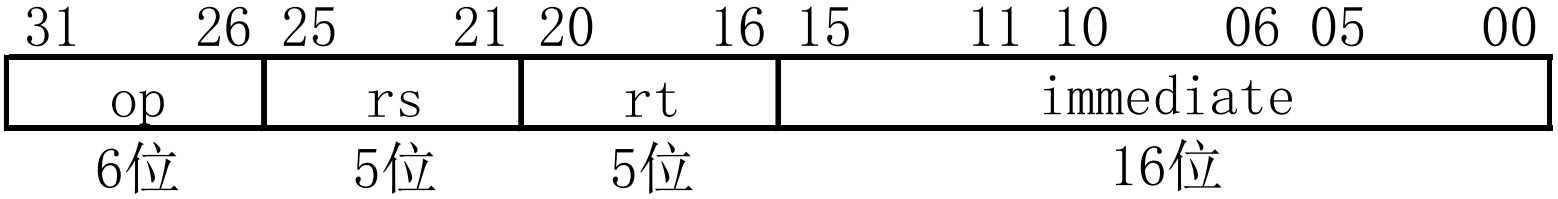
7.2.3 指令举例

(1) 计算类指令(Computational)

计算类指令用于执行算术操作，乘/除，逻辑操作和对寄存器进行移位操作。这些指令有两种类型:寄存器类型和立即数类型。寄存器类型的指令使用两个源寄存器的值作为源操作数，立即数类型使用一个寄存器和立即数作为源操作数。根据操作的不同，这些指令分为下面4种:

7.2.3 指令举例

1）ALU立即数指令



ADDI rt, rs, immediate ; rt←rs+immediate; 有符号加

- ADDI $0,$1,0001H ; $0←$1+0001H

001000 00001 00000 0000000000000001

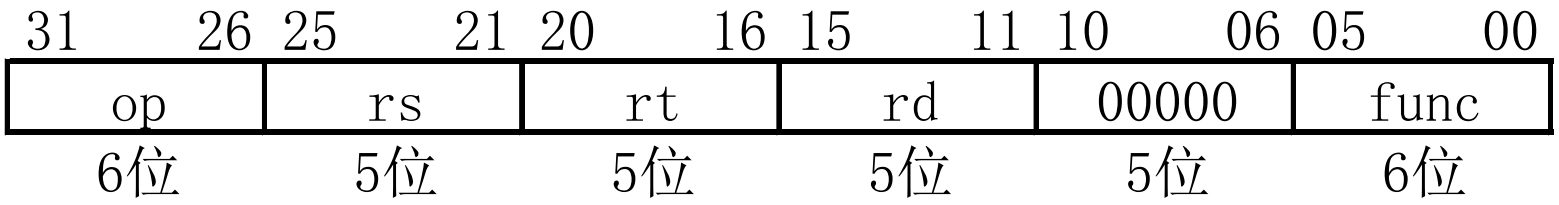
OP rs, rt,mmediate

- ADDI $2,$5,0010 ; $2←$5+0010H

001000 00101 00010 0000000000010000

7.2.3 指令举例

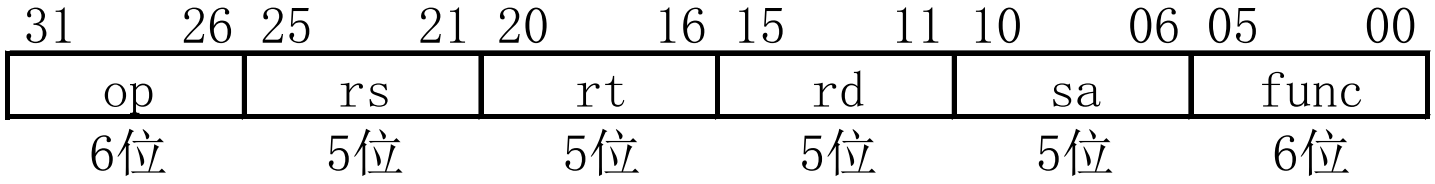
2）3操作数指令




```
ADD rd, rs, rt ; rd←rs+rt; 有符号加
ADD $0, $1, $2 ; $0←$1+$2
000000 00001 00010 00000 00000 100000
```

7.2.3 指令举例

3）移位指令



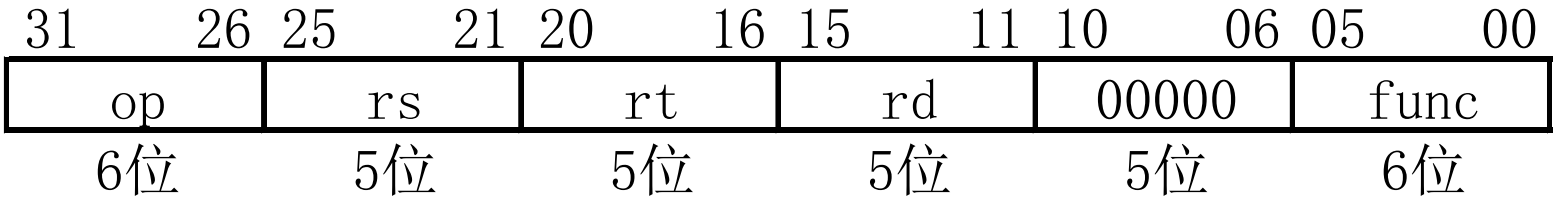
SLL rd, rt, sa ; rd←rt << sa; 逻辑左移sa位

SLL $4, $5, 6 ; $4←$5 << 6; 逻辑左移6位

000000 00000 00101 00100 00110 000000

7.2.3 指令举例

4）乘/除法指令



MUL rd, rs, rt ； rd←rs×rt（只存储结果低位）

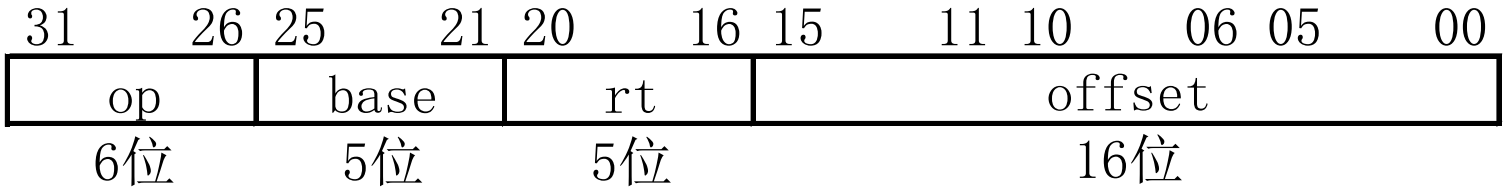
MUL $1, $2, $3 ； $1←$2×$3

011100 00010 00011 00001 00000 000010

(2) Load/Store指令

Load和Store指令都为立即数（I-type）类型，用来在存储器和通用寄存器之间的储存和装载数据。值得一提的是MIPS指令集只有该类指令访问内存，而其他指令都在寄存器之间进行，所以指令的执行速度较高。该类指令只有基址寄存器的值加上扩展的16位有符号立即数一种寻址模式，数据的存取方式可以是字节（byte）、字（word）和双字（Double word）。

7.2.3 指令举例



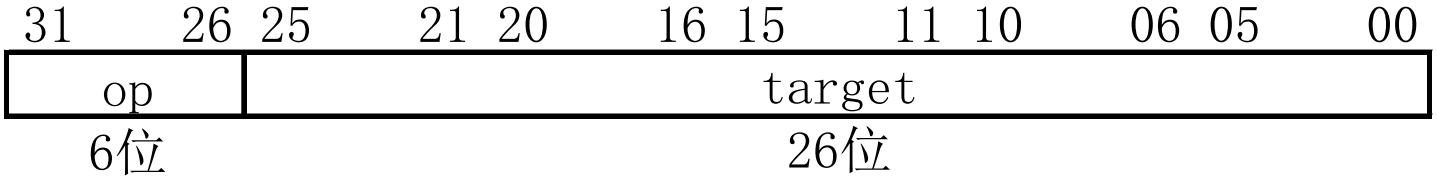

```
LW rt, offset(base) ; rt←memory[base+offset]; Load全字
Lw $5, 0x100($7)      ; $5←[$7+0x100]
100011 00111 00101 0000000100000000

SW rt, offset(base) ; memory[base+offset]←rt
```

(3) 跳转jump

所有的跳转目的的第一条指令从存储器中取出并执行。

Jump指令格式：



J target ； 在当前指令附近256MB的范围内跳转

7.2.3 指令举例

(4) 寄存器传送指令

寄存器传送指令用来在系统的通用寄存器（GPR）、乘除法专用寄存器（HI、LO）之间传送数据，这些指令分为有条件传送和无条件传送2种类型。

(5) 专用指令

专用指令用来产生软件中断，当执行这类指令的时候，CPU产生异常并转入中断处理程序。这些指令有系统调用（Syscall），暂停（Break）和Trap指令等，主要用于软件的异常处理。

7.2.3 指令举例

(6) 协处理器指令

协处理器指令对协处理器进行操作。协处理器的Load和Store指令是立即数类型，每个协处理器指令的格式依协处理器不同而不同。

(7) 系统控制协处理器（CP0）指令

系统控制协处理器（CP0）指令执行对CP0寄存器的操作来控制处理器的存储器并执行异常处理。

谢谢聆听

Thank You