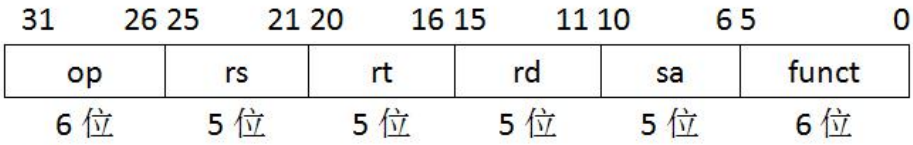
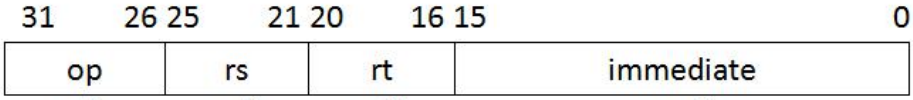
MIPS 架构 CPU 指令讲解（举例）

一、MIPS32 架构 CPU 指令的三种格式（指令代码格式）：指令格式中最高 6 位为操作码（op）。

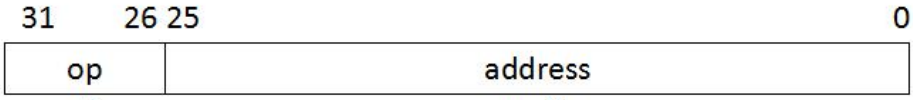
R 类型：



I 类型：



J 类型：



二、 指令举例 （以下操作码均为假设）

1、R 类型指令

（1）加法指令：add rd, rs, rt

举例：add $1, $2, $3；其中 rd = $1，rs = $2，rt = $3

操作码：000000，指令代码如：

000000	00010	00011	00001	000 0000 0000

功能：rd<－rs + rt。

如，$2 = 1， $3 = 2，执行指令后，则 $1 = 3。

（2）移位指令：sll rd, rt, sa

举例：sll $1, $2, 3; 其中 rd = $1, rt = $2, sa = 3

操作码：011000，指令代码如：

011000	00000	00010	00001	00011	00 0000

功能：rd<－rt<<(zero-extend)sa，左移 sa 位 ，(zero-extend)sa={{27{0}},sa}。

如，$2 = 1，sa 经扩展后为 0x00000003，执行指令后，$2 左移 3 位，则 $1 = 8。

（3）跳转指令：jr rs

举例：jr $31; 其中 rs = $31

操作码：111001，指令代码如：

111001	11111	00000	00000	00 0000 0000

功能：pc <－ rs，将转移到 rs 寄存器内容为地址的指令执行。

如，$31 = 0x00000018，执行指令后，则 pc = 0x00000018，指令将转移到地址为 0x00000018 的指令执行。

2、I 类型指令

（1）addi　rt, rs, immediate

举例：addi　$2, $0, 5; 其中 rs = $0, rt = $2, immediate = 5

操作码：000010，指令代码如：

000010	00000	00010	0000 0000 0000 0101

功能：rt<－rs + (sign-extend)**immediate，immediate 是符号数。**

如，$0 = 0，**immediate = 5（经扩展后为：0x00000005）**，执行指令后，则 $2 = 5（0x00000005）。

（2）beq　rs, rt, immediate

举例：beq　$1, $2, 3; 其中 rs = $1, rt = $2, immediate = 3

操作码：110100，指令代码如：

110100	00001	00010	0000 0000 0000 0011

说明：**immediate 是从 pc+4 开始和转移到的指令之间间隔 条数，符号数。**

功能：if(rs=rt) pc <－pc + 4 + (sign-extend)**immediate** <<2 else pc <－pc + 4。

如，

a. 相等：$1 = 1，$2 = 1，pc(原) = 0x00000008，**immediate 经**扩展再左移 2 位之后为 0x0000000c，

则 新 pc = 0x00000008 + 0x000000004+0x0000000c = 0x00000018，将跳转到该地址并该条指令；

b. 不等：$1 = 1，$2 = 2，pc(原) = 0x00000008，

则 pc = 0x00000008 + 0x000000004 = 0x0000000c，将顺序执行该地址指令。

bne 指令与 beq 指令情况相反，bgtz 类似。

（3）sw　rt, immediate(rs)

举例：sw　$1, 8($2); 其中 rs = $2,　rt = $1, immediate = 8

操作码：110000，指令代码如：

110000	00010	00001	0000 0000 0000 1000

说明：**sw 为字操作指令--写存储器，字长为 32 位。存储器中，数据存储的基本单位，一个字节，8 位二进制数长度。immediate 是立即数，符号数。**

功能：memory[rs+ (sign-extend)**immediate**]<－rt。即将 rt 寄存器的内容保存到 rs

寄存器内容和立即数符号扩展后的数相加作为地址的内存单元中。

如，$1 = 6，$2 = 0，immediate = 8（经扩展后为 0x00000008），执行指令后，

则 [0x00000000 + 0x00000008] = 0x00000006，

即 [0x00000008] = 0x00000006。“[addr]”表示地址为 addr（0x00000008）的内存单元内容。

内存地址	内存单元数据
0X00000000:	0X00
0X00000001:	0X00
0X00000002:	0X00
0X00000003:	0X00
…	…
0X00000008:	0X00
0X00000009:	0X00
0X0000000a:	0X00
0X0000000b:	0X06

数据存储使用大端存储模式，即高位数存储在低位地址单元中。（低位数存储在低位地址单元中，即小端存储模式）。至于，数据存储器中，数据存储使用哪种模式，只要在设计 CPU 时，对存储器写操作，规定好一种模式，即可。

（4）lw　rt, immediate(rs)

举例：lw　$3, -2($2)；其中 rs = $2, rt = $3, immediate = -2

操作码：110001，指令代码如：

110001	00010	00011	1111 1111 1111 1110(-2 补码)

说明：**lw 为字操作指令--读存储器，字长为 32 位。immediate 是立即数，符号数。**

功能：rt <－ memory[rs + (sign-extend)**immediate**]。即读取 rs 寄存器内容和立即数符号扩展后的数相加作为地址的内存单元中的数，然后保存到 rt 寄存器中。

如，$2 = 10，[0x00000008] = 8，immediate = -2（经扩展后为 0xfffffffe），

则 地址 addr = 0x0000000a + 0xfffffffe =0x00000008，执行指令后，

[addr] = [0x00000008] = 0x00000008，即 $3 = 0x00000008。

“[addr]”表示地址为 addr（0x00000008）的内存单元内容。

内存地址	内存单元数据
0X00000000:	0X00
0X00000001:	0X00
0X00000002:	0X00
0X00000003:	0X00
…	…
0X00000008:	0X00
0X00000009:	0X00
0X0000000a:	0X00
0X0000000b:	0X08

数据存储使用大端存储模式，即高位数存储在低位地址单元中。

3、J 类型指令

（1）j　addr

举例：j　0x00000018; 其中 addr = 0x00000018

操作码：111000，指令代码如：

111000	0000 0000 0000 0000 0000 0001 10

说明：由于 MIPS32 的指令代码长度占 4 个字节，所以指令地址二进制数最低 2 位均为 0，将指令地址放进指令代码中时，可省掉！这样，除了最高 6 位操作码外，还有 26 位可用于存放地址，事实上，可存放 28 位地址，剩下最高 4 位由 pc+4 最高 4 位拼接上。

功能：pc <－{(pc+4)[31..28],addr[27..2],0,0}，无条件转移。

如，addr = 0x00000018，则 pc = 0x00000018，这里 pc+4 的最高 4 位为 0。

（2）jal　addr

举例：jal　0x0000002c; 其中 addr = 0x0000002c

操作码：111010，指令代码如：

111010	0000 0000 0000 0000 0000 0010 11

功能：调用子程序，pc <－ {(pc+4)[31..28],addr[27..2],0,0}；$31<－pc+4，返回地址设置；子程序返回，需用指令 jr　$31。

执行指令后，则 pc = 0x0000002c，转移到地址为 0x0000002c 的指令执行。

pc 地址的形成方法同指令 j　addr，不同点是保存返回地址 $31<－pc+4。假设本条指令的地址为：0x00000020，则

$31 = 0x00000020 + 0x00000004=0x00000024。