|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 指令 | 指令意义 | Op[31:26] | Op2 [25:20] | [19:15] | [14:10] | [9:5] | [4:0] |
| add | 寄存器加法 | 000000 | 000001 | 00000 | rd | rs | rt |
| and | 寄存器与 | 000001 | 000001 | 00000 | rd | rs | rt |
| or | 寄存器或 | 000001 | 000010 | 00000 | rd | rs | rt |
| xor | 寄存器异或 | 000001 | 000100 | 00000 | rd | rs | rt |
| sra | 算术右移 | 000010 | 000001 | shift | rd | 00000 | rt |
| srl | 逻辑右移 | 000010 | 000010 | shift | rd | 00000 | rt |
| sll | 逻辑左移 | 000010 | 000011 | shift | rd | 00000 | rt |
| addi | 立即数加法 | 000101 | 16位immediate | | | rs | rt |
| andi | 立即数与 | 001001 | 16位immediate | | | rs | rt |
| ori | 立即数或 | 001010 | 16位immediate | | | rs | rt |
| xori | 立即数异或 | 001100 | 16位immediate | | | rs | rt |
| load | 取整数数据字 | 001101 | 16位offset | | | rs | rt |
| store | 存整数数据字 | 001110 | 16位offset | | | rs | rt |
| beq | 相等则跳转 | 001111 | 16位offset | | | rs | rt |
| bne | 不相等则跳转 | 010000 | 16位offset | | | rs | rt |
| jump | 无条件跳转 | 010010 | 26位address | | | | |

Op和Op2为操作码；

shift保存要移位的位数；

rd、rs、rt分别为寄存器的寄存器号；

immediate保存立即数的低16位；

offset为偏移量；

address为转移地址的一部分。

1、对于add/sub/mul/and/or/xor rd,rs,rt指令 //rd🡨rs op rt

其中rs和rt是两个源操作数的寄存器号，rd是目的寄存器号。

2、对于sll/srl/sra rd,rt,shift 指令 //rd🡨rt 移动 shift位

3、对于addi/muli rt,rs,imm 指令 //rt🡨rs+imm(符号拓展)

rt是目的寄存器号，立即数要做符号拓展到32位。

4、对于andi/ori/xori rt,rs,imm 指令 //rt🡨rs op imm(零拓展)

因为是逻辑指令，所以是零拓展。

5、对于load rt,offset(rs) 指令 //rt🡨 memory[rs+offset]

load是一条取存储器字的指令。寄存器rs的内容与符号拓展的offset想加，得到存储器地址。从存储器取来的数据存入rt寄存器。

6、对于store rt,offset(rs) 指令 // memory[rs+offset] 🡨 rt

store是一条存字指令。存储器地址的计算方法与load相同。

7、对于beq rs,rt,label指令 //if(rs==rt) PC🡨label

beq是一条条件转移指令。当寄存器rs内容与rt相等时，转移到label。如果程序计数器PC是beq的指令地址，则label=PC+4+offset<<2。offset左移两位导致PC的最低两位永远是0，这是因为PC是字节地址，而一条指令要占4个字节。offset要进行符号拓展，因为beq能实现向前和向后两种转移。

8、bne指令去beq类似，但是是在寄存器rs内容与rt不相等时，转移到label。

9、对于jump target指令 //PC🡨target

jump是一条跳转指令。target是转移的目标地址，32位，由3部分组成：最高4位来自于PC+4的高4位，中间26位是指令中的address，最低两位为0。