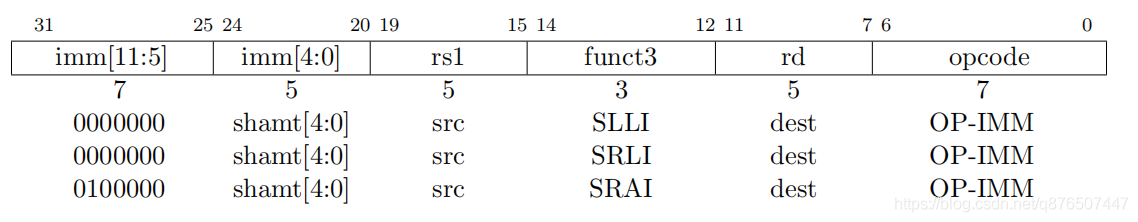


ADDI rd, rs1, imm：就是REG[rd] = REG[rs1] + $signed(imm);

SLTI (set less than immediate) rd, rs1, imm：就是当REG[rs1]比$signed(imm)小的时候，REG[rd]为1，否则为0，有符号比较。

SLTIU跟SLTI差不多，不过是无符号数比较。

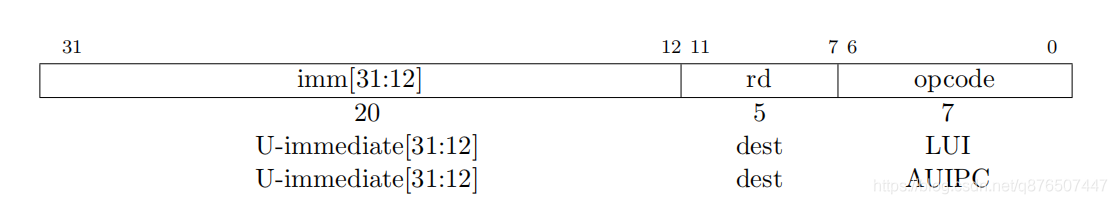


移位操作也是编译成I type的。移位的位数是imm[4:0]，所以一次指令最多可以移动32位，跟寄存器的位宽一样。

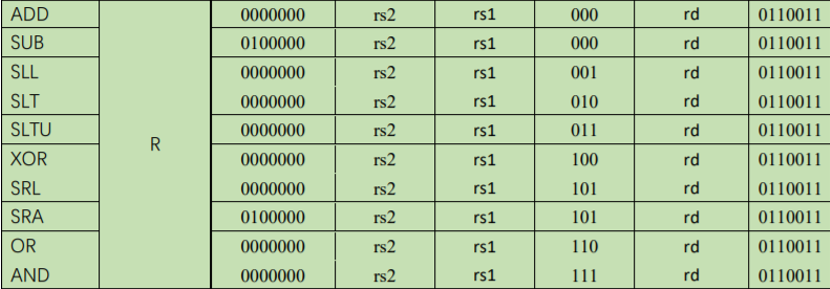
SLLI：是逻辑左移，低位补零

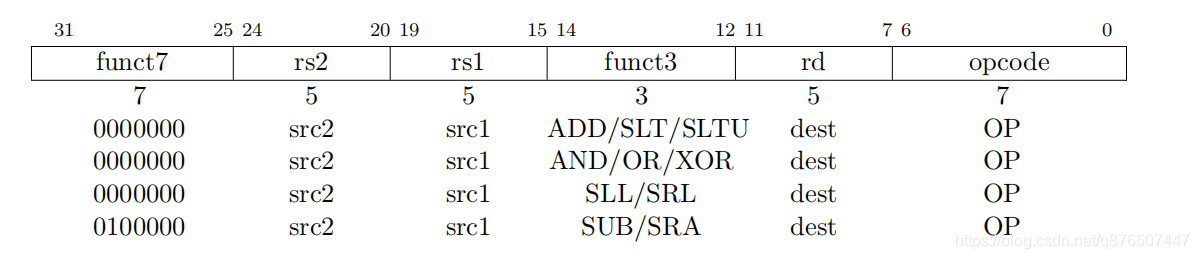
SRLI：是逻辑右移, 高位补零

SRAI：是算术右移，高位补原来高位。

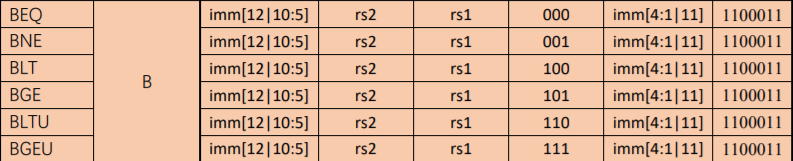


LUI (load upper immediate)： LUI rd imm。就是REG[rd] [31:12] = imm。REG[rd] [11:0] = 0。就是把imm赋给rd的高20位，然后低12位补零。一般后面加一个addi的指令，把低12位的值给上。这样子，通过两条指令，就把一个32位的imm赋给了寄存器啦。  
AUIPC (add upper immediate to pc)：就是在LUI的基础上，再加上这个命令的pc地址。同样后面再加一个12位的加法，就可以寻址到任何位置的指令的地址啦。





ADD就是加，SUB就是减，都不考虑溢出的哦。  
SLT和SLTU分别是有符号比较和无符号数比较，rs1<rs2的话，rd为1，否则为0。  
SLL, SRL, and SRA就是逻辑左，逻辑右，算术右的运算，是rs1进行移位，移动的位数是rs2的低5位。



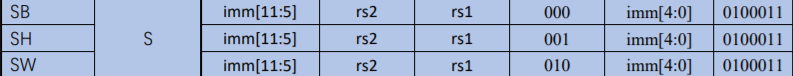
所有分支指令使用B类指令格式，12位立即数+pc作为目标

跳转范围为pc(+/-)4KB

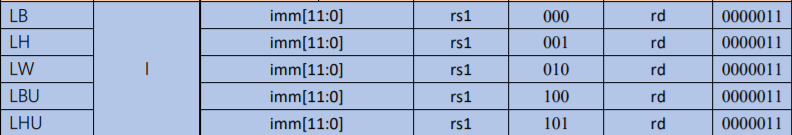
BEQ/BNE：rs1(==/!=)rs2, 分别在相等或者不等时，发生跳转

BLT：rs1 < rs2, 跳转

BGE：rs1 >= rs2, 跳转



Store Byte是将寄存器RS2的0位到7位内容保存在RS1为基地址，立即数进行有符号扩展后的S\_IMM为偏移量（可正可负）的地址。



LW 将32位值复制到rd中，LH从储存器中读取16位，然后将其符号扩展到32位，保存到rd中。LHU指令读取存储器16位，然后0扩展到32位，再保存到rd中。LB/LBU则是读取8位。SW/SH/SB分别将寄存器rs2中的低32/16/8/位到储存器中。



JAL：就是pc指针跳转到当前位置加上{$signed(imm[20:1])，0}，这是J型，你软件给说JAL rd 16，其实实现的是pc = pc + 32。然后将pc + 4这个值赋给rd存放好到时候要返回来的地址。imm是2个字节对齐的，20位就是2MB的寻址空间了，imm有符号数的哦。

JALR功能跟JAL差不多，比如JALR rd rs1 imm，那就是pc = REG[rs1] +imm。然后把最低有效的位置0

