第一部分：RV32I指令集的理解

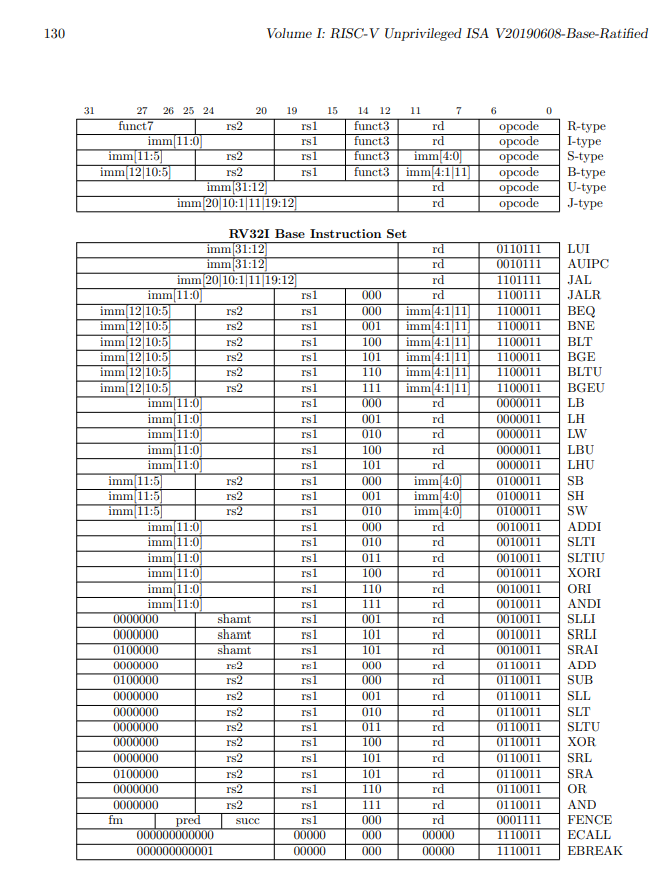
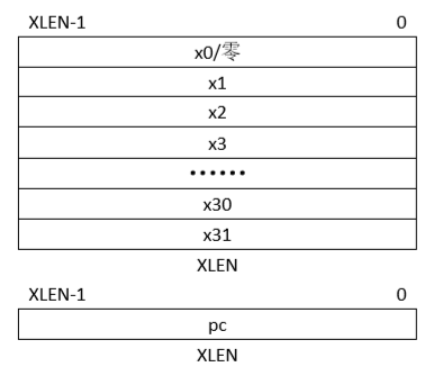


表 1 RV32I指令

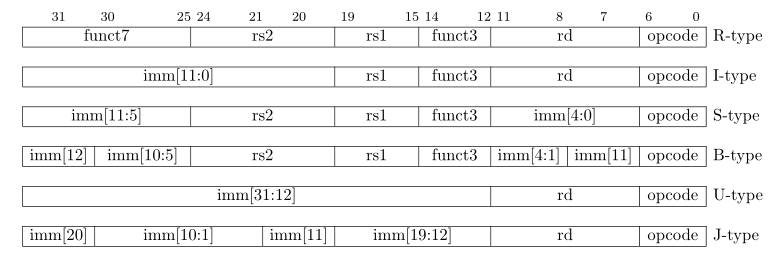
1. **寄存器定义**

RV32I共设置32个寄存器x0-x31，其中x0硬连线为0，x1-x31为通用寄存器。另外有一个pc的寄存器，用来保存当前指令的地址。位宽为32位。x1作为返回地址寄存器，x5作为备用链路寄存器。



1. **指令格式**

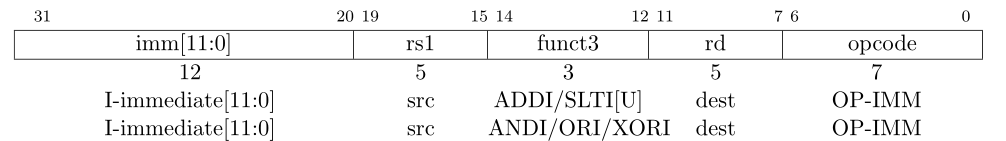
指令格式核心主要包括四种（R/I/S/U），基于立即数的扩展，还包括B和J两种变种。



1. **指令**

指令主要包括如下几类：

1. **整数计算指令**

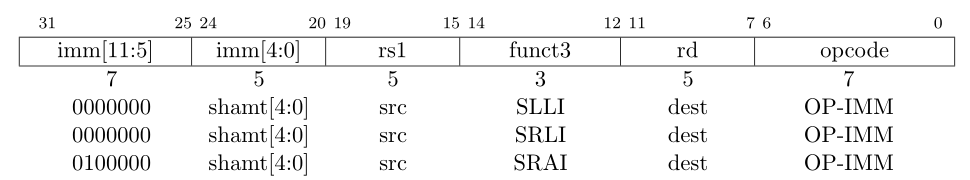


ADD rs1=imm(12bit)+rs1

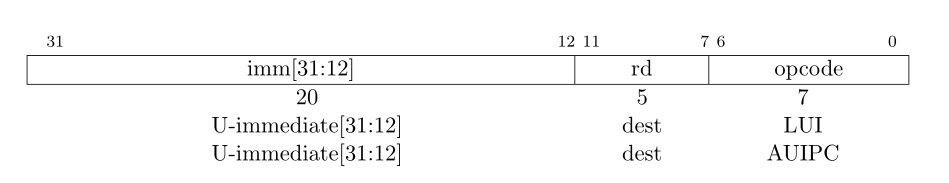
SLTI if(rs1<imm) rd=1; else rd=0;

SLTIU功能同SLTI，不过rs1和imm被当作无符号数

ANDI、ORI、XORI均为逻辑操作，在寄存器rs1和符号扩展位上的12bit按位AND、OR、XOR



rs1被移位imm[4:0]次，SLLI/SRLI分别是逻辑左右移（空出来的位填0），SRAI是算术右移（符号位的单独处理，正数填充0，负数填充1）。

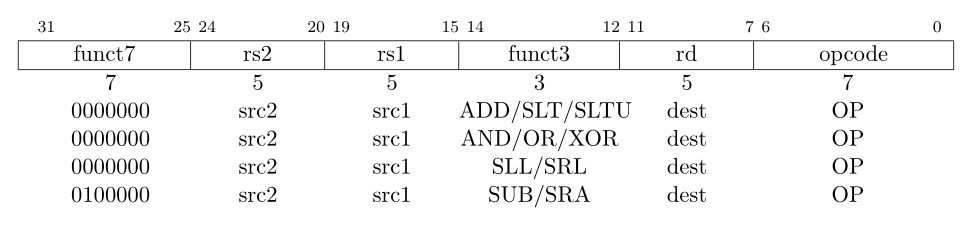


LUI用于构建32位常数，LUI将imm放到rd的高20位，低12位填0.

AUIPC(add upper immediate to pc) 用imm构建一个偏移量的高20位，低12位填0，并将此偏移加到pc上，将结果写入rd。

AUIPC+JALR可以将控制转移到任意32位相对地址，而加上一条12位立即数偏移的load/store指令就可以访问任意32位pc相对数据地址。

**（2）寄存器-寄存器指令**



ADD rd=rs1+rs2

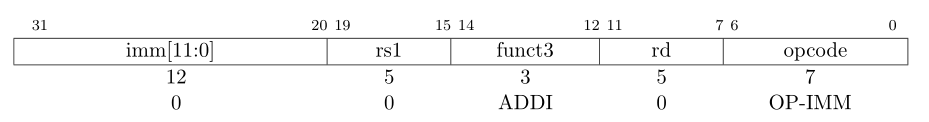
SUB rd=rs1-rs2

SLT 有符号位比较，如果rs1<rs2，rd=1，否则rd=0.

SLTU 无符号位比较，如果rs1<rs2，rd=1，否则rd=0.

AND、OR、XOR 按位进行逻辑与、或、异或

SLL、SRL、SRA 对寄存器rs1中的值执行逻辑左移、逻辑右移和算数右移，位移量保持在rs2的低五位

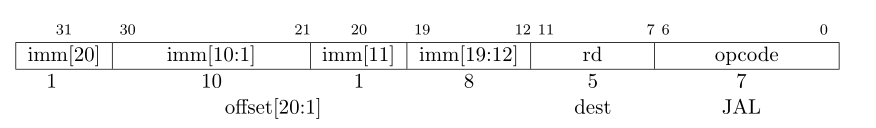


此指令不改变任何用户的可见状态，用于pc的向前推进。 被编码为 ADDI x0,x0,0

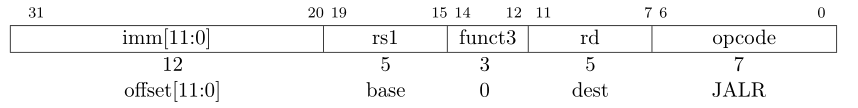
#### **（3）控制转移指令**

RV32I提供了两类控制转移指令，无条件跳转和条件分支。并且没有在体系结构中可见的分支延迟槽。

**无条件跳转**



JAL在立即数处编码了一个有符号偏移量，这个偏移量加到pc上后形成跳转目标地址，并将跳转指令后面指令的地址(pc+4)加载到rd，跳转范围为±1MB。标准软件调用约定使用寄存器x1来作为返回地址寄存器。



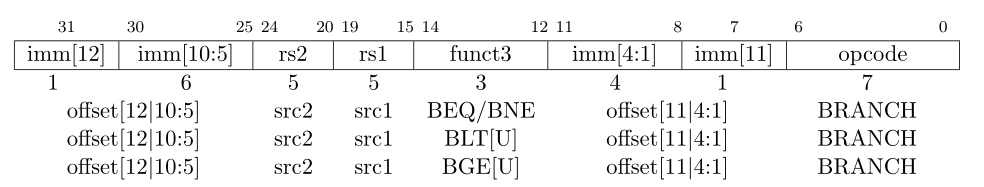
JALR(jump and link register) 通过有符号立即数加上rs1,然后将结果的最低位设置为0，作为目标地址，将跳转指令后面的地址存到rd中。

如果目标地址没有对齐到32位，JAL和JALR指令均会产生一个非对齐指令取址异常。

所有无条件跳转指令都是用pc相对寻址，有助于支持位置无关代码。JALR可以用来跳转到任何32位绝对地址空间。

首先LUI将目标地址的高20位加载到rs1中，然后JALR可以加上低12位。事实上，绝大多数JALR指令的使用要么是一个立即数0，要么就是配合LUI或者AUIPC来跳转到32位地址空间

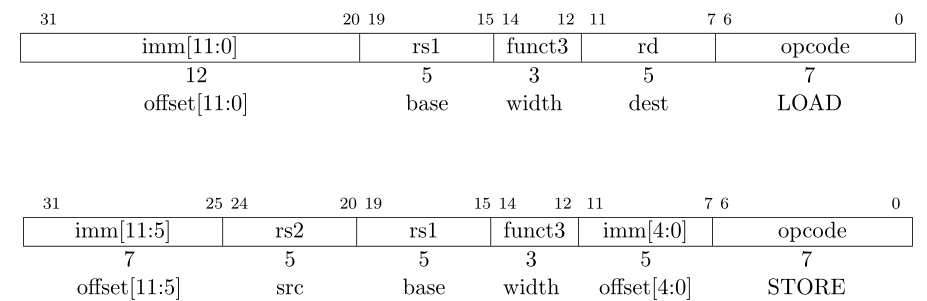
**条件分支**



BEQ/BNQ if (rs1==/!=rs2) Jmp  
  BLT/BLTU if(rs1<rs2) Jmp  
  BGT/BLEU if (ifrs1>rs2) Jmp

**（4）Load/Store指令**

RV32I中只有load和store指令可以访问储存器，其他指令只能操作寄存器。

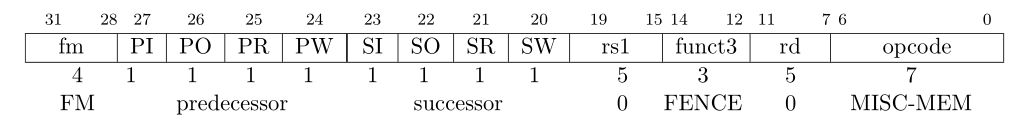


Load 为I类格式，而Store为S类指令格式。load类储存器地址是通过rs1+imm（偏移来实现），都是将储存器值复制到寄存器中。

Store将rs2中的值复制储存器中。

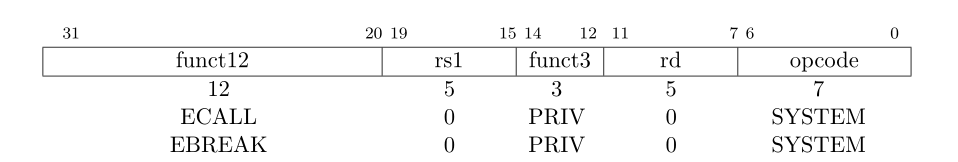
LW 将32位值复制到rd中，LH从储存器中读取16位，然后将其符号扩展到32位，保存到dr中。LHU指令读取存储器16位，然后0扩展到32位，再保存到rd中。LB/LBU则是读取8位。SW/SH/SB分别将寄存器rs2中的低32/16/8/位到储存器中。

1. **存储器模型（内存排序指令FENCE）**



主要用于其他RISCV harts和外部设备或者协处理查看的设备输入/输出和内存访问。与内存一致性模型相关。

1. **环境调用和断点**



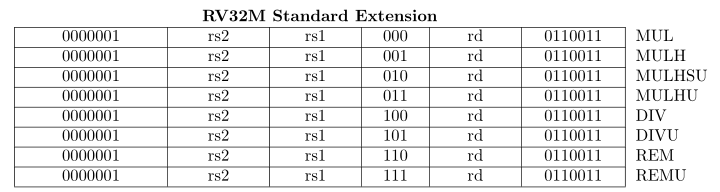
ECALL和EBREAK与特权级相关，当前软核暂且使用不到。

1. **HINT指令**

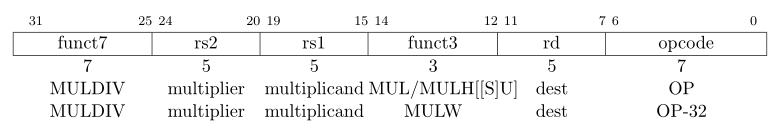
RV32I为HINT指令预留了很大的编码空间，通常用于向微体系结构传递性能提示。不会改变任何体系结构上可见的状态。

第二部分：扩展指令

经小组讨论，为实现功能完整性和充分发挥RISC-V性能优势，决定扩展指令使用M和C指令的内容。

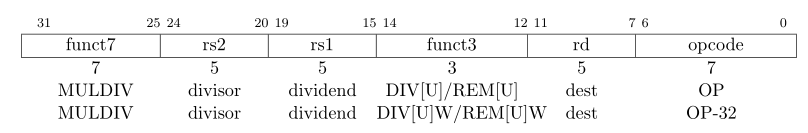


1. **M扩展**



（1）MUL rd=rs1\*rs2，并将较低位的XLEN位放入目标寄存器。

（2）MULH、MULHU、MULHSU执行相同的乘法，但分别返回完整的2\*XLEN位成绩的高XLEN位，用于有符号\*有符号、无符号\*无符号和有符号\*无符号乘法。



（3）DIV/DIVU 有符号/无符号rs1/rs2

（4）REM/REMU 有符号/无符号取余

1. **C扩展**

C表示标准压缩指令集扩展。它通过常见操作添加短的16位指令编码来减少静态和动态代码大小。通常来说，一个程序中50%-60%的RISC-V指令可以被RVC指令替换，导致代码大小减少25%-30%。

