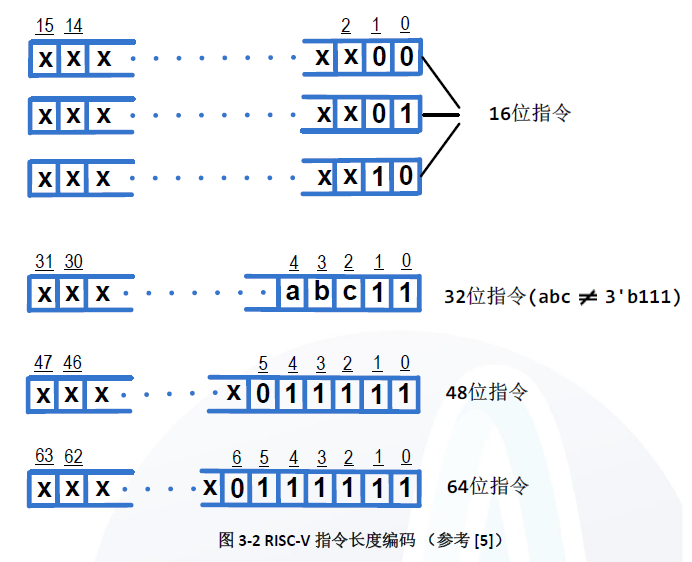
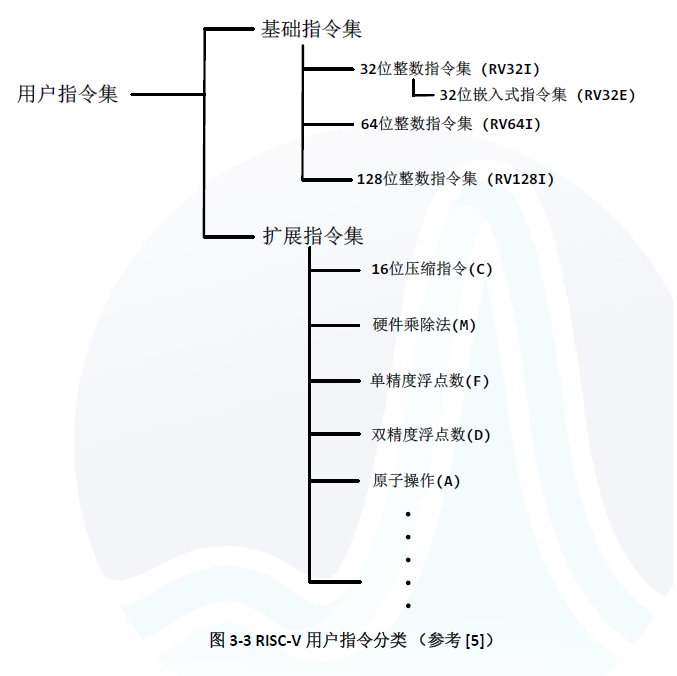


RISCV指令长度编码



RISC-V 指令集的官方标准主要分成两部分：用户指令集和特权架构。

RISC-V的用户指令集分为基础整数指令集 (Base Integer Instruction Set)和扩展指令集(Extension)。根据处理器字长的不同，基础整数指令集又有32位，64位和128位的分类。而扩展指令集则有“16位压缩指令”、“硬件乘除法”、“单精度浮点数”等多种不同的扩展。



RISC-V有3个独立的地址空间：

内存地址空间：内存地址空间可以用来分配给代码，数据或者作为寄存器的内存映射 (Memory Mapped Registers)。在物理实现时，代码和数据可以共用存储(Von Neumann 冯诺依曼架构)，也可以分别存储(Hardvard 哈佛架构)。和其它的处理器一样，RISC-V的处理器也是通过程序计数器 (Program Counter, PC)来指示当前正在执行的指令地址的。

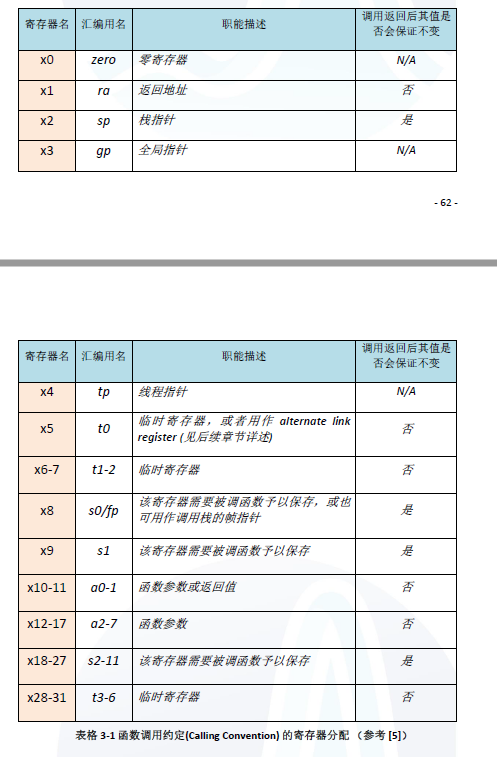
而在寄存器的内存映射部分，大部分的外部设备寄存器都会被映射到这个空间，其中也包括机器模式的定时器(mtime)和定时器触发值(mtimecmp)。

通用寄存器：RV32I指令集包含32个通用寄存器，而RV32E只有16个这样的寄存器。

控制与状态寄存器 (CSR)



RV32I通用寄存器与函数调用约定：



***每条指令实现单个功能***

在 RISC-V 标准刚刚推出时，32 位的基础指令集只有 RV32I，即 32 位整数指令集。后来

考虑到嵌入式系统资源稀缺的情况，又制定了 RV32E 基础指令集。这里的字母 E 即代表嵌

入式(Embedded)。RV32I 和 RV32E 的主要区别是在通用寄存器的数量上。在 RV32I 中，总

共有 32 个32 位宽的通用寄存器，而RV32E 只支持16 个32 位宽的通用寄存器。另外 RV32E

仅支持 M、A、C 三种指令扩展。

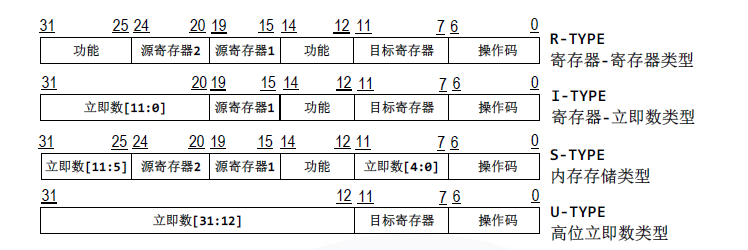
在 RISC-V 32 位基础整数指令集中(RV32I)，操作码总是占用低 7 位，而源寄存器 1 和 2

(rs1， rs2) 则分别占据 15 至 19 位 与 20 至 24 位。目标寄存器(rd)则占用 7 至 11 位（位索

引以 0 为参考起点）。这种规整的指令格式进一步简化了指令解码器和指令执行器的设计

。

RV32I 指令格式

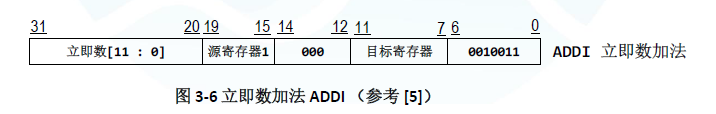


上面这些格式，除了R-TYPE 以外，其它的格式都需要把最高位（第31 比特）做符号扩

展(sign extension)来产生一个32 位的立即数，作为指令的操作数。

1. ***立即数指令***

立即数加法

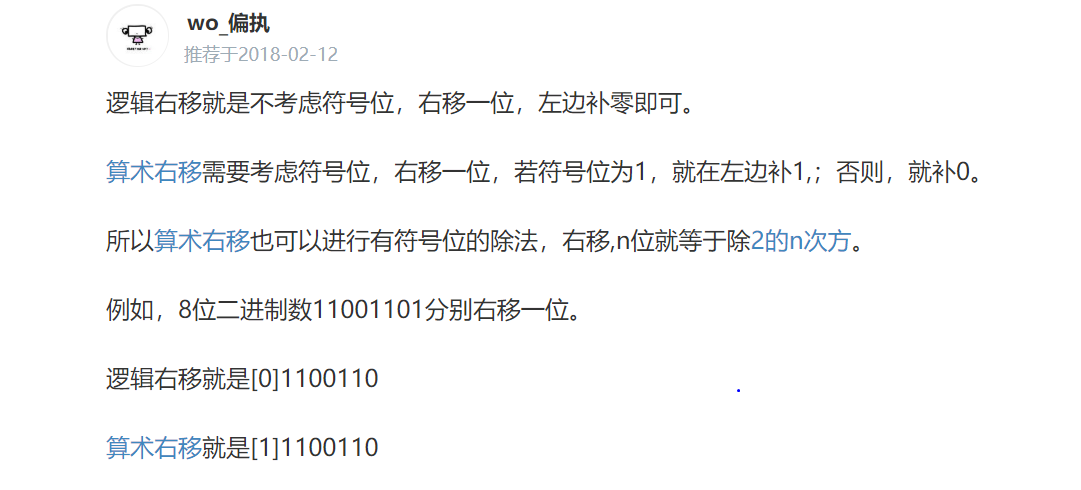


在 RV32I 当中并没有专门的状态寄存器和标记位来记录加法溢出，对加法溢出的判断是通过在加法指令之后安排比较和跳转指令来实现的。对符号数加法来说，只有正数加正数，或者负数加负数的情况才有可能发生溢出。所以溢出可以通过符号位（与零比较）来判断。而对无符号数来说，其和(sum)应该不小于被加数(addendum)，所以溢出也可据此判断。

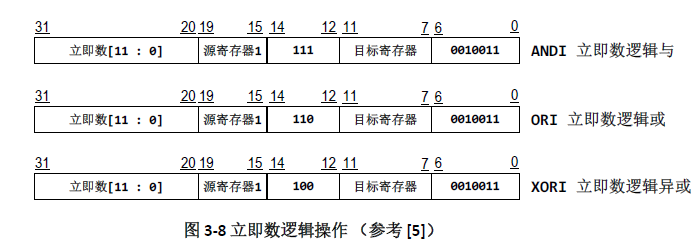
立即数比较

12 位立即数都应该通过符号位扩展变为 32 位立即数，然后根据指令 12

到 14 位中的功能定义，来决定比较方式（符号数/无符号数）。



立即数逻辑操作



RV32I 中并没有定义逻辑反操作(NOT)。实际上，逻辑反操作可以通过XORI 来实现。（只需将XORI 指令中的立即数置为全1 即可。）

立即数移位操作

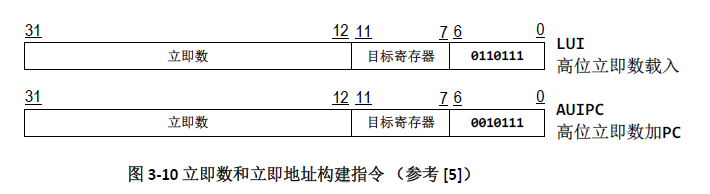


对逻辑左移操作，需要在低位(LSB) 补零。对逻辑右移操作，需要在高位(MSB)补零。而对算术右移操作，则需要在高位做符号位扩展(sign-extension)。

SRLI 与SRAI 的唯一的编码区别是在第30 位上。在处理器硬件实现上述移位

指令时，硬件只需判断此位便可加以区分。

32 位立即数构建与地址生成

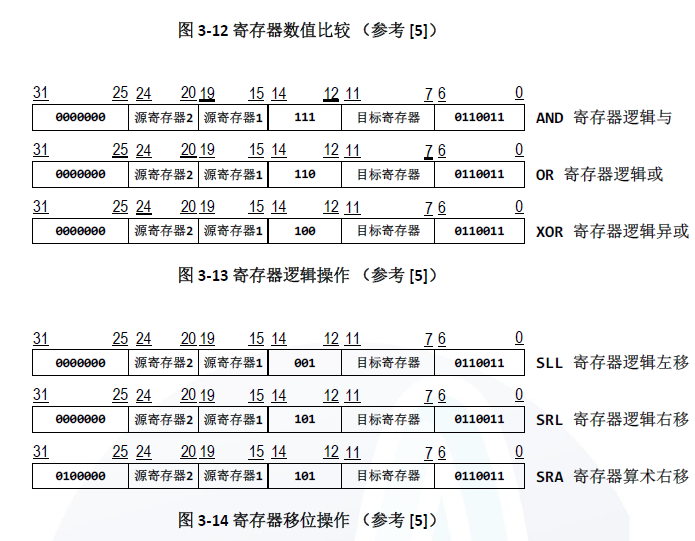


U-TYPE指令中的立即数有20位，而I-TYPE指令中的立即数有12位。所以32位立即数可以通过一条U-TYPE指令和一条I-TYPE指令来联合构建。图 3-10中的LUI (Load Upper Immediate) 指令会将其所携带的20位立即数载入目标寄存器的高位，而将目标寄存器的低12位置零。而如果在LUI指令之后紧随一条ADDI指令的话，则可以继续构建目标寄存器的低12位，从而拼接处完整的32位立即数。该32位立即数也可以作为32位的地址使用。

AUPIC 也会将其携带的20位立即数作为高位，而将低12位置零，以生成一个完整的32位数。然后该32位数会与当前指令计数器（PC， 32位寄存器）的值相加，并将结果存入目标寄存器。（和几乎所有其它的指令集一样，RV32I也用PC来存放当前活跃指令的内存地址。）

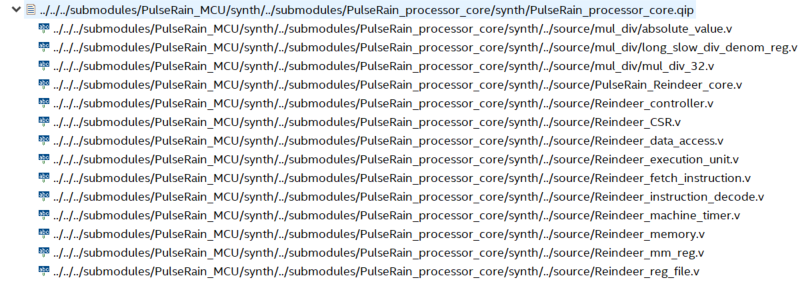
1. ***寄存器-寄存器指令***





这些指令的功能和前面的立即数指令相似。唯一的区别是立即数指令中的立即数被替换为源寄存器2。

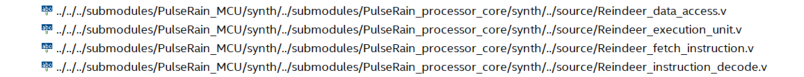
关于RISCV 在内核的代码：



PulseRain\_Reindeer\_core为主文件，调用其他文件。



以上文件为 RISC-V拓展指令集中的乘除法拓展，第一个为取绝对值，第二个为除法，第三个为乘除法。



以上文件为 RISC-V拓展指令集中的流水线的设计，PulseRain Reindeer处理器采用了如下的设计策略：

采用了2 x 2的流水线设计，内存布局采用冯·诺依曼架构，为了追求较高的时钟主频，PulseRain Reindeer处理器中包含有4级流水线：

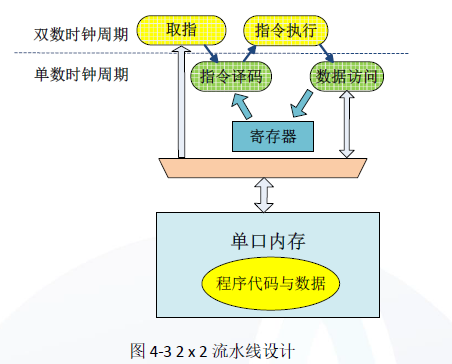
◼ Instruction Fetch （取指）

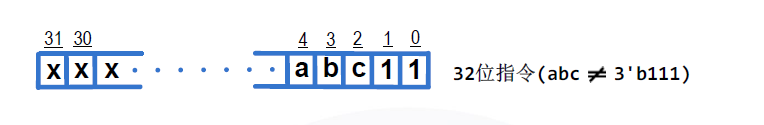
◼ Instruction Decode （指令译码）

◼ Execution （指令执行）

◼ Data Access （数据访问，包括寄存器的更新与内存的读写）

在双数时钟周期，只有“取指”和“指令执行”这两个阶段是活跃的。而在单数时钟周期，只有“指令译码”和“数据访问”这两个阶段是活跃的。



在指令译码器 (Instruction Decoder)中，指令比特[6 : 0]是操作码。在RV32IM下，比特[1 : 0] 总是0x3。所以在PulseRain Reindeer中只需要对比特[6 : 2]译码便可确定指令操作类型。

指令执行器(Instruction Execution)中，包含的指令有ALU、乘除法(M Extension)、无条件跳转指令 (JAL / JALR)、立即地址构建指令LUI / AUIPC (Load Upper Immediate / Add Upper Immediate to PC) 、CSR操作指令、BRANCH 指令，ECALL / EBREAK，LOAD / STORE。

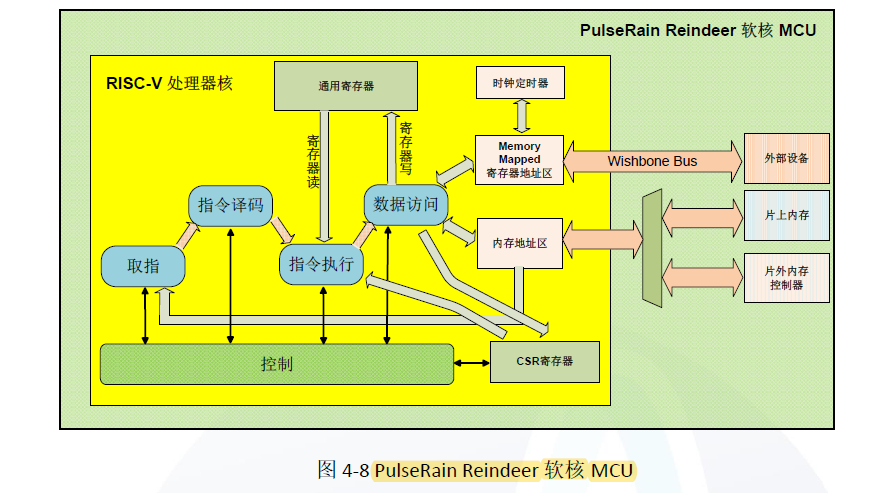


这个文件是RISC-V标准定义了两个与定时器有关的64位寄存器：mtime与mtimecmp，并且这两个寄存器在标准中被规定为Memory Mapped Register，其具体的地址由处理器设计者来自行决定。





以上文件分别对应寄存器地址区，通用寄存器和内存地址区。



流水线控制中，，FETCH\_EXE状态和DECODE\_DATA状态分别对应于图 4-3提到的双数时钟周期（“取指”与“指令执行”）和单数时钟周期（“指令译码”与“数据访问”）。其中跳转指令则会将流水线控制转入INIT状态，以重新加载流水线。而异常/中断的处理则需要一个额外的EXCEPTION状态，以根据异常/中断的具体种类，设置Exception Code （表格 3-18 与 表格 3-19），并确定异常/中断处理的返回地址（即mepc寄存器）

