CPU CS161E 手册

混乱沉睡，2025 年 5 月 25 日 编写
2025 年 7 月 8 日 第一次修订

目录

第一部分 简介

CS161E 是一块使用 RVE 指令集的 16 位哈佛结构 CPU。具有较低的性能占用，主要用于嵌入式领域，控制其它电路部件。

RVE 指令集是一种由 RISC-V 指令集（rv32i）简化得到的指令集，删除 auipc、除 lh 和 sh 以外的访存指令、除 beq 和 bne 以外的分支指令，指令长度 16 位，通用寄存器减少到 8 个，每个指令地址和数据地址对应两个字节（16位）。

CS161E 使用内存和接口统一编址，内存和其它输入输出部件都会被映射到不同的内存地址，每个部件会分配一个部件号（高 8 位内存地址），部件内使用低 8 位内存地址寻址。指令内存和数据内存使用不同的地址空间，无法互相访问。

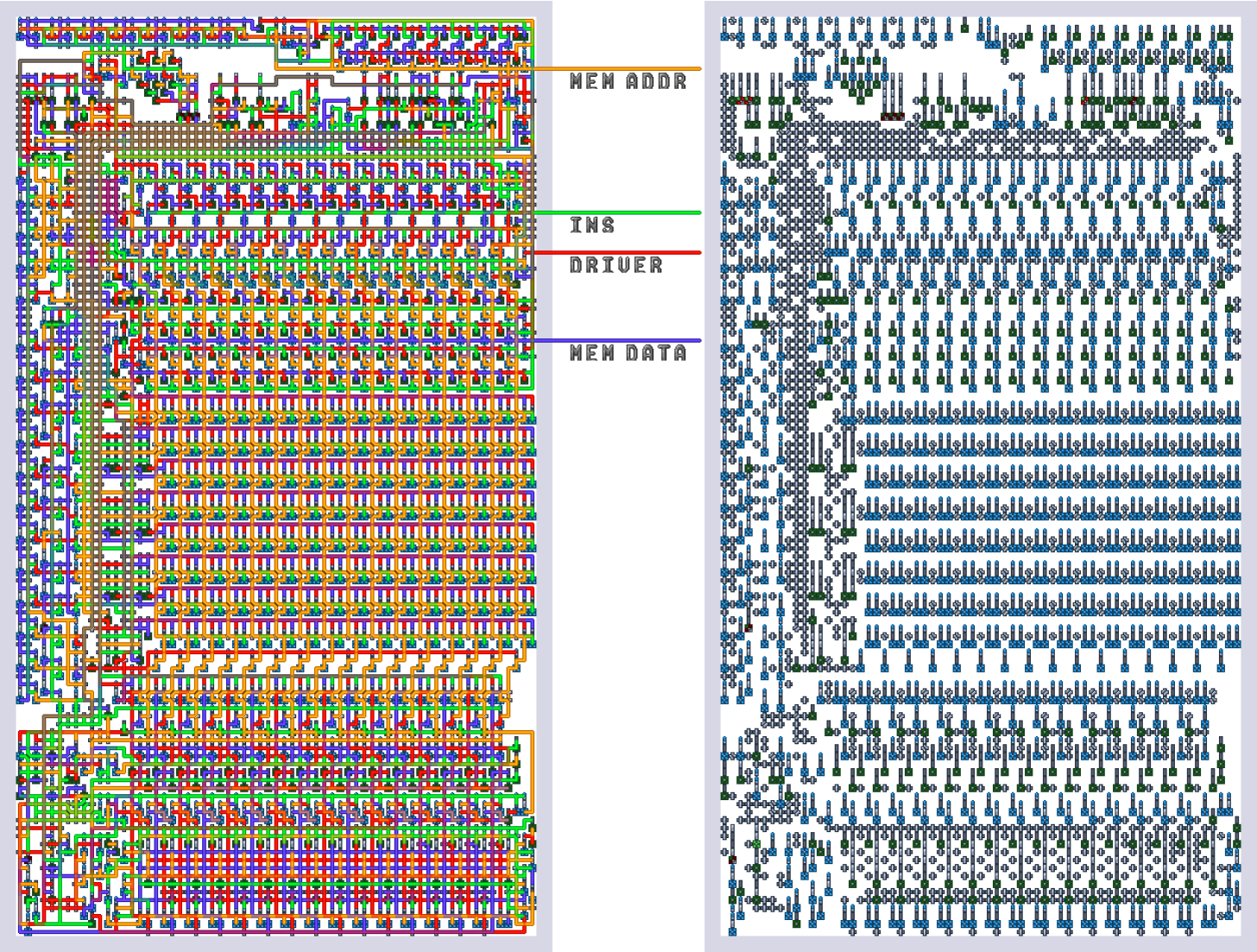
相较于上一代 CPU（CS100C）在电路方面的改进主要有以下几点：

- 使用累加器代替 ALU 中的加法器，减少逻辑门数量和电路面积；
- 内部总线使用并行总线代替原来的串行总线，减少逻辑门数量，提高性能；
- 让指令内存地址和数据内存地址一样可以被部件打断，简化指令内存地址译码电路；
- 使用集成的指令译码电路，提高模块化程度，便于制作与调试；

相较于前一型号 CPU（CS160E）修改了绿色总线（指令与中断）的格式，简化译码电路。

不同类型部件推荐使用不同颜色的背景区分：

- 计算部件：白色
- 指令内存：绿色
- 数据内存：黄色
- 输入部件：红色
- 输出部件：蓝色
- 驱动部件：橙色
- 测试部件：紫色



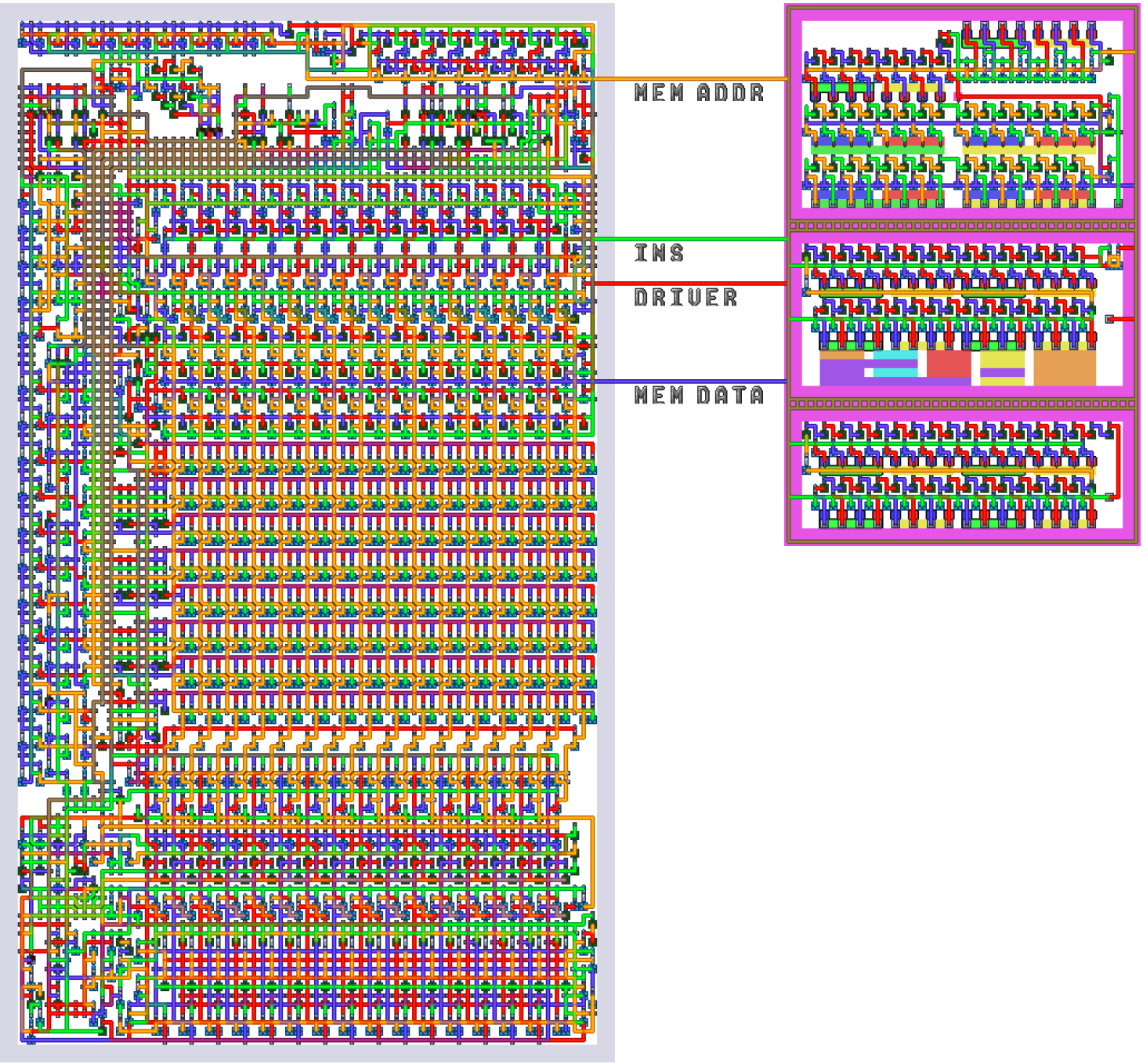
> 大小 65 × 115 格
> 逻辑门数量 1,072 个

第二部分 总线格式

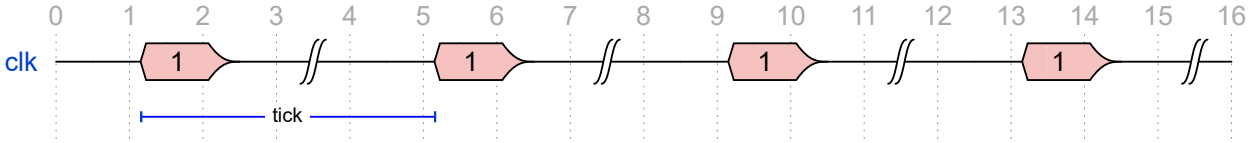
2.1 四色总线

CS161E 和连接的部件最多只需要连接红蓝绿黄四种颜色的四根电线，这四根电线被称为四色总线，部件连接到四色总线上既接入系统，四色总线在系统中任何位置的定义都相同。其中除了红色总线外，其余颜色的总线都使用状态串行传递数据。

- 红色总线：驱动（时钟）信号（驱动 => CPU）；
- 蓝色总线：数据内存数据信号（写：CPU => 数据内存 或 读：数据内存 => CPU）；
- 绿色总线：指令内存地址（CPU => 指令内存）和指令内存数据（指令内存 => CPU）信号，中断地址（中断部件 => CPU）；
- 黄色总线：数据内存地址及标志位信号（CPU => 数据内存）。



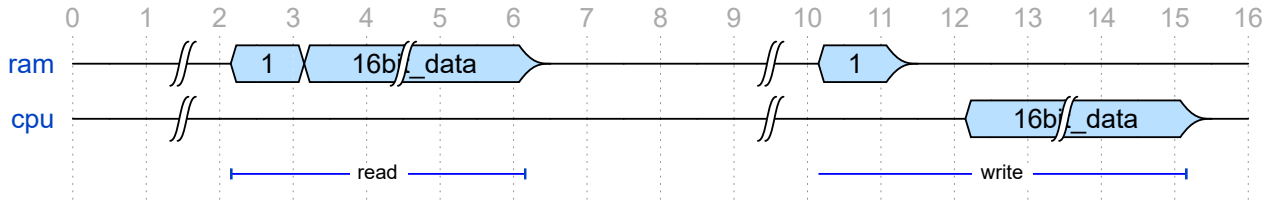
2.2 红色总线



红色总线连接到驱动（时钟），物理电源发出脉冲信号驱动 CPU 运行。从一个脉冲信号到另一个脉冲信号间为一个 CPU 时钟周期，执行一条指令。

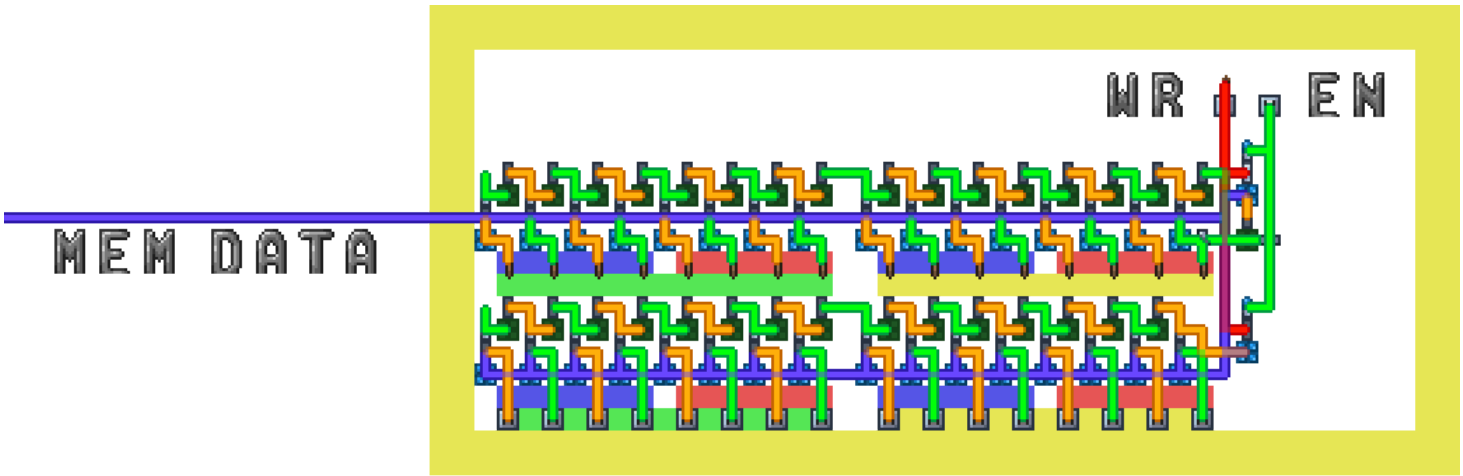
> 红色总线连接驱动示例

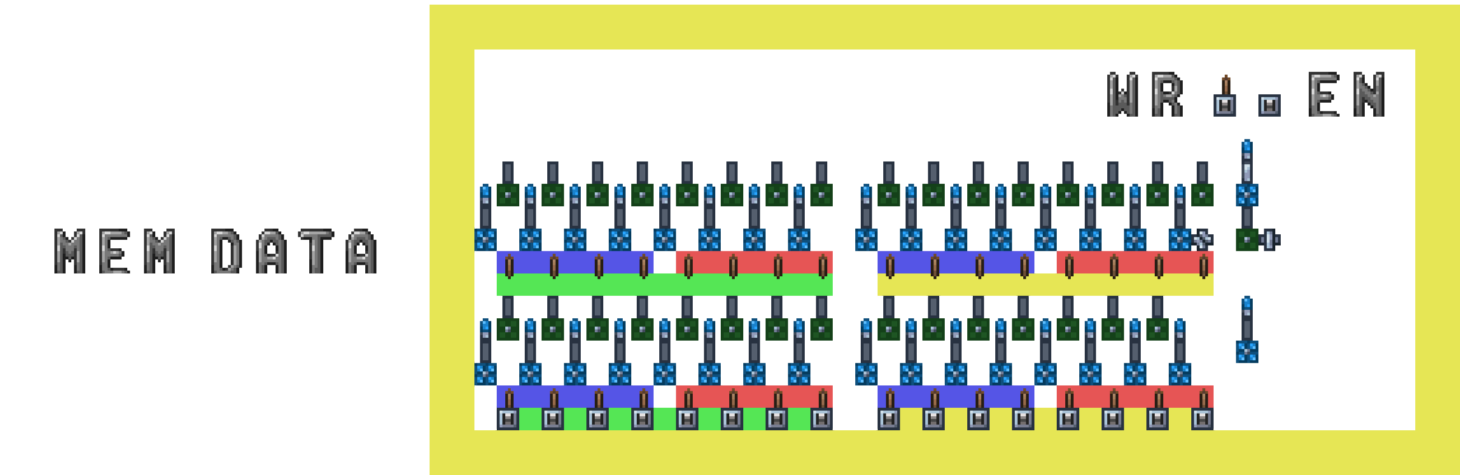
2.3 蓝色总线



蓝色总线传输数据内存的数据。在数据内存接收并处理内存地址后，向 CPU 发送一个脉冲信号，之后开始发送或接收数据。

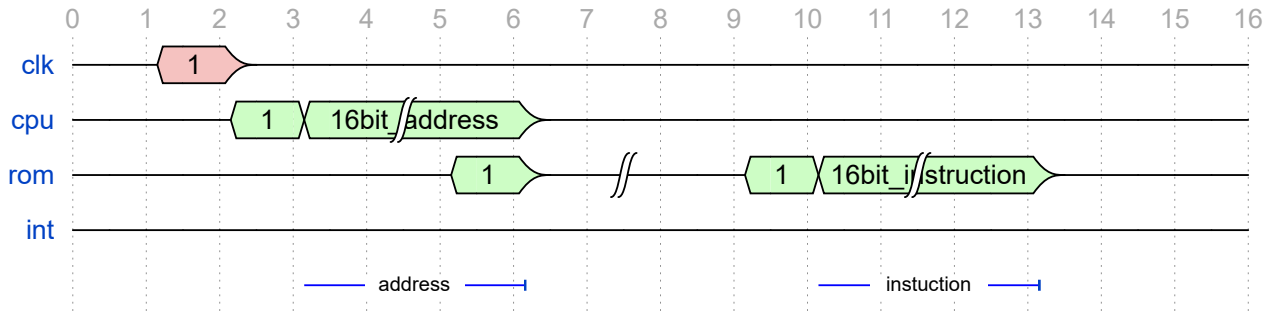
- 当数据传输方向为由数据内存到 CPU （读）时，在串行起始脉冲后马上开始传输数据；
- 当数据传输方向为由 CPU 到数据内存（写）时，在串行起始脉冲后一个逻辑帧开始传输数据。





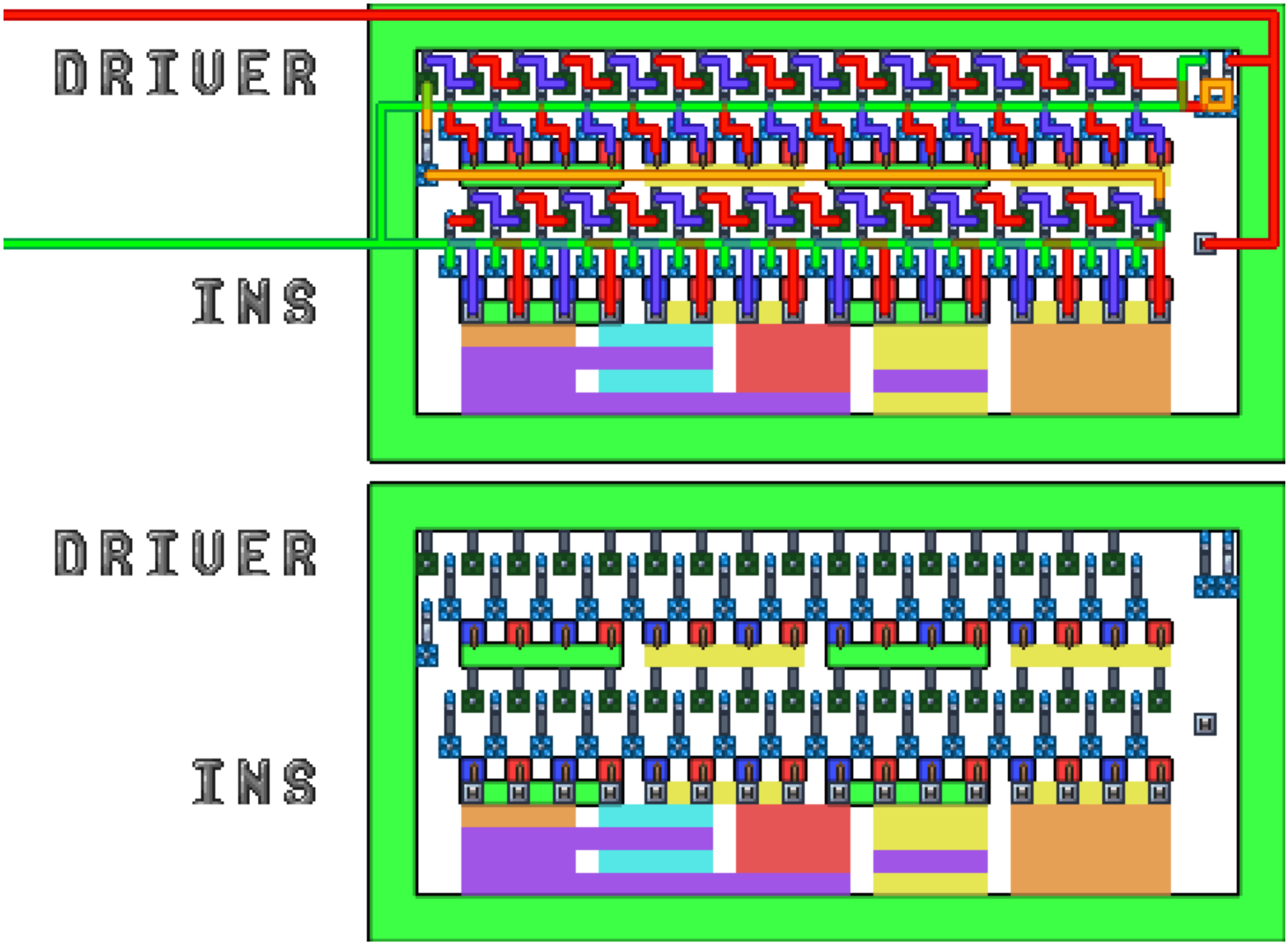
> 蓝色总线接收数据/发送数据示例

2.4 绿色总线

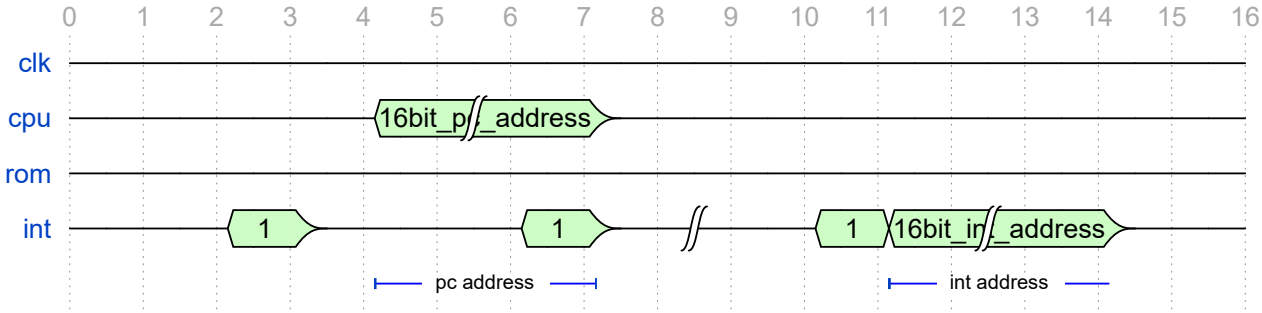


绿色总线传输指令内存地址与指令内存数据。

- 当驱动发送信号时， CPU 向指令内存发送 16 位指令内存地址，指令内存可以在接收任意一位后发送一个打断信号（脉冲）给 CPU，提前完成指令地址的接收；
- 发送完成后 CPU 进入接收态，指令内存就绪时向 CPU 发送 16 位指令数据。



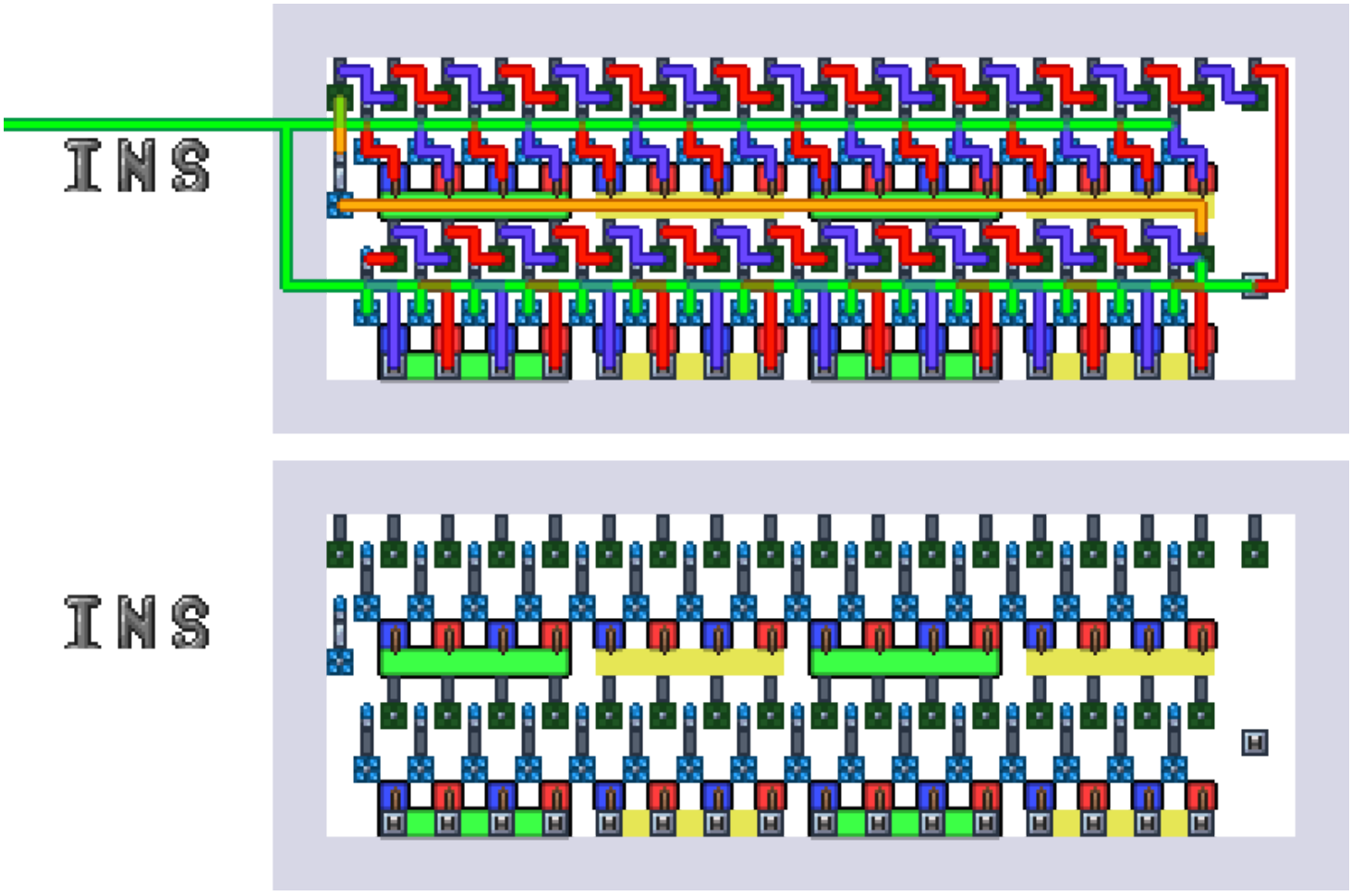
> 绿色总线接收指令地址/发送指令数据示例



同时绿色总线也用于传输中断信号：

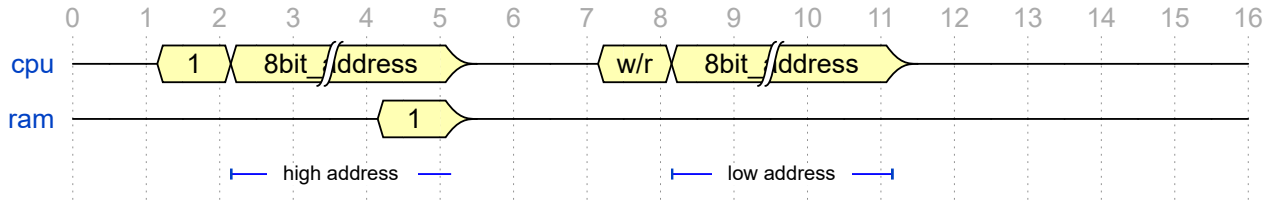
- 当驱动没有发出信号，但中断源向绿色总线发送脉冲时， CPU 在一逻辑帧后向中断部件发送 16 位指令内存地址，中断部件可以在接收任意一位后发送一个打断信号（脉冲）给 CPU，提前完成指令地址的接收；
- 发送完成后 CPU 进入接收态，之后指令内存向 CPU 发送 16 位中断地址，CPU 会将 PC 设置为中断地址的值，下一个时钟周期将从中断地址开始执行。

中断子程序的返回与调用类似，但是不需要保存当前 PC 地址。中断和返回不能由同步时钟源产生，可以为时钟源添加多路选择器生成用于驱动的时钟和用于中断与返回的时钟。



> 绿色总线发出中断示例

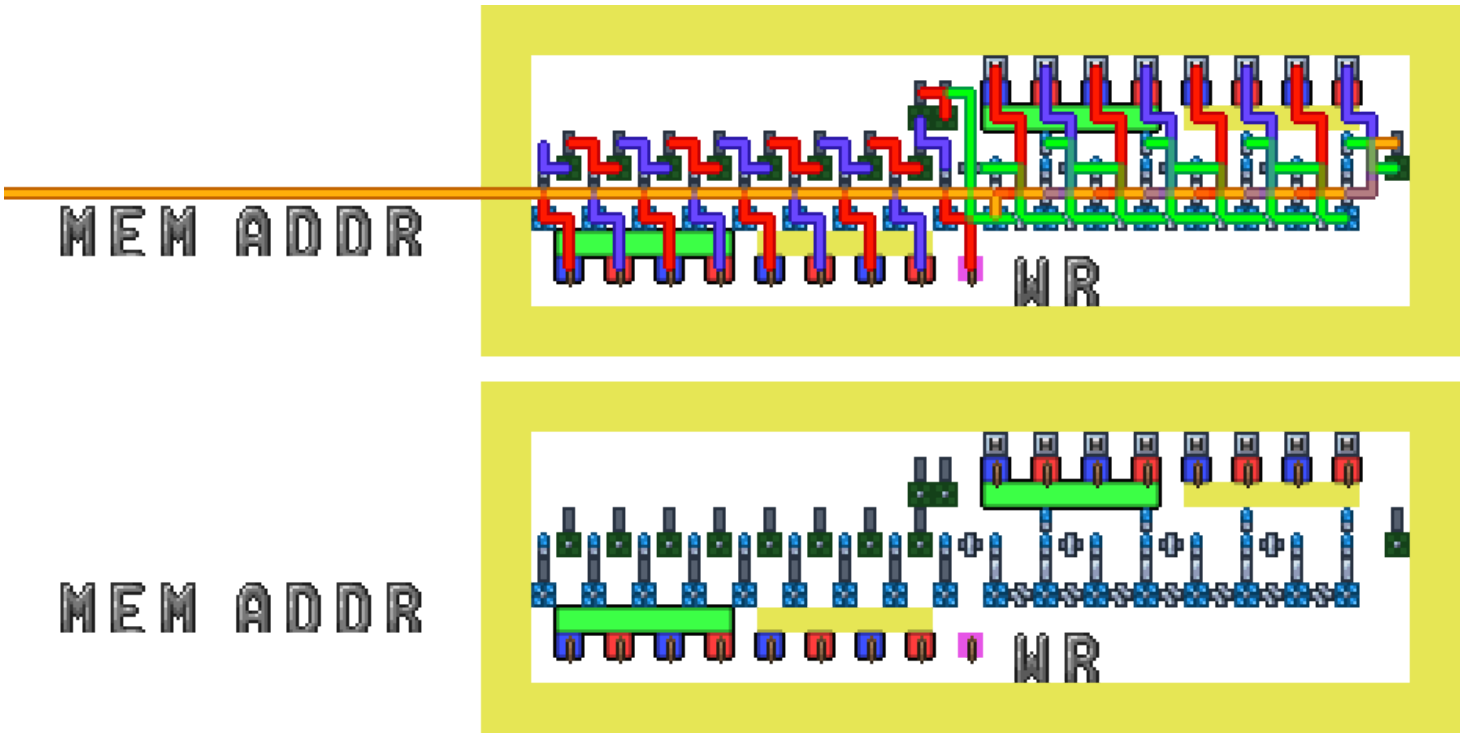
2.5 黄色总线



黄色总线传输数据内存地址与内存读写控制信号。

- 首先 CPU 向数据内存发送 8 位高位地址（部件号），数据内存可以在接收任意一位后发送一个打断信号（脉冲）给 CPU，提前完成高位地址的接收；
- 打断后，CPU 在两个逻辑帧后发送内存标志位，0 表示 CPU 将写入数据内存，1 表示 CPU 将读取数据内存；
- 发送标志位后，CPU 将发送 8 位低位地址（部件内偏移）。

高位地址和低位地址都可以不完整的接收与处理。



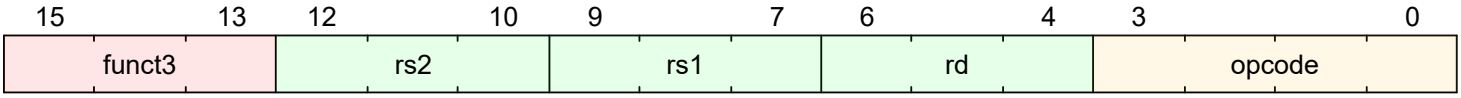
> 黄色总线接收地址示例

第三部分 指令类型

RVE 共有 4 种指令类型，26 条指令。

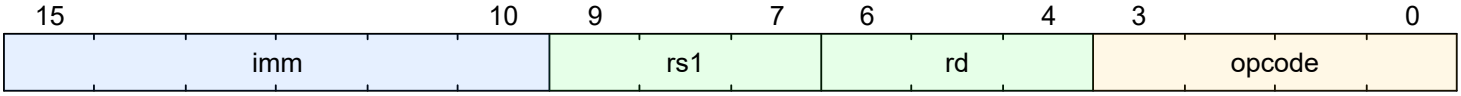
3.1 寄存器类型（R-type）

寄存器类型有 `add`、 `sub`、 `slt`、 `sltu`、 `xor`、 `or`、 `and`、 `sll`、 `srl`、 `sra` ，共 10 条指令。



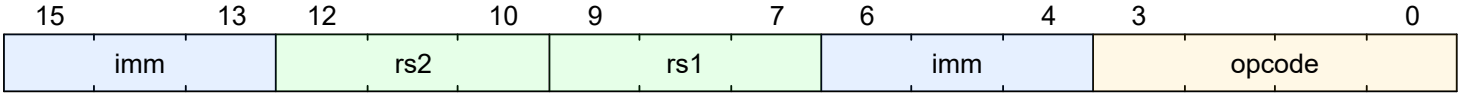
3.2 短立即数和访存读取类型（I-type）

短立即数和访存读取类型有 `addi`、 `slti`、 `sltiu`、 `xori`、 `ori`、 `andi`、 `slli`、 `srli`、 `srai`、 `lh`、 `jalr` ，共 11 条指令。



3.3 访存写入和条件跳转类型（SB-type）

访存写入和条件跳转类型有 `sh`、 `beq`、 `bne` ，共 3 条指令。



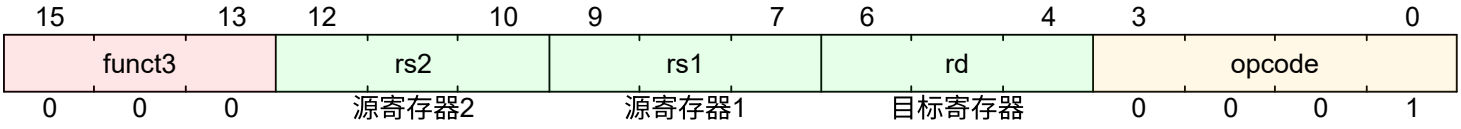
3.4 长立即数和无条件跳转类型（UJ-type）

长立即数和无条件跳转类型有 `lui`、 `jal` ，共 2 条指令。

第四部分 指令格式

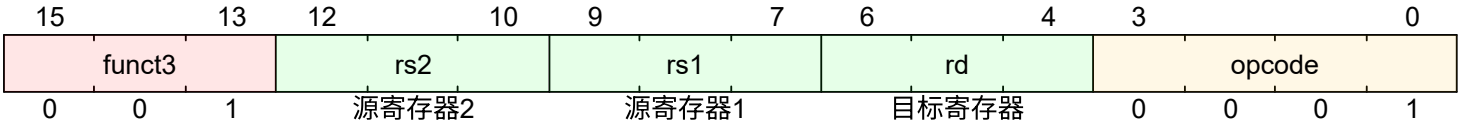
4.1 寄存器类型（R-type）

4.1.1 加（Add）



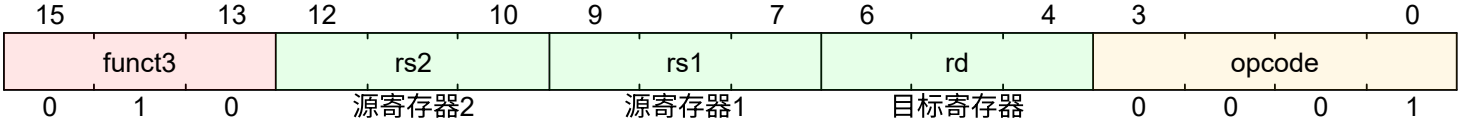
- 汇编指令： `add rd, rs1, rs2`
- 描述：x[rs1] 加上 x[rs2]，结果写入 x[rd]。忽略算术溢出。
- 寄存器传输级定义：x[rd] = x[rs1] + x[rs2]

4.1.2 减（Substract）



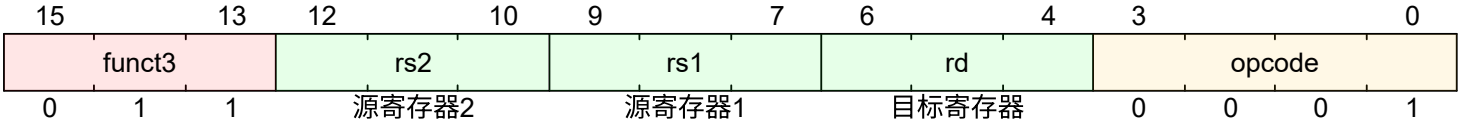
- 汇编指令： `sub rd, rs1, rs2`
- 描述：x[rs1] 减去 x[rs2]，结果写入 x[rd]。忽略算术溢出。
- 寄存器传输级定义：x[rd] = x[rs1] − x[rs2]

4.1.3 小于则置位（Set if Less Than）



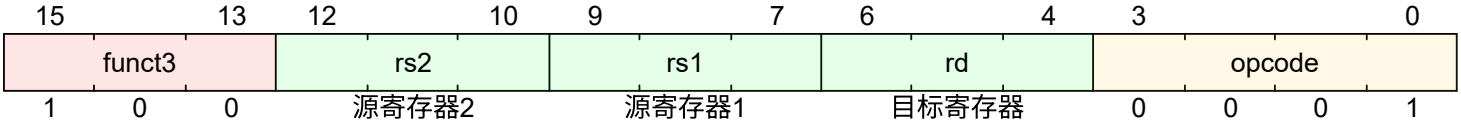
- 汇编指令： `slt rd, rs1, rs2`
- 描述：比较 x[rs1] 和 x[rs2] 中的数，如果 x[rs1] 更小，向 x[rd] 写入 1，否则写入 0。
- 寄存器传输级定义：x[rd] = (x[rs1] $<_s$ x[rs2])

4.1.4 无符号小于则置位（Set if Less Than, Unsigned）



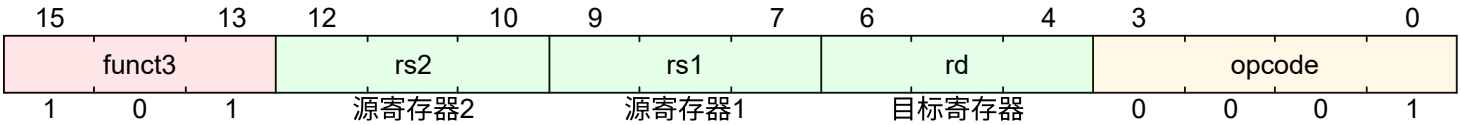
- 汇编指令： `sltu rd, rs1, rs2`
- 描述：比较 x[rs1] 和x [rs2]，比较时视为无符号数。如果 x[rs1] 更小，向 x[rd] 写入 1，否则写入 0。
- 寄存器传输级定义：x[rd] = (x[rs1] $<_u$ x[rs2])

4.1.5 异或（Exclusive-OR）



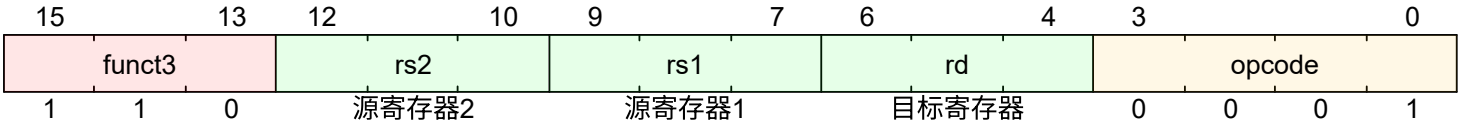
- 汇编指令： `xor rd, rs1, rs2`
- 描述：将 x[rs1] 和 x[rs2] 按位异或的结果写入 x[rd]。
- 寄存器传输级定义：x[rd] = x[rs1] ^ x[rs2]

4.1.6 或（OR）



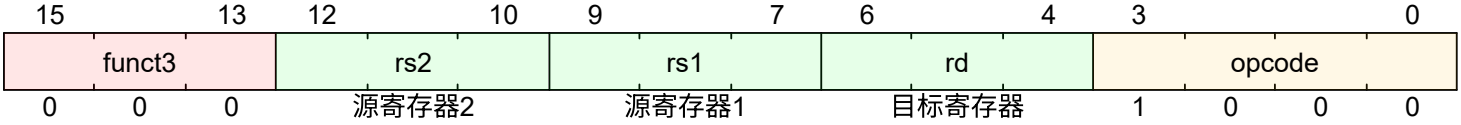
- 汇编指令： `or rd, rs1, rs2`
- 描述：将 x[rs1] 和 x[rs2] 按位或的结果写入 x[rd]。
- 寄存器传输级定义：x[rd] = x[rs1] | x[rs2]

4.1.7 与（And）



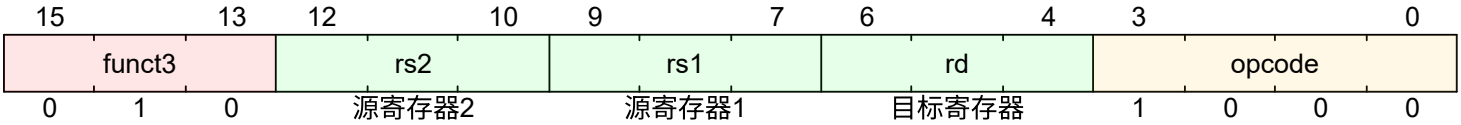
- 汇编指令： `and rd, rs1, rs2`
- 描述：将 x[rs1] 和 x[rs2] 按位与的结果写入 x[rd]。
- 寄存器传输级定义：x[rd] = x[rs1] & x[rs2]

4.1.8 逻辑左移（Shift Left Logical）



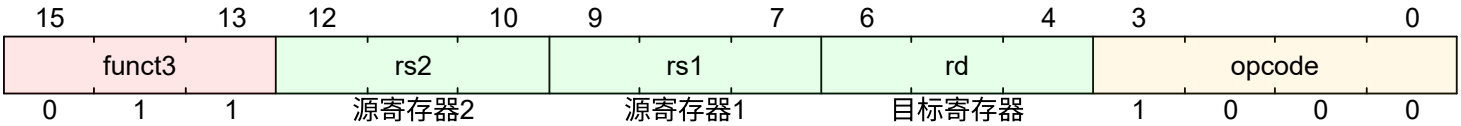
- 汇编指令： `sll rd, rs1, rs2`
- 描述：把 x[rs1] 左移 x[rs2] 位，空出的位置填入 0，结果写入x[rd]。x[rs2]的低 4 位代表移动位数，其高位则被忽略。
- 寄存器传输级定义：x[rd] = x[rs1] $\ll$ x[rs2]

4.1.9 逻辑右移（Shift Right Logical）



- 汇编指令： `srl rd, rs1, rs2`
- 描述：把 x[rs1] 右移 x[rs2] 位，空出的位置填入 0，结果写入 x[rd]。x[rs2] 的低 4 位代表移动位数，其高位则被忽略。
- 寄存器传输级定义：x[rd] = (x[rs1] $\gg_u$ x[rs2])

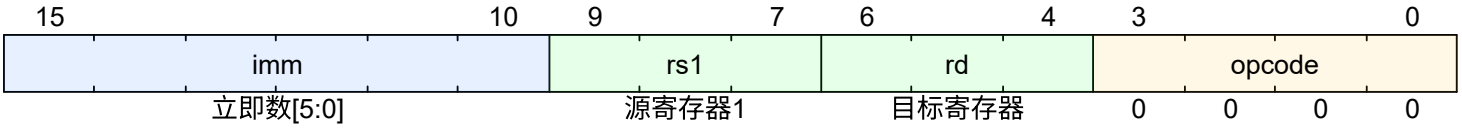
4.1.10 算术右移（Shift Right Arithmetic）



- 汇编指令： `sra rd, rs1, rs2`
- 描述：把 x[rs1] 右移 x[rs2] 位，空位用 x[rs1] 的最高位填充，结果写入x[rd]。x[rs2] 的低 4 位为移动位数，高位则被忽略。
- 寄存器传输级定义：x[rd] = x[rs1] $\gg_s$ x[rs2]

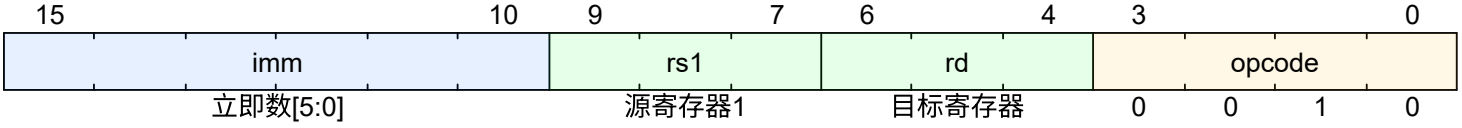
4.2 短立即数和访存读取类型（I-type）

4.2.1 加立即数（Add Immediate）



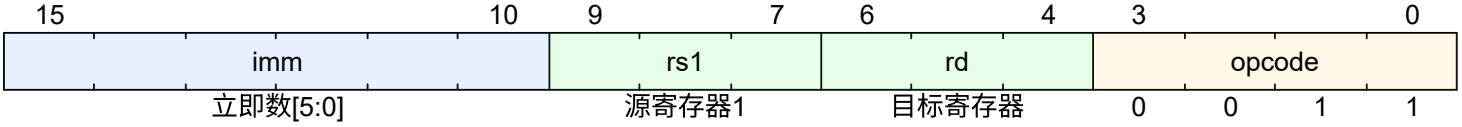
- 汇编指令： `addi rd, rs1, imm`
- 描述：x[rs1] 加上符号位扩展的立即数，结果写入 x[rd]。忽略算术溢出。
- 寄存器传输级定义：x[rd] = x[rs1] + sext(immediate)

4.2.2 小于立即数则置位（Set if Less Than Immediate）



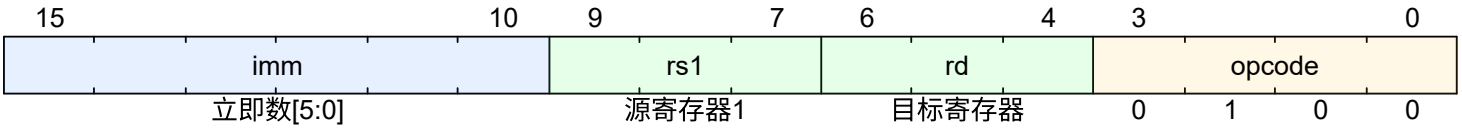
- 汇编指令： `slti rd, rs1, imm`
- 描述：比较 x[rs1] 和符号位扩展的立即数，如果 x[rs1] 更小，向 x[rd] 写入 1，否则写入 0。
- 寄存器传输级定义：x[rd] = (x[rs1] $<_s$ sext(immediate))

4.2.3 无符号小于立即数则置位（Set if Less Than Immediate, Unsigned）



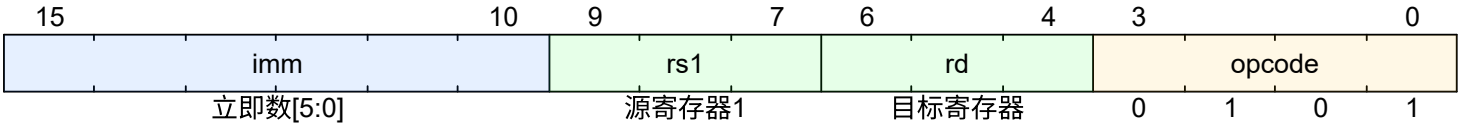
- 汇编指令： `sltiu rd, rs1, imm`
- 描述：比较 x[rs1] 和符号位扩展的立即数，比较时视为无符号数。如果 x[rs1] 更小，向 x[rd] 写入 1，否则写入 0。
- 寄存器传输级定义：x[rd] = (x[rs1] $<_u$ sext(immediate))

4.2.4 立即数异或（Exclusive-OR Immediate）



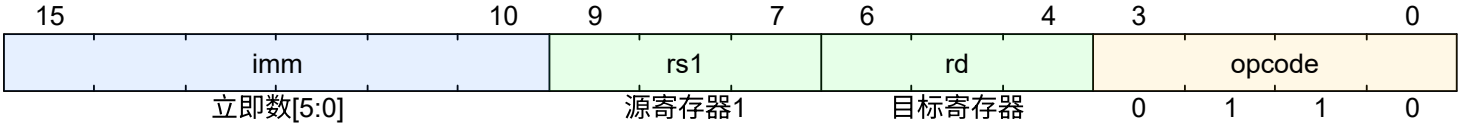
- 汇编指令： `xori rd, rs1, imm`
- 描述：把 x[rs1] 和符号位扩展的立即数按位异或的结果写入 x[rd]。
- 寄存器传输级定义：x[rd] = x[rs1] ^ sext(immediate)

4.2.5 立即数或（OR Immediate）



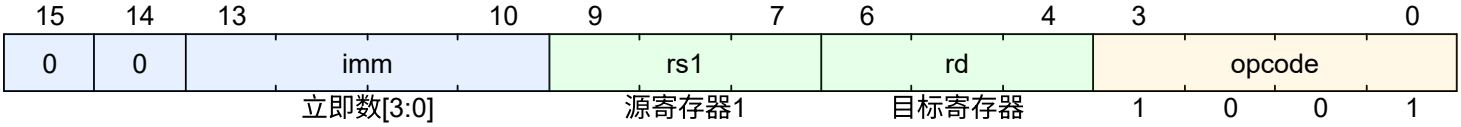
- 汇编指令： `ori rd, rs1, imm`
- 描述：把 x[rs1] 和符号位扩展的立即数按位或的结果写入 x[rd]。
- 寄存器传输级定义：x[rd] = x[rs1] | sext(immediate)

4.2.6 立即数与（And Immediate）



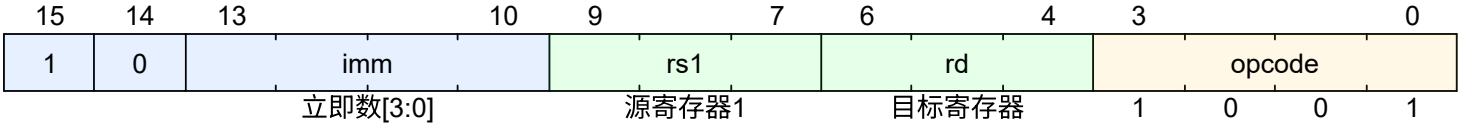
- 汇编指令： `andi rd, rs1, imm`
- 描述：把 x[rs1] 和符号位扩展的立即数按位与的结果写入 x[rd]。
- 寄存器传输级定义：x[rd] = x[rs1] & sext(immediate)

4.2.7 立即数逻辑左移（Shift Left Logical Immediate）



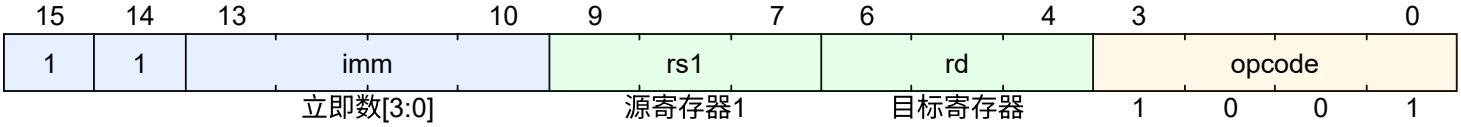
- 汇编指令： `slli rd, rs1, imm`
- 描述：把 x[rs1] 左移 shamt 位，空出的位置填入 0，结果写入 x[rd]。
- 寄存器传输级定义：x[rd] = x[rs1] $\ll$ shamt

4.2.8 立即数逻辑右移（Shift Right Logical Immediate）



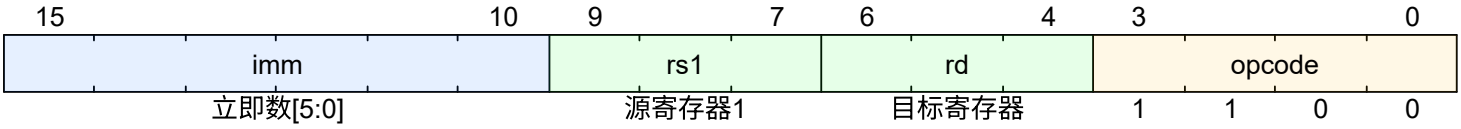
- 汇编指令： `srli rd, rs1, imm`
- 描述：把 x[rs1] 右移 shamt 位，空位用 x[rs1] 的最高位填充，结果写入 x[rd]。
- 寄存器传输级定义：x[rd] = x[rs1] $\gg_s$ shamt

4.2.9 立即数算术右移（Shift Right Arithmetic Immediate）



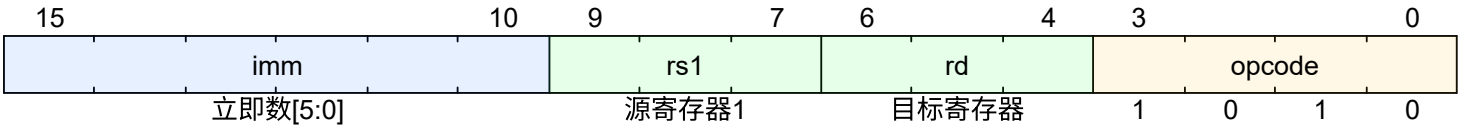
- 汇编指令： `srai rd, rs1, imm`
- 描述：把 x[rs1] 右移 shamt 位，空出的位置填入 0，结果写入 x[rd]。
- 寄存器传输级定义：x[rd] = x[rs1] $\gg_u$ shamt

4.2.10 半字加载 （Load Halfword）



- 汇编指令： `lh rd, offset(rs1)`
- 描述：从地址 x[rs1] + sext(offset) 读取两个字节。
- 寄存器传输级定义：x[rd] = M[x[rs1] + sext(offset)][15:0]

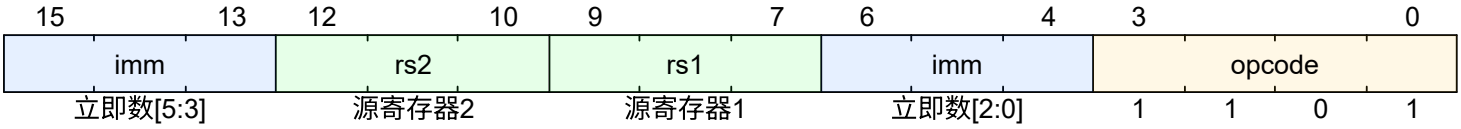
4.2.11 跳转并寄存器链接（Jump and Link Register）



- 汇编指令： `jalr rd, offset(rs1)`
- 描述：把 pc 设置为 x[rs1] + sext(offset)，把计算出的地址的最低有效位设为 0，并将原 pc + 1 的值写入 x[rd]。
- 寄存器传输级定义：x[rd] = pc + 1; pc = x[rs1] + sext(offset);

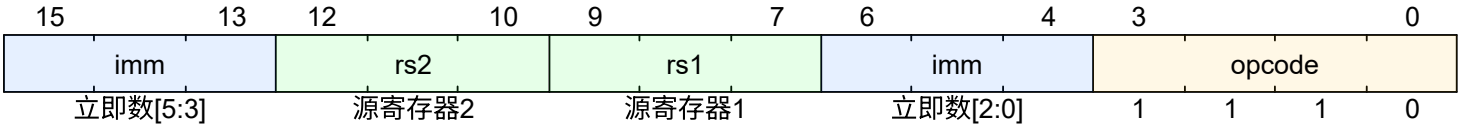
4.3 访存写入和条件跳转类型（SB-type）

4.3.1 存半字（Store Halfword）



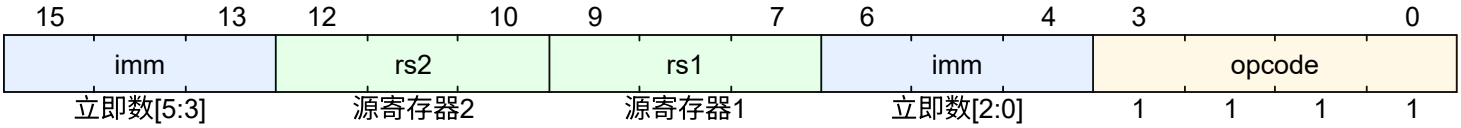
- 汇编指令： `sh rs2, offset(rs1)`
- 描述：将 x[rs2] 的 2 个字节存入内存地址 x[rs1] + sext(offset)。
- 寄存器传输级定义：M[x[rs1] + sext(offset)] = x[rs2]

4.3.2 相等时分支（Branch if Equal）



- 汇编指令： `beq rs1, rs2, offset`
- 描述：若 x[rs1] 和 x[rs2] 的值相等，把 pc 的值设为当前值加上符号位扩展的偏移 offset。
- 寄存器传输级定义：if (rs1 == rs2) pc += sext(offset); else pc += 1;

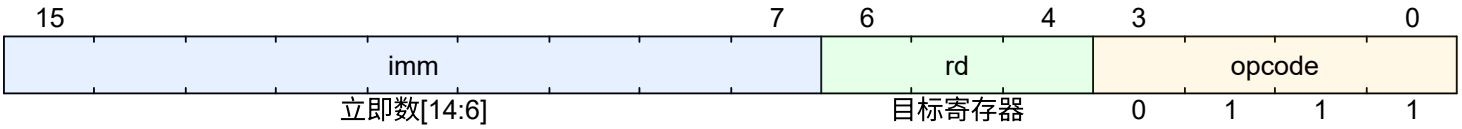
4.3.3 不相等时分支（Branch if Not Equal）



- 汇编指令： `bne rs1, rs2, offset`
- 描述：若 x[rs1] 和 x[rs2] 的值不相等，把 pc 的值设为当前值加上符号位扩展的偏移 offset。
- 寄存器传输级定义：if (rs1 $\neq$ rs2) pc += sext(offset); else pc += 1;

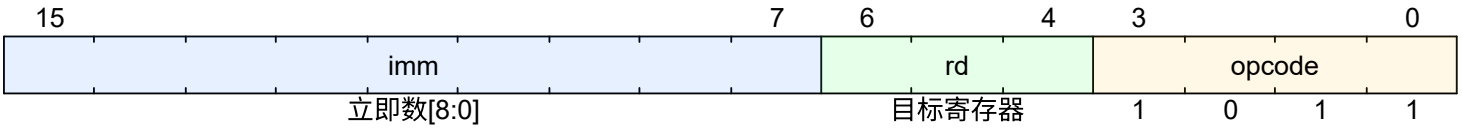
4.4 长立即数和无条件跳转类型（UJ-type）

4.4.1 高位立即数加载（Load Upper Immediate）



- 汇编指令： `lui rd, imm`
- 描述：高位立即数写入 x[rd] 中。
- 寄存器传输级定义：x[rd] = sext(immediate[14:6] $\ll$ 6)

4.4.2 跳转并链接（Jump and Link）



- 汇编指令： `jal rd, imm`
- 描述：把下一条指令的地址（pc + 1）的值写入 x[rd]，然后把 pc 设置为当前值加上符号位扩展的 offset。
- 寄存器传输级定义：x[rd] = pc + 1; pc += sext(offset);