值(0x016)。在此时，组合逻辑已经根据 addq 指令被更新了，但是状态还是保持着第二条 irmovq 指令(用浅灰色表示)设置的值。

当时钟上升开始周期 4 时(点 3)，会更新程序计数器、寄存器文件和条件码寄存器，因此我们用蓝色来表示，但是组合逻辑还没有对这些变化做出反应，所以用白色表示。在这个周期内，会取出并执行 je 指令(表中第 4 行)，在图中用深灰色表示。因为条件码 ZF 为 0，所以不会选择分支。在这个周期末尾(点 4)，程序计数器已经产生了新值 0x01f。组合逻辑已经根据 je 指令(用深灰色表示)被更新过了，但是直到下个周期开始之前，状态还是保持着 addq 指令(用蓝色表示)设置的值。

如此例所示，用时钟来控制状态单元的更新，以及值通过组合逻辑来传播，足够控制我们 SEQ 实现中每条指令执行的计算了。每次时钟由低变高时，处理器开始执行一条新指令。

4.3.4　SEQ 阶段的实现

本节会设计实现 SEQ 所需要的控制逻辑块的 HCL 描述。完整的 SEQ 的 HCL 描述请参见网络旁注 ARCH:HCL。在此，我们给出一些例子，而其他的作为练习题。建议你做做这些练习来检验你的理解，即这些块是如何与不同指令的计算需求相联系的。

我们没有讲的那部分 SEQ 的 HCL 描述，是不同整数和布尔信号的定义，它们可以作为 HCL 操作的参数。其中包括不同硬件信号的名字，以及不同指令代码、功能码、寄存器名字、ALU 操作和状态码的常数值。只列出了那些在控制逻辑中必须被显式引用的常数。图 4-26 列出了我们使用的常数。按照习惯，常数值都是大写的。

名称	值(十六进制)	含义
IHALT	0	halt 指令的代码
INOP	1	nop 指令的代码
IRRMOVQ	2	rrmovq 指令的代码
IIRMOVQ	3	irmovq 指令的代码
IRMMOVQ	4	rmmovq 指令的代码
IMRMOVQ	5	mrmovq 指令的代码
IOPL	6	整数运算指令的代码
IJXX	7	跳转指令的代码
ICALL	8	call 指令的代码
IRET	9	ret 指令的代码
IPUSHQ	A	pushq 指令的代码
IPOPQ	B	popq 指令的代码
FNONE	0	默认功能码
RRSP	4	% rsp 的寄存器 ID
RNONE	F	表明没有寄存器文件访问
ALUADD	0	加法运算的功能
SAOK	1	①正常操作状态码
SADR	2	②地址异常状态码
SINS	3	③非法指令异常状态码
SHLT	4	④halt 状态码

图 4-26　HCL 描述中使用的常数值。这些值表示的是指令、功能码、寄存器 ID、ALU 操作和状态码的编码

除了图 4-18～图 4-21 中所示的指令以外，还包括了对 nop 和 halt 指令的处理。nop