位：高 4 位是代码(code)部分，低 4 位是功能(function)部分。如图 4-2 所示，代码值为 0～0xB。功能值只有在一组相关指令共用一个代码时才有用。图 4-3 给出了整数操作、分支和条件传送指令的具体编码。可以观察到，rrmovq 与条件传送有同样的指令代码。可以把它看作是一个“无条件传送”，就好像 jmp 指令是无条件跳转一样，它们的功能代码都是 0。

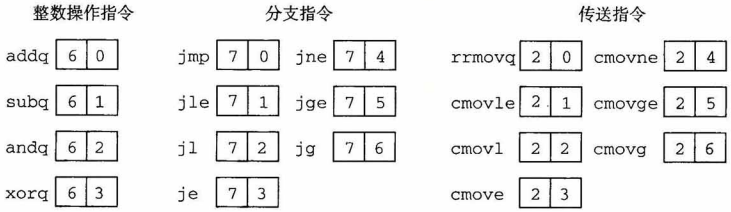

图 4-3　Y86-64 指令集的功能码。这些代码指明是某个整数操作、分支条件还是数据传送条件。这些指令是图 4-2 中所示的 OPq、jXX 和 cmovXX

如图 4-4 所示，15 个程序寄存器中每个都有一个相对应的范围在 0 到 0xE 之间的寄存器标识符(register ID)。Y86-64 中的寄存器编号跟 x86-64 中的相同。程序寄存器存在 CPU 中的一个寄存器文件中，这个寄存器文件就是一个小的、以寄存器 ID 作为地址的随机访问存储器。在指令编码中以及在我们的硬件设计中，当需要指明不应访问任何寄存器时，就用 ID 值 0xF 来表示。

数字	寄存器名字	数字	寄存器名字
0	%rax	8	%r8
1	%rcx	9	%r9
2	%rdx	A	%r10
3	%rbx	B	%r11
4	%rsp	C	%r12
5	%rbp	D	%r13
6	%rsi	E	%r14
7	%rdi	F	无寄存器

图 4-4　Y86-64 程序寄存器标识符。15 个程序寄存器中每个都有一个相对应的标识符(ID)，范围为 0～0xE。如果指令中某个寄存器字段的 ID 值为 0xF，就表明此处没有寄存器操作数

有的指令只有一个字节长，而有的需要操作数的指令编码就更长一些。首先，可能有附加的寄存器指示符字节(register specifier byte)，指定一个或两个寄存器。在图 4-2 中，这些寄存器字段称为 rA 和 rB。从指令的汇编代码表示中可以看到，根据指令类型，指令可以指定用于数据源和目的的寄存器，或是用于地址计算的基址寄存器。没有寄存器操作数的指令，例如分支指令和 call 指令，就没有寄存器指示符字节。那些只需要一个寄存器操作数的指令(irmovq、pushq 和 popq)将另一个寄存器指示符设为 0xF。这种约定在我们的处理器实现中非常有用。

有些指令需要一个附加的 4 字节常数字(constant word)。这个字能作为 irmovq 的立即数数据，rmmovq 和 mrmovq 的地址指示符的偏移量，以及分支指令和调用指令的目的地址。注意，分支指令和调用指令的目的是一个绝对地址，而不像 IA32 中那样使用 PC