地址	值
0x100	0xFF
0x104	0xAB
0x108	0x13
0x10C	0x11

寄存器	值
%rax	0x100
%rcx	0x1
%rdx	0x3

填写下表，给出所示操作数的值：

操作数	值
%rax	
0x104	
$0x108	
(%rax)	
4(%rax)	
9(%rax,%rdx)	
260(%rcx,%rdx)	
0xFC(,%rcx,4)	
(%rax,%rdx,4)	

3.4.2　数据传送指令

最频繁使用的指令是将数据从一个位置复制到另一个位置的指令。操作数表示的通用性使得一条简单的数据传送指令能够完成在许多机器中要好几条不同指令才能完成的功能。我们会介绍多种不同的数据传送指令，它们或者源和目的类型不同，或者执行的转换不同，或者具有的一些副作用不同。在我们的讲述中，把许多不同的指令划分成*指令类*，每一类中的指令执行相同的操作，只不过操作数大小不同。

图 3-4 列出的是最简单形式的数据传送指令——MOV 类。这些指令把数据从源位置复制到目的位置，不做任何变化。MOV 类由四条指令组成：movb、movw、movl 和 movq。这些指令都执行同样的操作；主要区别在于它们操作的数据大小不同：分别是 1、2、4 和 8 字节。

指令	效果	描述
MOV　*S*, *D*	*D*←*S*	传送
movb		传送字节
movw		传送字
movl		传送双字
movq		传送四字
movabsq　*I*, *R*	*R*←*I*	传送绝对的四字

图 3-4　简单的数据传送指令

源操作数指定的值是一个立即数，存储在寄存器中或者内存中。目的操作数指定一个位置，要么是一个寄存器或者，要么是一个内存地址。x86-64 加了一条限制，传送指令的两个操作数不能都指向内存位置。将一个值从一个内存位置复制到另一个内存位置需要两条指令——第一条指令将源值加载到寄存器中，第二条将该寄存器值写入目的位置。参考图 3-2，这些指令的寄存器操作数可以是 16 个寄存器有标号部分中的任意一个，寄存器部