4.1　Y86-64 指令集体系结构

定义一个指令集体系结构(例如 Y86-64)包括定义各种状态单元、指令集和它们的编码、一组编程规范和异常事件处理。

4.1.1　程序员可见的状态

如图 4-1 所示，Y86-64 程序中的每条指令都会读取或修改处理器状态的某些部分。这称为*程序员可见状态*，这里的“程序员”既可以是用汇编代码写程序的人，也可以是产生机器级代码的编译器。在处理器实现中，只要我们保证机器级程序能够访问程序员可见状态，就不需要完全按照 ISA 暗示的方式来表示和组织这个处理器状态。Y86-64 的状态类似于 x86-64。有 15 个*程序寄存器*：%rax、%rcx、%rdx、%rbx、%rsp、%rbp、%rsi、%rdi 和 %r8 到 %r14。(我们省略了 x86-64 的寄存器%r15 以简化指令的编码。)每个程序寄存器存储一个 64 位的字。寄存器%rsp 被入栈、出栈、调用和返回指令作为栈指针。除此之外，寄存器没有固定的含义或固定值。有 3 个一位的*条件码*：ZF、SF 和 OF，它们保存着最近的算术或逻辑指令所造成影响的有关信息。程序计数器(PC)存放当前正在执行指令的地址。

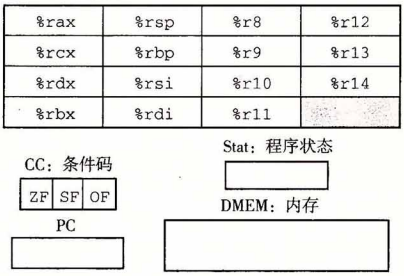

图 4-1　Y86-64 程序员可见状态。同 x86-64 一样，Y86-64 的程序可以访问和修改程序寄存器、条件码、程序计数器(PC)和内存。状态码指明程序是否运行正常，或者发生了某个特殊事件

内存从概念上来说就是一个很大的字节数组，保存着程序和数据。Y86-64 程序用*虚拟地址*来引用内存位置。硬件和操作系统软件联合起来将虚拟地址翻译成实际或物理地址，指明数据实际存在内存中哪个地方。第 9 章将更详细地研究虚拟内存。现在，我们只认为虚拟内存系统向 Y86-64 程序提供了一个单一的字节数组映像。

程序状态的最后一个部分是状态码 Stat，它表明程序执行的总体状态。它会指示是正常运行，还是出现了某种异常，例如当一条指令试图去读非法的内存地址时。在 4.1.4 节中会讲述可能的状态码以及异常处理。

4.1.2　Y86-64 指令

图 4-2 给出了 Y86-64 ISA 中各个指令的简单描述。这个指令集就是我们处理器实现的目标。Y86-64 指令集基本上是 x86-64 指令集的一个子集。它只包括 8 字节整数操作，寻址方式较少，操作也较少。因为我们只有 8 字节数据，所以称之为“字(word)”不会有任何歧义。在这个图中，左边是指令的汇编码表示，右边是字节编码。图 4-3 给出了其中一些指令更详细的内容。汇编代码格式类似于 x86-64 的 ATT 格式。

下面是 Y86-64 指令的一些细节。

- x86-64 的 movq 指令分成了 4 个不同的指令：irmovq、rrmovq、mrmovq 和 rmmovq，分别显式地指明源和目的的格式。源可以是立即数(i)、寄存器(r)或内存(m)。指令名字的第一个字母就表明了源的类型。目的可以是寄存器(r)或内存(m)。指令名字的第二个字母指明了目的的类型。在决定如何实现数据传送时，显式地指明数据传送的