线化程度需要大量的硬件。另一方面，除法相对不太常用，而且要想实现低延迟或完全流水线化是很困难的。

这些算术运算的延迟、发射时间和容量会影响合并函数的性能。我们用 CPE 值的两个基本界限来描述这种影响：

界限	整数		浮点数	
	+	*	+	*
延迟	1.00	3.00	3.00	5.00
吞吐量	0.50	1.00	1.00	0.50

延迟界限给出了任何必须按照严格顺序完成合并运算的函数所需要的最小 CPE 值。根据功能单元产生结果的最大速率，吞吐量界限给出了 CPE 的最小界限。例如，因为只有一个整数乘法器，它的发射时间为 1 个时钟周期，处理器不可能支持每个时钟周期大于 1 条乘法的速度。另一方面，四个功能单元都可以执行整数加法，处理器就有可能持续每个周期执行 4 个操作的速率。不幸的是，因为需要从内存读数据，这造成了另一个吞吐量界限。两个加载单元限制了处理器每个时钟周期最多只能读取两个数据值，从而使得吞吐量界限为 0.50。我们会展示延迟界限和吞吐量界限对合并函数不同版本的影响。

5.7.3　处理器操作的抽象模型

作为分析在现代处理器上执行的机器级程序性能的一个工具，我们会使用程序的*数据流*(data-flow)表示，这是一种图形化的表示方法，展现了不同操作之间的数据相关是如何限制它们的执行顺序的。这些限制形成了图中的*关键路径*(critical path)，这是执行一组机器指令所需时钟周期数的一个下界。

在继续技术细节之前，检查一下函数 combine4 的 CPE 测量值是很有帮助的，到目前为止 combine4 是最快的代码：

函数	方法	整数		浮点数	
		+	*	+	*
combine4	累积在临时变量中	1.27	3.01	3.01	5.01
延迟界限		1.00	3.00	3.00	5.00
吞吐量界限		0.50	1.00	1.00	0.50

我们可以看到，除了整数加法的情况，这些测量值与处理器的延迟界限是一样的。这不是巧合——它表明这些函数的性能是由所执行的求和或者乘积计算主宰的。计算 n 个元素的乘积或者和需要大约 $L \cdot n+K$ 个时钟周期，这里 L 是合并运算的延迟，而 K 表示调用函数和初始化以及终止循环的开销。因此，CPE 就等于延迟界限 L。

1. 从机器级代码到数据流图

程序的数据流表示是非正式的。我们只是想用它来形象地描述程序中的数据相关是如何主宰程序的性能的。以 combine4(图 5-10)为例来描述数据流表示法。我们将注意力集中在循环执行的计算上，因为对于大向量来说，这是决定性能的主要因素。我们考虑类型为 double 的数据、以乘法作为合并运算的情况，不过其他数据类型和运算的组合也有几乎一样的结构。这个循环编译出的代码由 4 条指令组成，寄存器%rdx 存放指向数组 data 中第 i 个元素的指针，%rax 存放指向数组末尾的指针，而%xmm0 存放累积值 acc。