A. 对于次数 n，这段代码执行多少次加法和多少次乘法运算？

B. 在我们的参考机上，算术运算的延迟如图 5-12 所示，我们测量了这个函数的 CPE 等于 5.00。根据由于实现函数第 7～8 行的操作迭代之间形成的数据相关，解释为什么会得到这样的 CPE。

练习题 5.6 我们继续探索练习题 5.5 中描述的多项式求值的方法。通过采用 Horner 法(以英国数学家 William G. Horner(1786—1837)命名)对多项式求值，我们可以减少乘法的数量。其思想是反复提出 x 的幂，得到下面的求值：

$$a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n)\cdots)) \tag{5.3}$$

使用 Horner 法，我们可以用下面的代码实现多项式求值：

```
1   /* Apply Horner's method */
2   double polyh(double a[], double x, long degree)
3   {
4       long i;
5       double result = a[degree];
6       for (i = degree-1; i >= 0; i--)
7           result = a[i] + x*result;
8       return result;
9   }
```

A. 对于次数 n，这段代码执行多少次加法和多少次乘法运算？

B. 在我们的参考机上，算术运算的延迟如图 5-12 所示，测量这个函数的 CPE 等于 8.00。根据由于实现函数第 7 行的操作迭代之间形成的数据相关，解释为什么会得到这样的 CPE。

C. 请解释虽然练习题 5.5 中所示的函数需要更多的操作，但是它是如何运行得更快的。

5.8 循环展开

循环展开是一种程序变换，通过增加每次迭代计算的元素的数量，减少循环的迭代次数。psum2 函数(见图 5-1)就是这样一个例子，其中每次迭代计算前置和的两个元素，因而将需要的迭代次数减半。循环展开能够从两个方面改进程序的性能。首先，它减少了不直接有助于程序结果的操作的数量，例如循环索引计算和条件分支。第二，它提供了一些方法，可以进一步变化代码，减少整个计算中关键路径上的操作数量。在本节中，我们会看一些简单的循环展开，不做任何进一步的变化。

图 5-16 是合并代码的使用“2×1 循环展开”的版本。第一个循环每次处理数组的两个元素。也就是每次迭代，循环索引 i 加 2，在一次迭代中，对数组元素 i 和 $i+1$ 使用合并运算。

一般来说，向量长度不一定是 2 的倍数。想要使我们的代码对任意向量长度都能正确工作，可以从两个方面来解释这个需求。首先，要确保第一次循环不会超出数组的界限。对于长度为 n 的向量，我们将循环界限设为 $n-1$。然后，保证只有当循环索引 i 满足 $i<n-1$ 时才会执行这个循环，因此最大数组索引 $i+1$ 满足 $i+1<(n-1)+1=n$。

把这个思想归纳为对一个循环按任意因子 k 进行展开，由此产生 $k×1$ 循环展开。为此，上限设为 $n-k+1$，在循环内对元素 i 到 $i+k-1$ 应用合并运算。每次迭代，循环索引 i 加 k。那么最大循环索引 $i+k-1$ 会小于 n。要使用第二个循环，以每次处理一个元素的方式处理向量的最后几个元素。这个循环体将会执行 $0\sim k-1$ 次。对于 $k=2$，我们能用一个简单的条件语句，可选地增加最后一次迭代，如函数 psum2(图 5-1)所示。对于 $k>2$，最后的这些情