小作业七：单机性能优化

计04 何秉翔 2020010944

1. 任务 0

1.1 测量结果

编译参数	Elapsed time	performance
-O0	1.0139 s	0.2648 $GFlops$
-O1	0.3447 s	0.7787 $GFlops$
-O2	0.3335 s	0.8050 $GFlops$
-O3	0.0487 s	5.5166 $GFlops$
-fast	0.0393 s	6.8323 $GFlops$

1.2 不同参数的优化

- -O0：不进行任何优化，代码生成速度快，但生成的代码执行速度较慢。
- -O1：进行基本优化，减小代码占据空间，包括：
  - Code Motion，即代码移动，比如将代码移动到循环之外。
  - Strength Reduction，即强度拆减，用简单的计算代替复杂的计算。
  - Instruction Scheduling，即指令调度，对指令进行重排，以最大化使用 CPU 的各种功能单元，提高程序的执行效率。
- -O2：进行更多的优化，会使编译时间略微增加，包括：
  - dead-code elimination，即死代码消除，消除不会被执行的代码。
  - copy propagation，即复制传播，消除语句之间的依赖关系。
  - variable renaming：即变量重命名，消除输出变量之间的依赖关系。
  - loop unrolling：即循环展开，减少循环变量的比较次数和分支跳转次数。
- -O3：进行更多的优化，但会增加编译时间，包括：
  - Fusion：循环融合，带来更少的分支跳转。
  - Collapsing IF Statements：即 IF 语句折叠，将嵌套 IF 语句通过逻辑连接符变成单一 IF 语句。
  - 向量化：采取很多向量化算法，提高代码的并行执行程度。
- -fast：提供了一个针对速度优化的选项，集成各种优化方法，相当于 -O3 加上一些额外的优化。
  - 快速浮点数算术
  - 快速函数调用
  - 快速存储器访问

2. 任务 1

2.1 测量结果

UNROLL_N	Elapsed time	performance
1	2.0679 s	15.8458 $GFlops$
2	1.9367 s	16.9195 $GFlops$
4	1.8022 s	18.1822 $GFlops$
8	1.7723 s	18.4893 $GFlops$
16	1.8216 s	17.9889 $GFlops$

2.2 循环展开的好处

通过将循环体内的迭代次数分成多个子迭代，使得每次迭代可以处理多个数据元素，从而减少了循环迭代次数，提高程序的性能：

1. 减少循环开销：循环体内的控制语句（比如循环的判断、加减等操作）会产生开销，循环展开可以减少这些开销，因为一个循环迭代可以处理多个数据元素。
2. 提高指令级并行性：循环展开使得循环体内的代码量增大，可以更好地利用现代处理器的流水线结构，提高指令级并行性，从而加速程序的执行。
3. 增加局部性：循环展开可以减少对内存的访问次数，从而增加局部性，减少缓存的失效率，提高程序的效率。
4. 便于向量化优化：循环展开可以将循环迭代次数分成多个子迭代，使得每个子迭代可以处理多个数据元素，更容易实现向量化优化，从而进一步提高程序的性能。