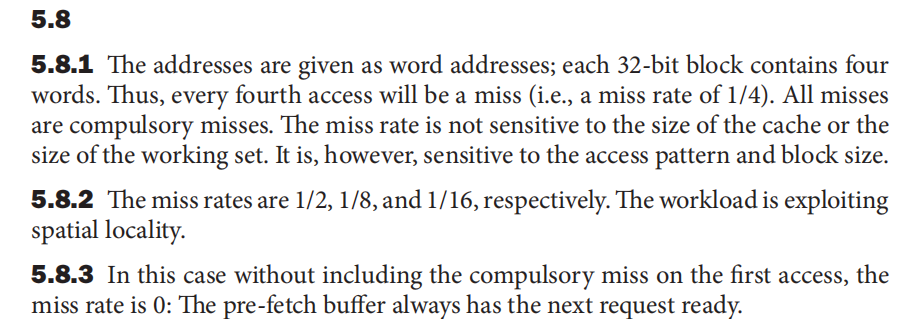
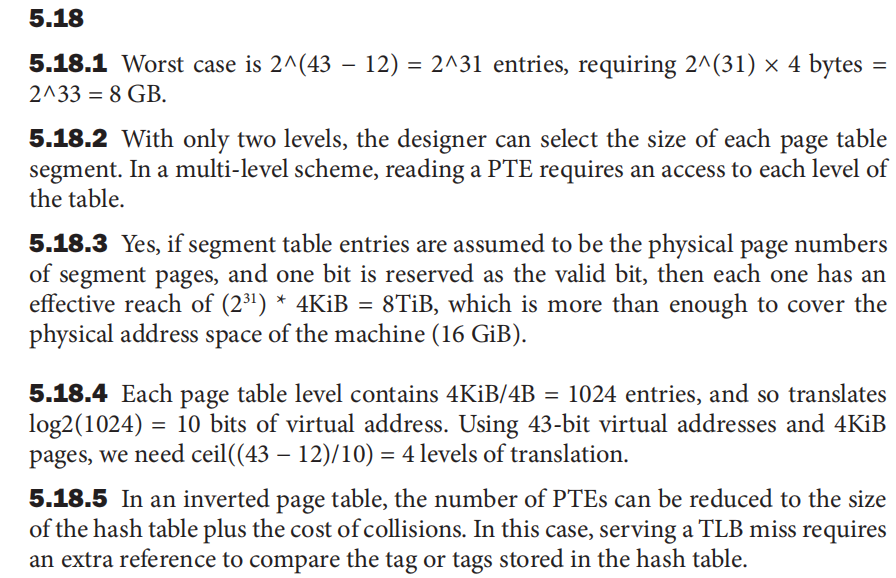
计算机组成原理第十次作业答案

5.8、518、5.23、5.27、6.4、6.6

5.8



5.18



5.23

模拟不同的指令集架构（ISA）需要针对该ISA的API进行特定处理。每个ISA都具有特定的行为，这些行为会在指令执行、中断处理、陷阱进入内核模式等情况下发生，因此必须对其进行仿真。这可能需要执行更多的指令来模拟每个指令，比目标ISA原本所需的指令数量要多得多。这会导致性能大幅下降，并使得与外部设备的正常通信变得困难。

然而，如果能够动态地检查并优化被仿真的代码，那么仿真系统理论上可以运行得比在其原生ISA上更快。例如，如果底层机器的ISA包含一条单一指令，能够处理被仿系统多条指令的执行，那么理论上可以减少被执行的指令数量。这类似于近期英特尔处理器所采用的微操作融合技术，该技术允许通过较少的指令来处理多条指令。

英文原答案如下：

Emulating a diff erent ISA requires specifi c handling of that ISA’s API. Each

ISA has specifi c behaviors that will happen upon instruction execution, interrupts,

trapping to kernel mode, etc. that therefore must be emulated. Th is can require manymore instructions to be executed to emulate each instruction than was originally

necessary in the target ISA. Th is can cause a large performance degradation and

make it diffi cult to properly communicate with external devices. An emulated

system can potentially run faster than on its native ISA if the emulated code can

be dynamically examined and optimized. For example, if the underlying machine’s

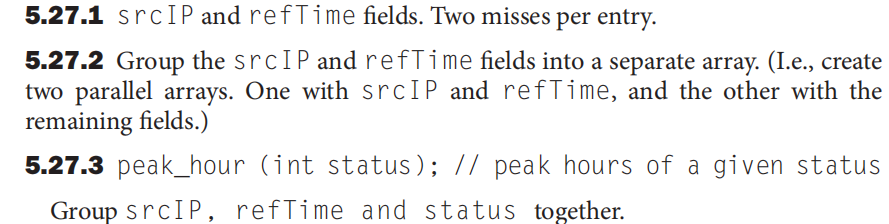
ISA has a single instruction that can handle the execution of several of the emulated

system’s instructions, then potentially the number of instructions executed can be

reduced. This is similar to the recent Intel processors that do micro-op fusion,

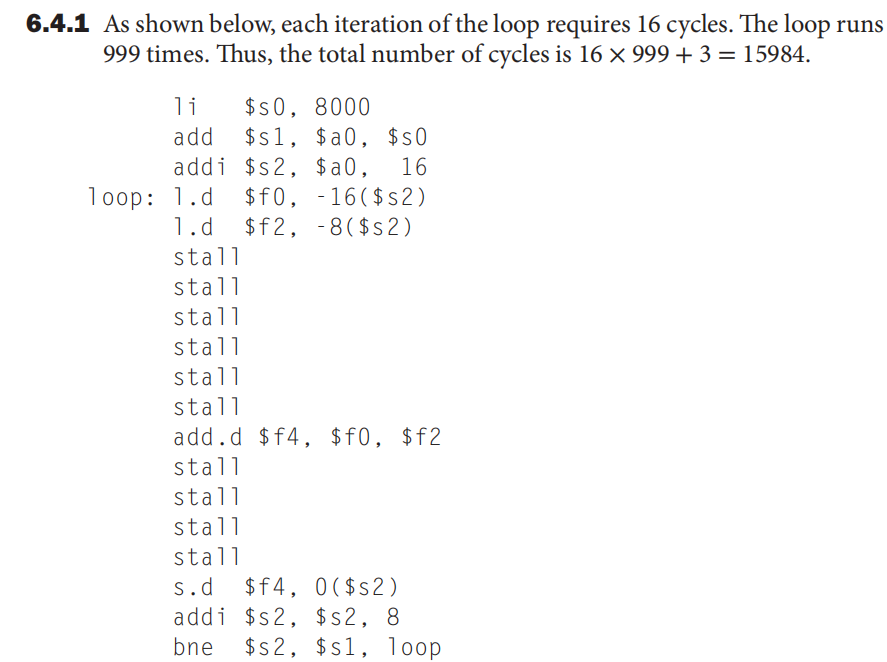
allowing several instructions to be handled by fewer instructions.

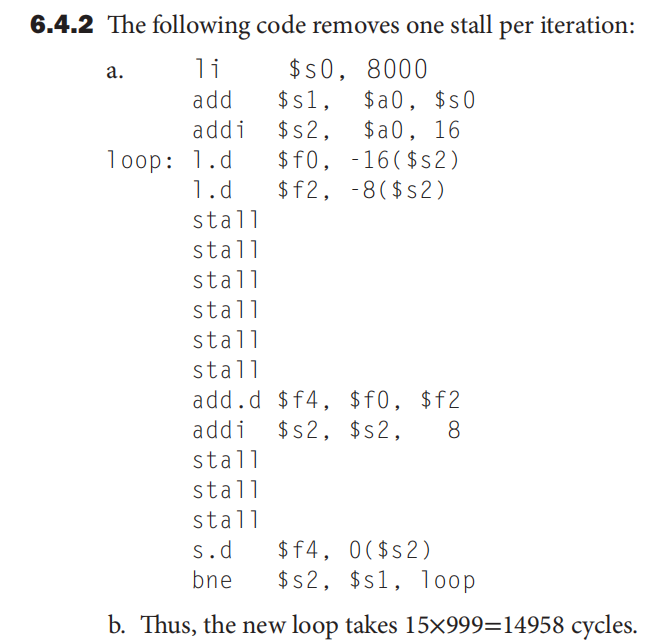
5.27

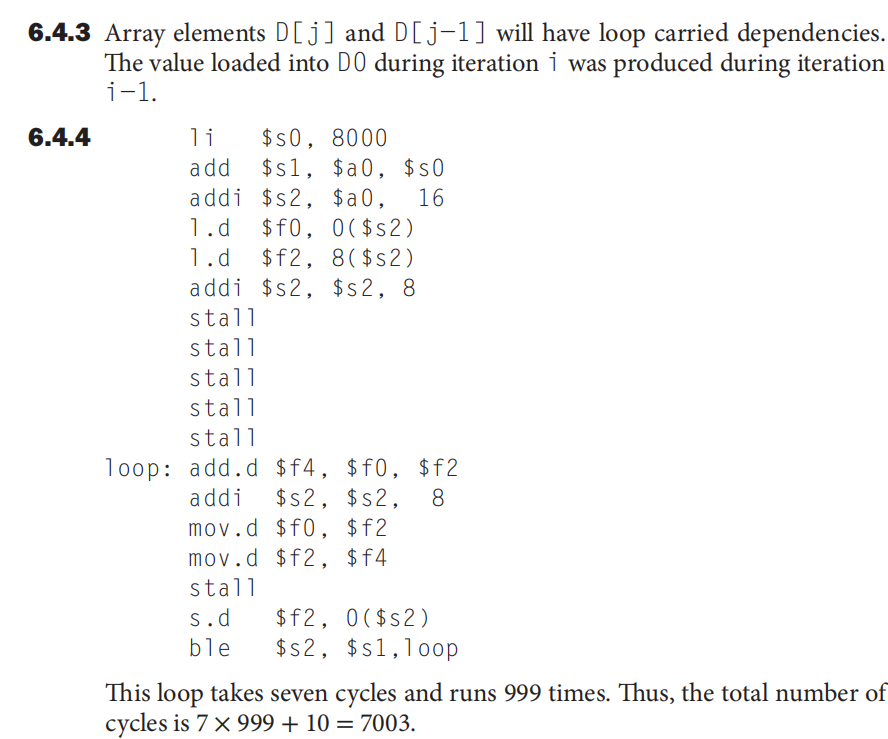


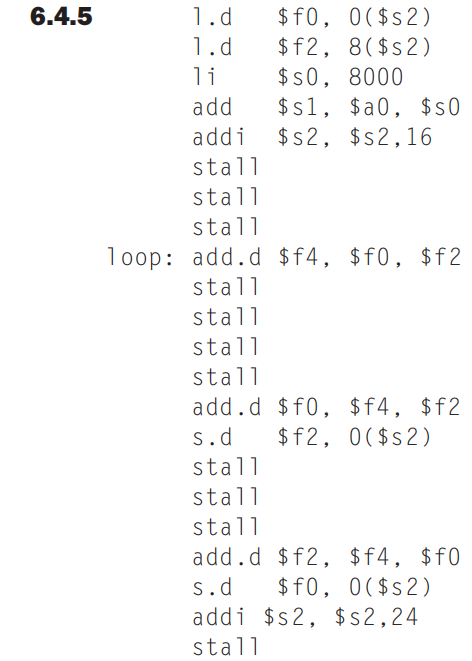
6.4

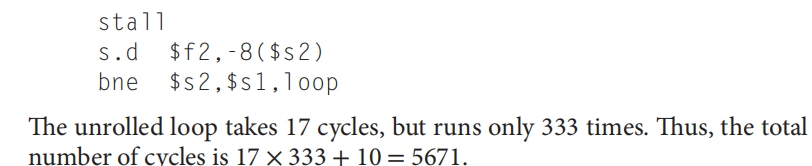
此题表明，如果串行代码经过重构，一些计算是可以并行进行的。但更重要的是，我们可能希望在我们的指令集架构（ISA）中提供SIMD（单指令多数据）操作，以便在对多个数据项执行相同操作时，能够利用数据级并行性。

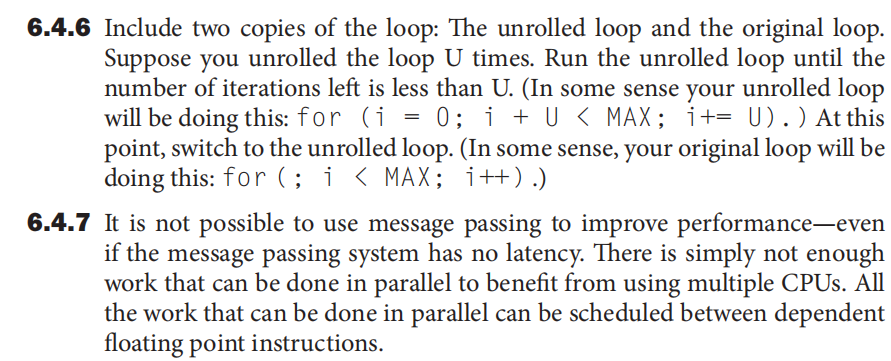












6.6

6.6.1

这个问题展示了一个“极度并行”的计算场景，并要求学生计算在四核系统上所能获得的加速比。所涉及的计算包括：（m × p × n）次乘法和（m × p × （n− 1））次加法。与矩阵C中单个元素相关的乘法和加法是相互依赖的（我们必须等到两个乘积都可用后，才能开始对一个元素的乘积求和）。所以在本问题中，理论上所能达到的加速比应该非常接近于4。

6.6.2

这个问题探讨了由于四个核心同时处理映射到同一缓存行的不同矩阵元素而导致的缓存未命中，是如何影响加速比的。每次更新都会带来缓存未命中的开销，因此会使得因服务缓存未命中的成本而降低的加速比进一步减小，其影响因子大约为未命中处理成本的3倍。

6.6.3

在这个问题中，我们被问及如何解决这一问题。解决伪共享问题最简便的方法是通过按列遍历而非按行遍历的方式来计算矩阵C中的元素（即，使用索引j而非索引i）。这样，这些元素将会被映射到不同的缓存行上。然后，我们只需要确保对计算出的矩阵索引（i, j）和（i + 1, j）在同一个核心上进行处理。这样做就可以消除伪共享的问题。

英文原答案：

