Для повышения производительности Pentium без внесения изменений в программный интерфейс рассматривались пять простейших способов.

Повышение тактовой частоты	порождает проблемы увеличения энергопотребления и перегрева микросхемы
Размещение на одной микросхеме двух процессоров	сопряжено с удвоением площади, занимаемой микросхемой. Если экономить на площади, то приходится использовать одну КЭШ-память на два процессора, что означает сокращение объема КЭШ-памяти на один процессор. Кроме того, возникает проблема распараллеливания вычислений
Введение новых функциональных блоков	в этом случае важно сохранить баланс. Нет смысла вводить большое число функциональных блоков, если скорость вызова команд ниже производительности блоков
Удлинение конвейера	с одной стороны повышает производительность, но с другой стороны увеличивает негативные последствия: ошибки прогнозирования ветвлений и промахи КЕШ—памяти. К тому же увеличение числа стадий требует увеличения тактовой частоты
Использование многопоточности	введение дополнительного программного потока, который позволяет использовать те аппаратные ресурсы, которые в противном случае простаивали бы

Основной принцип многопоточности – выполнение двух программных потоков (или процессов – процессор не отличает поток от процесса). Операционная система рассматривает гиперпоточный процессор как двухпроцессорный комплекс с общими КЭШами и основной памятью. Планирование для каждого потока операционная система выполняет отдельно.

Прикладные программы, предусматривающие возможность исполнения в виде нескольких программных потоков, могут задействовать оба виртуальных процессора.

Стратегии координации ресурсов:

Дублирование ресурсов	для каждого программного потока требуется свой счетчик команд. Вводится вторая таблица отображения архитектурных регистров (EAX, EBX и т.д.) на физические регистры, дублируется контроллер прерываний.
Жесткое разделение ресурсов	разделение слотов между потоками. Легко реализуется, обеспечивает полную независимость потоков. Один процессор фактически превращается в два. Недостатки: может сложиться ситуация, когда один из потоков не использует все ресурсы, а другому не хватает.
полное разделение ресурсов	доступ к ресурсам может получить любой программный поток, обслуживание осуществляется в порядке поступления запросов. Решает проблему неоптимального использования ресурсов, но создает проблему дисбаланса их использования. Недостатки: при наличии потоков с различными скоростями выполнения операций медленный поток выстроит очередь к конвейеру и полностью заполнит ее, а быстрый поток будет простаивать.
пороговое разделение ресурсов	Каждый поток получает требуемые ресурсы, но в допустимых пределах (например, 75%).

Жесткое разделение ресурсов не связано с большими издержками, а вот динамическое (особенно пороговое) требует отслеживания использования ресурсов на этапе исполнения. В некоторых случаях программы лучше работают без многопоточности, чем с ней (если для эффективной работы программы требуется более половины КЭШа. Тогда совместное выполнение этих двух программ приводит к постоянному обращению в память. Т.е. выполнение их последовательно было бы более эффективно).