Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Институт электронной техники и приборостроения

Кафедра Информационная безопасность автоматизированных систем

Направление 10.05.03 Информационная безопасность автоматизированных систем

**Практическая работа №2**

по дисциплине «Параллельные системы и их программирование»

по теме **«**Алгоритм поиска простых чисел»

|  |  |
| --- | --- |
|  | Выполнил: студент 4 курса  учебной группы с-ИБС42  очной формы обучения  Солодилов В.В.  Проверил: профессор. каф. ИБС  Кондратов Д.В. |

Саратов 2022

**Задание**

Реализовать последовательный и параллельные алгоритмы поиска простых чисел; выполнить анализ быстродействия алгоритмов при разном объеме данных, разном числе потоков; рассчитать ускорение и эффективность выполнения алгоритмов; сделать выводы о целесообразности применения параллельных алгоритмов и необходимости использования синхронизации.

**Вопросы**

1. Какими достоинствами и недостатками обладает каждый вариант распараллеливания?
2. Какие средства синхронизации можно использовать вместо конструкции lock? Какой вариант будет более эффективным?
3. Какой вариант ожидания завершения работ, запущенных пулом потоков, более эффективный и почему?

**Ответы на вопросы**

1. Все варианты распараллеливания наиболее эффективны только при большом количестве вычисляемых данных, так как при малом объеме элементов массива, накладываются расходы, связанные с организацией многопоточной обработки, которые существенно превышают выигрыш от параллельности обработки. Анализируя эффективность от распараллеливания задачи по частям и от параллельного выполнения цикла, можно сказать, что при первом варианте наибольшая эффективность достигается в случае, если число потоков будет совпадать с числом секций, а второй вариант – более универсальный, но сильно зависит от средств синхронизации потоков.
2. Помимо конструкции *lock* возможно использование конструкции *flush*, которая позволяет синхронизировать состояние памяти. Её выполнение предполагает следующие действия:

* значения всех переменных, временно хранящиеся в регистрах и кэш-памяти текущего потока, будут занесены в основную память;
* все изменения переменных, сделанные потоком во время работы, станут видимы остальным потокам;
* информация, хранящаяся в буферах вывода, будет удалена, т.е. буферы будут сброшены.

Выполнение данной директивы связано со значительными дополнительными расходами, и, если нужна гарантия согласованного представления не всех переменных, их можно явно перечислить в директиве списком. До полного завершения операции никакие действия с перечисленными в ней переменными не могут начаться. В связи с этим, эффективнее всего будет выполнение директивы *lock*.

1. Для ожидания завершения работ, запущенных пулом потоков, можно использовать конструкцию *barrier*. Данный вариант будет наиболее эффективным, поскольку возможно явно указать точку, где задача, запущенная пулом потоков, завершится.