Задание №1

1.1 Оценка максимальной производительности микропроцессора на заданных операциях.

Цель: оценить максимальную производительность микропроцессора на заданных операциях.

Постановка задачи
- Написать программу, выполняющую многократно (в цикле) заданную арифметическую операцию.
- Замерить время выполнения цикла. По результатам замера получить оценку производительности микропроцессора на заданной операции (в тактах процессора):
а) используя последовательность зависимых операций ("латентность"),
б) используя последовательность независимых операций ("темп выдачи результатов").

Для корректности теста необходимо добиться того, чтобы после компиляции внутри цикла не было обращений в память, и все вычисления проходили на регистрах. Полезно убедиться в этом путём анализа ассемблерного листинга. Кроме того, необходимо следить, чтобы компилятор не устранил из кода нужные операции. При необходимости можно использовать для тестирования модификацию ассемблерного листинга, сгенерированного компилятором.

Отчёт
В отчет необходимо включить:
- листинг программы на C/C++,
- по желанию весь листинг на ассемблере или фрагмент кода с циклом,
- полученные оценки производительности, их сравнение с теоретическими для случая без векторизации/оптимизации и для случая с векторизацией,
- вывод по результатам выполнения задания.

Варианты заданий (Номер в группе по модулю 8)
1) целочисленное умножение, векторное целочисленное умножение,
2) целочисленное деление, векторное целочисленное деление,
3) вещественное сложение, векторное вещественное сложение,
4) вещественное умножение, векторное вещественное умножение,
5) вещественное деление, векторное вещественное деление,
6) вещественное вычисление квадратного корня, векторное вещественное вычисление квадратного корня,
7) преобразование из целочисленного значения в вещественное (плюс вещественное сложение), векторное преобразование из целочисленного значения в вещественное (плюс векторное вещественное сложение).
8) Ваш выбор операции и его векторного выполнения

Для целочисленных операций использовать тип int, для вещественных - тип double. Векторные операции можно (желательно) реализовать с помощью intrinsics. Для замера времени можно использовать счётчик тактов процессора.

1.2 Освоение векторизации и распараллеливания программ на системе с общей памятью.

Цель: научиться выполнять оптимизацию (векторизация и распараллеливание на потоки) программ.

Постановка задачи
Оптимизировать заданную программу автоматически или полуавтоматически с помощью компилятора, а также при желании с помощью intrinsics. Сравнить времена работы следующих вариантов программы:
- исходная программа, без оптимизации
- исходная программа, оптимизированная только с помощью ключей компилятора,
- программа, векторизованная полуавтоматически (с помощью директив и ключей компилятора и незначительной правки кода),
- программа, векторизованная и распараллеленная полуавтоматически
...

Во всех случаях использовать ключи оптимизации, дающие наименьшее время работы программы, и наиболее позднее (эффективное) из доступных векторных расширений. По возможности обеспечить использование команд выровненного чтения и записи векторов. В программе с ручной векторизацией минимизировать количество обращений к памяти. На каждом этапе ручной оптимизации проверить, что время работы программы уменьшилось.

Отчёт
В отчет необходимо включить:
- Постановка задачи
- Описание используемой вычислительной системы и компилятора
- Результаты выполнения задания: описание выполненных оптимизаций и полученное время работы для указанных выше вариантов. Для варианта с ручной векторизацией или полуавтоматической привести времена расчёта, полученные на разных этапах оптимизации.
- Таблица результатов последовательных оптимизаций (векторизации, распараллеливания), сравнение с теоретическими результатами.
- График зависимости ускорения выполнения от числа потоков и сравнение с теоретической моделью (дополнительно)
- Обоснование класса задачи – compute-bound или memory-bound (дополнительно)
- Вывод по результатам выполнения задания
- Тексты программ в приложении

Варианты заданий (Номер в группе по модулю 3)
1) Интегрирование
2) Stencil,
3) Ваша программа
Во всех случаях использовать ключи оптимизации, дающие наименьшее время работы программы, и наиболее позднее (эффективное) из доступных векторное расширение. По возможности обеспечить использование команд выровненного чтения и записи векторов. В программе с ручной векторизацией минимизировать количество обращений к памяти. На каждом этапе ручной оптимизации проверить, что время работы программы уменьшилось.