

## Лабораторная работа № 2

Тема: Оценка производительности процессора.

Одним из основных показателей эффективности ЭВМ является производительность [1]. Под производительностью ЭВМ понимается ее способность обрабатывать информацию. Подробнее о качественных и количественных показателях технико-экономического функционирования вычислительной техники можно почитать в книге В.Г. Хорошевского «Архитектура вычислительных систем» [1].

Производительность процессора не может быть оценена его тактовой частотой. Развитие микроархитектуры процессора заключается в увеличении числа инструкций, выполняемых за один такт. Таким образом, производительность процессора может быть оценена числом инструкций, выполненных в единицу времени.

Теоретически можно оценить номинальное (или пиковое, или техническое) быстродействие (Nominal или Peak Speed) [1]. При наличии спецификации набора инструкций процессора, например тут [2], можно сделать выборку инструкций по работе только с оперативной памятью  $\{ins_1, ins_2, \dots, ins_i, \dots, ins_k\}$ . Для каждой инструкции известно время выполнения в тактах. Умножив длительность одного такта для конкретного процессора на число тактов, необходимых для выполнения инструкции  $ins_i$ , можно получить  $t_i$  – время выполнения инструкции  $ins_i$  в секундах. Предположив равновероятный выбор каждой инструкции, т.е.  $1/k$ , можно найти среднее время выполнения одной инструкции:  $\frac{\sum_i^k t_i}{k}$ . Номинальное быстродействие является обратной величиной среднему времени выполнения одной инструкции  $k(\sum_i^k t_i)^{-1}$ . Для нахождения более близких к реальности оценок можно применять вероятность использования каждой инструкции, например, смеси Гибсона. Быстродействие по Гибсону оценивается по формуле:  $(\sum_i^k t_i p_i)^{-1}$ , где  $p_i$  – вероятность использования инструкции  $ins_i$ . Распределение вероятностей  $\{p_i\}$ ,  $i = 1, k$ ,  $\sum_1^k p_i = 1$  зависит от характера конкретной задачи.

На практике при решении вычислительных задач используется весь набор инструкций процессора и учитывается множество факторов. Производительность оценивается специальными программами (benchmarks). Принцип их работы заключается в оценке времени выполнения набора эталонных тестовых задач близких к сфере применения вычислительного средства.

Пусть  $\{I_1, I_2, \dots, I_i, \dots, I_L\}$  – набор типовых задач (эталонных тестовых программ). Пусть также  $v_i$  – число операций, непосредственно входящих в программу решения задачи  $I_i$ ;  $t_i$  – время (в секундах) решения задачи  $I_i$  (в  $t_i$  входят время, расходуемое ЭВМ собственно на счет, и дополнительные затраты машинного времени при решении  $I_i$ ); быстродействием ЭВМ при решении типовой задачи  $I_i$ ,  $i \in \{1, 2, \dots, L\}$ , назовем  $\omega_i = v_i / t_i$ . Величина  $1/\omega_i$  является средним временем выполнения одной операции при решении задачи типа  $i$ ,  $i \in \{1, 2, \dots, L\}$ . Пусть  $\{\pi_1, \pi_2, \dots, \pi_i, \dots, \pi_L\}$  – распределение вероятностей спроса на типовые задачи  $I_i$ ,  $i = \overline{1, L}$ ,  $\sum_{i=1}^L \pi_i = 1$ , тогда  $\sum_{i=1}^L \pi_i / \omega_i$  будет средним временем выполнения одной операции при решении набора типовых задач. Тогда средним быстродействием будет величина  $\omega = \left( \sum_{i=1}^L \pi_i / \omega_i \right)^{-1}$ .

**Задание:** Реализовать программу для оценки производительности процессора (benchmark).

1. Написать программу(ы) (benchmark) на языке C/C++/C# для оценки производительности процессора. В качестве набора типовых задач использовать либо минимум 3 функции выполняющих математические вычисления, либо одну функцию по работе с матрицами и векторами данных с несколькими типами данных. Можно использовать готовые функции из математической библиотеки (math.h) [3], библиотеки BLAS [4] (англ. Basic Linear Algebra

Subprograms — базовые подпрограммы линейной алгебры) и/или библиотеки LAPACK [5] (Linear Algebra PACKage). Обеспечить возможность подать на вход программы общее число испытаний для каждой типовой задачи (минимум 10). Входные данные для типовой задачи сгенерировать случайным образом.

2. С помощью системного таймера (библиотека time.h, функции clock() или gettimeofday()) или с помощью процессорного регистра счетчика TSC реализовать оценку в секундах среднего времени испытания каждой типовой задачи. Оценить точность и погрешность (абсолютную и относительную) измерения времени (рассчитать дисперсию и среднеквадратическое отклонение).
3. Результаты испытаний в самой программе (или с помощью скрипта) сохранить в файл в формате CSV со следующей структурой:  
[PModel;Task;OpType;Opt;LNum;InsCount;Timer;AvTime;AbsErr;RelErr;TaskPerf], где

PModel – Processor Model, модель процессора, на котором проводятся испытания;

Task – название выбранной типовой задачи (например, sin, log, saxpy, dgemv, sgemm и др.);

OpType – Operand Type, тип операндов используемых при вычислениях типовой задачи;

Opt – Optimisations, используемые ключи оптимизации (None, O1, O2 и др.);

LNum – Launch Numer, число испытаний типовой задачи.

InsCount – Instruction Count, оценка числа инструкций при выполнении типовой задачи;

AvTime – Average Time, среднее время выполнения типовой задачи в секундах;

AbsError – Absolute Error, абсолютная погрешность измерения времени в секундах;

RelError – Relative Error, относительная погрешность измерения времени в %;

TaskPerf – Task Performance, производительность (быстродействие) процессора при выполнении типовой задачи.

3. \* Оценить среднее время испытания каждой типовой задачи с разным типом входных данных (целочисленные, с одинарной и двойной точностью).
3. \*\* Оценить среднее время испытания каждой типовой задачи с оптимизирующими преобразования исходного кода компилятором (ключи –O1, O2, O3 и др.).
3. \*\*\* Оценить и постараться минимизировать накладные расходы (время на вызов функций, влияние загрузки системы и т.п.) при испытании, то есть добиться максимальной точности измерений.
4. Построить сводную диаграмму производительности в зависимости от задач и выбранных исходных параметров испытаний. Оценить среднее быстродействие (производительность) для равновероятного использования типовых задач.

#### Литература.

1. *Хорошевский В.Г.* Архитектура вычислительных систем. М.: МГТУ им. Баумана, 2008, 520 с.
2. <https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html#architecture>
3. <https://www.opennet.ru/docs/RUS/mlib/>
4. <https://www.openblas.net/>
5. <https://www.nag.com/content/lapack-example-programs>