УДК 004.942

Практическое применение методов теоретической оценки вычислительной способности для процессоров Intel серии P5

А. А. Ракитский[1]

В работе Рябко Б.Я. «An information-theoretic approach to estimate the capacity of processing units» предложен новый подход к определению вычислительной способности компьютеров и им подобных устройств (кластеров, мобильных телефонов и т.п.). В статье этот метод используется для определения вычислительной способности компьютеров на базе процессоров семейства Intel, а также для сравнительного анализа влияния различных характеристик компьютеров на эту величину.

Ключевые слова: производительность, вычислительная способность.

1. Введение

На данный момент существует множество подходов для оценки и сравнения производительности вычислительных систем. В основном, все подходы сводятся к изучению ресурсов (время выполнения задачи, объём требуемой памяти, длина программы и т.д.), используемых для решения задачи. В настоящее время вычислительная способность компьютера оценивается эмпирически. Для этого на нём запускают набор тестовых вычислительных задач, так называемых бенчмарков, суммарное время выполнения которых на компьютере и является оценкой его вычислительной способности.

Очевидным недостатком этих оценок является то, что такое сравнение позволяет лишь оценить, как компьютер будет справляться с конкретной узконаправленной задачей. Важным фактом является и то, что задач, для выполнения которых предназначены вычислительные системы, бесконечно много и, следовательно, такая оценка не будет объективной и не может в полной мере оценить вычислительную способность компьютера. Для более общей оценки, как правило, используют одновременно несколько различных бенчмарков и выполняют сравнение, учитывая время выполнения каждого из них. Однако все эти оценки являются экспериментальными и требуют наличия конкретного оборудования для осуществления замеров времени выполнения.

Намного слабее изучены возможности теоретической оценки вычислительной способности вычислительных систем (компьютеров, мобильных телефонов, кластеров и т.д.). В работе [1] описан новый теоретический подход для определения вычислительной способности реальных компьютеров, опирающийся на такие их характеристики, как тактовая частота процессора, количество ядер процессора, организация памяти и набор инструкций процессора. Для оценки вычислительной способности компьютера предлагается использовать набор команд процессора и время их выполнения, включая временны́е задержки при обращении к разным видам памяти (кэш-памяти, оперативной памяти и т.д.), а также задержки, связан-

[1] Работа выполнена в рамках гранта по госконтракту № 8229 от 6 августа 2012 года.

ные с перезагрузкой конвейера и сменой контекста процессора. В основе данного подхода лежит концепция энтропии Шеннона, ёмкость дискретного канала без шума и другие идеи Шеннона, которые включает в себя теория информации.

Модель, описанная в работе [1], представляет большой практический интерес не только для теоретической оценки производительности уже существующих и используемых вычислительных систем, но также для расчёта и анализа вычислительной способности систем, находящихся на стадии проектирования, для которых не могут быть применены эмпирические методы оценки.

В работе ставится цель проверить указанную выше модель на практике и сравнить теоретические результаты, полученные при её использовании, с уже полученными и опубликованными ранее эмпирическими данными о производительности компьютеров. Для проверки модели были выбраны компьютеры с процессорами Intel, начиная с Intel 80486 и заканчивая Intel Pentium MMX, для которых известно полное описание и известны эмпирические данные об их производительности.

2. Вычислительная способность компьютера

2.1. Основные идеи и определения

В работе [1] описана упрощённая модель компьютера, согласно которой он состоит из набора инструкций процессора *I* и доступной памяти *M*. Каждая отдельная инструкция процессора $x \in I$ – это совокупность имени команды и операндов этой команды, таких как индексы регистров и адреса ячеек памяти. Под инструкцией подразумевается не только имя команды, но и значение операндов этой команды; следовательно, инструкции, имеющие одинаковые имена команд, но разные значения операндов, будут входить в *I* как разные.

Вычислительная задача *P* определяется некоторой последовательностью инструкций $X(P) = x_1\, x_2\, x_3 \ldots x_n$, где $x_i \in I$. Например, если задача *P* содержит цикл, который выполнится десять раз, последовательность *X* будет содержать тело цикла, повторяющееся десять раз. Не все последовательности инструкций являются допустимыми, могут быть запрещённые последовательности. Вполне вероятно существование недопустимых пар последовательных инструкций; другими словами, последовательность, возможно, должна удовлетворять некоторым ограничениям. Определим набор из всех допустимых последовательностей инструкций как S_c и будем рассматривать две разные последовательности из S_c как две разные вычислительные задачи.

Время выполнения инструкции x обозначается как $\tau(X)$. Тогда время выполнения $\tau(X)$ последовательности инструкций $X(P) = x_1\, x_2\, x_3 \ldots x_t$ определяется как

$$\tau(X) = \sum_{i=1}^{t} \tau(x_i).$$

Главное наблюдение заключается в том, что число разных вычислительных задач, чьё время выполнения равно *T,* эквивалентно размеру множества всех допустимых последовательностей инструкций, чьё время выполнения равно *T,* т.е.

$$\nu(T) = N(T), \tag{1}$$

где $\nu(T)$ – количество разных задач, чьё время выполнения равно *T,* и

$$N(T) = \left|\left\{X : \tau(X) = T\right\}\right|. \tag{2}$$

Поэтому

$$\log v(T) = \log N(T). \quad (3)$$

(Здесь и ниже T – это целое число, $\log x \equiv \log_2 x$ и $|Y|$ – это количество элементов Y, если Y – это множество, и длина, если Y – это слово.) Другими словами, общее количество задач, выполняющихся за время T, даётся выражением (2). Исходя из этих соображений даётся следующее определение.

Определение 1. Пусть есть компьютер с набором инструкций I и пусть $\tau(x)$ – время выполнения для инструкции $x \in I$. Вычислительная способность $C(I)$ определяется как

$$C(I) = \lim_{T \to \infty} \frac{\log N(T)}{T}. \quad (4)$$

Утверждение 1. Предел (4) существует, если I – конечное множество, времена выполнения $\tau(x)$, $x \in I$, – целые числа и наибольший общий делитель $\tau(x)$ равен 1.

Доказательство утверждения приводится в работе [1].

2.2. Метод для оценки вычислительной способности

Простейшая оценка вычислительной способности может быть получена, если предположить, что все последовательности инструкций являются допустимыми. Иными словами, рассмотрим набор инструкций I как алфавит и предположим, что все последовательности букв (инструкций) могут быть выполнены. В таком случае может быть использован метод вычисления пропускной способности канала без потерь, предложенный Шенноном в [2]. Важно отметить, что этот метод может быть использован для нахождения верхней границы вычислительной способности для всех других моделей, потому что для любого компьютера множество допустимых последовательностей инструкций – это подмножество всех слов над «алфавитом» I.

Рассмотрим компьютер с набором инструкций I, время выполнения которых $\tau(x)$, $x \in I$, и все последовательности инструкций являются разрешёнными. Другими словами, если набор I рассматривается как алфавит, то все возможные слова над этим алфавитом можно рассматривать в качестве допустимых последовательностей инструкций для компьютера. В этом случае оценить вычислительную способность (4) можно при помощи метода, предложенного Шенноном, который показал, что $C(I)$ равна логарифму от наибольшего действительного решения X_0 следующего уравнения:

$$X^{-\tau(x_1)} + X^{-\tau(x_2)} + ... + X^{-\tau(x_S)} = 1, \quad (5)$$

где $I = \{x_1, ..., x_S\}$. Иными словами, $C(I) = \log X_0$.

2.3. Оценка вычислительной способности многоядерных процессоров

Давайте рассмотрим многоядерный процессор (описание такого процессора можно найти в [11]). Для начала рассмотрим самый главный вопрос. Пусть у нас есть многоядерный процессор с l ($l \geq 1$), ядрами. Следующее утверждение показывает, что вычислительная способность данного процессора будет равна сумме вычислительных способностей соответствующих одноядерных процессоров.

Утверждение 1. *Пусть у нас есть многоядерный процессор с l ($l \geq 1$), ядрами $I_1, ..., I_l$, такой, что каждое ядро может быть использовано как индивидуальный процессор, независимо от других ядер, т.е. любая инструкция j-го ядра I_j может быть использована вне за-*

висимости от того, какие инструкции выполняются на других ядрах. Тогда вычислительная способность данного многоядерного процессора будет равна сумме вычислительных способностей одноядерных процессоров, соответствующих индивидуальным ядрам $I_1,...,I_l$. *Другими словами, если мы рассмотрим процессор с одним ядром* I_j *(т.е. с таким же набором инструкций и всеми видами памяти, как и у оригинального многоядерного процессора), тогда*

$$C(\otimes_{j=1}^{l} I_j) = C(I_1) + C(I_2) + ... + C(I_l), \quad (6)$$

где $C(\otimes_{j=1}^{l} I_j)$ *– вычислительная способность многоядерного процессора,* $C(I_j)$ *– вычислительная способность одноядерного процессора с набором инструкций* I_j *и всеми видами памяти заданного многоядерного процессора.*

Доказательство этого утверждения представлено в [1].

3. Практическое применение метода

3.1 Пример расчета вычислительной способности абстрактного компьютера

Рассмотрим описанный выше метод на примере примитивного абстрактного компьютера, который может складывать и вычитать небольшие числа, а также обмениваться данными между регистрами и памятью. Пусть наш компьютер будет иметь четыре регистра по 8 бит каждый (назовём их *A,B,C,D*). Память состоит из 1024 ячеек, каждая размером 8 бит. В таком случае набор команд компьютера будет следующим:

- put *addr* – команда копирует значение из регистра *A* в ячейку памяти по адресу *addr*, передаваемому ей в качестве операнда;
- get *addr* – команда копирует значение из ячейки памяти по адресу *addr*, передаваемому ей в качестве операнда, в регистр *A*;
- mov *reg1*, *reg2* – команда перемещает значение из регистра *reg2* в регистр *reg1*, регистры *reg1* и *reg2* передаются команде в качестве операндов (команда mov *A*, *A* и аналогичные команды с другими регистрами являются допустимыми);
- add *reg* – команда складывает регистр *A* с регистром *reg*, переданным в качестве операнда, и заносит результат сложения в регистр *A* (команда add *A* является допустимой);
- sub *reg* − команда вычитает из регистра *A* регистр *reg*, переданный в качестве операнда, и заносит результат вычитания в регистр *A* (команда sub *A* является допустимой);
- clr – команда, сбрасывающая значение регистра *A* в ноль и не имеющая операндов.

Зададим следующее время выполнения этих команд: put – 14 тактов, get – 10 тактов, mov и clr – 1 такт, add – 2 такта, sub – 3 такта.

С помощью данного набора инструкций, можно составить любое количество последовательностей инструкций, причём каждая последовательность будет корректна и допустима.

Когда мы задали набор инструкций и знаем время выполнения каждой из них, мы можем перейти к составлению уравнения (5) для определения вычислительной способности описанного процессора. Запишем слагаемые для каждой инструкции по отдельности. Рассмотрим команду put: у этой команды всего один операнд, указывающий, в какую ячейку памяти поместить содержимое регистра *A*, всего у нас есть 1024 ячейки памяти; следовательно, можно записать ровно 1024 различных вариантов команды put или 1024 инструкции, у которых имя

команды совпадает. Тогда слагаемое для команды put запишем как $1024\times X^{-14}$. Здесь 1024 – это количество всех инструкций с именем команды put и в степени записано время выполнения команды со знаком минус.

Команда get также имеет один операнд, который указывает, из какой ячейки памяти загружать значение в регистр *A*. По аналогии слагаемое для команды get будет иметь вид $1024\times X^{-10}$.

Команда mov имеет два операнда, каждый из которых может быть одним из четырёх регистров. Следовательно, всего разных вариантов инструкции может быть $4\times 4=16$. Таким образом, получаем, что слагаемое будет иметь вид $16\times X^{-1}$.

Команда add имеет один операнд, которым является один из четырёх регистров. Таким образом, различных инструкций с именем команды add будет ровно 4. Время выполнения команды – 2 такта, значит, слагаемое равно $4\times X^{-2}$.

Команда sub аналогична команде add, однако время её выполнения равно трём тактам; значит, слагаемое будет $4\times X^{-3}$.

Команда clr не имеет операндов; это значит, что есть только одна инструкция с именем команды clr и временем выполнения – 1 такт. Слагаемое для команды clr будет следующим: X^{-1}.

Теперь составим уравнение (5) для описанного компьютера из рассмотренных слагаемых. Записывать слагаемые будем в порядке их описания:

$$1024\times X^{-14}+1024\times X^{-10}+16\times X^{-1}+4\times X^{-2}+4\times X^{-3}+X^{-1}=1,$$

здесь первое слагаемое описывает команду put, второе – команду get, третье – mov, четвёртое – add, пятое – sub, шестое – clr.

Решение этого уравнения: $X_0\approx 17.24$.

Тогда вычислительная способность рассматриваемого компьютера будет равна $C(I)=\log_2 X_0\approx\log_2 17.24\approx 4.11$ бит/такт.

Далее рассмотрим применение метода к реальным компьютерам с процессорами фирмы Intel.

3.2 Оценка производительности компьютера на базе процессора Intel 80286

В первую очередь, мы рассмотрим компьютер на базе процессора Intel 80286 исходя из того факта, что все последующие рассматриваемые в этой статье процессоры в некоторой степени основываются на нём. Этот процессор имеет значительно более простую структуру, чем последующие, поэтому на нём мы остановимся более подробно и тщательно рассмотрим процесс составления уравнения (5). Во-первых, Intel 80286 – это 16-битный x86-совместимый микропроцессор второго поколения фирмы Intel. Дата выпуска: 1 февраля 1982 года.

Для расчёта вычислительной способности компьютера необходим список команд процессора и время их выполнения. Найти полный перечень команд можно в [3]. Формат команд архитектуры x86 подробно описан в [4]. Для построения уравнения необходимо подробнее рассмотреть команды, которые будут участвовать в нём.

Во-первых, введём в уравнение слагаемые для команд без операндов и разобьём их на группы по времени выполнения: (CBW, CLC, CLD, CMC, CWD, LAHF, SAHF, STC, STD) – время выполнения 2 такта; (AAA, AAS, DAA, DAS, PUSHF) – время выполнения 3 такта; (POPF) – выполняется за 5 тактов; (AAD) – 14 тактов; (AAM) – 16 тактов; (POPA, PUSHA) – 19 тактов процессорного времени.

Дальше у нас идут команды с одним операндом – регистром. Тут мы распишем группы более подробно: команды (DEC, INC, NEG, NOT, RCL, RCR, ROL, ROR, SAL, SAR, SHL, SHR), которые используют регистры (AX, CX, DX, BX) и выполняются за 2 такта; команда

(PUSH), которая использует регистры (AX, CX, DX, BX, SP, BP, SI, DI) и выполняется за 3 такта; команда (POP), которая использует регистры (AX, CX, DX, BX, SP, BP, SI, DI) и выполняется за 5 тактов; команды (IMUL, MUL), которые используют регистры (AX, CX, DX, BX) и выполняются за 21 такт; команда (DIV), которая использует регистры (AX, CX, DX, BX) и выполняется за 22 такта; команда (IDIV), которая использует регистры (AX, CX, DX, BX) и выполняется за 25 тактов.

Третьей группой слагаемых пойдут команды, операндом у которых выступает память. В этой группе мы разобьём команды на две подгруппы: команды, которые адресуют ячейки памяти одного сегмента (64 Кбайт или 2^{15} слов), и команды, которые адресуют всю доступную память (1 Мбайт или 2^{19} слов), но для этого необходима смена значения сегментного регистра командой MOV, так как в Intel 80286 использовалась сегментная модель памяти. Первая подгруппа: команды (POP, PUSH), выполняющиеся за 5 тактов; команды (DEC, INC, NEG, NOT, RCL, RCR, ROL, ROR, SAL, SAR, SHL, SHR), выполняющиеся за 7 тактов; команды (IMUL, MUL), выполняющиеся за 24 тактов; команда (DIV), выполняющаяся за 25 тактов; команда (IDIV), выполняющаяся за 30 тактов. Во второй подгруппе к времени выполнения команд, описанных в первой подгруппе, добавляется 2 такта, за которые выполняется команда MOV, меняющая значение сегментного регистра.

Четвёртая группа слагаемых состоит из команд, которые имеют два операнда регистра. Перечислим эти команды: (ADC, ADD, AND, CMP , OR, SBB, SUB, TEST, XOR), которые выполняются за 2 такта и в качестве операндов им могут быть переданы регистры (AX, CX, DX, BX); команда (MOV), которая выполняется за 2 такта и её операнды – регистры (AX, CX, DX, BX, SP, BP, SI, DI); команда (XCHG), её время 3 такта и операнды – (AX, CX, DX, BX, SP, BP, SI, DI).

Пятой группой идут команды, которые имеют два операнда: регистр-память или память-регистр. На первом месте всегда стоит операнд, в который сохраняется результат операции. Например, команды ADD reg, mem и ADD mem, reg будут выполнять одно и то же действие, но в первом случае результат будет сохранен в регистр, а во втором в память. Поэтому такие команды рассматриваются как разные инструкции процессора.

Сначала рассмотрим команды, у которых на первом месте стоит операнд-регистр: команды (XCHG, MOV), у которых первый операнд – это регистр (AX, CX, DX, BX, SP, BP, SI, DI), а второй – это один сегмент памяти (2^{15} ячеек памяти); команды (CMP, TEST), у которых первый операнд – регистр (AX, CX, DX, BX), а второй операнд – один сегмент памяти; команды (ADC, ADD, AND, OR, SBB, SUB, XOR), где первый операнд – регистр (AX, CX, DX, BX), а второй – один сегмент памяти; команды (LDS, LES, LEA), у которых первый операнд – регистр (AX, CX, DX, BX, SP, BP, SI, DI), а второй – один сегмент памяти. К этой же группе отнесём те же самые команды, но с добавочной командой MOV для смены сегментного регистра. Эта команда ко времени всех команд добавит 2 такта.

Теперь рассмотрим команды, у которых на первом месте стоит операнд-память: команда (MOV), время выполнения которой 3 такта, первый операнд – один сегмент памяти, а второй – один из регистров (AX, CX, DX, BX, SP, BP, SI, DI); команда (XCHG), у которой время выполнения – 5 тактов, первый операнд – это сегмент памяти, а второй – один из регистров (AX, CX, DX, BX, SP, BP, SI, DI); команда (TEST), где время выполнения команды – 6 тактов, первый операнд – сегмент памяти и второй – регистр (AX, CX, DX, BX); команды (ADC, ADD, AND, CMP, OR, SBB, SUB, XOR), у которых время выполнения равно 7 тактов, первый операнд – сегмент памяти, а второй – регистр (AX, CX, DX, BX). И опять же необходимо добавить к этой группе все эти команды с добавленными 2 тактами на смену сегментного регистра.

Шестая группа состоит из команд, которые работают со строками. Строка – это набор ячеек, которые располагаются в памяти последовательно. Работать со строками могут команды (CMPS, LODS, MOVS, SCAS, STOS), каждая из этих команд выполняет соответствующую операцию над двумя операндами, в зависимости от команды операндом может быть

либо элемент строки, либо регистр. Для того чтобы происходило циклическое выполнение команды над всеми элементами строки, необходимо использовать команду REP. Использовать команду REP отдельно от строковых операций не имеет смысла, поэтому время выполнения команды REP – 2 такта – было прибавлено ко времени выполнения команд, работающих со строками. Перечислим слагаемые из этой группы: команда CMPS, которая выполняется за 8 тактов и имеет два операнда: первый и второй – строки; команда LODS, время выполнения – 5 тактов, первый операнд – регистр AX, второй – строка; команда MOVS, время выполнения – 5 тактов, оба операнда – строки; команда SCAS, выполнение которой занимает 7 тактов, первый операнд – регистр AX, второй – строка; команда STOS, выполняющейся за 3 такта, первый операнд – строка, второй – регистр AX. К этой же группе относятся описанные выше команды, но с прибавленными ко времени выполнения 2 тактами, необходимыми для смены сегментного регистра.

К последней группе относятся команды условных и безусловных переходов, а также команды циклов. Ход выполнения команд условного перехода может развиваться по двум сценариям. Первый – если условие перехода выполнилось, тогда очередь выполнения команд сбивается и требуется загрузить команду, на которую произошёл переход; время загрузки команды зависит от её размера, то есть ко времени выполнения команды условного перехода (7 тактов) необходимо прибавить время, требующееся на загрузку команды (от 1 до 4 тактов). Всего команд условного перехода 32 и одна команда безусловного перехода JMP. Второй – если условие перехода не выполнилось, тогда выполняется следом идущая команда из очереди, в этом случае время выполнения команд условного перехода равно 3 тактам. И необходимо ещё рассмотреть аналогичные случаи для команд циклов, которых в Intel 80286 пять. Если условие цикла выполняется, то время команды будет 8 тактов плюс от 1 до 4 тактов на загрузку команды, в противном случае – 4 такта.

Далее перечислим команды, которые не использовались при составлении уравнения, их три типа. Первый из них – это команды вызова процедур и прерываний (BOUND, CALL, ENTER, INT, INTO, IRET, LEAVE, RET). Так как метод направлен на оценку верхнего порога вычислительной способности (пиковую производительность системы), то не учитывать эти команды является вполне логичным. Задача процедур заключается в минимизации размера кода программы и его структуризации, что плохо сказывается на скорости выполнения этого кода. Второй – это системные команды (ARPL, CLTS, ESC, HLT, LAR, LGDT, LIDT, LLDT, LMSW, LOCK, LSL, LTR, SGDT, SIDT, SLDT, SMSW, STR, VERR, VERW, WAIT), используемые в основном операционной системой для управления процессором и не использующиеся в прикладных задачах. Третий – команды ввода/вывода с портов (IN, INS, OUT, OUTS). Время выполнения этих команд зависит от множества факторов, связанных с оборудованием, установленным на вычислительную систему, и поэтому определить его затруднительно, но известно, что оно будет значительно больше времени выполнения арифметических и им подобных команд процессора, поэтому их вклад в оценку вычислительной способности компьютера будет пренебрежимо мал. Получив в итоге характеристическое уравнение и решив его, мы получим $X_0 \approx 65.49$ $C(I) \approx \log_2 65.49 \approx 6$ бит/такт. Частота выбранного процессора равна 12.5 МГц; следовательно $C_{i286}(I) \approx 75$ Мбит/сек.

3.3 Оценка производительности компьютера на базе процессора Intel 80486

Основное отличие процессора 80486 от 80286 заключается в том, что на этом процессоре впервые появился конвейер и два уровня кэш-памяти.

Кэш-память первого уровня располагалась на кристалле процессора и была объёмом сначала 8 Кбайт, а в дальнейшем 16 Кбайт, работала она на частоте ядра процессора. Процессор также имел кэш-память второго уровня, которая хоть и имела больший размер, чем первая,

но время чтения-записи в неё было значительно больше. Наличие кэш-памяти позволило значительно ускорить доступ к часто используемым ячейкам оперативной памяти.

Начиная с Intel 80486 кэш-память присутствует во всех процессорах семейства Intel. В связи с наличием кэш-памяти необходимо разобрать её влияние на построение характеристического уравнения для определения вычислительной способности. Для всех последующих процессоров Intel, разобранных в этой статье, кэш-память будет учитываться аналогичным образом. В первую очередь необходимо разобраться с принципом работы двухуровневой кэш-памяти. Предположим, что процессор имеет доступ к кэш-памяти первого уровня L_1 за время τ_1, а к кэш-памяти второго уровня L_2 за время τ_2 и к оперативной памяти M за время τ_3. Будем считать, что при организации кэш-памяти соблюдаются следующие неравенства: $L_1 < L_2 < M$ и $\tau_1 < \tau_2 < \tau_3$, в противном случае преимущества от её использования не будет. Рассмотрим теперь ход выполнения команды INC mem, которая увеличивает на единицу значение ячейки памяти по адресу mem (при этом mem – это адрес ячейки в оперативной памяти компьютера). Кэш-память первого и второго уровня хранит только те ячейки, к которым процессор обращается наиболее часто, то есть такие ячейки продублированы в кэш-памяти и в оперативной памяти. Чтобы получить значение ячейки по адресу mem, компьютер в первую очередь обратится к кэш-памяти первого уровня. Если результат поиска будет отрицательным, тогда процессор обратится с этим же запросом к кэш-памяти второго уровня. Если же в кэш-памяти второго уровня ячейка с таким адресом не будет обнаружена, то процессор обратится уже к оперативной памяти и возьмет значение оттуда. Вполне очевидно, что в таком случае время поиска в кэш-памяти первого уровня будет равно τ_1, время поиска в кэш-памяти второго уровня будет $\tau_1 + \tau_2$, а время поиска данных в оперативной памяти будет соответственно равняться $\tau_1 + \tau_2 + \tau_3$. В таком случае слагаемое характеристического уравнения для команды INC мы запишем следующим образом:

$$L_1 \times X^{-\tau_1} + (L_2 - L_1) \times X^{-(\tau_1+\tau_2)} + (M - L_2) \times X^{-(\tau_1+\tau_2+\tau_3)},$$

здесь первое слагаемое описывает все инструкции, которые запросят и получат данные из кэш-памяти первого уровня, второе слагаемое – инструкции, которые обратятся к кэш-памяти второго уровня, и третье – инструкции, обращающиеся к оперативной памяти.

В Intel 80486 появился механизм, позволяющий значительно ускорить выполнение инструкций – так называемый конвейер. Он позволяет выполнять инструкцию в несколько этапов, тем самым значительно повышая производительность процессора. Конвейер состоит из пяти ступеней: выборка инструкции, декодирование инструкции, декодирование адресов операндов инструкции, выполнение команды и запись результата выполнения инструкции. Очевидно, что на каждой ступени конвейера может находиться по одной команде процессора; следовательно, в идеальном случае конвейер может одновременно обрабатывать пять совершенно разных команд. Именно это и позволило значительно увеличить производительность процессора за счёт снижения времени выполнения команд в потоке (т.е. за счёт повышения пропускной способности процессора).

С учётом наличия конвейера необходимо теперь определить, как рассчитывать время выполнения команды, необходимое для составления характеристического уравнения. В конвейерных системах принято использовать понятие пропускная способность для измерения скорости прохождения команд через конвейер. Пропускная способность – это количество инструкций, выполняемых процессором за 1 такт. Рассмотрим в качестве примера конвейер, на котором каждая инструкция выполняется на каждой ступени за один такт, тогда пропускная способность конвейера будет равна одной команде за такт, тогда как на самом деле команда будет выполняться столько тактов, сколько есть ступеней в конвейере. В реальности команда на каждой ступени конвейера будет выполняться разное время. Например, инструкция без операндов будет декодироваться быстрее, чем инструкция с двумя операндами; инструкция с операндом регистр будет выполняться быстрее, чем инструкция с операндом

память; и т.д. В таком случае, если на каком-то этапе команда выполняется достаточно долго, то на всех предыдущих ступенях уже выполнившиеся команды будут простаивать. Поэтому под временем выполнения команды в конвейерных системах подразумевается задержка, которая произошла в конвейере по вине этой команды.

Ещё одна особенность подобных систем – это сброс конвейера. Такая ситуация может произойти, если при выполнении операции ветвления был неправильно предсказан адрес условного перехода. В таком случае происходит очистка конвейера и команды начинают загружаться в него с нового адреса. Очевидно, что это влечёт большие задержки и простой конвейера. Для уменьшения количества сбросов в процессорах реализован алгоритм предсказания переходов, точность работы такого алгоритма составляет свыше 90%. Насыщенность исполняемой процессором последовательности команд условными переходами связана с типом решаемой задачи, квалификацией программиста, написавшего последовательность, и другими факторами. То есть могут существовать допустимые последовательности команд, не содержащие в себе инструкций условных переходов, так же как могут быть построены допустимые последовательности команд, состоящие практически из одних инструкций условных переходов. Так как нам необходимо произвести оценку верхней границы вычислительной способности, то мы будем рассматривать наилучший из вариантов и считать, что все переходы предсказываются верно.

В процессор Intel 80486 были добавлены следующие команды: INVD, INVLPG, WBINVD, BSWAP, CMPXCHG, CPUID.

В результате решения полученного характеристического уравнения получаем: $X_0 \approx 1734.13 \quad C(I) \approx \log_2 1734.13 \approx 10.76$ бит/такт. Частота выбранного процессора равна 66 МГц; следовательно, $C_{i486}(I) \approx 710$ Мбит/сек.

3.4 Оценка производительности компьютера на базе процессора Intel Pentium (P5)

Процессор Intel P5 был представлен 22 марта 1993 года. Его микроархитектура относится к пятому поколению процессоров Intel, и это был первый суперскалярный x86-совместимый процессор. В этом процессоре используются два параллельных конвейера для выполнения целочисленных операций и математический сопроцессор FPU для операций с плавающей запятой. Рассмотрим работу этих двух конвейеров. Назовем их U-конвейер и V-конвейер. При выполнении определённых условий возможно выполнение одновременно двух инструкций, и это может удвоить скорость работы процессора. Однако эти условия выполняются далеко не всегда.

Согласно условиям выполнения операций на U и V конвейерах, все целочисленные операции можно разбить на четыре группы:

- инструкции, которые могут быть распараллелены и выполняться на обоих конвейерах;
- инструкции, которые могут быть распараллелены, но могут выполняться только на U-конвейере;
- инструкции, которые могут выполняться на обоих конвейерах, но распараллеливаться могут, только если выполняются на V-конвейере;
- инструкции, которые не могут выполняться параллельно с другими и выполняются только на U-конвейере.

В данном случае под словом распараллеливаться мы понимаем возможность инструкции выполняться параллельно с другой инструкцией на втором конвейере. Согласно этому разбиению перечислим инструкции, которые относятся к той или иной группе. К первой группе относятся инструкции: MOV, PUSH, POP, LEA, NOP, INC, DEC, ADD, SUB, CMP, AND, OR, XOR и некоторые виды инструкции TEST. Ко второй группе относятся инструкции: ADC, SBB, (SHR, SAR, SHL, SAL с константным значением), ROR, ROL, RCR, RCL. В третьей

группе содержатся все операции близкого вызова, близкого и короткого перехода и быстрого и короткого условного перехода. Все остальные инструкции относятся к четвёртой группе.

Для того чтобы учесть эту особенность процессора при построении характеристического уравнения, мы произведём изменение списка инструкций. Давайте рассмотрим последовательность инструкций $X(P) = x_1 x_2 x_3 \ldots x_n$, $x_i \in I$. Мы знаем, что некоторые подряд идущие пары инструкций могут выполняться одновременно, поэтому эти пары инструкций можно рассматривать как индивидуальные инструкции.

Это означает, что наш набор инструкций *I* может быть преобразован в набор инструкций *I'*, который содержит не только одиночные инструкции из оригинального набора, но и пары инструкций, которые могут выполняться одновременно (первая – на U-конвейере, а вторая – на V-конвейере).

Теперь, когда список инструкций определён, мы легко можем найти наибольшее целочисленное решение X_0 характеристического уравнения (5) и определить вычислительную способность.

Отдельно рассмотрим инструкции для выполнения операций с плавающей запятой. Эти инструкции входят в блок FPU и также могут быть распараллелены. В этом блоке инструкций образуется две группы распараллеливания:

- инструкции, которые могут выполняться параллельно с инструкцией FXCH;
- инструкции, которые выполняются последовательно и только на U-конвейере.

Построив характеристическое уравнение и решив его, мы получим $C(I) \approx 25.56$ бит/такт. Частота выбранного процессора равна 100 МГц, а, следовательно, $C_{iP5}(I) \approx 2556$ Мбит/сек.

3.5 Оценка производительности компьютера на базе процессора Intel Pentium MMX

Процессор Intel Pentium MMX относится к пятому поколению процессоров Intel P5 и был представлен в 1996 году. MMX (MultiMedia eXtension) – это дополнительный набор инструкций, позволяющий выполнять характерные для кодирования (декодирования) видео- и аудиооперации. Обработка этих команд производится с помощью математического сопроцессора путём перевода его в режим MMX, в котором все регистры сопроцессора начинают восприниматься как набор регистров MMX. И эта особенность сильно влияет на определение вычислительной способности для подобных процессоров. Помимо этого отличия, в процессорах Intel Pentium MMX был вдвое увеличен объём кэш-памяти первого уровня.

Разберемся, как же определять вычислительную способность процессоров P5MMX. Инструкции MMX могут выполняться параллельно не только между собой, но и параллельно с целочисленными инструкциями. Однако между собой они тоже имеют довольно сложное взаимодействие. Рассмотрим группы инструкций:

- Все инструкции, которые в качестве операндов используют адреса ячеек памяти или стандартные регистры процессора, могут выполняться только на U-конвейере и выполняться параллельно только с инструкциями MMX. Ограничения, описанные в этой группе, относятся также и ко всем инструкциям из других групп
- Инструкции сдвига, упаковки и распаковки могут выполняться на любом конвейере, но не могут выполняться параллельно с операциями такого же типа
- Инструкции умножения могут выполняться на любом конвейере, но не могут выполняться одновременно с другими инструкциями умножения MMX
- Все остальные инструкции могут выполняться на любом конвейере и параллельно с любой допустимой инструкцией

Следуя этой формуле, построим характеристические уравнения, решим их и получим: $C(I) \approx 28.35$ бит/такт. Частота выбранного процессора равна 200 МГц; следовательно, $C_{iPMMX}(I) \approx 5670$ Мбит/сек.

4. Анализ полученных результатов

В табл. 1 приведены технические характеристики и оценки вычислительной способности для компьютеров на базе рассмотренных нами процессоров фирмы Intel.

Таблица 1. Вычислительная способность компьютеров на базе процессоров Intel

Название процессора	Частота процес-сора, МГц	Кол-во ядер	Объём ОЗУ, Мбайт	Размер L1 кэша, Кбайт	Размер L2 кэша, Кбайт	Размер L3 кэша, Мбайт	Выч. способ., Гбит/сек
Intel 80286	12,5	1	1	-	-	-	0.075
Intel 80486	66	1	8	8	128	-	0.710
Pentium	100	1	32	16	512	-	2.556
Pentium	150	1	32	16	512	-	3.834
Pentium MMX	200	1	64	32	512	-	5.670
Pentium MMX	233	1	64	32	1024	-	6.606

Описывать уравнения, благодаря которым были получены результаты, не представляется возможным, потому как для всех процессоров P5 уравнения содержат несколько тысяч слагаемых и их построение, как и вычисление, производилось с помощью программ.

Особый интерес представляет сравнение полученных данных с результатами эмпирических оценок производительности. Для этого мы будем использовать бенчмарк, разработанный компанией Intel, ICOMP. Так как единицы измерения не совпадают, то мы будем производить сравнение относительных оценок, что сможет нам показать динамику роста.

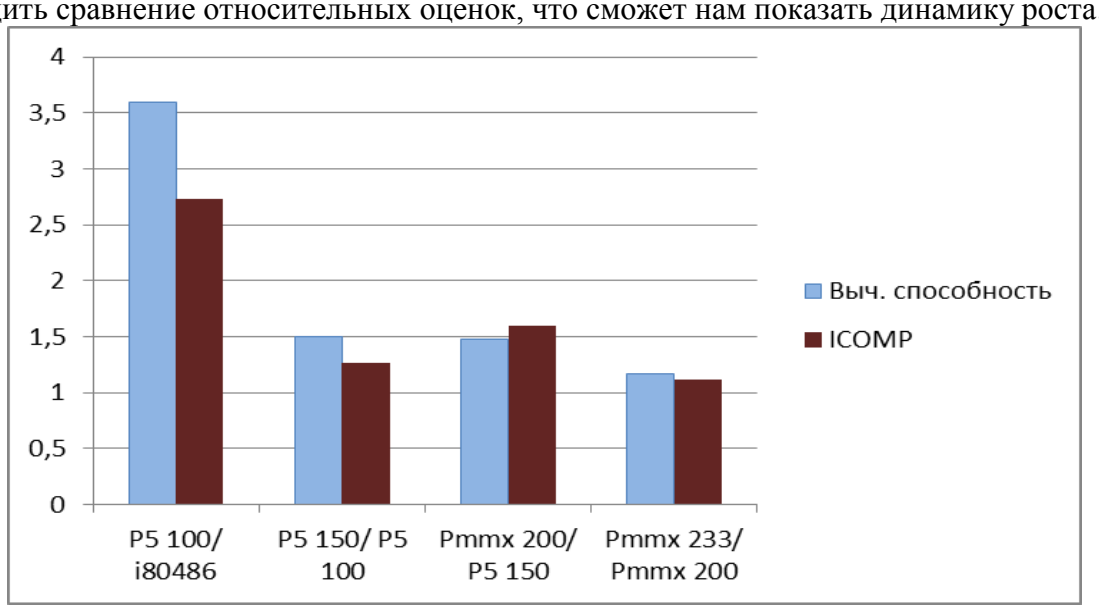

Рис. 1. Сравнение результатов с индексами ICOMP

На рис. 1 видно, что темпы роста производительности компьютеров имеют общие черты, хотя есть некоторая погрешность. Это объясняется тем, что бенчмарк рассчитывает задачи, которые его производители посчитали наиболее важными, и на их основании дает оценку производительности компьютера. Предлагаемый нами метод, наоборот, направлен на оценку верхнего порога производительности всего множества задач, которые может решать компьютер. Из-за этого и появляются разногласия между значениями, рассчитанными нами, и индексами ICOMP.

Из всего вышесказанного можно сделать вывод, что модель, предложенная в работе [1], является вполне обоснованной и может применяться для сравнения существующих процессоров, а также для оценки вычислительной способности разрабатываемых вычислительных систем.

Литература

1. *Ryabko B.* An information-theoretic approach to estimate the capacity of processing units // Performance Evaluation. 2012. V. 69, P. 267-273.
2. *Shannon C. E.* A mathematical theory of communication // Bell Sys. Tech. J. 1948. V. 27, P. 379-423, P. 623-656.
3. Intel x86 Quick Reference Instruction Manual - 8086/80186/80286/80386/80486. URL: `http://www.intel-assembler.it/portale/5/x86-instruction-reference-manual/x86-instruction-reference-manual.asp` (Дата обращения: 04.12.2012).
4. *Касперски К.* Секреты поваров компьютерной кухни или ПК: решение проблем. BHV, 2003. 560 с.
5. *Fog A.* Lists of instruction latencies, throughputs and microoperation breakdowns for Intel, AMD and VIA CPUs. Copenhagen University College of Engineering. 2012. URL: http://www.agner.org/optimize/ (Дата обращения: 04.12.2012).
6. Intel 64 and IA-32 Architectures Software Developers Manual Volume 1: Basic Architecture. Intel Corp. URL: `http://www.intel.ru/content/www/ru/ru/architecture-and-technology/64-ia-32-architectures-software-developer-vol-1-manual.html` (Дата обращения: 04.12.2012).
7. Intel 64 and IA-32 Architectures Software Developers Manual Volume 2. Intel Corp. URL: `http://www.intel.ru/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-2a-2b-instruction-set-a-z-manual.html` (Дата обращения: 15.03.2012).
8. *Tanenbaum A.S.* Structured computer organization. Prentice Hall PTR, 2001. 514 p.
9. *Marr D. T., Binns F., Hill D. L., Hinton G., Koufaty D. A., Miller J. A. and Upton M.* Hyper-threading technology architecture and microarchitecture // Intel Technology Journal. 2001. V. 06, Issue 01.

Статья поступила в редакцию 03.09.2012;
переработанный вариант – 12.12.2012

Ракитский Антон Андреевич
ассистент кафедры прикладной математики и кибернетики СибГУТИ,
тел.+7-923-101-1936, e-mail: rakitsky.anton@gmail.com

Practical application of theoretical estimation methods of computational power for Intel processor P5 series

A. Rakitskiy

In Ryabko's paper «Information-theoretic approach to estimate the capacity of processing units» was presented a new approach of evaluating computational power of computers and other similar devices (clusters, mobile phones etc.). In the article, this method is used for evaluation of computational power for Intel processors as well as for the comparative analysis of the influence of different characteristics of computers on this value.

Keywords: computational power, performance.