|  |
| --- |
| Министерство образования РФ  **Череповецкий государственный университет**  Кафедра ПО ЭВМ  Дисциплина:  «Организация ЭВМ и систем» КУРСОВАЯ РАБОТА **Тема**: «Тест: быстродействие микропроцессора»  Выполнил студент: Лужинский Андрей  Группа: 1ПО-32  Принял преподаватель: Зуев А.Н. Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Дата сдачи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Череповец 2001 |

Содержание

§ 1. Задание 3

§2. Обзор существующих систем 3

§3. Недостатки существующих систем 13

§4. Алгоритм решения 14

§5. Описание алгоритма 16

§6. Листинг программы 16

§7. Описание программы 20

§8. Экспериментальные данные 20

§9. Полученные результаты 21

§10. Выводы 23

§11. Список литературы 25

# 

# § 1. Задание

Сделать тест, который проверяет быстродействие микропроцессора. Проанализировать несколько микропроцессоров с различными характеристиками.

# §2. Обзор существующих систем

2.1. *Введение*

Создание фирмой Intel первого микропроцессора в 1971 году положило начало эпохе компьютеризации. «Благодаря микропроцессорам компьютеры стали массовым общедоступным продуктом», - заявил Тед Хофф (Ted Hoff), один из изобретателей первого микропроцессора.

За чуть более чем 25 лет процессоры прошли поистине гигантский путь. Первый чип Intel 4004 работал на частоте 750 Кгц, содержал 2300 транзисторов и стоил около $200. Производительность его оценивалась в 60 тыс. операций в секунду. На сегодняшний день реальная производительность микропроцессоров превзошла 1300 Мгц.

Сравнение приведённых значений подтверждают оценку успехов микропроцессорной индустрии, данную основателем и председателем совета директоров фирмы Intel Гордоном Муром (Gordon Moor): «Если бы автомобилестроение эволюционировало со скоростью полупроводниковой промышленности, то сегодня «Роллс-Ройс» стоил бы всего 3 доллара, мог бы проехать полмиллиона миль на одном галлоне бензина, и было бы дешевле его выбросить, чем заплатить за парковку».

Такое интенсивное развитие технологий в обществе, где основным предметом труда становится информация, является следствием растущего спроса на основные орудия труда – компьютеры. На сегодняшний день компьютеризация является одним из главных направлений научно-технического прогресса. Количество и качество производимых в стране компьютеров, степень насыщенности вычислительной техники самых разных отраслей становится одним из основных критериев её экономического и военного потенциала.

В микропроцессорах – наиболее сложных микроэлектронных устройствах - воплощены самые передовые достижения инженерной мысли. В условиях свойственной данной отрасли производства жёсткой конкуренции и огромных капиталовложений, выпуск каждой новой модели микропроцессора – так или иначе, связан с очередным научным, конструкторским, технологическим прорывом.

2.2. *Микропроцессор Pentium.*

На сегодняшний день микропроцессоры Pentium компании Intel занимают лидирующее положение на рынке микропроцессоров. Посмотрим архитектуру этих процессоров.

Семейство микропроцессоров Pentium (II, Ш, 4) включает в себя высокопроизводительные процессоры Intel, работающие на тактовых частотах от 60 до 2000 Мгц. Процессор Pentium полностью программно совместим с предыдущими микропроцессорами Intel, и позволяет применять ранее разработанные программы

В отличие от предыдущих микропроцессоров с системой команд х86, процессоры семейства Pentium обладают рядом целых технических новшеств, к числу которых относятся:

* суперскалярная архитектура;
* раздельные кэш-памяти для команд и данных;
* предсказание переходов;
* высокопроизводительные операции с плавающей точкой;
* усовершенствованная 64-разрядная шина данных;
* средства обеспечения целостности данных;
* SL-технология со средствами управления энергопотреблением;
* поддержка многопроцессорности;
* мониторинг производительности;
* поддержка различных страниц памяти.

Рассмотрим эти новшества более подробно.

1. Суперскалярная архитектура

Два конвейера процессора могут выполнять две команды одновременно. Команды выполняются в 5 этапов:

* 1. предвыборка;
  2. декодирование 1;
  3. декодирование 2;
  4. выполнение;
  5. запись результатов;

При этом несколько команд могут находиться на разных этапах выполнения.

Однако два конвейера не являются независимыми. При остановке одного останавливается и другой.

#### Раздельные кэш-памяти команд и данных

Каждая кэш память процессора имеет размер 8 Кбайт (для версий Pentium I) и 16 Кбайт (для версий Pentium II, MMX, III). Кэш памяти являются частично ассоциативными. Поиск требуемой информации выполняется в стандартных 32-байтовых строках.

Буфер трансляции адресов (TLB) преобразует ячейки внешней памяти в соответствующий адрес данных в кэш памяти.

#### Предсказание переходов

Процессор Pentium – первый х86 совместимый микропроцессор. Выполняет утверждение переходов, используя буфер BTB (Branch Target Buffer) и два буфера предварительной выборки.

Алгоритм упреждения переходов процессора Pentium не только прогнозирует выбор простых ветвей, но поддерживает и более сложное прогнозирование. Это делается с помощью хранения в буфере BTB нескольких адресов переходов. BTB хранит до 256 результатов переходов, что позволяет выполнять правильное предсказание с вероятностью не менее 0,8.

1. Высокопроизводительный блок операций с плавающей точкой

В процессоре применяется блок вычислений с плавающей точкой, использующий сложные восьмиступенчатые конвейеры и внутренние функции. Большинство команд с плавающей точкой начинают выполняться в одном из целочисленных конвейеров, а затем передаются на конвейеры с плавающей точкой. Умножение и деление реализованы как внутренние функции.

2.3. *Архитектурные особенности микропроцессоров*

**Типы команд микропроцессора.**

В ходе эволюционного развития архитектур процессоров в состав системы команд вводились и закреплялись сложные команды, которые, по мнению разработчиков, соответствовали решаемым задачам. Мерой этого соответствия чаще всего был объём двоичного кода программы.

Команды бывают разных типов: «регистр, регистр🡪регистр»; «память, память🡪память»; «регистр🡪память» и др. Сложные команды модифицируют содержимое групп регистров и ячеек памяти, и для их реализации при приемлемых затратах оборудования, как правило, применяется микропрограммирование.

Команды называются *скалярными,* если входные операнды и результаты являются числами (скалярами).

Команды называются *векторными*, если входные операнды и, возможно, результаты являются вектором (массивом) чисел, а для преобразования данных массива (вектора) используется одна векторная команда.

Само появление векторных команд обусловлено стремлением ускорить обработку массивов данных за счёт исключения затраты времени на выборку и дешифрацию команд обработки, одинаковых для всех компонент входных массивов.

При сохранении последовательных программ для ускорения обработки применяются суперскалярные процессоры, в которых за счёт параллельной работы функциональных устройств процессора в одном такте вырабатывается несколько скалярных результатов.

**Структурный параллелизм микропроцессоров.**

Повышение производительности микропроцессоров достигается за счёт увеличения тактовой частоты, совершенствования параллельной и конвейерной обработки данных, а так же уменьшения времени доступа к памяти. Современные микропроцессоры содержат десять и более обрабатываемых устройств, каждое из которых предоставляет собой конвейер. Эффективная загрузка параллельно функционирующих конвейеров обеспечивается либо аппаратурой процессора, либо компилятором, на вход которого поступают программы на последовательном языке программирования, либо совместно с аппаратурой и компилятором.

В компиляторах используется изощрённая техника извлечения параллелизма из последовательных программ. Аппаратура микропроцессоров ориентированна на выделение более простых форм параллелизма, в том числе естественного. Стремление использовать присущий большинству программ естественный параллелизм вычисления целочисленных адресных выражений и собственно обработки данных в формате с плавающей точкой привело к появлению разнесенных структур (decoupled architecture). Условно микропроцессоры с разнесённой архитектурой называются А - процессорами и Е- процессорами.

Разнесённая архитектура позволяет достигать при скалярной обработке производительности, характерной для векторных процессоров, за счёт предвыборки данных из памяти и автоматической развёртки нескольких последовательных витков цикла в А – процессоре. Проблемы расщепления программы на подпрограммы для А - и Е- процессора решаются на уровне компилятора или специальным блоком-расщепителем.

**Структурные методы уменьшения времени доступа к памяти.**

Совершенствование архитектуры микропроцессоров и механизма доступа к памяти не может происходить без принятия совокупности соглашений о структуре программ и обрабатываемых данных. Это обусловлено, например, тем, что время доступа в основную память в десять и более раз больше, чем время выполнения преобразования данных в регистрах процессора. Поэтому необходимо находить решения по уменьшению времени доступа, учитывающие этот факт.

*Иерархическая структура памяти.* Идеальная память должна обеспечивать процессор командами так, чтобы не вызывать простоев процессора. При этом память должна иметь большую емкость. В современных условиях уменьшение времени доступа достигается введением многоуровневой иерархии памяти. Время доступа зависит от объема и типа используемой памяти.

Современная типовая иерархия памяти имеет следующую структуру:

* регистры 64 – 256 слов со временем доступа 1 такт процессора;
* кэш 1 уровня – 8к слов со временем доступа 1 – 2 такта;
* кэш 2-го уровня – 256к слов со временем доступа 3 – 5 тактов;
* основная память – до 4 Гигаслов со временем доступа 12 – 55 тактов.

Использую помимо основной памяти небольшую и более быструю буферную память, можно значительно сократить количество обращений к основной памяти, за счёт аккумуляции фрагмента программного кода в буферной памяти. Создание иерархической памяти за время, пока предшествующие блоки программ и данных между уровнями памяти за время пока предшествующие блоки обрабатываются микропроцессором, позволяет существенно сократить простои процессора в ожидании данных. При этом эффект уменьшения времени доступа к памяти будет тем больше, чем больше время обработки данных в буферной памяти по сравнению со временем пересылки между буферной и основной памятью. Это достигается при локальности обрабатываемых данных, когда процессор многократно использует одни и те же данные для выработки некоторого результата.

В связи с тем, что локально обрабатываемые данные могут возникать в динамике вычислений, и не обязательно сконцентрированы в одной области при статическом размещении в основной памяти, буферную память организуют как ассоциативную, в которой данные содержатся в совокупности с их адресом в основной памяти. Такая буферная память получила название *кэш-память.*

Кэш имеет совокупность строк, каждая из которых состоит из фиксированного количества единиц в памяти (байтов, слов с последовательными адресами Типичный размер строки: 16,64,256 байтов).

Наиболее часто используются три способа организации кэш-памяти, отличающиеся объёмом аппаратуры, требуемой для их реализации. Это так называемая кэш-память с прямым отображением (direct-mapped cache), частично ассоциативная кэш-память (set-associative cache) и ассоциативная кэш память (full associative cache).

*Расслоение памяти.* Другим структурным способом уменьшения времени доступа к памяти служит расслоение памяти. В предположении, что выборка из памяти выполняется по последовательным адресам возможно использование k блоков памяти с размещением в блоке I, где I=0,…, k-1, слов с адресами n=I+krrp mod k, где p=0,1,…,M. В этом случае возможно k параллельных обращение в память по адресам, принадлежащим различным блокам. Поэтому выборка команд программы за исключением команд, выбираемых как результат ветвления, может быть ускорена применением расслоения памяти. Аналогично может быть ускорена обработка массивов данных

Многоуровневая иерархия и расслоение памяти могут использоваться совместно.

**Способы измерения производительности**

Пиковая (техническая) производительность представляют собой теоретический максимум быстродействия компьютера при идеальных условиях. Данный максимум определяется как число вычислительных операций, выполняемых в единицу времени всеми имеющимися в процессоре обрабатывающими логико-арифметическими устройствами. Предельное быстродействие достигается при обработке бесконечной последовательности не связанных между собой и не конфликтующих при доступе в память команд (т.е. когда результат любой операции не зависит от действий, выполненных другими командами). При этом предполагается, что все операнды выбираются из внутрикристальной кэш-памяти данных, а команды – из кэш-памяти команд. Разумеется, подобная ситуация чисто гипотетическая, и на практике ни одна система не в состоянии работать сколь-нибудь длительное время с пиковой производительностью, хотя и может приближаться к этой величине. Например, на текстах Linpack при больших размерностях обрабатываемых матриц (10001000) почти все компьютеры демонстрируют производительность в диапазоне от 0,8 до 0,95 от пикового значения.

Пиковая производительность является единственной по-настоящему объективным параметром (для его определения необходимо знать несколько параметров процессора) и совершенно не зависит от типа выполняемой программы. Речь идёт о тактовой частоте процессора, которая для подавляющего большинства современных компьютеров определяет темп формирования результатов на выходе арифметического конвейера, и числе арифметических конвейеров. Не трудно установить соответствие между единицами измерения тактовой частоты и производительности процессора. Очевидно, что 1 Мгц соответствует 1 MFLOPS или 1 MIPS-пиковой производительности одного конвейера, в зависимости от того, какой тип операций рассматривается – с плавающей или с фиксированной точкой. Не следует забывать также о разрядности обрабатываемых чисел, чтобы избежать не корректного сравнения производительности для 32- 64- разрядных данных. Но в любом случае правила вычисления пиковой производительности, при всей их внешней простоте, могут оказаться чрезвычайно полезными при обсуждении достоинств компьютеров. Чтобы определить пиковую производительность машин, надо умножить тактовую частоту на количество параллельно выполняемых операций. Например, арифметическое устройство Pentium каждый такт может формировать один результат полноразрядной (64 бит) операций с плавающей точкой или два результата 32-х разрядных целочисленных операций. Следовательно, для Pentium/90 пиковая производительность равна 90 MFLOPS при выполнении вычислений с плавающей точкой и 180 MIPS при целочисленной 32-х разрядной обработке.

Особенно удобно использовать показатели предельного быстродействия для сравнения возможностей процессоров в первом приближении. Американские специалисты по контролю за экспортом вооружений при оценке уровня производительности компьютеров используют показатель основной теоретической производительности (CTP), измеряемой в Mtops – миллионах теоретических операций в секунду. CTP зависит только от аппаратных средств компьютера (тактовой частоты, набора функциональных устройств, пропускной способности и набора внутренних шин, длины разрядной сетки и так далее).

**Реальная производительность**

При выполнении реальных прикладных программ эффективная (реальная) производительность микропроцессора может весьма существенно (до нескольких раз) быть меньше пиковой. Это объясняется тем, что современные высокопроизводительные микропроцессоры имеют весьма сложную архитектуру (суперконвейерная и суперскалярная обработка, многоуровневая система памяти и так далее). Характеристики их функционирования на уровне внутренних устройств существенно зависят от программы и обрабатываемых данных. Поэтому не возможно с необходимой точностью оценить производительность только на основании тактовой частоты их работы, числа затрачиваемых на выполнение одной команды тактов процессора и числа устройств обработки.

Для оценки производительности различных вычислительных средств в мировой практике наибольшее распространение получило использование наборов характерных для той или иной области применения вычислительной техники задач. Время выполнения одной каждой из задач набора составляет основу для расчёта индекса производительности исследуемой вычислительной установки. Индекс производительности является относительной оценкой, несущей информацию о том, на сколько быстрее или медленнее исследуемая вычислительная установка выполняет подобные задачи по сравнению с некоторой широко распространенной ЭВМ (последнюю часто называют базовой или эталонной).

**Способы измерения реальной производительности**

При оценке производительности на тестах приходится решать три проблемы, связанные с анализом результатов контрольного тестирования производительности:

* отделение показателей, которым можно доверять безоговорочно, от тех, которые должны восприниматься с известной долей настороженности (проблема достоверности оценок);
* выбор конторольно-оценочных тестов, наиболее точно характеризующих производительность при обработке типовых задач пользователя (проблема адекватности оценок);
* правильное истолкование результатов тестирования производительности, особенно если они выражены в довольно экзотических единицах типа MWIPS, Drystoones/s и т.д. (проблема интерпретации).

Первую группу тестов измерения производительности составляют тесты производителей, разрабатываемые компаниями-изготовителями компьютеров для «внутреннего» применения – оценивая качества собственных продуктов. Главная особенность этих тестов заключается в том, что они ориентированы на сравнение ограниченного множества однотипных компьютеров, часто относящихся к одному семейству.

Пользовательские тесты создаются крупными компаниями, специализирующимися на внедрение компьютерных технологий, или совместными усилиями групп пользователей, объединённых сходством решаемых задач. Эти средства предназначены для выбора компьютеров и ПО, наиболее подходящие под определённые прикладные задачи. В принципе такой подход позволяет получить наиболее точные оценки производительности для конкретного класса приложений, хотя и сопряжён со значительными усилиями пользователей по созданию тестовых программ и проведению испытаний компьютеров.

Сегодня наиболее распространенными являются наборы тестов компании SPEC (Standard Performance Evolution Corporation) – SPEC (89, 92,95), которые и будут рассматриваться ниже

Пакет тестовых программ SPEC 89 включает в себя два тестовых набора – Cint 89, состоящий из четырёх программ с целочисленной обработкой, и Cfp 89, объединяющий шесть программ со значительным объемом информации над числами с плавающей точкой двойной точности.

Все десять модулей представляют собой достаточно сложные программы на языках C и FORTRAN с широким спектром решаемых задач – от оптимизации матриц булевой алгебры до моделирования замещения атомов в квантовой химии.

**Пакет тестовых программ SPEC 92**

SPEC 92 состоит из наборов тестовых пакетов Сint92 и Cfp92.

Пакет оценочных программ Cint92 предназначен для оценки производительности микропроцессоров при выполнении целочисленных операций, преимущественно в коммерческой области применения. В его состав входят 6 эталонных тестов, написанных на языке Си и представляющих собой задачу из теории сетей, интерпретатор языка Lisp, задачу логического проектирования, Unix – утилиту упаковки тестового файла размером 1 Мбайт \, который 20 раз подвергается сжатию, операции со строками и столбцами электронной таблицы и компилятор языка Си.

Пакет оценочных программ Cfp92 предназначен для оценки производительности микропроцессоров на операциях с плавающей точкой, преимущественно в технической и научной областях.

В его состав входят 14 различных прикладных программ, две из которых написаны на языках Си и 12 – на языке Фортран. В пакет входят программы схемного проектирования, проектирования термодинамики ядерного реактора методом Monte-Carlo, задачи квантовой химии и физики, решение уравнения Максвелла, преобразование координат, трассировка оптических лучей, задачи робототехники и нейросетей, моделирование человеческого уха, решение уравнения Навье-Стокса для определения параметра межгалактического газа, семь библиотечных функций обработки матриц (умножение, обращение и т.д.) и ряд других.

Пакет SPEC 92 имеет одно новшество. Речь идёт о характеристиках мультипрограммной обработки SPECrate, формируемых в рамках метода одной нагрузки. Суть последнего заключается в следующем: тестируемый компьютер выполняет задание, состоящее из множества копий одной программы, а показателем производительности многопроцессорной обработки служит количество копий, завершенных за определённый интервал времени. Для получения оценки SPECrate используются те же программы, что и для расчёта показателей SPECint92 и SPECfp92. Разница только в том, что тестовый модуль реализуется как несколько копий, образующих одно задание, а результатом измерений является нормированное общее время выполнения всех копий задания. Подобной процедуре подвергается каждая из 20 тестовых программ, что позволяет получить шесть частных оценок SPECratio, для программ целочисленной обработки, и 14 – для программ обработки вещественных чисел.

Таким образом, SPECrateint 92 и SPECratfp 92 оценивают среднюю скорость выполнения задачи в многопроцессорном режиме работы системы. Кроме того, эти показатели позволяют получить представление о возможностях компилятора по организации параллельного мультизадачного кода, а также операционной системы – по эффективному динамическому распределению ресурсов системы (в частности, процессоров) между выполняемыми параллельными программами. Это делает оценки SPECrateint 92 и SPECratfp 92 особенно представительными для SMP-систем коллективного пользования, работающих в пакетном режиме.

Сводные сведения о программах, входящих в SPEC 89 и SPEC 92, приведены ниже в таблице.

*Перечень программ наборов SPEC89, SPEC92.*

|  |  |
| --- | --- |
| Программа | **Тест, язык, тип данных** |
| Моделирование ПЛМ (PLA) | 89+92, С, ФТ |
| Lisp – интерпретатор | 89+92, С, ФТ |
| Формирование логических таблиц истинности | 89+92, С, ФТ |
| Unix – утилита упаковки тестового файла размером 1Мбайт, который 20 раз подвергается сжатию | 92, С, ФТ |
| Операция со строками и столбцами электронной таблицы | 92, С, ФТ |
| Компилятор GNU, трансляция 19 программ на Си в оптимизированный код ассемблера | 89+92, С, ФТ |
| Моделирование аналоговых цепей с высокой интенсивностью обменов с памятью | 89+92, Фортран, ПТ, ДВ |
| Моделирование термодинамики ядерного реактора методом Монте-Карло; содержит большое число ветвлений и коротких циклов | 89+92, Фортран, ПТ, ДВ |
| Задача квантовой химии для системы из 500 атомов | 92, Фортран, ПТ, ДВ |
| Версия теста Mdijdp2 для одинарной точности | 92, Фортран, ПТ, ОТ |
| Решение уравнений Максвелла | 92, Фортран, ПТ, ДВ |
| Генерация сетки при моделировании процессов обтекания; программа ориентирована на тестирование параллельных систем | 89+92, Фортран, ПТ, ДВ |
| Моделирование управления движением робота с использованием видеосистемы | 92, С, ПТ |
| Решение сеточной задачи shallow – water для сетки 256х256 | 92, Фортран, ПТ |
| Задача квантовой физики вычисления массы элементарных частиц с использованием метода Монте-Карло; хорошо векторизуется и ориентирована на тестирование параллельных систем | 92, Фортран, ПТ, ДВ |
| Решение уравнение Навье-Стокса для определения параметра межгалактического газа; программа хорошо векторизуется и ориентирована на тестирование параллельных систем | 92, Фортран, ПТ, ДВ |
| Семь библиотечных функций обработки матриц (умножение, обращение и т.д.) | 89+92, Фортран, ПТ, ДВ |
| Моделирование процесса замещения атомов на сериях Гаусса; программа плохо распараллеливается и содержит большой объем ввода/вывода | 89+92, Фортран, ПТ, ДВ |
| Синтетический тест, имитирующий различные алгоритмы умножения матриц | 89, Фортран, ПТ, ДВ |
| Моделирование уха человека | 92, С, ПТ, ДВ |

**Пакет тестовых программ SPEC 95**

Появление нового набора программ обусловлено развитием микропроцессоров (повышение производительности и увеличением объёма внутрикристальной памяти), совершенствование компиляторов, стремление учесть требования стандартов открытых систем, атак же скорректированным после появления SPEC 92 представлением об актуальности различных областей приложения.

Современные процессоры выполняют решение тестов SPEC 92 в течение интервалов времени от одной до нескольких секунд, что вносит большие погрешности в измерения производительности. Объём кода программ и данных SPEC 92 таков, что программы и данные могут разместиться в кэш памяти процессора. Это вообще не позволяет получить сколько-нибудь достоверные оценки реальной производительности.

Индексы производительности в SPEC 95 даются по отношению к эталонной машине SPARC-station 10/40 в конфигурации с кэш памятью второго уровня.

Используются два тестовых набора CINT 95 и CFP 95, состоящие из 8 и 10 программ соответственно.

*Таблица 2. Перечень программ набора CINT*

|  |  |
| --- | --- |
| Область приложения | Спецификация задачи |
| Моделирование | Моделирование кристалла Motorola 88100 |
| Компиляция | Компиляция программы на Си и компиляция оптимизированного кода для процессора SPARC |
| Искусственный интеллект | Игра Го – игра сама против себя |
| Компрессия | Компрессия текстового файла размером 16 Мбайт |
| Интерпретация | Lisp-интерпретатор |
| Обработка изображений | Сжатие графических объектов(JPEG) с различными параметрами |
| Манипулирование текстовыми строками | Shell-интерпретатор |
| Базы данных | Построение таблиц и манипулирование с ними |

При испытаниях компьютеров формируются:

* индексы производительности SPEC int 95, SPEC fp 95 и SPEC int base 95, SPEC fp base 95, для фиксированной и плавающей точки в оптимизированном режиме компиляции и без оптимизации соответственно.
* индексы пропускной способности SPEC int rate 95, SPEC fp rate 95 и SPEC int rate base 95, SPEC fp rate base 95 для оценки многозадачных режимов и SMP архитектур при оптимизированном режиме компиляции и без оптимизации соответственно.

Все интегральные индексы производительности формируются как среднее геометрическое индексов по отдельным тестам.

Корпорация SPEC не даёт никаких рекомендаций по установления соответствия между значениями индексов SPEC 92 и ЫЗУС 95.

*Таблица3. Перечень набора программ набора CFP*

|  |  |
| --- | --- |
| **Область приложения** | **Спецификация задачи** |
| Гидродинамика, геометрический аспект | Генерация двумерной координатной сетки в произвольной области |
| Предсказание погоды | Моделирование водной поверхности методом конечных элементов (вещественная арифметика с одинарной точностью) |
| Квантовая физика | Вычисление масс элементарных частиц методом Монте-Карло |
| Астрофизика | Расчет межгалактических газов по уравнениям Навье-Стокса |
| Электромагнетизм | Расчет трехмерного поля потенциалов |
| Гидродинамика | Решение системы уравнений в частных производных |
| Моделирование | Моделирование турбулентности в кубическом объеме |
| Предсказание погоды | Вычисление статистики температур, воздушных потоков и уровней загрязнения |
| Квантовая химия | Порождение потока электронов |
| Электромагнетизм | Решение уравнения Максвелла |

# §3. Недостатки существующих систем

* 1. При решении различных задач прикладными программами ресурсы микропроцессора расходуются зачастую неэкономно. В результате этого процессору приходится совершать огромное количество команд, которые могли быть заменены несколькими. Вина лежит не на программных продуктах, а на тех системах программирования, с помощью которых эти программы создавались.

Решение:создавать средства для создания программ максимально эффективным способом, обращая внимание на используемые процессором адресации и размещение данных.

* 1. На быстродействие так же влияет и то, какое напряжение подаётся на микропроцессор. При большом напряжении происходит нагрев процессора. В результате этого основа, на которой размещаются транзисторы, начинает греться и, соответственно он выходит из строя.

Решение:расстояние между транзисторами необходимо уменьшить. На сегодняшний день минимальную технологию изготовления имеют процессоры Intel Pentium 4, и она составляет 0,13 мкм (для сравнения: процессор AMD с архитектурой К7, тактовой частотой 1000Мгц имеет 0,18 мкм. Обычная температура работы ~ 65°С). Расстояние между транзисторами уменьшается, соответственно сопротивление, а отсюда они требуют меньшее энергопотребление – температура нагрева заметно снижается.

* 1. Тормозит работу процессора и то, что затрачивается много времени на обмен данными, которые находятся в постоянной памяти. И даже при небольших вычислениях микропроцессор обращается к ОЗУ за данными, а мог бы данные хранить в собственных регистрах

Решение:по возможности, часто используемые данные желательно размещать либо в регистрах процессора, либо в быстрой кэш-памяти. Многие компиляторы не заботятся о том, чтобы разместить данные так, чтобы наблюдалась максимальная скорость работы программы.

* 1. Так же скорость вычисления уменьшается из-за того, что процессору приходится останавливать свою работу и выполнять аппаратные прерывания. Реакция процессора на прерывания:

а) приходит прерывание;

б) процессор прекращает выполнять основную задачу;

в) происходит сохранение положения, на котором остановилась основная программа;

г) выполняется прерывание;

д) процессор восстанавливает запомненное состояние;

е) продолжает выполнение основной задачи.

Надо заметить, что процессор тратит много времени на сохранение своего текущего состояния и восстановление.

Решение: во время вычисления больших объёмов информации можно запретить некоторые аппаратные прерывания ниже IRQ 0 (таймер). Запрещённые прерывания не будут останавливать процессор, соответственно скорость выполнения возрастёт. *Недостатки: могут потеряться некоторые данные, например символы, которые вводятся с клавиатуры – прерывание IRQ 1.*

# §4. Алгоритм решения

Рассмотрим недостаток № 3. *Затрачивание времени на обмен данными.*



1

2

3

4

5

6

7

8

9

10

11

12

13

14

# §5. Описание алгоритма

*Блок 1-7.* Подсчёт времени на выполнение выбранной операции на языке высокого уровня. В данном случае применяется язык Pascal.

*Блок 1.* Начало программы.

*Блок 2.* Вводятся два числа с клавиатуры.

*Блок 3.* Выбираем операцию из предложенных на экране. Если это операция сложения, то выполняется алгоритм суммирования ряда, если умножение, то выполняется вычисление факториала.

Ввели N, то рад сложения выглядит следующим образом:

1+2+3+…+N=СУММА

Умножения

1\*2\*3\*…\*N=ПРОИЗВЕДЕНИЕ

*Блок 4.* Включаем таймер и запоминаем начальное значение.

*Блок 5.* Выполняем выбранную операцию на Pascal.

*Блок 6.* Выключаем таймер и запоминаем считанное значение

*Блок 7.* Находим разницу между начальным значением времени и конечным

*Блок 8-14.* Подсчёт времени на выполнение выбранной операции на языке низкого уровня. Применяется язык Assembler.

*Блок 8.* Переменные, которые мы вводим с клавиатуры, находятся в памяти. Чтобы начать вычисления мы должны из памяти (ОЗУ) поместить значения в регистры процессора. Это мы делаем до включения таймера, чтобы время, затраченное на это действие, не вошло в результат.

*Блок 9.* Считываем начальное значение времени.

*Блок 10.* Выполняем операцию

*Блок 11.* Считываем конечное время

*Блок 12.* Находим время выполнения операции – это разница между конечным значением и начальным.

*Блок 13.* Выводим значение времен на экран и делаем выводы.

*Блок 14.* Выход из программы.

# §6. Листинг программы

Program Test\_of\_CPU\_for\_Zuev;

uses crt;

var

vid:byte;

t1,t2,NN:word;

time1,time2:real;

Procedure Add\_Pascal(N:word); {Сложение на Pascal}

var i:word;

Summa:integer;

Begin

summa:=0;

t1:=0;

t2:=0;

asm

in al,40h

mov bl,al

in al,40h {Считываем начальное значение тиков от таймера}

mov bh,al

mov t1,bx

end;

for i:=1 to N do

summa:=summa+i;

asm

in al,40h

mov bl,al {Конечное количество тиков}

in al,40h

mov bh,al

mov t2,bx

end;

Time1:=t1/(65536\*18.2);

time2:=t2/(65536\*18.2);

writeln('Время выполнения сложения на Pascal: ',(time2-time1):6:12);

end;

Procedure Add\_Asm(N:word);

var i:word;

Proisv:integer;

Begin

t1:=0;

t2:=0;

asm

in al,40h

mov ah,al

in al,40h

xchg ah,al

mov t1,ax

xor ax,ax

xor dx,dx

@@q: {Выполняем сложение}

inc dx

add ax,dx

loop @@q

in al,40h

mov cl,al

in al,40h

mov ch,al

mov t2,cx

end;

Time1:=t1/(65536\*18.2);

time2:=t2/(65536\*18.2);

writeln('Время выполнения сложения на Assembler: ',(time2-time1):6:12);

end;

{=================================================================}

Procedure Mul\_Pascal(N:word); {Умножение на Assembler}

var i:word;

Proisv:integer;

Begin

t1:=0;t2:=0;

asm

in al,40h

mov bl,al

in al,40h {Считываем начальное значение тиков от таймера}

mov bh,al

mov t1,bx

end;

for i:=1 to N do

proisv:=proisv\*i;

asm

in al,40h

mov bl,al {Конечное количество тиков}

in al,40h

mov bh,al

mov t2,bx

end;

Time1:=t1/(65536\*18.2);

time2:=t2/(65536\*18.2);

writeln('Время выполнения умножения на Pascal: ',(time2-time1):6:12);

end;

Procedure Mul\_Asm(N:word);

var Proisv:integer;

Begin

t1:=0;

t2:=0;

asm

in al,40h

mov bl,al

in al,40h {Считываем начальное значение тиков от таймера}

mov bh,al

mov t1,bx

xor ax,ax

xor bx,bx

@@q: {Выполняем сложение}

inc dx

mul dx

loop @@q

in al,40h

mov bl,al

in al,40h {Конечное количество тиков}

mov bh,al

mov t2,bx

end;

Time1:=t1/(65536\*18.2);

time2:=t2/(65536\*18.2);

writeln('Время выполнения умножения на Assembler: ',(time2-time1):6:12);

end;

Begin

clrscr;

{Определяем вид операции}

textcolor(11);

Writeln('Сделайте выбор операции и нажмите клавишу ENTER:');

Writeln('1) Сложение');

Writeln('2) Умножение');

Readln(vid);

textcolor(10);

Case vid of

1: begin

write('Введите сумму ряда, не более 2-х байт (65535: )');

Readln(NN);

Writeln;

textcolor(3);

Add\_Pascal(NN);

Add\_Asm(NN);

end;

2: begin

write('Введите произведение ряда, не более 2-х байт (65535: )');

Readln(NN);

Writeln;

textcolor(3);

Mul\_Pascal(NN);

Mul\_Asm(NN);

end;

end;

readln;

End.

# §7. Описание программы

Программа написана на алгоритмическом языке Pascal. Состоит из двух блоков. В каждом блоке две процедуры: первая – это выполнение операции на языке Pascal, а вторая – выполнение этой же операции на языке Assembler.

*Первый блок.*

Состоит из двух процедур, которые выполняют сложение на языке Pascal и Assembler. Производится суммирование ряда чисел. Измерение времени происходит с помощью микросхемы таймера 8253. Сначала считываем начальное число тиков, затем выполняем операцию и снова считываем число с микросхемы. Разница, разделенная на произведение 65535\*18,2 и будет временем выполнения операции.

asm

in al,40h

mov bl,al

in al,40h {Считываем начальное значение тиков от таймера}

mov bh,al

mov t1,bx

end;

##### *{Выполняем операцию}*

asm

in al,40h

mov bl,al

in al,40h {Конечное количество тиков}

mov bh,al

mov t2,bx

end;

Time = t2-t1/(65536\*18.2); {ВРЕМЯ ВЫПОЛНЕНИЯ ОПЕРАЦИИ}

*Второй блок.*

Аналогичен первому блоку, с той лишь разницей, что здесь выполняется другая операция – вычисление факториала.

# 

# §8. Экспериментальные данные

Для проведения эксперимента вводятся числа:

*10;100;1000;5000;10000;15000;20000;25000;30000;35000;50000;60000;65000;*

Данные вводятся как для суммирования ряда, так и для вычисления факториала.

# §9. Полученные результаты

Все вычисления выполнялись на микропроцессоре AMD Athlon 1000 MHz

Средняя температура процессора при вычислении 64 °С

*Вычисление суммы ряда.*

|  |  |  |
| --- | --- | --- |
| **СЛОЖЕНИЕ** | | |
| ***Введенное знвчение*** | Pascal | ***Assembler*** |
| **10** | 1,34143E-05 | 1,84447E-05 |
| **100** | 1,67679E-05 | 1,34143E-05 |
| **1000** | 4,19198E-05 | 2,17983E-05 |
| **5000** | 0,000157618 | 5,19805E-05 |
| **10000** | 0,000301822 | 9,22234E-05 |
| **15000** | 0,000444349 | 0,000132466 |
| **20000** | 0,00059023 | 0,000171033 |
| **25000** | 0,000734434 | 0,000211276 |
| **30000** | 0,000875284 | 0,000249842 |
| **35000** | 0,001021165 | 0,000290085 |
| **50000** | 0,001483959 | 0,000410814 |
| **60000** | 0,001742185 | 0,00050136 |
| **65000** | 0,001884712 | 0,000528189 |



*Вычисление факториала.*

|  |  |  |
| --- | --- | --- |
| **УМНОЖЕНИЕ** | | |
| ***Введенное знвчение*** | ***Pascal*** | ***Assembler*** |
| **10** | 1,50911E-05 | 1,34143E-05 |
| **100** | 2,85054E-05 | 1,34143E-05 |
| **1000** | 0,000159295 | 2,51519E-05 |
| **5000** | 0,000749525 | 7,2102E-05 |
| **10000** | 0,001482282 | 0,000132466 |
| **15000** | 0,002216716 | 0,000191154 |
| **20000** | 0,002994747 | 0,000251519 |
| **25000** | 0,003687261 | 0,000338712 |
| **30000** | 0,00443511 | 0,000368894 |
| **35000** | 0,005157806 | 0,000429258 |
| **50000** | 0,007403028 | 0,000608675 |
| **60000** | 0,00887525 | 0,000727727 |
| **65000** | 0,009579501 | 0,000786415 |



# §10. Выводы

Начиная анализировать полученные результаты, хотелось бы привести график роста производительности при увеличении числа итераций.



Проанализируем для каждого из двух случаев:

*Сложение.*

При применении языка Assembler рост производительности наблюдается от 10 итераций до 25000 и дальше стремиться к постоянному значению равному ~3,7. Даже такое увеличение производительности заметно скажется на работе некоторых приложений, использующих большие вычисления.

*Умножение.*

С умножением дело обстоит немного иначе: производительность резко увеличивается (от 1 до 10,3) при изменении числа итераций от 10 до 5000 и дальше выравнивается, стремясь к постоянному значению. Можно заметить, что даже при небольшом объеме вычисления наблюдается колоссальный рост производительности. По сравнению со сложением, производительность гораздо выше, хотя время выполнения этих операций в отдельности имеет несущественное различие, но при выполнении этих операций в цикле лидирующее положение занимает умножение.

При имеющихся у нас данных мы можем смело сделать выводы, что компилятор языка Pascal (и многих других языков программирования) расходует ресурсы микропроцессора неэкономно, тратя много времени на различные «лишние» действия, которые приводят, как мы заметили, к заметному снижению производительности.

Проанализируем порядок выполнения операции на языке Pascal, чтобы найти так называемые «лишние» действия.

Вводятся значения и располагаются в оперативной памяти компьютера. Далее мы должны выделить память, так же в ОЗУ, для переменной, в которой будет накапливаться результат (для нашего случая – сумма ряда, либо значение факториала). Затем при выполнении операции мы должны наши входные значения переписать в соответствующие регистры, выполнить операцию и вернуть значение обратно в память. Далее возвращённое значение потребуется для совершения следующей итерации – необходимо опять все значения из ОЗУ переписать в регистры и выполнить операцию и т.д.

Как следствие всех этих операций много времени затрачивается на пересылку данных в регистры для совершения соответствующих операций и возвращение их обратно.

В случае же с языком Assembler мы можем сами распорядиться тем, где хранить промежуточные данные, применяя при этом самую быструю регистровую адресацию, т.е. перемещая значения из одного регистра в другой и не тратя время на бесполезное сохранение промежуточных результатов в ОЗУ. К оперативной памяти обращение происходит всего лишь дважды: первый раз – считываем значения с клавиатуры в память, а затем из памяти в регистр; второй раз – возвращаем результаты в память для дальнейшего использования. Но и этих операций можно избежать, считывая вводимые значения, если это предусмотрено алгоритмом программы, прямо в регистры, избегая попадания их в память. Ввод можно осуществить аналогично: данные из регистров поместить прямо на экран.

Сделаем окончательный вывод: на современных компьютерах, которые имеют почти безграничный потенциал для вычисления при использовании обычного программного обеспечения, применение описанного метода увеличения производительности скажется не так заметно, как хотелось бы. Но в недалёком будущем сложность программных продуктов будет расти, повысятся требования ко времени выполнения поставленных перед ПО задач, тогда производители языков программирования, в частности компиляторов, будут обязаны обращать внимание на такие тонкости как экономия времени при вычислении, будут экономнее и более качественно использовать ресурсы микропроцессорного оборудования.

# §11. Список литературы

1. Юров В. «Assembler». Учебник — СПб.: Издательский дом «Питер», 2001 г. 624с.
2. Юров В. «Assembler: специальный справочник» — СПб.: Издательский дом «Питер», 2001 г. 496с.
3. Корнеев В.В., Киселёв А.В. «Современные микропроцессоры» — М. «Нолидж», 1998г. — 240с.
4. Джордейн Р. «справочник программиста персональных компьютеров тип IBM PC, XT и AT: Пер. с англ. /Предисл. Н. В. Гайского. — М.: «Финансы и статистика», 1991 г. —544 с.
5. Пирогов В. Ю. «Assembler. Учебный курс». — М.: «Нолидж», 2001 г. — 848с.