РОССИЙСКАЯ АКАДЕМИЯ НАУК
Ордена Ленина Институт прикладной математики
им. М.В. Келдыша

В. А. Фисун

Прикладное программирование в задачах математической физики Архитектурные принципы построения ЭВМ

Учебное пособие

Москва - 2007

Аннотация

Первая часть двухсеместрового курса лекций по дисциплине “Прикладное программирование в задачах математической физики” В этом курсе рассмотрены общие вопросы машинных вычислений, связанные с решением задач математической физики: структура и программное обеспечение микропроцессоров, архитектура современных мультипроцессорных ЭВМ, функциональные принципы организации ЭВМ.

V.A. Fisun

Application programming for mathematics physics problem.

Abstract

The first part of two-semester lecture courts on discipline “Application programming for mathematics physics problem physics”. In the course computer organization and architecture are presented.

Предисловие

При современном состоянии математической физики машинный эксперимент как виртуальная модель изучаемых явлений является исключительно важным инструментом познавания мира, а в некоторых областях и единственным способом изучения реальных процессов и явлений. Характерным примером достижений в области вычислительной математики может служить многолетний мораторий на проведение ядерных испытаний: “взрывы ” на ЭВМ оказываются адекватными полигонным экспериментам. Однако, необходимость учета более тонких физических эффектов, расширение круга изучаемых явлений потребовали резко увеличить объем машинных вычислений, а усложнение вычислительных алгоритмов и увеличение размерностей задач математической физики привело к необходимости повысить требования к методам программирования. Поэтому в учебных планах технических университетов организации научных вычислений на ЭВМ уделяется большое внимание.

В настоящее время технология конструирования и производства аппаратного оборудования реализуется индустриальными методами, обеспечивая, в соответствии с законом Мура, удвоение числа элементов на кристалле – чипе и соответствующее увеличение производительности компьютеров каждые полтора года.

“Персональные компьютеры так быстро устаревают, что их нужно постоянно достраивать и обновлять. Компьютер открыт в будущее. У него нет набора неизменных качеств и свойств. Мы не знаем, на что он окажется способным завтра, более того, мы даже не знаем, на что он будет похож. Поэтому мы вынуждены описывать компьютер не как вещь, а как процесс”. /20/

В основе каждой области науки и техники лежат принципиальные законы: сохранения, переноса, Ома, Фарадея и т.д., правила: Лопиталя, правило левой руки для определения направления силы Ампера или, по крайней мере, правило буравчика. Выбор архитектурных решений при конструировании ЭВМ также основывается на ряде принципов, которые не зависят от технологических новаций, например, принципов фон Неймана, локальности вычислений Деннинга, конвейерной, параллельной, потоковой обработки информации. В вычислительной технике действуют законы Мура, Гроша, Амдала, учитываются постулаты Минского, аномалии Биледи и т.д. Технология машинных вычислений базируется на результатах широкого круга научных открытий и инженерных решений.

Особенностью данного курса является комплексное рассмотрение вопросов научных вычислений, связанных с численным решением задач математической физики: архитектура современных микропроцессоров и мультипроцессорных супер-ЭВМ, реализация параллельных алгоритмов и программ, средства программирования для задач математической физики, методы оптимального использования ресурсов персональных компьютеров.

Отбор материала для включения в курс лекций “Прикладное программирование в задачах математической физики” произведен с учетом предметной области. Ссылки на используемые материалы приводятся только при текстуальном цитировании, авторы других материалов перечислены в списке литературы.

Данная работа является конспектом лекций со всеми особенностями статуса данного материала. Так, некоторые разделы приводятся конспективно, без объяснений и аргументации. Это связано с ограниченностью объема материала, а также с прямым умыслом. Конспект, по мнению автора, не должен заменять лекции, он только предоставляет студенту исходный материал, который студент должен затем усвоить на занятиях и самостоятельно с помощью других источников знаний.

В процессе прохождения занятий по курсу студенты выполняют ряд лабораторных работ. Для контроля процесса обучения студенты получают задания для самостоятельного изучения характеристик и особенностей программирования современных вычислительных систем и готовят рефераты по заданным темам.

Автор выражает глубокую благодарность М.П. Галанину за большую помощь при написании данной работы.

Введение

В первой части курса “Прикладное программирование в задачах математической физики” излагаются основные архитектурные принципы конструирования ЭВМ и их влияние на эффективность научных вычислений. Содержание каждой главы курса состоит из описания отдельного этапа выполнения программ на ЭВМ, в курсе содержатся также сведения по истории вопроса, тенденции развития и ссылки на литературу для углубленного изучения материала.

В первой главе рассматриваются общие вопросы информатики: информация и средства ее обработки. Приводятся простейшие метрики, примеры измерения количества информации для определения объема памяти при кодировании и передаче информации. Перечисляются теоретические проблемы оценки информации. За основные виды ЭВМ принимаются аналоговые вычислительные машины и цифровые вычислительные машины. Эволюционная классификация ЭВМ используется для иллюстрации истории развития отечественной вычислительной техники и программирования.

Вторая глава посвящена принципам классификации ЭВМ. Перечисляются принципы фон-Неймана, являющиеся основой архитектур всех современных компьютеров. Реализация ЭВМ традиционной фон-неймановской архитектуры позволяет повышать производительность микропроцессоров в два раза каждые полтора года, однако такие узкие места этой архитектуры, как программное управление вычислениями, ограничивают возможность выполнения распределенных вычислений. Обсуждаются вопросы влияния архитектуры ЭВМ на программирование приложений.

В третьей главе рассматриваются номенклатура запоминающих устройств, физические принципы методов хранения информации. Память ЭВМ представляется в виде многоуровневой пирамиды, вверху которой располагаются сверхоперативная, затем оперативная память, внизу массовая память. Оперативная память является одним из важных факторов эффективности вычислений, так как для реализации одной арифметической или логической команды могут потребоваться четыре обращения к оперативной памяти. Поэтому детально рассматриваются различные схемы ускорения выборки данных и команд такие, как расслоение памяти и механизмы кэш-памяти. Виртуальная память служит для организации мультипрограммной работы ЭВМ, а также для облегчения программирования приложений.

Четвертая глава описывает один из важных методов повышения быстродействия ЭВМ – конвейеризацию работ и учет особенностей совмещенного по времени выполнения команд. Приводятся оценки повышения производительности вычислений для конвейерных арифметико-логических устройств, параметры векторно-конвейерных вычислителей. Рассматриваются статический и динамический механизмы предсказания переходов для конвейеризации потока команд и их влияние на эффективность выполнения вычислений.

В пятой главе классифицируются наборы команд различных процессоров: сокращенный и полный наборы команд, многофункциональные и сверхдлинные команды. Рассматриваются скалярные, суперскалярные вычислители, особенности и достоинства архитектуры современных микропроцессоров - EPIC с гибкими сверхдлинными командами и особенности выполнения программ для различных архитектур микропроцессоров. Описывается эволюция микропроцессоров серии Intel, приводится перечень микропроцессоров современных серий.

В шестой главе описываются метрики измерения производительности ЭВМ. Приводятся методы оценки пиковой и реальной производительности вычислительных систем, перечисляются стандартные пакеты для измерения производительности. Описывается структура топ-листа рейтинга наиболее производительных ЭВМ.

1. Глава 1
Информатика

1.1. Бинарные коды и информация

Одно из самых распространенных определений ЭВМ как устройства для обработки информации есть заблуждение или, по крайней мере, сильное преувеличение. Электронная вычислительная машина – это просто оборудование для хранения и переработки текстов. Текст – это последовательность символов в некотором алфавите. В вычислительной технике чаще всего используется символы бинарного алфавита. Наименьшая единица этого текста - элемент двухсимвольного алфавита - в вычислительной технике называется бит – bit. Термин является аббревиатурой выражения "binary digit" (двоичный разряд) и представляется сочетанием чисел – 0 и 1 (символов ‘0’ или ‘1’, ‘+’ или ‘-‘, слов 'да' или 'нет'). Для измерения бинарного текста используется обычно его объем, выраженный в байтах. Байт – единица измерения текстов, равная 8 битам. Такая разметка кодирует текст символами алфавита, состоящего из 256 (2^8) кодов. В вычислительной технике принято измерять объемы текстов при их хранении в двоичной системе счисления, т.е. Килобайт – это 2^{10} , Мегабайт – это 2^{20} , Гигабайт – 2^{30} байтов и т.д.

Для измерений в цифровых коммуникационных системах - например скорости передачи текстов - используются такие же метрики. Бит в секунду – bits per second, bps – единица измерения скорости передачи (компьютерной обработки) текстов с учетом всех передаваемых битов данных, как содержательных, так и служебных. Для измерения скорости передачи только полезной информации используется показатель "символы в секунду" (characters per second, cps). Для указания объемов текстов в телекоммуникациях используется уже десятичная система.

Электронному тексту или его фрагментам может быть определена семантика, придающая тексту новые качества. Так, разметка текста на 32-битовые элементы и принятие соответствующих соглашений об интерпретации разрядов в элементе могут означать запись массива вещественных чисел. ‘‘Машинное слово’’ – квант кода, считываемый из оперативной памяти одной операцией, в ряде ЭВМ может трактоваться как команда или как ее аргумент. Байтовая разметка текста широко используется для кодировки алфавитно-цифрового текста, причем каждому символу текста отводится байт целиком (стандарт ANSII). В один Гигабайт текста может быть переписано содержимое грузовика книг, в десять Терабайтов – книжный фонд большой библиотеки.

К текстам, которым придается семантика, может относиться понятие ‘‘информация’’(information).

В прикладном, телекоммуникационном аспекте информация трактуется как сведения, разъяснение, информирование, изложение, сбор и обработка, хранение, прием и передача сигналов и сообщений. Толковый словарь компьютерных терминов А. Синклера определяет информацию как данные, организованные таким образом, что имеют смысл для имеющего с ними дело человека. Понятие информации нельзя считать лишь техническим термином. Информация - это фундаментальная философская категория. Вот некоторые концепции данной области знаний.

М. Брой называет информацией абстрактное содержание, семантику какого либо высказывания, описания, указания, сообщения или известия. Конкретная, внешняя форма информации называется “представлением”. Одна и та же информация может иметь различные формы представления. Переход от представления к абстрактной информации, к значению представления, называется “интерпретацией”. Наконец, Брой вводит понятие “понимание”, как процесс установления соотношения между содержащейся в представлении информацией и окружающим миром, т. е. толкование значения информации /23/.

Категория “понимание” Броя очень близко к понятию “знания”, для которого также предложено несколько определений. Например, знания определяются как усвоенная человеком структурированная информации. Или знаниями называют накопленную реципиентом информацию, которая может быть использована для получения новых фактов. А если в процессе понимания структуры информации будут постигнуты и принципы организации этих структур, то это уже “мудрость” (wisdom) /Ackoft R.L./.

По определению президента Международной академии информатизации И. Юзвишина "информация - это фундаментальный генерализационно-единый безначально-бесконечный законопроцесс резонансно-сотового, частотно-квантового и волнового отношения, взаимодействия, взаимопревращения и взаимносохранения (в пространстве и времени) энергии, движения, массы и антимассы на основе материализации и дематериализации в микро - и макроструктурах Вселенной" /18/.

Отец кибернетики Н. Винер говорил: ‘‘информация есть информация, а не материя и не энергия’’, т.е., она не принадлежит миру вещей и на неё не распространяется законы сохранения.

Дискуссии ученых о философских аспектах информации показали несводимость информации ни к одной из предложенных категорий. Корогодины сравнивают положение исследователей в данной области с ситуацией в некой религии, адепты которой признают наличие у Бога нескольких имен, но ни одно из них нельзя произносить вслух /19/.

1.2. Измерение информации

Один из конструктивных методов рассмотрения содержание понятия "информация" основан на теории множеств и этот подход предложил Р. Хартли (R. Hartley) в 1928 г. Он обосновал использование логарифмической меры для измерения количества информации. Текст из n символов двоичного алфавита (телеграфных знаков: точек и тире) может принимать вид одного из $N=2^n$ возможных сообщений, однако время передачи одного такого сообщения пропорционально его длине – n, двоичному логарифму от числа возможных сообщений. Хартли предложил использовать для измерения количества информации не число всевозможных сообщений N, а функцию от него: логарифм. Пусть имеется N состояний объекта или N опытов с различными, равновозможными состояниями системы. Для кодировки состояний можно использовать числовые коды в той или иной системе счисления. Естественно, что число разрядов в кодах при этом зависит от выбранной системы счисления. Допустим, что имеется n-разрядный код, использующий m различных символов. Так как количество всевозможных кодовых комбинаций будет $N=m^n$, то при равновероятности появления любой из них, Хартли предложил измерять количество информации в этом коде по формуле: $I= \log N = \log m^n = n*\log m$. Если в качестве основания логарифма принять m, то I = n. Так, для двоичной системы счисления количество информации в n-разрядном двоичном коде по Р.Хартли ($I= n *\log_2 2 = n$) будет равно n единиц количества информации, которые для $\log_2 2$ называются “битами”. В данном случае количество информации - n битов в бинарном тексте будет равно объему данных этого текста и числу двоичных разрядов в любом коде из N возможных. Если за основание логарифма принять e, то единица измерения информации будет называться “натом”: I=ln N (нат), если десять, то “дитом”: I=lg N (дит).

Для текстов – числовых кодов зависимость количества информации от основания системы счисления можно рассмотреть на примере. Исчислим, сколько бит информации несет каждое двузначное десятичное число XX со всеми значащими цифрами, измеряя десятичные цифры двоичной мерой. Так как таких чисел может быть всего 90 (10 - 99), то количество информации будет $I=\log_2 90$ или приблизительно I=6.5. Отсюда можно сделать вывод о том, что сообщение в одну десятичную единицу несет в себе приблизительно в 3.32 больше информации, чем в одну двоичную единицу (чем $\log_2 2=1$), и что для представления рассматриваемого двузначного десятичного числа в двоичном коде нужно двоичное слово из 7 разрядов. Действительно, десятичное число 99 имеет в двоичном виде код 1 100 011. Семиразрядный двоичный код кодирует больше 99 чисел, причем мера чисел, больших чем 99, но меньших чем двоичное число 1 1111 1111, известна и равна приблизительно 0.5 бита (7 – 6.5).

Справедливо утверждение Хартли: если во множестве $X=\{x_1, x_2, ..., x_n\}$ выделить произвольный элемент $x_i \in X$, то для того, чтобы найти его, необходимо получить не менее $\log_a n$ (единиц) информации. Так, для определения минимального числа вопросов для самого худшего случая, которые следует задать для угадывания заданного элемента из некого множества, следует перенумеровать элементы этого множества в двоичной системе счисления. Тогда число двоичных разрядов в представлении и есть искомое число вопросов. Вопросы для угадывания можно задавать так: “очередной разряд в представлении заданного числа равен 1?”.

К.Шеннон (C.Snannon) предложил в 1948 г. другую, более общую формулу определения количества информации, учитывающую возможную неодинаковую вероятность некоторых состояний в наборе из N сообщений.

$I = - (P1* \log P1 + P2*\log P2 + \ldots + Pn*\log Pn)$,

где Pi – вероятность того, что именно i-е сообщение выделено в наборе из N сообщений. Если все $Pi = 1/N$, то формула Шеннона превращается в формулу Хартли.

Существуют и другие определения измерения количества информации, например “алгоритмическое”. А.Н. Колмогоров предложил определять сложности системы по ее “программному” описанию. Это определение заключается в том, что "количество информации в тексте приблизительно равно логарифму длины самой короткой программы, которая может напечатать этот текст". Иными словами алгоритм, печатающий число "100000", должен быть проще алгоритма, печатающего число "234769". Этот принцип положен в основу компьютерных программ-архиваторов, сжимающих файлы за счёт сокращения последовательностей одинаковых символов. Отсюда же следует, что сложность программ возрастает быстрее, чем количество кодирующей их информации.

Для субъективного измерения семантических свойств информации используется понятие “тезауруса” – совокупность сведений, которым располагает реципиент. В зависимости от компетентности пользователя смысловое содержание поступающей информации может быть понятно пользователю и оно даст ему новые сведения, которые повлияют на его тезаурус. Для некомпетентного пользователя (реципиента с другим тезаурусом) аналогичная информация может оказаться бессмысленной. Наконец, поступающая информация может оказаться бесполезной для пользователя, если она ему уже известна. Итак, количество семантической информации, количество “знаний” некоторого текста является величиной относительной. Она зависит от соотношения между смысловым содержанием поступающей информации и тезауросом пользователя. Мерой количества поступившей семантической информации может служить коэффициент содержательности сообщения – отношение количества воспринятой реципиентом информации ко всему объему сообщения.

1.3. Основные виды ЭВМ

Электронные Вычислительные Машины (ЭВМ) принято разделять на аналоговые или непрерывные машины и дискретные машины, реализующие цифровые вычисления.

В машины первого типа информация может быть введена в виде значений физических величин: электрические сопротивления в виде резисторов, емкости в виде конденсаторов т.д., а результаты обработки информации могут выдаваться на самопишущие устройства или на экран осциллографа. Машины этого типа называются моделирующими или Аналоговыми Вычислительными Машинами (АВМ) или машинами непрерывного действия. Электронных элементов в АВМ может не быть вовсе, их включили в класс ЭВМ в качестве примера вычислителей, альтернативных дискретным вычислителям.

Схема такой машины строится как модель изучаемого процесса. Спидометр на щитке автомобиля - пример тривиальной АВМ. Этот прибор - вольтметр, измеряющий силу тока, выработанного генератором, вращающимся вместе с колесом авто. Здесь методика измерения расстояния, проходимая машиной за единицу времени, подменяется электромеханической моделью. В 1911 г. академик А.Н. Крылов построил первую в мире машину для решения дифференциональных уравнений. Эта технология получила развитие для решения задач, называемых сейчас задачами математической физики. Существовали аналоговые интеграторы (1952 г. выпускались серийно ИГ-1, ИГ-3, ИГ-4), в которых параметры интегрируемых функций задавались сопротивлениями, емкостями на специальной коммуникационной доске. Автопилоты самолетов также создавались первоначально на принципах АВМ. Машины этого типа как самостоятельный класс ЭВМ в настоящее время не рассматриваются, но могут входить в состав систем реального времени. Достоинство этой архитектуры в быстродействии (иногда в мгновенном срабатывании), в относительной дешевизне. К недостаткам следует отнести: низкую точность результатов, сложность программирования - настройки. АВМ сохраняют значение в качестве узлов в гибридных системах. Аналоговая элементная база востребована в нейросетевой архитектуре.

Другой, наиболее распространенный тип ЭВМ - цифровые вычислительные машины (ЦВМ) или машины дискретного действия. В машинах такого типа вся информация представляется в виде цифровых кодов, и ее обработка производится последовательностью арифметических и логических операций. ЦВМ иногда называют универсальными, имея в виду сферу их использования, но существуют различные методы их классификации, в том числе и по сферам приложения. Можно считать, что термины: "универсальная ЦВМ", "ЭВМ", "вычислительная машина", "компьютер", "вычислительная платформа" - эквивалентны. В этом классе ЭВМ есть подраздел специализированных ЭВМ таких, как бортовые, сконструированные с учетом специфики выполняемых функций и рабочей среды.

1.4. Эволюционная классификация ЭВМ

Одна из форм классификации ЭВМ - по "поколениям" - связана с эволюцией аппаратного и программного оборудования, причем, основным классификационным параметром является технология производства ЭВМ. Классификация иллюстрируется этапами развития отечественной техники, что дает возможность перечислить хотя бы основных творцов отечественной информационной технологии. История отечественных исследований в данной области пока малоизвестна. Это связано с тем, что работы в данной области длительное время носили закрытый характер. В России (в СССР) начало эры вычислительной техники принято вести от 1946 г., когда под руководством Сергея Алексеевича Лебедева был закончен проект малой электронной счетной машины (МЭСМ - 50 оп/сек, ОЗУ на 63 команды и 31 константы) В 1950/51 гг. она пущена в эксплуатацию. Далее приводятся только некоторые крупные отечественные достижения в области вычислительной техники.

Первое поколение ЭВМ /1946-1957 гг/ использовало в качестве основного элемента электронную лампу. Быстродействие их не превышало 2-3 т. оп/сек, емкость ОЗУ - 2-4 Кслов. Это ЭВМ: БЭСМ-1 (В.А. Мельников, 1955 г.), Минск-1 (И.С. Брук 1952/59 гг.), Урал-4 (Б.И. Рамеев), Стрела (Ю.Я. Базилевский 1953 г.), М-20 (М.К.Сулим 1860 г.). А.Н. Мямлиным была разработана и несколько лет успешно эксплуатировалась "самая большая в мире ЭВМ этого поколения" - машина Восток. Программирование для этих машин: однозадачный, пакетный режим, машинный язык, ассемблер.

В ЭВМ **второго поколения** /1958-1964 гг/ элементной базой служили транзисторы. Отечественные: Урал-14, Минск-22, БЭСМ-4, М-220, Мир-2, Наири и БЭСМ-6 (1 млн. оп/сек , 128 Кслов), Весна (В.С. Полин, В.К. Левин), М-10 (М.А. Карцев). ПС-2000, ПС-3000, УМШМ, АСВТ, Сетунь. Программирование: мультипрограммный режим, языки высокого уровня, библиотеки подпрограмм.

Элементная база ЭВМ **третьего поколения** /1965-1971 гг/ - это интегральная схема - логически законченный функциональный блок, выполненный печатным монтажом. Отечественные ЭВМ этого поколения ЭВМ ЕС (Единой Системы): ЕС-1010, ЕС-1020, ЕС-1066 (2 млн. оп/сек , 8192 Кбайт) и др. Программирование: мультипрограммный, диалоговый режимы, ОС, виртуальная память.

В 1996 г. в России работали 5 тысяч ЕС ЭВМ из 15 т., установленных во времена СССР. НИИЦЭВТ на базе комплектующих компонентов IBM/390 разработал 23 модели производительностью от 1.5 до 167 Мфлоп (ЕС1270, ЕС1200, аналоги серверов 9672). IBM предоставляет также лицензионные программные продукты (ОС-390). Используются в России для сохранения программного задела прикладных систем (проблема наследия ЕС ЭВМ).

ЭВМ **четвертого поколения** /1972-1977 гг/ базируются на "больших интегральных схемах" (БИС) и "микропроцессорах". Отечественные - проект "Эльбрус", ПК. Программирование: диалоговые режимы, сетевая архитектура, экспертные системы.

ЭВМ **пятого поколения** /начиная с 1978 г/ используют "сверхбольшие интегральные схемы" (СБИС). Выполненные по такой технологии процессорные элементы на одном кристалле могут быть основным компонентом различных платформ - серверов: от супер-ЭВМ (вычислительных серверов), до интеллектуальных коммутаторов в файл-серверах.

На этом поколении технологические новации приостанавливаются и в восьмидесятые годы в ряде стран появляются проекты создания новых вычислительных систем на новых архитектурных принципах. Так, в 1982 году японские разработчики приступили к проекту "компьютерные системы пятого поколения", ориентируясь на принципы искусственного интеллекта. Этот проект получил название “великий вызов”, он оказал большое влияние на развитие всей вычислительной техники. В СССР под руководством А.Н. Мямлина /15/ в рамках такого проекта велась разработка вычислительной системы, состоящей из специализированных процессоров: процессора ввода/вывода, вычислительного, символьного и архивного процессоров. В 1991 году японское министерство труда и промышленности принимает решение о прекращении работ по компьютерам пятого поколения; вместо этого запланировано приступить к разработке компьютеров шестого поколения на основе нейронных сетей.

В настоящее время в России создаются мультисистемы на базе зарубежных микропроцессоров: вычислительные кластеры (НИИЦЭВТ), супер-ЭВМ МВС-1000 (В.К. Левин, А.В. Забродин). Под руководством Б.А.Бабаяна проектируется микропроцессор Мерсед-архитектектуры. В.С. Бурцев разрабатывал проект супер-ЭВМ на принципах потоковых машин.

Эволюция отечественного программного обеспечения непосредственно связана с эволюцией архитектуры ЭВМ. Первая Программирующая Программа - ПП, Интерпретирующая Система - ИС создавались для М-20 (ИПМ, школа М.Р. Шуры-Буры). Для ЭВМ этого семейства были реализованы компиляторы с Алгола: ТА-1 (С.С. Лавров), ТА-2 (М.Р. Шура-Бура), Альфа (А.П. Ершов).

Для БЭСМ-6 создан ряд операционные системы: от Д-68 до ОС ИПМ (Л.Н. Королев, В.П. Иванников, А.Н. Томилин, В.Ф.Тюрин, Н.Н. Говорун, Э.З. Любимский).

Под руководством С.С. Камынина и Э.З. Любимского был реализован проект Алмо: создание машинно-ориентированного языка и на его базе системы мобильных трансляторов.

В.Ф. Турчин предложил функциональный язык Рефал. Системы программирования на базе этого языка используются для описания алгоритмов текстовых преобразований при реализации трансляторов, конверторов, а также в исследованиях в области мета - вычислений.

2. Глава 2
Принципы фон-Неймана

2.1. Классическая ЭВМ

Принято считать, что основные архитектурные принципы построения электронных цифровых (дискретных) вычислительных систем (ЭЦВМ) были впервые опубликованы в 1946 г в отчете А. Беркса (A. W. Burks) , Г. Голдстейна (H. H. Goldstein), Дж. Неймана (John Louis von Neumann): "Предварительное обсуждение логического конструирования электронного вычислительного устройства". Классическая (Von Neumann architecture) ЭВМ имеет следующую структуру:

АЛУ + УУ <кд...кд.....кд > ОЗУ, где

- ОЗУ (Оперативное Запоминающего Устройства, ММ – Main Memory) - память для хранения программ и данных. Можно представлять эту память как таблицу, каждая строка которой имеет свой порядковый номер и может содержать команду или число в двоичной системе счисления.

- АЛУ (Арифметико - Логическое Устройство, ALU – Arithmetic and Logic Unit) - устройство, которое выполняет операции над числами или битовыми данными. Аргументы и результаты операции считываются и записываются из (в) ОЗУ.

- УУ (Устройства Управления, CU – Control Unit) - устройство, которое последовательно выбирает команды из ОЗУ, дешифрирует их и организует выполнение заданных операций в АЛУ.

- <кд.....кд.....кд > на схеме обозначают коммуникационную среду – шины для пересылки последовательности команд и данных. Данные как читаются из ОЗУ, так и туда же записываются. В большинстве ЭВМ команды только считываются из ОЗУ.

В приведенной схеме не отражены устройства ввода/вывода информации, блоки массовой памяти для постоянного хранения информации.

Совокупность АЛУ и УУ принято называть процессором (ЦПУ, CPU – Central Processing Unit). Для обозначения полного вычислительного комплекса используется слова ЭВМ, ПЭ. (По словарю А. Синклера "processor" - блок компьютера, выполняющий вычислительные действия). В современных микропроцессорах, процессор размещается на одном кристалле (чипе), это: УУ + АЛУ + набор регистров + кэш память.

Принципы построения ЭВМ данной архитектуры признаны классикой и получили название фон-неймановских принципов. В работе /7/ эти принципы перечисляются в порядке, описываемом далее.

2.1.1. Принцип программного управления вычислениями

Первый и основной принцип фон-Неймана заключается в том, что ЭВМ выполняет вычислительные директивы последовательно в соответствии с заданной программой.

Программа ЭВМ состоит из команд, хранимых в Оперативном Запоминающем Устройстве. Каждая команда задает единичный акт преобразования информации. Устройство Управления поочередно выбирает команды программы и выполняет предписанные в них дискретные вычисления. В любой момент времени работы ЭВМ выполняется только одна команда программы. Пусть принятый уровень абстракции предполагает считать единичными актами преобразования информации, команды ЭВМ, исполняющие арифметические операции. Тогда алгоритм вычисления площади трапеции с основаниями A и B, высотой H {S=0.5*(A+B)*H} можно представить в виде последовательности элементарных вычислений - арифметических команд ЭВМ (трехадресных).

Команды	Комментарий
+, A, B,P1;	P1=A+B
*, P1,H,P1;	P1=P1*H
*,P1,’’0.5’’,S;	S=P1*0.5.

Но, в общем случае, элементарным шагом вычислений можно считать вычисления вида: P = f(A), где A – число, а f - функция вида sin или sqrt (корень квадратный) и т.д. В качестве функции f можно выбрать функцию редукции (например, нахождение суммы элементов вектора), тогда аргумент функции A – вектор. Если в качестве аргументов арифметических операций допустить использование векторов, то единичным актом преобразования информации может быть поэлементная операция сложения (или умножения) векторов.

Итак, при обсуждении архитектуры дискретных вычислителей всегда необходимо уточнять семантику элементарных актов преобразования информации, команд (инструкций) ЭВМ.

2.1.2. Принцип условного перехода.

Принцип условного перехода заключается в обеспечении возможности перехода в процессе вычислений на тот или иной участок программы в зависимости от промежуточных, получаемых в ходе вычислений, результатов.

Команды условного перехода могут нарушать последовательный порядок выборки команд программы и указывать команду для последующего выполнения в случае выполнения условий заданного соотношения. Так, определение максимального числа может быть выполнено программой:

```
    MAX = B
    IF (A<B)  GOTO L
    MAX =A
L ..............
```

Команды перехода позволяет реализовывать в программах ЭВМ циклы с автоматическим выходом из них.

2.1.3. Принцип хранимой программы

Принцип заключается в том, что команды представляются в битовой форме и хранятся в том же Оперативном Запоминающем Устройстве, что и обрабатываемые данные.

ОЗУ можно представить состоящим из последовательно пронумерованных слов – ячеек памяти. Пусть каждая команда или обрабатываемое число занимает ровно одну ячейку памяти ЭВМ. Над программой можно производить арифметические действия, изменяя ее динамически. Пусть в ОЗУ по адресу A размещен первый элемент вектора a, по адресу B – первый элемент вектора в; для размещения первого элемента вектора c – результата поэлементного суммирования векторов a+в отведено место в памяти, начиная с адреса C. Тогда, программа суммирования n элементных векторов может иметь вид:

```
Команды                 Комментарий
   i = 0;
L: +, A, B, C;   Суммирование первых (очередных) элементов векторов
   ^ , L,  K,  L;  Модификация команды L для сложения очередных элементов
   +, i, “1”, i;     Коррекция счетчика
   IF (i < n)  GOTO L;  Проверка на завершения цикла
   L = M;          Восстановление  команды L для повторной работы цикла
.....................................
K: 0, 1, 1, 1;      Поле констант
M: +, A, B, C;
```

В данном примере ^ обозначает операцию целочисленного сложения кодов.

2.1.4. Принцип использования двоичной системы счисления

Этот принцип заключается в использовании двоичной системы счисления для представления чисел и команд и реализации аппаратного оборудования ЭВМ на элементах бинарной логики.

Элементарной единицей кодирования информации является бит, принимающий одно из двух значений 0 или 1. В двоичной системе счисления представляются целые и вещественные числа, над которыми ЭВМ производит вычисления, команды программ. Технические параметры ЭВМ, такие, как размер ОЗУ и величина строки ОЗУ, число банков расслоенной памяти и т. д., измеряются обычно в единицах, кратных степени двойки: 2, 4, 8, 16, 32, 64, 128, ... 2^n , причем нумерация параметров производится, обычно, начиная от нуля.

Программа, хранимая в памяти ЭВМ, задает последовательность выполнения команд, причем в каждый данный момент времени в машине обрабатывается одна команда. Это свойство и есть главная особенность ЭВМ фон-неймановской архитектуры.

Единичный шаг преобразования информации можно трактовать как двуместную арифметическую или логическую операцию, процедуру извлечение корня, а также векторную операцию, например, операцию сложения векторов. Последняя трактовка единичного акта шага работы ЭВМ вносит дополнительную сумятицу в классификатор ЭВМ по Флинну при идентификации векторных машин.

Возможность динамической модификации программ, допустимая в силу третьего принципа, противоречит философии структурного программирования. Более того, в некоторых ЭВМ (Эльбрус) ОЗУ содержит теги: разряды, уточняющие тип хранимой информации, что позволяет предотвратить попытки передачи управления на данные в случае ошибки - использования текстов программ в качестве данных. М.Р.Шура-Бура до сих пор гордится экономией одной ячейки памяти, которую он произвел, используя код некой команды программы в качестве константы, при составлении для системы ИС-2 одной из первой в СССР стандартной программы перевода чисел из десятичной системы в двоичную систему. Исключением могут считаться функциональные языки Лисп, Рефал, для которых концепция единства данных и программ поддержана принципом хранимой программой. Архитектура памяти современных микропроцессоров, определяющая размещение в памяти кода программ и данных и стиль работы с этими элементами, реализуется одним их двух вариантов. Традиционная, фон-неймановская архитектура, без учета семантики хранимых данных, получила название Принстонской архитектурой памяти. В ней правила хранения данных и кода одинаковы. Раздельное размещение кодов программ и данных используется в Гарвардском варианте архитектуры. В современных ЭВМ используются совместно оба варианта памяти. Так, процессорная кэш-память первого уровня разделены на память данных и память инструкций, но запросы к ОЗУ уже обезличены.

Принцип использования двоичной системы счисления для представления информации нарушен при проектировании ЭВМ Сетунь, в которой использована троичная система счисления. Эта система так же, как и десятичная, шестнадцатеричная, признаны нецелесообразными для реализации ЭВМ.

Все современные микропроцессоры имеют фон-неймановскую архитектуру, ключевым параметром которых является производительность, скорость вычислений, которую грубо можно определить как число команд, выполняемых за единицу времени, обычно за секунду.

2.2. Принципы фон Неймана и проблемы программирования

Архитектура вычислителя фон Неймана допускает простую и универсальную аппаратную интерпретацию. Усложнения оборудования при реализации классических вычислителей относятся, главным образом, к необходимости учитывать и преодолевать ограничения объемов и быстродействия оперативной памяти.

Производительность ЭВМ повышается также за счет реализации параллельной и конвейерной схем обработки данных. Так как архитектурные схемы ЭВМ давно отработаны, то технология проектирования микропроцессоров реализуется на уровне индустриального производства. Рост производительности вычислителей в соответствии с законом Мура (G.Moore) - удвоение производительности вычислителей каждые полтора года - обеспечивается, главным образом, технологическим прогрессом: увеличением плотности размещения транзисторов на кристалле, снижением толщины слоев монтажа. В 1953 г. Х.Грош (H.R.J.Grosch) предложил формулу оценки роста стоимости вычислительной системы - С в зависимости от роста производительности системы - Р: $P = Const * C^2$. Это означает, что за удвоенную цену можно купить вычислительную систему с учетверёнными возможностями.

Принципы фон Неймана обеспечивают абсолютное превосходство микропроцессоров перед всеми моделями нетрадиционных, альтернативных фон-неймановской схеме вычислителей, обороты компьютерного рынка постоянно увеличиваются. Однако известно, что уже десятки лет сохраняется тенденция к увеличению в финансовом обороте компьютерного рынка доходов от продажи программных средств по сравнению с доходами от продажи аппаратных средств. Такое нарушение пропорций возникает из-за низких темпов роста производительности программирования при высоких темпах спроса на информационные продукты. В отличие от индустриальных средств проектирования и изготовления вычислительной аппаратуры, программирование до сих пор рассматривается как искусство. Наращивание объемов производства программ на базе современных технологий программирования таких, как использование языков программирование высокого уровня, модульное, объектно-ориентированное программирование и т.д., не обеспечивает потребностей пользователей ЭВМ.

Консерватизм технологий программирования отчасти определен традиционной моделью последовательных вычислений. Математически тривиальная модель вычислителя фон Неймана при реализации усложняется механизмами, особенности которых учитывают системные программисты при разработке операционных систем, оболочек. А прикладным программистам предлагается адаптировать модель последовательного счета к условиям реальной машинной среды. Отображение многомерных массивов, структурированных данных и списков на одномерную, линейную память также должны учитываться технологией программирования. Другой недостаток традиционных ЭВМ: при переходе от программ на языках высокого уровня к объектным кодам происходит значительное усложнение и увеличение этих кодов из-за линейной структуры памяти и примитивного набора команд. Современные мультисистемы усугубляют проблемы управления памятью, добавляют проблемы синхронизации параллельных вычислительных процессов.

3. Глава 3
Архитектура памяти ЭВМ

Производительность ЭВМ определяется технологией исполнения электронных схем и характеризуется максимальной частотой срабатывания ее дискретных элементов. Сейчас это свыше 3 миллиардов тактов в секунду, один такт за каждые 3 наносекунды. Для сравнения: за одну наносекунду луч света проходит расстояние всего в 29.98 см. Современные микропроцессоры могут вырабатывать результаты вычислений арифметических операций над числами с плавающей точкой на каждом такте. При выполнении последовательных арифметических вычислений без потери эффективности оборудования ЭВМ следует обеспечить доставку в арифметическое устройство машины (АЛУ) на каждом такте двух операндов, сохранение результата операции и выборку следующей команды, т.е. реализовать четыре обращения к памяти за такт. Запрос к оперативной памяти ОЗУ за единицей информации “стоит” не менее 30 тактов. Естественно, что архитектура виртуальной машины фон Неймана не может сгладить этот дисбаланс быстродействия между АЛУ и ОЗУ примерно в два порядка. Поэтому все современные микропроцессоры содержат аппаратные схемы для ускорения потоков данных, основанные на использовании буферной сверхбыстрой памяти, конвейерной и параллельной обработки данных. Эти схемы реализуются специальными интерфейсами сопряжения функциональных блоков ЭВМ. Решающее значение для достижения полной загрузки АЛУ, то есть высокой производительности ЭВМ имеет архитектура оперативной памяти и дисциплина работы с ней. При программировании приложений можно ускорить выполнение задач математической физики за счет знания и учета особенностей функционирования оперативной памяти ЭВМ.

3.1. Виды запоминающих устройств

Хранилищем информации в вычислительных системах служит память, состоящая из различных видов запоминающих устройств (ЗУ). ЗУ различаются по ряду признаков: по физическим принципам реализации, по объему хранимой информации, по времени и стилю доступа к информации, и даже по месту расположения и способам сопряжения с процессорами ЭВМ.

Основным, но не всегда главным, параметром ЗУ является размер памяти – максимальное количество информации, которые может быть размещено для хранения в устройстве. Размер памяти принято измерять в байтах, причем, для указания объема памяти используются числа восьмеричной системы счисления.

1 К (Кило) байт = 2^{10} байт (1024 в десятичной системе).
1 М (Мега) байт = 2^{20} байт (1 048 576 в десятичной системе).
1 Г (Гига) байт = 2^{30} байт (1 073 741 824 в десятичной системе).
1 Т (Тера) байт = 2^{40} байт.
1 П (Пета) байт = 2^{50} байт.

Внимание! Префиксы Кило, Мега, и др. в метриках для измерения пропускной способности каналов передачи информации и оценки производительности ЭВМ относятся к десятичной системе счисления. Емкость магнитных дисков (винчстеров) также чаще всего измеряется в десятичной системе. Зависимость значений вышеуказанных префиксов от контекста применения неудобно, так как может вызвать двусмысленное толкование. Одна из попыток унифицировать отношение префиксов к основанию системы счисления заключается в предложении добавлять к соответствующему двоичному префиксу окончание binary, а сами единицы обозначать Ki, Mi, Gi и т. д. Тогда один килобайт памяти будет называться кибибайт. Обсуждаемая двусмысленность не имеет большого практического значения, ибо разница между значениями десятичного миллиона и двоичного миллиона составляет около 5%, а между соответствующими миллиардами – не более 8%.

Элементарной единицей в любом ЗУ является хранилище для одного бита – бистабильная ячейка памяти, которая может принимать только одно из двух значений. В зависимости от физических принципов, положенных в основу реализации хранилища для бита, можно построить нижеследующую цепочку видов запоминающих устройств. Примерно в таком же порядке ЗУ распределены по такому параметру, как плотность записи.

1. Электронные виды ЗУ. Значение бита фиксируется одним из бинарных состояний триггерной транзисторной схемы (статическая электронная память) или наличием (отсутствием) заряда в конденсаторе (динамическая электронная память). Управление процессами записи и чтения информации в таких ЗУ производится электронными схемами, которые размещаются на той же микросхеме, что и наборы элементарных единиц памяти хранилищ информации. Электронная память являются обязательной составной частью любого микропроцессора, на этом виде памяти реализуется оперативное запоминающее устройство - ОЗУ.

2. Оптические виды ЗУ. Значение бита – прозрачная или затемненная частица оптического диска.

3. Электромагнитные виды ЗУ. Здесь значение бита задает намагниченная или нейтральная частица ферросплава магнитного диска, барабана или магнитной ленты.

4. Механические виды ЗУ. Значение бита – дырка или её отсутствие на соответствующей позиции перфокарты или перфоленты. Данный вид ЗУ включен с перечень исключительно из методических соображений, они утратили свое значение так же, как запоминающие устройства на электронно-лучевых трубках или на ферритовых кольцах.

В запоминающих устройствах 2,3,4 видов запись и чтение информации производится сканером, к которому путем механического перемещения носителя информации доставляется требуемая часть носителя. ЗУ данного вида называются Внешними Запоминающими Устройствами – ВЗУ (иногда – периферийными), они предназначаются для долговременного хранения информации.

Только магнитные барабаны и магнитные диски (винчестеры) включаются в постоянный состав оборудования ЭВМ, их носители информации также не отделимы от устройства. Остальные виды ВЗУ могут включаться в состав оборудования ЭВМ, а также могут быть автономными абонентами сетевой среды. Носители информации: диски, ленты, карты являются заменяемыми элементами. Они используется в качестве массовой памяти для долговременного хранения программ и данных вне ЭВМ. Эта память может использоваться для однократной записи и постоянного хранения информации, но большинство типов носителей позволяют производить многократную перезапись.

Номенклатура внешних ЗУ следующая. Первыми ВЗУ были устройства для чтения (записи): перфокарты, перфоленты, перфорированной киноленты, магнитной ленты (МЛ), магнитных барабанов (МБ), магнитных дисков (МД). Затем появляются устройства для флоппи-дисков и оптических дисков, магнитные диски – винчестеры, ВЗУ на электронике - флеш-память.

С 1995 г. установлена следующая классификация семейства винчестеров, магнитных дисковых накопителей с несменяемыми дисками. Первый жесткий диск на 16 Кбайт разработан фирмой ИБМ в 1973 г. Он имел 30 цилиндров (дорожек) по 30 секторов - 30/30, что соответствует калибру винтовки - винчестера. Емкость современных накопителей данного вида превышает сотни гигабайт. Известен прогноз, что к 2030 году на магнитных носителях может быть достигнута плотность записи в 10^9 бит на квадратный дюйм (на 6.45 кв. см).

Магнитные диски: флоппи-диски с заменяемыми носителями, с магнитными мини-дисками, дискетами. Дискеты имеют размеры и емкость: 5,25 дюйма (360 - 1200 Кбайт) и 3.5 дюйма (720 - 1440 Кбайт). Вариации в объеме определяются плотностью записи. Приводы для них могут быть как раздельными, так и комбинированные в одном корпусе.

Оптические дисковые накопители работают со сменными дискетами размером 5.25 дюйма (22 см), емкость которых достигает 10 Гбайт.

Стример (streamer) - лентопротяжный механизм, работающий в инерционном режиме. Высокая плотность записи на ленту обеспечивается отказом от стартстопного режима работы, лента в таких устройствах движется с постоянной скоростью. Различаются форматом используемой магнитной ленты: полудюймовые ленты (девятидорожечные), четвертьдюймовые (QIC - картридж), 8-мм , 4-мм (DAT). Используются для резервного копирования и архивирования данных, емкость таких устройств достигает сотен гигабайт.

Электронные ВЗУ представлены флеш-памятью. Флэш (Flash) – память – это энергонезависимое перезаписываемое ЗУ. Flasn - короткий кадр фильма (запись/считывание производится блоками), вспышка, мигание. In a flash – в мгновение ока (скорость стирания информации). Flashing - засвечивание, прожигание (процесс записи). Физически однобитовая ячейка такой памяти состоит из одного или двух полевых транзисторов с “плавающим” затвором, сохраняющая заряд длительное время, годами (20 – 100 лет).

Многоуровневые ячейки флеш–памяти (MLC – multilevel cell) используют аналоговый принцип записи – в плавающий затвор может заноситься (и считываться) различное количество заряда. Выпуск микросхем с такой технологией, позволяющей хранить в ячейке два, четыре бита еще более повышают емкость памяти. По сравнению с оперативной электронной памятью флеш-память медленнее и имеет ограниченное число циклов перезаписи (от 10.000 до 1.000.000). Флеш-память компактнее механических ВЗУ и потребляет меньше электроэнергии, но значительно дороже.

Электронная память служит для кратковременного хранения информации, она есть память с произвольным доступом (Random Access Memory - RAM), которая допускает обращение к любому ее элементу в произвольном порядке. Она энергозависима, т.е. при отключении питания хранимая в ней информация теряется. Для памяти этого вида главными параметрами являются время доступа и цикл памяти. Время доступа – время от запроса до получения данного или занесение его в память. Цикл памяти – минимальное время между двумя последовательными обращениями к памяти. По физическим принципам организации электронная память разделяется на динамическую и статическую память.

Динамическая память (DRAM) базируется на элементах, способных хранить полученный электрический заряд на конденсаторе. Ёмкость такого конденсатора равна 30 феймоФарад – $30*10^{-15}$ Фарад, конденсатор современной DRAM памяти имеет микроскопические размеры и для его заряда достаточно 40 000 электронов. Однако, характерной особенностью такого оборудования является необходимость поддерживать сохраняемый электрический заряд. Конденсаторы могут самопроизвольно разряжаться со временем и заряд необходимо поддерживать, “освежать” динамически, приблизительно каждые 5 - 18 мс (для этого существует процедура refresh – процедура регенерации информации). Но наличие емкости ограничивает быстродействие схемы. При считывании данных из DRAM памяти заряд также разрушается, поэтому его необходимо обновлять после такой операции. Микросхемы динамической памяти можно размещать на кристалле – чипе с высокой плотностью, они самый дешевый вид ОЗУ, однако, не самый быстрый.

Первые микросхемы памяти данного вида работали на частоте 40 МГц и могли обработать очередной запрос на получение одного бита информации примерно за 120 нс. Скорость выборки определяется не только частотой работы микросхемы, но и таким параметром микросхемы памяти, как тайминг. Тайминг x-yyy - число тактов микросхемы: x – для получения первого бита, y – последующих битов. Тайминг стандартной DRAM памяти – 5-555. Для ускорения выборки разработано несколько модификаций памяти, в которых обращение к любому биту всегда обеспечивает также последующий ускоренный доступ к соседним битам. Время доступа к схемам FPM DRAM – около 60 нс (5-333), SDRAM - менее 7 нс (5-111).

Тайминг показывает локальные особенности микросхемы, для интегральной оценки параметров ОЗУ используется такой параметр, как ско-

рость передачи данных в память или из неё. Микросхемы RAMBUS (Rambus DRAM) - новый, один из самых быстрых видов динамической памяти, который значительно увеличивает пропускную способность памяти. В них предусмотрена "удвоенная" шина передачи данных, частота шины увеличена до 800 МГц, а пропускная способность равна 1,6 Гбайт/с. Для повышения производительности используются двух- и четырехканальные RDRAM, которые позволяют увеличить скорость передачи данных до 3,2 или 6,4 Гбайт/с соответственно. (данные 2006 г)

В статической памяти (SRAM) элементарная ячейка памяти (хранилище бита) представляется триггером – электронной схемой с двумя устойчивыми состояниями на биполярных или МОП-транзисторах (4-6 элементов). Схемы с шестью элементами имеют повышенную устойчивость к внешним источникам излучений и увеличенную производительность. Получив заряд один раз, такая ячейка способна хранить и идентифицировать его произвольно долго, пока поддерживается питание. Память данного вида может работать по формуле 3-111 при высоких частотах, время доступа 4 – 12 нс. Так, микропроцессоры Pentium комплектуются статической оперативной памятью с пакетным конвейерным доступом (Pipelined Burst SRAM) и временем доступа 4.5 – 8 нс. На этом виде памяти реализуются регистровая память и кэш-память - элементы памяти, интегрированные в кристалл вместе с процессором. Время выборки из этой памяти сравнимо с частотой работы микропроцессора, иногда её называют СОЗУ – сверхоперативным ЗУ. По сравнению с ней ячейка динамической памяти содержит только один транзистор, кроме конденсатора, и поэтому она занимает меньше места на кристалле. Такая микросхема в целом потребляет меньше энергии и слабее греется, следовательно, она долговечнее и дешевле. Однако, в настоящее время статическая память SRAM обладает наивысшим быстродействием, она уступает DRAM памяти только по перечисленным выше параметрам. Основная электронная память ЭВМ – ОЗУ реализуется на элементах динамической памяти (DRAM) и располагается на отдельном кристалле.

Для ускорения работы микросхемы памяти используются различные системы управления доступа к данным. Основные способы повышения производительности следующие:

1. Многопортовые системы памяти (multiport memory access).

Многопортовые микросхемы допускают одновременную обработку нескольких запросов к памяти. Так, в двухпортовой памяти дублируются адресные, информационные и управляющие шины. Число возможных параллельных запросов, число портов, определяется коэффициентом размножения шин. Проблемы корректности записи в ячейку памяти одновременно с попыткой прочитать её содержимое решается аппаратной схемой арбитража.

2. Пакетный режим (burst mode) обращения к памяти.

Большинство команд считываются последовательно из соседних ячеек памяти. Последовательная выборка и запись данных является массовой операцией ряда вычислительных алгоритмов. Если обеспечить одновременный

доступ к содержимому соседних ячеек памяти и выбор большого кванта памяти целиком в регистры, то команды и данные последовательных запросов можно брать из этого пакета.

3. Конвейерная (pipelining) обработка запросов к памяти.

Архитектура памяти позволяет начать цикл считывания из памяти, не дожидаясь окончания считывания текущего. При этом очередной доступ выполняется, не дожидаясь окончания предыдущего (если запросы независимы).

Так как память разбита на множество банков с независимыми системами управления, то можно управлять очередью запросов к памяти. Если запрашиваются данные из разных банков, то последующий запрос можно делать, не дожидаясь окончания обработки предыдущего запроса.

4. Чередование адресов (расслоение) памяти (interleaved memory).

Расслоение памяти реализуется для памяти, состоящей из банков с независимыми устройствами управления. Адресация байтов в М банках организуется так, что при регулярной выборке данных повторное обращение к одному банку произойдет через М обращений, поэтому возможно совмещение времени выборки. Такая адресация называется иногда карусельной. Для четырех банков одинаковой емкости: Б1, Б2, Б3, Б4 при расслоении распределение адресов памяти по банкам будет таким.

Адреса в банках	Б1	Б2	Б3	Б 4	
0	0	1	2	3	Распределение первых
1	4	5	6	7	12 адресов памяти
2	8	9	10	11	

При такой схеме расслоения два младших бита в адресе байта определяют банк памяти, из которого по старшим битам адреса будет происходить обращение к байту. Расслоение можно использовать как при пакетном, так и конвейерном доступе к памяти. При конвейерном доступе при М-кратном расслоении и регулярной выборке данных доступ к памяти возможен с интервалом, равным 1/М времени цикла памяти. Но возможны конфликты по доступу, если шаг регулярной выборки коррелируется с числом банков памяти. Обычно коэффициент расслоения выбирается кратным двойки. Поэтому выборку данных с шагом, кратным двойке, следует избегать. Механизм расслоения реализуется как для оперативной памяти, так и для кэш - память, причем выбор коэффициента расслоения для банков кэша - не тривиальная задача.

3.2. Адресная и ассоциативная память

Единицей программно адресуемых данных в оперативной памяти – ОЗУ - является байт. Такой вид памяти, при котором необходимая информация определяется номером строки памяти (байта), называется "адресной". При размещении данных в составных элементах: в полуслове (16 бит – 2 байта),

в слове (32 бита – 4 байта), в двойном слове (64 бита – 8 байтов), - возникает проблема выбора старшинства байтов в этих элементах. Пусть код 32-битового целого числа без знака, записанного в обычном порядке, при котором бит с наименьшим весом записывается последним, состоит из байтов Б1, Б2, Б3, Б4. Порядок записи этого кода, при которой старший байт (Б1) расположен, начиная с младшего адреса памяти, называется big-endian (тупоконечный). Такой порядок принят в микропроцессорах фирм IBM, HP и еще в некоторых других. Тогда целое число “один” будет храниться в двухбайтовом формате в виде 0000 0001, в четырехбайтовом – 0000 0000 0000 0001.

Альтернативный порядок размещения байтов (Б4, Б3, Б2, Б1) – младшие разряды в младших адресах памяти - называется little-endian (остроконечный), он принят фирмами Intel (IA32), VAX и др. Здесь единица будет храниться в двухбайтовом формате записи в виде 0001 0000, в четырехбайтовом формате – 0001 0000 0000 0000.

Достоинства каждого порядка размещения байтов преувеличивается разработчиками оборудования, в котором применятся тот или иной порядок старшинства записи байтов в памяти. Некоторые процессоры поддерживают оба порядка записи, а выбором порядка размещения байтов составных элементов можно управлять программно. Экзотические названия порядков размещения байтов в составных элементах взяты у Д. Свифта, Гулливер которого насмехался над расколом в неком обществе, возникшем по поводу выбора конца вареного яйца, с которого следует его разбивать. При побитовом программировании, например, при формировании логических шкал, следует иметь в виду, что выбранный порядок размещения байтов в составных структурах может не совпадать с порядком нумерации битов в байтах. Содержимое этого восьмибитового кода может интерпретироваться как целое число без знака из диапазона 0-256, как логическая шкала или литера алфавита, например, стандарта ANSII. В коммутационных системах уточняется старшинство битов в байте. Так, целое число 7 может быть закодировано байтом 00000111, а также – 11100000, то при побитовых пересылках байта учитывается (задается) порядок пересылки битов. А если учесть существование схем адресации, при которых составной элемент ОЗУ (страница памяти, например) идентифицируется не первым, а последним байтом, разбираться со схемами адресации памяти на уровне битов непросто. Поэтому к приведенным выше терминам о тупо и остроконечности можно добавить сентенцию того же автора: “Все истинные верующие должны разбивать яйца с правильного конца”.

К составным элементам относятся также символьные строки, причем первый символ имеет наибольший вес, выбор порядка последовательности записи символов строки такой же, как и для чисел.

Из составных элементов можно формировать массивы, тогда ОЗУ будет представляться двумерным массивом. Обращение за содержимым строки производится заданием номера (адреса) нужной строки - Ai. При записи,

кроме адреса строки, указывается регистр, содержимое которого следует записать в эту строку. Пусть в памяти из трех строк хранятся номера телефонов:

1924021
9448304
3336167

Для получения номера телефона второго абонента следует обратиться: load (A2) и получить в регистре ответа 9448304.

Другой вид оперативной памяти – "ассоциативное запоминающее устройство" (АЗУ) - можно рассматривать также как двумерную таблицу, но у каждой строки таблицы есть дополнительное поле, поле ключа. Например:

Поле ключа	Содержимое
Иванов	1924021
Петров	9448304
Сидоров	3336167

После обращение к ассоциативной памяти с запросом: load (Петров) получим ответ: 9448304. Здесь задание координаты строки памяти производится не по адресу, а по ключу - Петров. Но при состоянии ассоциативной памяти:

Ключ	Содержимое
A1	1924021
A2	9448304
A3	3336167

можно получить номер телефона из второй строки запросом: load (A2), по ключу A2. Этот пример служит для иллюстрации возможной имитации адресной памяти на ассоциативной памяти.

Ассоциативная память имеет очевидное преимущество по сравнению с адресной памятью. Однако у нее есть большой недостаток - ее аппаратная реализация при современном состоянии техники невозможна для памяти большого объема. Привлекательность ассоциативной схемы хранения и поиска информации вызвала к жизни целый набор механизмов имитации ассоциативной памяти при помощи адресной. В общем случае задача сводятся к нахождению так называемой хэш-функции f. Эта функция отображения ключа K в адрес памяти (адресной) a = f(K).

Пример тривиальной хеш-функции для организации телефонной книги. Адресная память, выделенная для реализации телефонного архива, делится пополам.

В первой части памяти записаны ключи (фамилии), во второй - соответствующие номера телефонов. Алгоритм работы хэш-функции очевиден: сначала перебором содержимого первой части памяти находится индекс, который соответствует заданной фамилии. Затем по этому индексу из второй части памяти выбирается номер телефона. Такая функция непригодна для систем, критичных к времени выборки.

Опуская промежуточные схемы, реализация быстрой схемы выборки данных из адресной памяти по ассоциативному принципу такова. Опять тривиальная хэш-функция: текст ключа – фамилия - рассматривается как двоичное число и используется в качестве адреса в обычной адресной памяти, в которой хранятся номера телефонов. Обеспечивая скорость выборки данных, равной по характеристикам обычной, адресной памяти, такая функция очень расточительно использует память. Поэтому хэш-функции выбирают, используя особенности множеств, из которых формируются ключи. Так, если ключи – тексты фамилии в байтовой кодировке - разрядность кодов ключей может быть сокращена при сжатии текстовой записи.

3.3. Виртуальная память

Виртуальная память – линейная модель памяти, она же сплошная, плоская (flat), позволяет обращаться к памяти ЭВМ единообразно. Механизм виртуальной памяти позволяет: снять ограничения, связанные с объемом памяти, при разработке алгоритмов; предоставлять программисту область памяти в виде логически непрерывного пространства; способствовать более эффективному управлению физической памятью.

По определению, исполнительный адрес, сформированный в процессоре, может указывать на любой элемент памяти виртуального адресного пространства (байта) из диапазона: 0 – (N-1), где $N=2^n$, а n ассоциируется с разрядностью внутрипроцессорных интерфейсов. Современные процессоры имеют 32-разрядные шины, а некоторые - уже 64-разрядные. Так, для i486 процессоров (архитектура IA32) с 32-разрядной шиной объем виртуального адресного пространства равен 4 Гбайт. Программа, выполняемая на процессоре, имеет право использовать для адресации элементов своего объектного кода и данных в общем случае весь объем виртуальной памяти. Обычно виртуальное пространство программ укорачивается на размер общедоступных системных процедур (библиотек, оболочек, пакетов), к которым обращаются программы всех пользователей. Реальное физическое пространство для отображения всего объема виртуального пространства программы может быть обеспечено только на жестких магнитных дисках, физическая память процессора - ОЗУ, которая используется при выполнении программы, имеет гораздо меньший объем (1-32 Мбайт). Однако, согласно принципу существования в вычислительных приложениях точек сгущения, можно предположить, что программы в течение некоторого времени используют локальные фрагменты текста программы и ее данных, возможно, расположенных на различных участках виртуального пространства.

Поэтому можно использовать физическую оперативную память процессора в качестве буфера, куда будут переписываться требуемые фрагменты программного приложения с жесткого диска по мере необходимости. Фрагментация виртуальной памяти программы может производиться путем разделения её на блоки фиксированной длины, называемые страницами, или на блоки переменной длины, на сегменты. Размер сегментов определяется

структурой задачи, каждый сегмент представляет собой отдельную логическую единицу информации, содержащую совокупность данных или программ. К сегментам можно обращаться по именам, в каждом из них устанавливается собственная нумерация слов, начиная с нуля. В микропроцессорах архитектуры IA32 используется комбинированная схема фрагментации виртуальной памяти: память разделяется на сегменты, которые, в свою очередь, состоят из страниц.

В современных микропроцессорах чаще всего используется только страничная фрагментация памяти. В каждый момент выполнения программы на физическую память в ОЗУ отображаются только те виртуальные страницы, которые используются программой в текущий момент.

Остальные страницы виртуальной памяти хранятся во внешней памяти в накопителях на жестких магнитных дисках. Управление оперативной памятью реализуется путем разбиения виртуальной и физической памяти на одинаковые страницы размером в 4 Кбайта (здесь и далее используются параметры микропроцессоров архитектуры IA32) и такими квантами производится перепись фрагментов виртуального пространства задачи с диска в оперативную память процессора. Такими же квантами производится перепись на диск результатов счета. После переписи фрагмента программ с магнитного диска в ОЗУ возникает проблема соответствия адресов виртуального пространства задачи - виртуальных адресов - адресам реального физического пространства микропроцессора - физическим адресам ОЗУ.

Необходимое при этом отображение виртуальных адресов в физические адреса оперативной памяти обеспечивается специальным механизмом трансляции адресов. При страничной организации памяти виртуальное пространство 4 Гбайт разделяется на блоки – страницы (page) размером в 4 Кбайт, т.е. на миллион виртуальных страниц. Виртуальный 32-разрядный адрес байта представляется в виде двух полей: 20 старших разрядов определяют номер страницы, а 12 младших разрядов - сдвиг, смещение задают номер байта внутри этой страницы. Трансляция адресов заключается в замене номера виртуальной страницы на соответствующий номер страницы оперативной памяти – физической страницы.

Трансляция адресов может производиться при помощи служебной таблицы страниц. Каждой странице виртуальной памяти соответствует строка в таблице страниц. В i строке таблицы для страниц, отображенных на физическую память, хранятся: N страницы физической памяти, которая соответствует данной виртуальной, статус доступа (чтение, запись), признак записи.

Такая схема организации таблицы соответствия виртуальных и физических адресов носит модельный характер и не используется на практике из-за большого объема таблицы, число строк в которой – миллион, большая часть которых хранит признак “отсутствует в ОЗУ”. Так как при мультипрограммном режиме работы на ЭВМ одновременно выполняются несколько программ, то общий размер таких служебных таблиц превысил бы разумные пределы.

Реальные схемы трансляции адресов используют двухуровневую систему таблиц, при которой виртуальный адрес страницы рассматривается как два десятиразрядных поля. Первое поле указывает на строку каталога разделов, служебную страницу оперативной памяти, приданной каждой задаче. Строка каталога, в свою очередь, хранит ссылку на таблицу страниц, индексация которой вторым полем виртуального адреса страницы и дает искомый адрес физической страницы. Таким образом, для выборки заданного байта, в общем случае, необходимо три обращения к оперативной памяти (без использования аппарата кэш-памяти).

Каталог раздела и таблицы страниц также имеют размер страницы, причем таблицы страниц формируются динамически, при обращении к очередному фрагменту виртуального пространства, который еще не представлен в ОЗУ. Формирование новой таблицы страниц производится при инициализации новой строки каталога, число служебных страниц возрастает только при большом разбросе значений виртуального адреса. Такая схема позволяет экономить память для служебных таблиц, плата – дополнительные обращения к памяти.

Механизм трансляции виртуальных адресов оптимизируется также с учетом принципа существования точек сгущения в исполняемых программах. Факт обращения к текущей странице и нескольким предшествующим виртуальным страницам фиксируется в буфере TLB (Translation Localside Buffer). В буфере сохраняются также адреса соответствующих физических страниц, найденных по предыдущей схеме. Процесс трансляции виртуального адреса начинается с обращения к этому буферу. Механизм TLB за один такт обращения к буферу выдает физический адрес страницы, если информация о запрашиваемой виртуальной странице находится в буфере. Время работы TLB совмещается с выборкой заказанной информации из кэш-памяти, обращение к которой производится по содержимому поля смещения.

При обращении к виртуальной странице, которой нет в оперативной памяти, она переписывается в физическую, возможно, предварительно отправив неиспользуемую в данный момент страницу на диск процедурой “своппинг”. Вопрос выбора страницы оперативной памяти для вытеснения из пула рабочих страниц определяется выбранной стратегией подкачки страниц.

Наиболее часто реализуется механизм LRU (Least Recently Used), по которому вытесняется страница с самым “старым” обращением к ней из последних зафиксированных. Модельная реализация этого механизма такова. С каждой страницей пула физических страниц задачи связывается счетчик; все счетчики увеличиваются на единицу через каждую фиксированную единицу времени. При обращении к странице её счетчик обнуляется. При данной стратегии вытесняется страница с максимальным значением счетчика. Следующий вариант реализации данного механизма основан на списковых структурах. Можно устроить очередь из ссылок на страницы пула, пе-

ремещая в хвост очереди ссылку на страницу при каждом обращении к ней (к странице). Тогда в голове очереди будет ссылка на страницу с самым “древним” обращением – кандидат на вытеснение.

Алгоритм замещения страниц “первый вошел, первый вышел” FIFO (First In First Out) обеспечивается вытеснение страницы, находящейся в пуле дольше всех. Реализуется на списковых структурах по аналогии с предыдущей схемой, однако здесь положение страницы в очереди не изменяется при последующих обращениях к странице.

Алгоритм LFU (Least Frequently Used) обеспечивает вытеснение наименее часто используемой страницы, страницы с минимальным числом обращений. Каждая страница имеет счетчик, увеличивающий на единицу свое значение при каждом обращении к странице.

Наконец можно вытеснять страницы произвольным образом, по содержимому датчика случайных чисел.

Некоторые особенности перечисленных стратегий подкачек страниц.
Для наиболее часто реализуемого механизма LRU Э.Таненбаум приводит пример ”патологической ” ситуации. Пусть задача, использующая пять страниц, есть большой цикл, последовательно перебирающий виртуальные страницы: 1,2,3,4,5,1,2,3,4,5,1,.Если такой задаче дать пул из четырех физических страниц, то очередь первых четырех страниц в начале работы задачи имеет вид: <хвост списка> 4-3-2-1< голова списка> страниц. При переходе на пятую страницу вытесним первую страницу (подкачка пятой страницы на поле первой) из головы и очередь страниц будет иметь вид: 5-4-3-2. Переход в начало цикла, на первую страницу, также вызовет подкачку и очередь страниц будет иметь вид: 1-5-4-3. Далее везде, переход программы на новую страницу, всегда вызывает подкачку страницы.

Для алгоритма FIFO: “первый пришёл – первым ушёл” известен пример неэффективного использования ресурсов (аномалия Биледи). Если задача использует пять страниц и обращается к ним в таком порядке: 1-2-3-4-1-2-5-1-2-3-4-5, то в случае выделения ей четырех физических страниц в процессе выполнения задачи произойдет 10 подкачек, а на трех физических страницах памяти задаче потребуется для работы даже на одну подкачку меньше, только девять подкачек.

Алгоритм LFU (Least Frequently Used) может обеспечить сохранение в оперативной памяти страницы с интенсивно используемым кодом – с циклом. Однако, накопленные значения счетчиков будут грузом и при выходе из цикла будут блокировать вытеснение этих страниц. Модификация данной стратегии заключается в одновременном сдвиге всех счетчиков на один разряд в сторону младших разрядов через заданные промежутки времени. Данная схема может несколько скомпенсировать вес счетчиков мертвых страниц.

Описанные механизмы управления физической памятью реализуется средствами операционных систем процессора автоматически, они ”прозрачны”, не “видимы” для прикладного программиста.

3.4. Иерархическая структура оперативной памяти

Структуру памяти ЭВМ принято представлять в виде пирамиды, на гранях которой записана вся номенклатура памяти сообразно ее быстродействию и емкости. Ближе к вершине расположены наиболее быстрые компоненты, в основании пирамиды самые медленные, но зато самые емкие. Вот этот перечень:

1. Регистровая память.
2. Процессорный кэш (кэш первого уровня).
3. Кэши 2, 3 уровней.
4. ОЗУ.
5. Массовая память.

Память 1-4 уровней служит для кратковременного хранения информации, она есть память с произвольным доступом (Random Access Memory - RAM), которая допускает обращение к любому ее элементу в произвольном порядке. Ее реализация основана на электронике, то есть на управлении процессами записи и хранения информации электронными схемами. Для массовой памяти используются внешние запоминающие устройства.

Для ускорения доступа к оперативной памяти используется буферизация данных и объектного кода в специальной сверхоперативной памяти, скорость которой значительно выше основной оперативной памяти. Исходной посылкой для буферизации служит гипотеза о наличии в вычислительном процессе объективных свойств: пространственной и временной локальности (точек сгущения). Процессы стремятся исполнять команды небольшими порциями, которые именуются программными циклами или подпрограммами, используемые ими указатели группируются в информационном пространстве процесса. В этом состоит суть так называемого принципа "локальности" Деннинга [Denning]. Модель локальности состоит в том, что, когда процесс выполняется, он двигается от одной локальности к другой. Локальность в терминах виртуального пространства - набор страниц, которые активно используются вместе. Программа обычно состоит из нескольких различных локальностей, которые могут перекрываться.

Например, когда вызвана процедура, она определяет новую локальность, состоящую из инструкций процедуры, ее локальных переменных и множества глобальных переменных. После ее завершения процесс оставляет эту локальность, но может вернуться к ней вновь. Таким образом, локальность определяется кодом и данными программы. Итак, модель локальности - принцип, положенный в основу работы буферов. Если бы доступ к любым типам данных был случайным, то буферизация была бы бесполезной. Эффект от буферизации можно определить по среднему времени выборки: $t = t2*p + t1*(1-p)$, где $t1$ - среднее время доступа к данным основной памяти, $t2$ - среднее время доступа к данным из буфера ($t2<t1$), p - вероятность наличия данного в буфере. Очевидно, что среднее время зависит от вероятности p и изменяется от среднего времени доступа к основной

памяти (при p=0) до среднего времени доступа непосредственно в буфер (при p=1). В качестве буфера используются регистровая память и кэш-память. Размер регистровой памяти может достигать нескольких Кбайт, кэш – более медленная память, но ее размер может измеряться Мегабайтами. Если регистровой памятью управляет программист в явном виде, то кэш-память "прозрачна" (невидима) для программиста, кэширование производится автоматически.

Прозрачностью кэш-памяти и обязана интерпретация слова кэш (cache), как тайник, тайный склад. По определению, эффект кэширования основан на предположении о многократном использовании данных (Data reuse) из кэш-памяти. Принято различать две формы многократного использования данных кэша: временное использование (temporal reuse) и пространственное использование (spatial reuse). Временное использование означает, что некоторое данное, загруженное в кэш, может использоваться, по крайней мере, более двух раз. Пространственное использование кэша предполагает возможность использовать некоторый пространственный набор данных, загруженных в кэш.

3.5. Кэш-память.

Механизм кэширования обеспечивает ускорение работы оперативного запоминающего устройства путем иерархического сочетания различных по быстродействию видов памяти и использования алгоритма кэширования для размещения данных. Кэш-память (cache, cache memory) – это память, как правило, на порядок более быстрая, чем основная, размещается в качестве буферной, между процессором и основной памятью - ОЗУ, и служит для временного хранения (в рамках своего объема) всех данных, потребляемых или генерируемых процессором. В многоуровневых кэшах элементами связки "процессор - основная память" могут выступать сами кэши. Считается, что технология кэш-памяти предложена Морисом Уилксом на основе идеи Гордона Скеротта в 1965 г.

Алгоритм кэширования состоит в следующем:

1. По каждому запросу процессора на чтение из ОЗУ команды или данного (числа, текстовой информации) происходит поиск требуемого данного в кэш-памяти (или места в кэше для записи генерируемого данного).
2. Если данное (место для записи) есть в кэше – кэш-попадание (cache hit) - то оно передается в процессор из кэша (записывается в кэш).
3. Если нужного данного нет в кэше – кэш-промах (cache miss) – то данное выбирается из основной памяти и пересылается в кэш память, а затем передается процессору. При переполнении кэша (нет места для записи) из него удаляется (модифицированные данные сохраняются в основной памяти) часть данных, обычно наименее востребованных.

Любой микропроцессор имеет "процессорный" кэш или кэш первого уровня (L1 Cache). Это буферная память объемом от 4 Кбайт до 16 Мбайт, в которой размещаются все данные и команды программы, адресуемые

процессором, и из которой данные (команды) поставляются процессору. Эта память значительно быстрее основной памяти - ОЗУ, но меньшего объема, поэтому механизм кэширования обеспечивает обновление кэша, обычно сохраняя в нем только наиболее часто употребляемые данные. Обмен между основной и кэш памятями производится квантами, объемами 4 - 120 Кбайт. Копируются целиком "строки кэша" (cache line), содержащие данные (команды), адресуемое процессором.

Все адресуемые данные процессора, включая агрегатированные (двойные слова, вещественные числа одинарной и двойной точности), размещаются в строке кэша целиком (для режима с выравниванием данных в строках); в свою очередь, строка кэша может содержать несколько данных. Сегменты программного кода, на которые возможна передача управления, размещаются в начале строки кэша. Такое размещение информации по строкам кэша производится системами программирования автоматически, этот процесс называется “выравниванием данных”, он может быть заблокирован. Обычно программный код помещается в особый кэш, I-кэш – кэш-память, которая отделена от кэша данных - D-кэша. Выборка данных из кэша (hit time) производится обычно за один такт синхронизации (оценки: 1 - 4 такта), потери при кэш-промахе оцениваются в 8 - 32 такта, доля промахов (miss rate) в 1% -20%.

Кэш-память можно рассматривать как таблицу, содержащую копии последовательности байтов из оперативной памяти, причем с каждой строкой кэша при её заполнении ассоциируется адрес строки оперативной памяти. Оперативная память при этом также будет рассматриваться как двумерная таблица, состоящая из строк, каждая строка имеет свой номер – тег. Тег для процессоров архитектуры IA32 состоит из 25 битов, а размер строки кэша равен 32 байтам. Соответственно, адрес ОЗУ состоит из поля тега, содержимое которого задает номер строки кэша в памяти и поля сдвига, которое содержит указатель на заданный байт в этой строке. В заполненном кэше с каждой его строкой должен быть связан тег - номер соответствующей строки кэша в ОЗУ.

Типы связи определяют архитектуру кэша, которая разделяется на схемы: полностью ассоциативная, частично ассоциативная и кэш-память с прямым отображением.

Полностью ассоциативная кэш память (fully associative) реализуется механизмами ассоциативной памяти - АОУ. Кэш можно рассматривать как таблицу, состоящую из элементов вида: <тег><строка кэша>, где тег есть поле ключа, а строка кэши – данные, которые механизм ассоциативной памяти выдает при совпадении значения поля тега с запрашиваемым ключом – адресом строки кэша оперативной памяти. Каждая новая строка оперативной памяти будет размещаться в кэше до тех пор, пока не произойдет полного заполнения кэш-памяти. В этом случае, для размещения очередной строки будет вытеснена одна из занятых строк, алгоритмы вытеснения аналогичны алгоритмам подкачки страниц. Данный механизм по сравнению с

другими наиболее полно использует кэш-память, но имеет громадный недостаток – объем ассоциативной памяти не может быть большим. Остальные схемы кэша моделируют ассоциативную память на адресной памяти, то есть реализуют хэш-функцию в той или иной форме.

Архитектура кэш-памяти с прямым отображением (direct-mapped) характеризуется наличием явной зависимости между адресами буферной и оперативной памяти, причем каждому адресу кэша соответствуют адреса оперативной памяти, кратные размеру кэш-памяти. Модель данного механизма такова. Пусть размер кэш-памяти равен размеру оперативной памяти. Тогда хэш-функция тривиальная: ключ - адрес строки кэша - совпадает с адресом строки ОЗУ. Затем добавим такой же блок оперативной памяти и сохраним правила размещения строк памяти в кэше (прямое отображение): первая строка кэша заполняется из первых строк блоков ОЗУ, вторая -из вторых и т.д. Соответственно к адресу строки ОЗУ добавляется старший разряд, уточняющий номер блока, который теперь не помещается в ключ. Для размещения старшего бита ОЗУ в каждой строке кэш-памяти отводится дополнительный, служебный разряд - тег, который заполняется при переписи строки кэша из ОЗУ. Тривиальная схема теперь усложняется, после считывания по ключу – по младшим разрядам адреса (без одного, старшего) ОЗУ строки кэша необходимо проверить значение разряда тега в выбранной строке. Кэш-попадание будет в случае, когда значение этого тега совпадает со старшим разрядом адреса, который не поместился в формат ключа. При кэш-промахе следует заменить строку кэша на строку с таким же номером из другого блока ОЗУ. Уже на таком примере виден главный недостаток данной схемы. Когда выбираются данные, размещенные по одинаковым адресам этих двух блоков ОЗУ, то происходит вытеснение ранее занесенных строк, несмотря на возможное наличие свободных строк в кэше. Обобщение этой схемы очевидно: ОЗУ размечается на блоки, кратные размеру кэш-памяти, и номера строк которых совпадают с номерами строк кэша. Номера этих блоков – теги - заносятся в строки кэша одновременно при заполнении кэша.

Наконец, теги можно разместить в особой памяти – в памяти тегов и всё той же хэш-функцией считывать значения тегов из этой памяти одновременно с данными из строк кэш-памяти. Если считанное значение тега совпало с полем тега адреса, то имеем кэш-попадание и считанная строка кэша – искомая.

В кэше с прямым отображением схема его заполнения может стать причиной перезагрузки кэша, если последовательно используемые процессором данные расположены в оперативной памяти по адресам, кратным размеру памяти кэша (кэш-трэшинг). Компромиссной схемой адресации кэша является частично ассоциативная схема.

Частично-ассоциативная схема (n-way set associative) кэш памяти получается при размножении с заданным коэффициентом (n) кэш-памяти с прямым отображением. Продублируем кэш-память с прямым отображени-

ем, четыре раза (4-way). Теперь, при вызове кэш-строки ОЗУ в кэш-память она может быть размещена в одном из четырех блоков памяти данных. Соответственно, тег этой строки заносится в память тегов. И только при вызове пятой строки ОЗУ, претендующей на одно из четырех занятых строк, происходит кэш-трешинг, вытеснение одной из занятых строк при вызове претендента на это место. Частично-ассоциативная схема использует кэш-память более оптимально по сравнению с кэшем прямого отображения.

При ассоциативной организации кэша (полностью или частично) выбор строки для вытеснения определяется механизмом LPU (Least Recently Used). Однако, при увеличении размера кэша затраты на работу LRU также увеличиваются и в таких случаях более эффективно использовать простой механизм случайного выбора.

Для записи в кэш-память данного, измененного процессором, используются две стратегии обновления основной памяти (ОЗУ): стратегия сквозной записи и стратегия обратной записи. Кэш-память с немедленной (сквозной) записью в ОЗУ (write-through) при операции "запись" всегда записывает измененное данное как в кэш, так и в основную память, в ОЗУ. Кэш-память с отсроченной записью в ОЗУ (write-back) определяет другой порядок обновления оперативной памяти при изменении содержимого кэша. По этой схеме, при изменении строки кэша, её новое значение записывается только в кэш и не дублируется в ОЗУ. Обновление же содержимого оперативной памяти для кэшей с отсроченной записью производится автоматически:

- при выталкивании строки кэша из-за необходимости использования этой строки для работы с другими данными оперативной памяти, адреса которых также соответствуют этой строке кэша;
- при запросе данного, хранящегося в кэше в измененном виде, но не продублированного в оперативной памяти другими абонентами, например, устройствами вывода информации, которым доступна эта память.

Выталкивание строки процессорного кэша данных при адресации данного из строки другими процессорами мультипроцессорных систем обеспечивается механизмом "когерентности" кэшей.

Естественно, при первом способе записи кэш-эффект отсутствует, но реализация второй стратегии обновления ОЗУ дороже. Сквозной метод записи в ОЗУ оптимизируется схемой с буферизацией (buffered write through), в которой строка кэша сначала помещается в буфер, из которого перепись в ОЗУ производится при отсутствии к ней других запросов.

4. Глава 4
Конвейерные технологии

4.1. Параллелизм конвейерных работ

Обсуждение данной проблемы принято начинать с описания конвейера, изобретенного и внедренного Г. Фордом для сборки автомобилей. Этот механизм, используемый на автозаводах и по настоящее время, состоит из последовательности рабочих мест автосборщиков - постов, которые связаны непрерывной конвейерной лентой. На первый пост поступает шасси автомобиля, здесь к нему прикрепляются элементы подвески, и шасси по ленте передвигается на следующий пост, где к нему прикручивают колеса, на следующем посту – устанавливают мотор и т.д. Лента движется рывками, останавливаясь на одно и то же время – такт работы конвейера - и за это время каждый сборщик должен выполнить свою часть - этап работы. Работы на каждом посту организованы так, чтобы могли быть выполнены за время такта работы конвейера. Время, необходимое для сборки одного автомобиля, равно полному времени прохождения шасси авто от первого до последнего поста. Допустим, что число постов, длина конвейера, равно m, и пусть полное время сборки будет равно T часов. Тогда при постоянном поступлении на первый пост очередного шасси (круглосуточная работа завода) новый автомобиль будет выходить с завода за каждый такт, равный времени выполнения работ на одном посту, то есть за время $t = T/m$ часов. Итак, несмотря на то, что полное время сборки одного автомобиля осталось равным T, конвейерная организация работ обеспечивает сборку за это же время m автомобилей.

Конвейеризация (pipelining) - технологический прием (режим) выполнения работы путем разделения ее на последовательность автономных подработ (этапов) так, чтобы операции каждого этапа работы могли бы выполняться параллельно с операциями других этапов и за одинаковое время. Конвейерное оборудование проектируется таким образом, чтобы результаты выполнения очередного этапа работы могли передаваться как входные на следующий этап, а оборудование текущего этапа после этого было бы готово начать обработку следующей работы.

Наиболее важным параметром конвейеров является не длина конвейера, а его производительность. Немногие знают все конвейерные этапы прохождения водопроводной воды от Истринского водозабора до крана в ванной, однако скорость потока воды в кране интересна всем. Рассказывают /А.А. Гладков/, что в известные времена при посещении нового автозавода у членов делегации изымали часы, дабы сохранить в тайне время конвейерного такта - производительность завода. Класс конвейеризуемых работ очень широк и эти работы не обязательно направлены на генерацию сложных механизмов из элементарных составляющих.

Так /H.G. Cragon/ пишет, что идея сборочного конвейера зародилась у Г.Форда на мясоперерабатывающей фабрике в Чикаго при наблюдении процесса разделки коровьих туш. Форд будто бы заметил, что если что-то можно последовательно разбирать на части, то можно делать это и в обратном порядке.

4.2. Ускорение арифметических вычислений

Конвейеризация широко используется для ускорения вычислений на ЭВМ, причем на всех этапах выполнения программы. Эффективность конвейерной обработки чисел в АЛУ может быть показана так.

Пусть работа АЛУ - Арифметического Логического Устройства - при сложении чисел вещественного типа выполняется за три такта вычислителя. Для конвейеризации этой работы разделим её на три этапа, а оборудование АЛУ на три последовательных блока (Р1-Р2-Р3), где Р1 - выравнивание порядков операндов, Р2 - собственно операция (сложение мантисс), Р3 - нормализация результата, причем блоки работают параллельно и каждый блок может выполнять свою работу за один такт вычислителя. И пусть на таком АЛУ выполняются следующие вычисления:

A1 = B1+C1
A2 = B2+C2
A3 = B3+C3
A4 = B4+C4

Тогда временная диаграмма работы конвейеризованного АЛУ имеет вид:

Этапы	1 такт	2 такт	3 такт	4 такт	5 такт	6 такт
Р1	B1+C1	B2+C2	B3+C3	B4+C4	н/р	н/р
Р2	н/р	B1+C1	B2+C2	B3+C3	B4+C4	н/р
Р3	н/р	н/р	B1+C1	B2+C2	B3+C3	B4+C4

Здесь н/р – нет работы.

Вычисления A1-A4 выполнены за 6 тактов работы вычислителя, причем только во время двух тактов (3-4) работы оборудование загружено полностью. Однако не конвейеризованное АЛУ потратило бы на эти вычисления 12 тактов, то есть в два раза больше времени.

В общем случае работа операционного блока АЛУ разбивается на m последовательных частей (стадий, выполняемые за одинаковое время), на которых арифметические команды ЭВМ выполняются в конвейерном режиме. Тогда, если на выполнение одной команды блоку требуется время T, то на обработку n команд сложения потребуется время $Tn = (n + m) * (T / m)$. Следовательно, если $m << n$, то ускорение вычислений по сравнению с работой АЛУ без конвейерной организации (его время для той же работы равно T*n) будет в m раз.

Фактором, снижающим производительность конвейеров АЛУ, может быть взаимозависимость выполняемых на конвейерном устройстве операций - конвейерный конфликт.

Так, программа:

Команды	Комментарии
A1 = B1+C1	
A2 = A1+C2	A2 = (B1+C1)+C2
A3 = B3+C3	A3 = B3+C3

A4 = B4+C4 A4 = B4+C4

будет выполняться на 3 этапном АЛУ на два такта дольше, чем предыдущая:

Этапы	1такт	2 такт	3 такт	4 такт	5 такт	6 такт	7 такт	8 такт
P1	B1+C1	н/р	н/р	A1+C2	B3+C3	B4+C4	н/р	н/р
P2	н/р	B1+C1	н/р	н/р	A1+C2	B3+C3	B4+C4	н/р
P3	н/р	н/р	B1+C1	н/р	н/р	A1+C2	B3+C3	B4+C4

На этой диаграмме видно, что количество холостых тактов работы оборудования “н/р” – “конвейерных пузырей “ (pipeline bubble) стало больше.

В общем случае конвейерные конфликты принято разделять на три группы.

1. RAW – чтение после записи (ЧПЗ), реальная зависимость.
2. WAR – запись после чтения (ЗПЧ), антизависимость.
3. WAW – запись после записи (ЗПЗ), выходная зависимость.

Реальная зависимость - ”чтение после записи” (RAW конфликт) возникает, когда команде для её выполнения требуется результат работы предыдущей команды. Приведенный выше пример иллюстрирует конфликт данного вида и способ его разрешения. Реализация такой схемы производится с помощью аппаратуры внутренних блокировок конвейера (pipeline interlook). Другим аппаратным способом разрешения таких конфликтов является техника продвижения данных (data forwarding). Логика этой техники состоит в передаче результата операции непосредственно на вход конвейера параллельно с записью результата в память. При этом сокращается число пропущенных рабочих тактов, что повышает производительность вычислителя. В некоторых системах результат арифметических операций всегда записывается в один из буферов на входе в АЛУ сразу после его получения.

Антизависимость - “запись после чтения” (WAR конфликт) возникает при нарушении последовательности выполнения команд, когда последующая команда записывает результат в операнд-источник ещё до того, как предыдущая команда прочитала этот результат. Такая ситуация может возникнуть при аппаратной оптимизации вычислений.

Пример данной коллизии /H.G. Cragon/. Пусть процессор умеет выдавать на исполнение команды вне очереди и он исполняет следующий фрагмент программы.

i A2 = A1 + A8
j A9 = A4 + A2
k A4 = A7 + A8

Между командами i и j имеет место взаимозависимость типа RAW по переменной A2 и выполнение команды j будет приостановлено. Команда k не имеет взаимозависимостей своих операндов и может быть выполнена вне очереди. Но она может завершиться и записать результат в A4 еще до того, как “старое ”значение A4 будет востребовано командой j, и тогда значение A9 будет некорректным – WAR конфликт.

Взаимозависимость можно предотвратить путем буферизации операндов-источников в регистрах.

A4 -> B1
i A2 = B2 = A1 + A8
j A9 = B1 + B2
k A4 = A7 + A8

Буферизация входных данных для команды j устраняет взаимозависимости, теперь команда k может свободно записывать в A4. Управление порядком записи/чтения в регистры производится при помощи управляющих битов. Такое использование буферов на входах АЛУ является частным случаем общего аппарата, называемого “переименование регистров”.

Выходная зависимость - “запись после записи” (WAW конфликт) возникает тогда, когда команды осуществляют запись в тот же приемник и одна из этих команд выполняется также без очереди. Пусть при выполнении фрагмента программы:

i A2 = A1 + A8
j A2 = A4 + A2
k A4 = A2 + A8

имеются взаимозависимости, приостанавливающие выполнение команды i, тогда возможен такой порядок фактического выполнения программы.

j A2 = A4 + A2
i A2 = A1 + A8
k A4 = A2 + A8

При этом команда К получит неправильный операнд в A2. Для разрешения конфликта такого рода можно использовать буферизацию.

i A2 = B1 = A1 + A8
j A2 = B2 = A4 + A2
k A4 = B2 + A8

Для разрешения конвейерных конфликтов внутри АЛУ используется техника, направленная, в первую очередь, на обеспечение корректности вычислений, а для минимизации времени простоев конвейера используются методы динамической оптимизации, основанные на изменении порядка вычислений. Методы оптимизации вычислений (out-of-order – неупорядоченное выполнение, out-of-order issue – неупорядоченная выдача) путем изменения порядка вычислений и механизмы обеспечения внеочередного выполнения команд для сглаживания конфликтов в АЛУ, разделяются на статические и динамические методы.

Статическим преобразованием программ на уровне трансляции – изменением порядка арифметических вычислений в программе в ряде случаев можно избежать конвейерных конфликтов. Преобразование последовательности вычислений из предыдущего раздела к виду:

A1 = B1+C1
A3 = B3+C3 A3 = B3+C3
A4 = B4+C4 A4 = B4+C4

A2 = A1+C2 A2 = (B1+C1)+ C2

позволит проводить эти вычисления без пропуска рабочих тактов АЛУ.

Преобразования такого вида могут производиться соответствующими блоками трансляторов, однако статическая оптимизация не может исключить все случаи конвейерных конфликтов. Так, оператор L: A2=A1+C2 уже нельзя переносить на другое место, для таких фрагментов программы с помеченными операторами статические методы изменения порядка вычислений на работают. Оптимизационные преобразования последовательности вычислений могут проводиться динамически, аппаратурой микропроцессоров, путем анализа динамической цепочки выполняемых операторов. Однако область динамического анализа ограничена фрагментом программы, который помещается в буфер команд, статические же методы преобразования программ мощнее, они могут проводить даже межпроцедурный анализ текстов программ.

4.3. Векторно-конвейерные вычисления

Время выполнения отдельной (скалярной) арифметической операции на конвейерном вычислителе (конвейеризованном АЛУ) равно T = S + t, где t - время работы, за которое конвейер выдает очередной результат, а S - время запуска конвейера, время заполнения конвейера, которое (без учета времени подготовки операндов) равно S = t*(m-1), где m - число ступеней конвейера. Производительность конвейерного вычислителя на скалярных операциях (число результатов, выдаваемых за единицу времени) равна: R = 1/(S + t). Время выполнения массовой – векторной - операции на конвейерном вычислителе равно T = S + t*n, где n – длина вектора, а S – время на дополнительные, служебные издержки организации векторной операции. Производительность конвейерного вычислителя при векторной работе (число результатов, выдаваемых за единицу времени) равна R = n/(S + t*n), это значение производительности при бесконечном увеличении длины вектора равно Rб = 1/t. Эта величина – “максимальная” или “асимптотическая” производительность – есть один из двух характеристических параметров первого порядка оценки производительности ЭВМ. Скорость выполнения операций для АЛУ с плавающей точкой измеряется в Мфлопс. (миллион операций с плавающей точкой в секунду). Например, при t = 10 нс, Rб = 10^8 результатов/сек, т.е. 100 мегафлопсов.

Графики достижения асимптотической производительности при увеличении длины вектора показывают их зависимость также от S. На графиках для S = 100 нс и S= 1000 нс видно, что они имеют различный характер достижения асимптоты, так при S = 100 нс конвейерный вычислитель показывает более высокую производительность при работе с короткими векторами. Для оценки этого эффекта используется величина N/2 - “длина полупроизводительности”, определяемая как длина вектора, при вычислении которого достигается половина максимальной производительности оборудования. Этот параметр был введен Р. Хокни (R. W. Hockney) для оценки эффек-

тивности векторных операций, он является вторым характеристическим параметром оценки производительности ЭВМ. Для приведенного выше примера N/2 = 100 для S =1000 и N/2 = 10 для S = 100. При работе с короткими векторами параметр N/2 является важным фактором, влияющий на производительность вычислителей.

Другой характеристикой конвейерных вычислителей является граничная длина - Nc (длина вектора), для которой скорость работы с векторами совпадает с темпом обработкой скаляров (Для ЭВМ с векторной и со скалярной арифметикой). Пусть скалярная арифметика выполняется со скоростью 10 мегафлопсов, а векторная - 100. Тогда при S = 1000 можно определить Nc = 12. Для векторов, длина которых меньше 12, векторная арифметика медленнее скалярной арифметики. Вычислено, что при S = 100 значение Nc = 2. Для 5 мегафлопсной скалярной арифметики: Nc = 6 для S=1000 и Nc = 1 при S = 100.

Для достижения максимальной производительности конвейерных арифметических вычислителей (и ЭВМ в целом) необходимо обеспечивать для них загрузку очередных операндов на каждом такте работы оборудования. Такой стиль перебора аргументов характерен для массовых, векторных операций, реализуемых командами цикла, например, такого:

```
DO  L = 1,N
   A(I) = B(I)+C(I)
ENDDO
```

Для подобных циклов с регулярной структурой обработки данных накладные расходы и препятствие опережающему просмотру при реализации команд организации цикла (счетчик и переход) на традиционных ЭВМ явились причиной появления специализированного векторно-конвейерного вычислительного оборудования, ранее называемого матричными процессорами. Если вместо приведенного выше цикла использовать запись этого алгоритма в виде векторной команды сложения вида: VADD(B,C,A,N), тогда конвейерное арифметическое устройство, выполняющее такие команды, будет работать как векторный вычислитель и может вырабатывать результаты вычислений Ai на каждом такте своей работы. Так как класс задач с регулярной обработкой данных достаточно широкий, то для них созданы специальные вычислители, дополняющие обычные скалярные ЭВМ.

В таком вычислителе имеется конвейерный процессор, выполняющий векторные команды путем засылки элементов обрабатываемых векторов в арифметический конвейер с интервалом, равным длительности прохождения одной стадии обработки. В векторных процессорах операндами и результатами операций могут служить векторные регистры или сверхоперативная память, в которые следует предварительно загрузить данные из ОЗУ, а данные из векторов с результатами вычислений переписываются в ОЗУ отдельной командой. Обработка длинных векторов при этом производится квантами, сообразно размеру векторных конвейеров. Вычислительный векторный блок ЭВМ может содержать несколько конвейеров.

Для их совместной работы используется “принцип зацепления конвейеров”, когда выход одного из них является входом для другого. В вычислениях зацепление операций производится методом образования цепочек из последовательных арифметических операций.

Примером одной из первой векторно–конвейерной ЭВМ может служить “матричный процессор” AP-120B фирмы Floating Point System, выпущенный в 1976 г. (отечественный аналог – ЕС 2706, Изот). Так как он был в десять раз медленнее и в пятнадцать раз дешевле машины Cray-1, то получил прозвище “Cray для бедных”. Этот процессор имел большой коммерческий успех, к 1980 г было установлено 1000 машин. Данная ЭВМ имела двух - стадийный конвейер для сложения и трех - стадийный конвейер для умножения чисел с плавающей запятой. ЭВМ подключалась к главной – хост-машине - как периферийное оборудование и использовалась для выполнения программ с массовыми векторными операциями. Её производительность равна 12 Мегафлопс, что по тем временам было недоступно универсальным ЭВМ. Одной из особенностей этого процессора была система команд – это была VLIW машина. Для неё было создано разнообразное программное обеспечение: автокод, транслятор с Фортрана, векторная библиотека.

Некоторое время на принципах матричного сателлита использовались векторные ускорители для ПК, например, плата RORTEX.

Векторными блоками дополнялось оборудование традиционных, скалярных ЭВМ. Программы таких ЭВМ могли содержать как скалярные, так и векторные команды (С-120, ЕС-1095, ПС-3000).

В 1976 г. начала работать ЭВМ Cray-1 (конструктор Саймур Крей) с революционной для того времени векторной архитектурой. Вычислители данного класса (PVP, parallel vector processing) в настоящее время являются основой мультипроцессорных систем, характеристики которых неоднократно находились на вершине рейтинга супер-ЭВМ.

Основой Cray архитектуры являются две идеи:

- функциональные устройства (ФУ, АЛУ) являются специализированными однофункциональными векторными конвейерами;

- аргументами вычислений являются специальные векторные регистры, хранящих массивы (векторы) данных.

ЭВМ Крей-1 имеет 12 ФУ, восемь из которых могут работать в режиме зацепления конвейеров и кроме, обычных регистров, имеет восемь 64-разрядных V регистра по 64 слова. Память содержит 1 млн. 64 разрядных слов, расслоение по 16 банкам. Время такта ЭВМ 12.5 нс., цикл памяти – 50 нс. Производительность ЭВМ 20 – 60 млн. операций в секунду.

В настоящее время только на вычислителях этой архитектуры можно добиваться сверхвысокую производительность векторных программ, работы в данном направлении проводятся в фирмах Cray и NEC. Архитектура NEC SX-6 (35.86 Tflop) интересна тем, что одна модель этой системы некоторое время возглавляла рейтинг самых высокопроизводительных вычислительных систем - TOP500.

Данная ЭВМ - Earth Simulator состоит из 640 процессорных узлов, которые соединены между собой переключателем (12.3 Gв/s * 2). Процессорный узел состоит из 8 арифметических процессоров и оперативной памяти объемом в 16 Гбайт, разделенной на 2048 банков (SMP-архитектура). Арифметический процессор (производительность 8 Гфлопс) имеет суперскалярный вычислитель с кэш-памятью 2*64 Кбайт и 8 векторных вычислителей. Каждый векторный вычислитель имеет 72 векторных регистра по 256 элементов каждый и 6 конвейерных устройств (для сложения, умножения, деления, логических операций, операций маскирования, чтения/записи). Форматы чисел с плавающей запятой 32, 64, 128 (только для скалярного процессора) бит. Система имеет дисковые массивы на 250 Тбайт и ленточную библиотеку на 1.5 Пбайт (1 Пбайт = 10^{15} байт).

По такой же архитектуре выполнена супер-ЭВМ SX-8, которая на октябрь 2004 г. являлась самой мощной ЭВМ в мире. Ее производительность – 65 триллионов операций в секунду (Tflops).

Достоинствами векторно-конвейерных вычислителей является высокая и сверхвысокая производительность, однако эта производительность достигается только на регулярных вычислениях, так как эти вычислители есть специализированные машины.

4.4. Конвейеризация команд

В конвейерных архитектурах ЭВМ устройство обработки команд УУ является таким же конвейером, как и другие функциональные устройства. Работа устройства управления, так же как и АЛУ, разбивается на независимые этапы выполнения. Грубая схема разделения процесса выполнения команды на независимые конвейерные этапы такова.

1. Выборка команды.
2. Декодирование и выборка аргументов команды.
3. Исполнение команды.
4. Запись результатов.

В такой схеме при отсутствии конвейерных конфликтов время исполнения программы будет ускорено в четыре раза относительно одноэтапного УУ.

Большинство конвейеров выполняет работу каждого этапа за минимальное время, причем одинаковое на всех этапах оборудования, обычно за машинный такт синхронизации. Однако замечено, что выполнение некоторых конвейерных операций занимает менее половины машинного такта. Тогда при удвоении частоты внутренних тактовых импульсов можно было бы выполнить две работы за один машинный такт. Такая схема конвейера называется “суперконвейерной” (superpipelined), примером её реализации может служить микропроцессор MIPS R4000. Конвейеры процессоров с суперконвейерной архитектурой имеют еще большее число ступеней, что позволяет упростить каждую из них и, следовательно, сократить время пребывания в них инструкций. Считается, что в таком оборудовании каждый этап конвей-

ера разбивается еще на внутренние микроэтапы, Каждый микроэтап исполняется за время этих внутриэтапных временных интервалов. Так как время конвейерной работы должно быть одинаково на всех этапах конвейера, включая и микроэтапы, то тогда все этапы конвейера должны содержать одинаковое число микроэтапов и, следовательно, можно принять внутиэтапную частоты за единую меру. Когда предлагается в суперконвейерной схеме сохранить две тактовой частоты: машинный такт и внутриэтапный такт, то это напоминает преодоление пропасти двумя прыжками. Эту не очень убедительную схему (а почему бы не выставить в единую конвейерную цепочку и микроэтапы?) часто показывают в схемах конвейеров векторных машин. Длина конвейера команд различна и определяется архитектурой ЭВМ. В процессоре Pentium 4 применена так называемая “гиперконвейерная” архитектура: конвейер, по которому проходят инструкции от момента считывания кода инструкции из памяти до ее завершения, состоит из очень большого числа ступеней. Гиперконвейер Pentium 4 состоит из 20 ступеней; для сравнения – “суперконвейер” процессоров P6 имеет 10 ступеней, а конвейер Pentium - всего 5. Здесь "супер" и "гипер" - определения, используемые фирмой изготовителя оборудования. Наверное, следует называть гиперконвейерной архитектурой мега-конвейер всего мультипроцессора - сумму всех исполняемых при выполнении программы ступеней УУ и АЛУ. А термин “суперконвейерная архитектура” оставить только для обозначения микропроцессоров с умножением частоты работы внутри конвейерных этапов.

Увеличение числа ступеней – этапов конвейера команд приводит к увеличению числа конвейерных конфликтов. Конвейерные конфликты УУ имеют те же свойства, что и для АЛУ, и они разрешаются одинаковыми методами, например, исполнением с изменением последовательности инструкций (out-of-order execution). Однако имеется и ряд особенностей. Например, при конвейерной обработке команд код операции декодируется на второй (по крайней мере, не на первой) стадии, поэтому команда безусловного перехода будет идентифицирована после того, как на первую стадию конвейера будет выбрана команда, текстуально следующая после команды перехода.

Для оптимизации программы здесь применяется техника отсрочки ветвления – после команды перехода в программу помещается пустая команда. Естественно, оптимизирующие алгоритмы компиляторов ищут для этой позиции полезные команды программы, которые могут быть выполнены, не нарушая правильность программы, то есть они включаются в схему переупорядочивания последовательности команд.

Положение усложняется для команд условного перехода. Так, для условного оператора:

IF (A<B) GO TO L; S1; L:S2;

еще до вычисления значения условного выражения (A<B) необходимо решать задачу о выборе ветки исполнения программы: заполнении конвейера

команд кодами операторов S1 или S2. В процессорах прежних поколений инструкция перехода приостанавливала конвейер (выборку инструкций) до исполнения собственно команды перехода. При этом, естественно, терялась производительность. Банальная схема ускорения - вычисление условия перенести, по возможности, на ранние стадии выполнения программы. Статистика показывает, что такое переупорядочение порядка выполнения команд возможно не часто. Следующее простое решение – начать обработку одной из ветвей (любой) не дожидаясь окончания вычисления условия. Признавать же результаты этой обработки действительными следует только после вычисления условия. Если случайно выбранная ветка не должна обрабатываться, то следует пропуск рабочих тактов оборудования, так как необходимо коды неверно выбранной ветки убрать из конвейера. Выполнение условного перехода, при котором происходит обработка одной из веток оператора заранее, называется “спекулятивным”. При этом производится исполнение выбранного фрагмента программы, предвосхищая результаты неизвестного в этом момент вычисления условия перехода. Дальнейшее развитие идеи опережающего выполнения фрагментов условных операторов привело к появлению механизмов предсказания результатов вычисления условных выражений в условных операторах.

Предсказание переходов в условных операторах программы (branch prediction) позволяет продолжать выборку и декодирование потока инструкций после выборки инструкции ветвления (условного перехода), не дожидаясь проверки самого условия. Механизм предсказания переходов направляет поток выборки и декодирования команд программы по одной из ветвей. Исполнение по предположению, называемое также спекулятивным (speculative execution), идет дальше - предсказанные после перехода инструкции не только декодируются, но и по возможности исполняются до проверки условия перехода. Если предсказание сбывается, то работа предсказателя оказывается оправданной, если не сбывается - конвейер оказывается недогруженным и простаивает. Методы реализации предсказателей разделяются на статические и динамические методы.

Статический метод предсказания реализуется по алгоритмам, заложенным в компиляторы в процессе трансляции программ. Известны особенности программ, характерные признаки того, что переходы по одним условиям, вероятнее всего, произойдут, а по другим - нет. Так, для циклических конструкций вероятность перехода на повторение цикла выше вероятности выхода из него. Некоторые системы программирования дают возможность программисту указывать предполагаемую вероятность перехода непосредственно в программе на языке высокого уровня. Статическая схема выбора альтернативной ветки вычислений (результат предсказания переходов) в условных операторах фиксируются в объектных кодах следующим образом. Некоторые ЭВМ продолжают выбор в конвейер команд, текстуально следующих за командами передачи управления, тогда реализовать статический метод предсказания переходов можно, помещая код с наибольшей вероятно-

стью перехода непосредственно за командой условного перехода (в примере, приведенном выше, это код оператора S1). Для такого оборудования компиляторы могут формировать объектный код с размещением наиболее вероятно выполняемого фрагмента программы непосредственно за командой условного перехода. Некоторые системы команд содержат в коде условных команд разряды, флаги для указания статических предположений о вероятности перехода по метке в условных операторах.

Одним из методов уменьшения потерь на выполнение команд переходов, как условных, так и безусловных, является использования кэш-памяти прогнозирования переходов, называемой таблицей, буфером целевых адресов переходов - BTB (Branch Target Buffer). Буфер содержат информацию о командах переходов, последних по времени выполнения программы. Если выполняемая команда перехода не представлена в таблице (пусть её объем равен 2^n), то в i-строку таблицы записывается TAG и <целевые адреса команды>. Номер строки таблицы – i есть значения n-младших разрядов адреса команды перехода (счетчика команд), а старшие разряды адреса этой команды перехода заносятся в поле TAG. Поле <целевые адреса команды> может состоять из исполнительного адреса перехода, который формируется при выполнении команды перехода. Если при выборки очередной команды выясняется факт нахождения ее в таблице BTB, тогда для команды безусловного перехода сразу известен исполнительный адрес перехода еще до дешифрации выбранной команды. Выбранная команда игнорируется, в счетчик команд заносится исполнительный адрес перехода из таблицы. Более того, в поле <целевые адреса команды> можно хранить код команды, на которую указывает исполнительный адрес перехода. Для обработки команд условного перехода BTB используется совместно с аппаратом динамического предсказания переходов.

Динамическое предсказание переходов опирается на предысторию выполнения вычислительного процесса - для каждого конкретного случая перехода накапливается статистика поведения.

Аппаратный механизм учета вероятности перехода (динамический) состоит из блока предсказания переходов. Этот блок, кроме учета статически определенных предпочтений для ветвлений, использует таблицу переходов, в которой хранится история переходов для каждого (в рамках объема таблицы) перехода программы. Большинство современных микропроцессоров обещают точность предсказаний переходов этим способом выше 90%.

Для динамического прогнозирования переходов аппаратура собирает статистику переходов, которая помещается в – BHT (Branch History Table). Реализация этой схемы имеет ряд красивых архитектурных решений. Таблица BHT содержит статистику о результатах вычисления условных выражений: каждая её строка соответствует выполненным условным операторам и она содержит информацию о том, как выполнялись переходы при предыдущих обращениях к данному оператору. Такой информацией в общем случае

является значение n-битового счетчика в строке таблицы и принципиальная схема прогнозирования перехода для текущего условного оператора такова.

1. Если значение счетчика больше нуля, то переход прогнозируется как выполняемый. Если направление перехода предсказано верно, к значению счетчика прибавляется единица, иначе значение счетчика уменьшается на единицу.

2. Если значение счетчика меньше нуля, то переход прогнозируется как невыполняемый. Если направление перехода предсказано верно, из значения счетчика вычитается единица, при неверном прогнозе - к значению счетчика единица добавляется.

Для упрощения работы со счетчиком в качестве “нуля”, нейтральной точки отсчета, обычно выбирается точка из середины интервала значений счетчика (например, значение 2n – 1). При этом отказ от работы с отрицательными числами не отменяет процедуры проверки выхода значения счетчика за пределы интервала.

Исследования n-битовых схем прогнозирования на тестовых задачах SPEC показало, что наилучшие результаты прогнозирования обеспечивают трех-битовые счетчики, причем результаты двух-битовых отличаются от трех-битовых незначительно. Большинство схем прогнозирования используют двух-битовые счетчики, причем для работы с ними используется алгоритм Смита. В соответствии с этим алгоритмом, значения счетчиков определяют четыре вероятности: ветвь часто исполняется (strongly taken, код – 11), ветвь выполняется (taken, код – 10), ветвь не выполняется (not taken, код - 01), ветвь часто не выполняется (strongly not taken, код – 00). В этой схеме прогнозирования переход прогнозируется, если первый, левый бит равен единице. Правый бит в этой схеме показывает результат вычисления условия на предыдущем шаге.

Другой проблемой является выбор размера таблицы BHT. Статистические исследования показывают, что таблица, состоящая из 4096 строк, дает высокую точность прогнозов и на ряде тестовых пакетах работает практически также как и таблица бесконечного размера.

Для идентификации строк таблицы BHT редко используют "честные" схемы с хранением в таблице значений счетчиков команд условного перехода (в виде тегов) из-за большого расхода памяти. Обычно таблица BHT индексируется младшими разрядами адреса команды перехода. Возникающий при этом эффект наложения (aliasing), адресации одной строки таблицы двумя или более разными командами перехода может быть конструктивным, деструктивным и нейтральным. Считается, что при многократном выполнении команды условного перехода влияние эффекта наложения будет незначительным для прогнозирования работы этого оператора.

Рассмотренная схема прогнозирования переходов является примером одноуровневой схемы. Более точный прогноз обеспечивают двухуровневые, коррелированные и гибридные схемы предсказания переходов. В таких схемах кроме статистических данных используется дополнительно качест-

венная история переходов при помощи регистров переходов. Регистр глобальной истории (Global History Register) есть логическая шкала, в младший разряд которой после сдвига шкалы заносится результат вычисления условного выражения текущего условного оператора перехода. Затем эта шкала складывается по модулю 2 с младшими разрядами адреса команды перехода и полученное значение используется для индексации таблицы BHT. Наконец, для каждой команды перехода можно хранить локальную историю переходов и её также использовать для предсказания переходов.

Гибридные схемы предсказания переходов используют одновременно несколько схем предсказания – предикторов, причем для предсказания перехода динамически выбирают результаты работы наиболее точного предиктора. При этом выбор предиктора, наиболее адекватного ходу выполнения программы, может производиться суперпредиктором по вышеперечисленным алгоритмам.

Как динамический, так и статический подходы к предсказанию переходов в условных операторах имеют свои преимущества и недостатки. Преимущество статического подхода - отсутствие необходимости интегрировать на чипе микропроцессора дополнительную аппаратуру предсказания переходов. Большинство современных микропроцессоров оборудованы различными средствами динамического предсказания переходов, производимого на базе анализа "предыстории". Динамическое предсказание переходов "мощнее" статического. Однако у динамического предсказания есть и свои слабые места - это проблемы, возникающие из-за ограниченности ресурсов для сбора статистики. Статические методы предсказания используются совместно с динамическими методами.

5. Глава 5
Архитектура ЭВМ

5.1. Системы команд

Архитектура набора команд (ISA), называемая иногда системой команд ЭВМ, восходит к фон-неймановской архитектуре, содержавшей всего 21 команду. Эволюция системы команд непосредственно связана с развитием всей архитектуры ЭВМ и будет рассматриваться, в основном, на примере наиболее характерных команд обработки данных - инструкций выполнения арифметических или логических функций, инструкций управления. Команда ЭВМ - это инструкция для аппаратной реализация единичного акта преобразования информации в ЭВМ, состоящая из кода операции команды (имени функции преобразования) и адреса (ов) - ссылки на аргумен-

ты и на результат функции. Кроме основного результата, каждая команда вырабатывает дополнительный, который служит служебной информацией для последующих команд. Служебная информация фиксируется в системных регистрах – битах индикации. В них хранится информация об особенностях выполнения текущей команды, например, был ли результат арифметической операции равен нулю или отрицателен. К служебной информации относится также значение программного счетчика команд (СК) – адреса размещения в ОЗУ исполняемой команды. Набор команд разделяется на команды, предоставляемые прикладным программистам, и привилегированные команды.

Команды для программирования приложений обеспечивают реализацию арифметических и логических операций, команд управления, пересылок. Состав арифметических команд, кроме базовых операций: +, -, *, /, может содержать команды вычисления тригонометрических функций и др. Набор аргументов таких операций трансформировался от чисел с фиксированной запятой до вещественных чисел обычной и двойной точности, от скалярных аргументов до векторных аргументов. Объявлены планы реализации ЭВМ для арифметики с иными системами счисления, например, интервальной арифметики. Логические команды реализуют операции Булевой алгебры, формирования логических шкал: сдвиги, сборки, разборки и директивы для по-байтовой обработки текста. Команды управления служат для управления ходом выполнения программы. Команды безусловной передачи управления – требование начать выполнение команды, адрес размещения которой в ОЗУ указан в команде. Команды условной передачи нарушают последовательный перебор команд для исполнения только тогда, когда характеристика результата выполнения предыдущей команды соответствует заданному условию. К командам управления относятся и команды организации циклов, примитивы для проверки окончания цикла с одновременной индексацией параметра цикла. Особую группу команд управления составляют команды обращения к подпрограммам и возврата из них.

Адреса команд – аргументы операции - бывают следующих типов. Литералы - это задание, запись в команде аргумента в явном виде. Например, в трехадресной команде можно задать в виде (*, A1, “2”, A3) арифметическую операцию – число из A1 умножить на число 2 и отправить в A3. Частой, а иногда и единственной, формой адресов в некоторых архитектурах ЭВМ является ссылки на регистры, элементы из набора ячеек сверхоперативной памяти процессора. В таких регистрах хранятся аргументы операции, в них же может записываться результат. Наконец, адресом может служить ссылка на байт оперативной памяти, который определяет первую компоненту аргумента (если аргумент длиннее одного байта). На этом примере прямой адресации простота адресации операндов в командах ЭВМ заканчивается, появляется возможность косвенной адресации данных. В таких схемах адрес памяти, по которому хранится аргумент - исполнительный (эффективный) адрес - формируется по заданным правилам из элемен-

тов, указанных в адресе команды. Косвенная адресация (адрес в адресе) состоит в том, что по указанному в команде адресу (ссылка на слово памяти в регистре или в ОЗУ) хранится не сам аргумент - операнд, а адрес памяти, где и содержится требуемый операнд. Исполнительный адрес в команде может вычисляться, например путем сложения содержимого заданного регистра с константой, которая указана в адресе команды. В архитектуре IA32 определяются 12 способов формирования исполнительных адресов. “А смерть Кащеева находится в яйце, которое хранится в ларце, который и т.д.”.

Форматы команд ведут историю от первобытных ЭВМ, когда все команды имели одинаковую длину и формат. Наиболее удобной для программирования формой описания арифметических вычислений являются трехадресная команда, формата (ЭВМ М-20, БЭСМ-4, Весна): <КОП><А1><А2><А3>, при выполнении которой над операндами А1 и А2 производится вычисление функции, определенной <КОП>ом, а результат фиксируется в А3. С увеличением размеров ОЗУ размещение трех адресов, апеллирующих непосредственно к оперативной памяти, в формате команды стало затруднительным и в командах стало меньше операндов. Так, в двухадресных командах ЭВМ оперирует парой операндов - операндом назначения dest (destination) и операндом-источником src (source). Базовая схема выполнения команды: dest = F(dest, src), где F - некоторая функция от двух переменных. Это означает, что при выполнении команды процессор извлекает из указанных в команде адресов (регистр, память, константа в самой инструкции) пару двоичных аргументов и результат действия над ними записывает на место одного из них (dest). Для выполнения той же функции над следующей парой чисел требуется повторное исполнение команды, но уже с другой парой операндов. Такой принцип исполнения используется для архитектуры процессоров IA-32.

Если зафиксировать в качестве операнда назначения dest сумматор, то его содержимое можно брать по умолчанию и формат команды станет одноадресный, как это было реализовано в БЭСМ-6. Если аргументы операции занести в стек, который используется также в качестве операнда назначения, то задавать порядок вычислений можно безадресными командами. Итак, команды ЭВМ бывают трех, двух, одно адресными, а также – четырех адресными (четвертый задает следующую команду для исполнения), нуль и полуторно адресные. Кроме набора команд с фиксированной постоянной длиной, имеются реализации архитектур микропроцессоров с командами переменной длины. Так, команды процессоров IA32 могут занимать от одного до четырех байтов, формат, длина команды фиксируется в известных полях первого байта команды. Соответственно, программа для ЭВМ данной архитектуры с искаженными или утраченными значениями первых байтов текста превращается в китайскую грамоту.

Набор арифметических команд в современных процессорах дополняется векторными командами, которые ведут начало от процессоров Pentium, в них было введено расширение – группа команд MMX, направленное на ускорение обработки потоков и массивов данных. Это расширение стало реализацией принципа SIMD (Simple Instruction - Multiple Data, одна инструкция на множество данных). Здесь вводятся новые упакованные форматы данных: в один регистр MMX можно помещать не только один операнд (64-битное число), но и пару 32-битных, четверку 16-битных или восьмерку 8-битных чисел. Одна инструкция MMX выполняет однотипные действия сразу над всеми числами, упакованными в регистры MMX, заданные адресной частью данной команды.

Все множество архитектур процессоров по типу системы команд делятся на две глобальные категории - RISC и CISC, к которым добавляются комбинированные схемы. Традиционные процессоры универсальных ЭВМ называются CISC (Complex (Complicated) Instruction Set Computer) - компьютерами с полной системой команд (семейство Intel ЭВМ от серии ix86 до Pentium и Pentium Pro, архитектура IA32). Помимо арифметических и логических операций в систему команд (аппаратно реализуемые функции) включались сложные операции, такие, как извлечения корня; реализуются сложные схемы формирования исполнительных адресов (в i486 доступно до 12 способов). Расширение спектра операций облегчает программирование и отладку программ, однако увеличивает трудоемкость разработки процессоров. Так ошибка в АЛУ одного из процессоров фирмы Intel была выявлена пользователями только после выхода процессора на рынок. Время выполнения даже таких операций, как умножение и деление чисел с плавающей запятой, варьируется в зависимости от значений аргументов операций, что ограничивает возможности аппаратного планирования совмещенного выполнения команд. CISC набор команд нельзя было разместить целиком на одном кристалле – чипе первых СБИС.

Исполнение процессоров на кристалле потребовало ревизии системы команд, что привело к реализации компьютеров с сокращенным набором команд - RISC (Reduced Inctruction Set Computing) архитектуры. Основными чертами микропроцессоров данной архитектуры принято считать следующие: выполнение одной команды за цикл; использование простых видов адресации; трехадресные арифметические команды над операндами в регистрах; фиксированный формат команд. Традиционное число команд здесь - 128. Выполнение команд, не входящих в этот состав, производится программно, для этого в архитектуре предусматривается развитый механизм реализации подпрограмм: реализация запоминания-восстановления регистров, стеки и т.д.

Упрощение оборудования, конвейеризация выполнения команд, позволили RISC процессорам резко повысить производительность. Сокращенный набор команд означает также ряд других упрощений оборудования для достижения высокой производительности. Команды в архитектуре RISC имеют

фиксированную и небольшую длину, они не нуждаются в интерпретации. Использование трехадресных команд упрощает их дешифрацию и дает возможность сохранять большее число переменных в регистрах без перезагрузки. По мере развития архитектуры RISC она обзаводилась все новыми возможностями наращивания производительности. Ключевыми из них являются суперскалярная или многоконвейерная обработка, внеочередное выполнение команд, появление смешанных или групповых команд для выполнения часто повторяющихся последовательностей.

Массовому переходу на RISC процессоры препятствует большой объем накопленного фонда алгоритмов и программ в кодах самой массовой ЭВМ, в кодах IA32. Интеграция обсуждаемых архитектур была приведена в процессоре Pentium Pro. Этот процессор – реализация CISC-среды на RISC оборудовании ядра. Поступающие в процессор на исполнение объектные коды, написанные в системе команд IA32, аппаратно перекодируются на входе во внутреннее представление, в коды команд внутреннего RISC представления. Таким образом достигается совместимость накопленного фонда алгоритмов и программ с современными технологическими новациями.

5.2. Суперскалярные микропроцессоры

Конвейеризация (pipelining) предполагает разделение выполнения каждой инструкции (команды) на несколько этапов, причем каждый этап выполняется на своей ступени конвейера процессора. При выполнении инструкция продвигается по конвейеру по мере освобождения последующих ступеней. Таким образом, на конвейере одновременно может обрабатываться несколько последовательных инструкций и производительность процессора можно оценивать темпом выхода выполненных инструкций со всех его конвейеров. Конвейеры процессоров имеют большее число ступеней, что позволяет повысить производительность микропроцессоров, однако при этом повышается вероятность роста конвейерных конфликтов.

В типовой конвейерной схеме работы обычного “скалярного” УУ первые два этапа выполняют предварительную, подготовительную работу исполнения команды. Затем функциональная часть команды - третий этап - реализуется на одном из операционных блоков (опять модельная схема): на АЛУ с плавающей точкой, на целочисленном АЛУ и в блоке анализа условных операторов. Можно заметить, что при работе третьего этапа загружен всегда только один блок из трех. Микропроцессоры с единственным конвейером для выборки и декодирования команд называются “скалярными”, к этому типу относятся 486 процессоры фирмы Intel.

Другое направление повышения производительности микропроцессоров находится в области реализации параллелизма на уровне команд (instruction level parallelism, ILP), в архитектуре “суперскалярных” (superscalar) микропроцессорах. На уплотнении загрузки функциональных блоков путем дублирования конвейера и выполнении в функциональном блоке сразу двух команд одновременно и основана суперскалярная версия

конвейера, реализованная в процессорах Pentium. Суперскалярный процессор имеет более одного (Pentium - два) конвейера, способных обрабатывать инструкции параллельно. Эти процессоры содержат два пятиэтапных конвейера: главный конвейер, u-конвейер может выполнять любую команду, а второй, дополнительный v-конвейер- только некоторые. Конвейерные блоки первого этапа выбирают сразу две команды, причем они ассиметричны, право изменять счетчик выбираемых команд имеет только один из этих блоков. Общий блок выборки команд компонует из потока команд по-парно совместимые команды и передает их на исполнение в два конвейера. Команды всегда выполняются по заданному порядку, несовместимые команды выполняются последовательно на главном конвейере. Реализация процессоров с 3-4 конвейерами признана нецелесообразной, суперскалярная архитектура современных микропроцессоров опирается на более общую схему.

В такой схеме мультиплицируется оборудование подготовительных этапов конвейера команд для выборки команд и их декодирования. Коэффициент дублирования подготовительных этапов называют иногда степенью суперскаляра. Состав функциональных блоков также может расширяться. Так, у Pentium III блоки целочисленной арифметики и арифметики с плавающей запятой дублированы, а у Pentium IV размножены с коэффициентом три. После декодирования команд и преобразования их во внутреннее представление диспетчер организует выполнение вычислительных работ с учетом возможного параллелизма на уровне команд и обеспечения максимальной загрузки функциональных устройств. Возможные конвейерные конфликты разрешаются способами, описанными выше. Переименование регистров (register renaming) позволяет обойти архитектурное ограничение на возможность параллельного исполнения инструкций. Продвижение данных (data forwarding) подразумевает начало исполнения инструкции до готовности всех операндов.

При этом выполняются все возможные действия и декодированная инструкция с одним операндом помещается в исполнительное устройство, где дожидается готовности второго операнда. Предсказание переходов (branch prediction) позволяет продолжать выборку и декодирование потока инструкций после выборки инструкции ветвления (условного перехода), не дожидаясь проверки самого условия.

Суперскалярная схема выполнения программы реализует алгоритмы внеочередного и спекулятивного порядка выполнения команд. При этом возможно совмещенное по времени выполнение независимых команд. Однако эта схема не позволяет учитывать параллелизм на уровне команд, который выявляется на уровне трансляции программ.

5.3. Сверхдлинные команды

Идеи VLIW архитектуры ЭВМ просты: для достижения большей производительности логично прибегнуть к статическому распараллеливанию вычислений и заставить один физический процессор параллельно обраба-

тывать несколько инструкций. Считается, что начало работ по архитектуре ЭВМ со сверхдлинными командами VLIW (Very Long Instruction Word) положили в начале 80-х Алан Чарльзворт (Alan Charlesworth), Джош Фишер (Josh Fisher) и Боб Рау (Bob Rau). Такая архитектура подразумевает применение специальных компиляторов, способных обнаруживать в тексте программы независимые инструкции и формировать из них длинные машинные слова инструкций. В архитектурах VLIW - машин команды могут иметь широкий формат (длину) и команда может содержать несколько содержательных инструкций, выполнение которых детально регламентируется в терминах тактов работы АЛУ (допускается параллельное выполнение нескольких инструкций). В таких архитектурах имеется возможность программировать вычислительные алгоритмы (включая векторные) с максимальной производительностью для данной аппаратуры. В них вся работа по оптимальному программированию возлагается на системы программирования (или ручное программирование). Главные особенности VLIW архитектуры:

- существование одного центрального управляющего устройства (УУ), обрабатывающее за один такт одну длинную команду;
- наличие нескольких функциональных устройств (ФУ);
- разделение длинной команды на поля, каждое из которых содержит команду управления некоторым функциональным устройством или команду обращения к памяти;
- фиксированная длина в тактах времени исполнения каждой операции, причем операции ФУ могут быть конвейеризованными;
- ОЗУ имеет большой коэффициент расслоения памяти, банки памяти идентифицируются при программировании;
- сложность ручного программирования в силу большого числа комбинаторных задач.

Достоинством данной архитектуры является возможность статически задавать работу всем ФУ на каждом такте. Однако упрощения в архитектуре управления приводят к значительному возрастанию сложности задачи планирования выдачи команд, так как программными средствами должна быть обеспечена точная синхронизация считывания и записи данных. При этом необходимо так планировать параллельное выполнение операций машины, чтобы выполнялись определенные ограничения на число одновременно считываний и записей в наборы регистров, использование ФУ и т.д. Размер командного слова в машинах данной архитектуры - FPS (AP-120B) - 64 бита, Multilow Tract – 1024 бита.

В 1997 г. на проходившем совещании - Microprocessor Forum - появились первые подробности относительно разрабатываемой архитектуры и тогда же появился термин EPIC (Explicitly Parallel Instruction Computing). Набор инструкций для новой архитектуры получил название IA64. Компании Intel потребовалось еще 4 года, прежде чем она смогла представить на рынке готовый образец процессора. Процессоры IA64 с архитектурой EPIC,

патент США 96/24895, имеют 128-битовую внутреннею шину, но в отличие от VLIW-программ объектный код IA64 не зависит от числа специализированных конвейеров в конкретном процессоре. Merced, одно из первых воплощений новой архитектуры, содержит два блока по 128 регистров для хранения целых и плавающих чисел, независимые кэши для инструкций и данных, специализированные конвейерные устройства (шесть для процессоров "нижнего уровня"). В микропроцессоре реализована схема динамического распределения регистров, условной (спекулятивной) загрузки данных из памяти при обработке ветвлений. Команда – связка (bundie) - длиной 128 бит здесь имеет три слота, задающие три операции и поле маски, уточняющее порядок выполнения этих операций. В мае 2001 г. появился первый Itanium, а набор инструкций IA64 для него из маркетинговых соображений переименовали в IPF (Itanium Processor Family).

Уровень вычислительной мощности новых процессоров весьма высок и позволяет успешно конкурировать с RISC-системами. Недавно суперкомпьютер "Колумбия" из 10240 процессоров Itanium-2 в списке TOP500 занимал второе место, а его производительность достигла 52 ТФлоп в штатном режиме или более 60 ТФлоп в пиковом. Распространению архитектуры EPIC в немалой степени способствует и продолжающийся рост количества оптимизированных для нее приложений. Системы на базе процессоров Itanium могут работать под управлением различных ОС - Windows, Linux, HP-UX, Solaris и др.

Отличием архитектуры EPIC по сравнение с современными суперскалярными архитектурами, с одной стороны, и VLIW архитектуры, с другой, является попытка совместить достоинств обеих архитектур при решении задач распараллеливания программы на уровне инструкций.

Данная архитектура обеспечивает:

- статическую проверку зависимостей между командами ЭВМ для выявления тех из них, которые можно сгруппировать для параллельного выполнения, поиск микропроцессов;
- динамическое распределение вычислительных работ между функциональными блоками процессора.

В архитектуре EPIC, в отличие от RISC архитектуры, задача возможного статического распараллеливания программ возложена на компилятор. Компилятор оптимизирует выполнение программы целиком, а не отдельного ее фрагмента, который доступен процессору в случае аппаратного распараллеливания. Это позволяет упростить структуру процессора и, как следствие, добиться более высокой его производительности при меньшей степени сложности. Архитектура VLIW, подобно EPIC, представляет собой крайний случай использования статического параллелизма. Это эффективный способ поднять производительность, особенно если к ней добавить технологию суперскалярной организации вычислений, что и сделано в архитектуре EPIC.

5.4. Современные микропроцессоры.

Микропроцессор - производное от слов "микросхема" (“microchip”) и процессор ("processor"), это кремниевый кристалл - “чип”, на котором размещен центральный процессор (ЦПУ, CPU - Central Processor Unit). Его две основные компоненты: арифметико-логическое устройство (АЛУ, ALU – Arithmetic and Logic Unit), которое выполняет логические и арифметические операции, и устройство управления (УУ, CU – Control Unit). УУ извлекает команды из памяти, декодирует их и исполняет. На кристалле размещаются также регистры и кэш-память, а, иногда, и другие элементы оперативной памяти. Развитие технологии конструирования и производства элементов ЭВМ позволяет размещать на кристалле все больше и больше элементов, в частности, кэш память второго уровня.

Оперативное запоминающее устройство (ОЗУ, MM – Main Memory) – место хранения программ и данных размещается на другом кристалле, которое связано с микропроцессором “системной шиной”. Микропроцессор обычно называют просто “процессор”, а вместе с ОЗУ называют, иногда, “процессорным элементом”. Процессор в комплекте с внешними устройствами (массовая память, монитор, и т.д.) называется персональным компьютером (ПК).

Показано, что даже при предельной, пиковой работе микропроцессора задействуется не более 25% транзисторов на кристалле. Так на чипе EV78 процессора Alpha 21364 размещено 100 млн транзисторов, из которых 92 млн – кэш-память. Для увеличения нагрузки оборудованию на кристалле в микропроцессоры добавляют дополнительно отдельные процессоры для обработки данных. Наверное, первыми такими элементами стали сопроцессоры. Это арифметические конвейерные устройства для быстрой обработки векторных операндов (техника MMX и SSE).

Развитие технологии обработки видеоинформации привело к появлению в составе оборудования микропроцессоров видеокарт, вычислительная мощность которых превосходит на векторной обработке производительность самого процессора.

В настоящее время активно реализуется многоядерная архитектура микропроцессоров, клонирование процессоров на одном кристалле. Многопроцессорные кристаллы (Chip MultiProcessor, CMP), двух-, четырехядерные процессоры позволяют значительно увеличить вычислительную производительность микропроцессоров, их используют в качестве процессорных элементов в современных мультисистемах. Двухъядерный (dial-core) процессор POWER4 корпорации IBM – это два микропроцессора с с индивидуальными кэшами первого уровня 2*64 Кбайт. На кристалле размещается также общая кеш-память второго уровня емкостью 1.4 Мбайт. Стоимость такого процессора значительно ниже, чем у двух процессоров, имеющих по одному ядру. Двухъядерные процессоры занимают меньше места, потребляют меньше энергии и рассеивают меньше тепла. Все ведущие разработчики микропроцессоров клонируют свои процессоры.

Другой тенденцией развития микропроцессорной архитектуры является стремление к достижению максимальной загрузки оборудования при мультипрограммной работе. Технология Hyper-Threading – (HT технология), разработанная корпорацией Intel, дает возможность одновременного выполнения сразу двух последовательностей команд или “потоков” на одном ядре, что повышает производительность микропроцессоров; в одной машине появляются два логических процессора. При одновременном выполнении двух потоков большую долю времени обычно занимает работа операционной системы для переключения задач. Разместив на кристалле механизмы аппаратной реализации алгоритмов ОС, которые в некоторых реализациях занимают всего 2% оборудования, можно получить ускорение счета на классе задач до 20%. Но следует иметь в виду, что выполняемые потоки – это разные задачи, для получения реального ускорения работы одного приложения необходимо распараллеливать исполняемую программу.

Обобщением такой схемы являются мультитредовые процессоры (MTA – MultiThrading Architecture). Грубая схема мультитредовых машин такова: размножаются только конвейеры команд, а остальные ресурсы машины (операционные устройства) предоставляются тредам по-очереди. В них допускается выполнение одновременно несколько тредов (потоков) – вычислительных процессов, называемых легковесными процессами. Треды, в отличие от настоящих вычислительных процессов (задач), выполняются в одном виртуальном адресном пространстве, поэтому переключение вычислений между тредами очень просто. Каждому треду выделяется собственный набор регистров, благодаря чему в процессе переключения между потоками команд не приходится тратить время на сохранение и восстановление состояния служебной информации.

Ресурсы машины выделяются динамически и они будут использоваться с полной загрузкой, если выполнение медленных инструкций потока будет прикрыто обслуживанием других потоков. Считается, что при такой организации вычислений все ресурсы машины будут работать с предельной производительностью. Здесь имеется полная аналогия с работой многозадачной ОС. Мультитредовая архитектура процессоров с одновременным выполнением тредов называется SMT (Simultaneous Mulni-Threading). Одной из первых разработок в этой области была машина MTA фирмы Tera. Имеются сообщения, что фирма Sun представит новую модификацию мультитредового процессора Niagara во второй половине 2008 г. Это процессор Rock, состоящий из четырех блоков, по четыре процессорных ядра в каждом. Каждое ядро Rock способно поддерживать одновременную работу двух потоков (всего 32 потока на процессор), кроме того, каждое ядро имеет собственный модуль для работы с вещественной арифметикой и графическими инструкциями. Каждая группа из четырех ядер будет обладать собственным кэшем первого уровня, 32 Кбайт для инструкций и 32 Кбайт для данных. Каждый из четырех кэшей второго уровня будет иметь емкость в 512 Кбайт.

Так как все современные суперскалярные процессоры содержат развитые механизмы обеспечения “параллелизма на уровне команд”, то оказалось, что реализация MTA, особенно в рамках технологии HT, требует лишь относительно небольших аппаратных доработок. Поэтому, для многоядерных процессоров удвоение числа логических процессоров и, соответственно, увеличение пропускной способности процессора становится нормой. Многоядерные кристаллы и мультитредовые процессоры могут значительно увеличить производительность приложений, если последние были написаны с учетом особенностей их архитектуры.

5.5. Поставщики микропроцессоров

Главным поставщиком микропроцессоров на рынок многие годы была компания Intel с моделями x86, процессорами классической CISC архитектуры. Начиная с микропроцессора Pentium, который пришел на смену Intel486, данная архитектура приобрела имя IA32 (Intel Architectura 32-разрядная) и все черты современного вычислителя: суперскалярную архитектуру, раздельные кэш-памяти команд и данных, механизмы предсказания переходов. Однако эта классическая архитектура процессора с полным набором команд CISC служила тормозом для повышения производительности вычислений и фирма делает поворот при реализации микропроцессора Pentium шестого поколения - P6. Этот вычислитель Pentium Pro - процессор с подлинной RISC организацией, допускающий обработку программ IA32. Программа, состоящая из команд микропроцессоров IA32 архитектуры, поступает на один из трех параллельных декодеров ID микропроцессора и перекодируется в наборы внутренних микроопераций, команд RISC архитектуры. Далее, состав оборудования может использовать все достоинства работы с сокращенным набором команд. Эти работы обеспечивают возможность использования накопленного фонда программ для IA32. Для написания новых программ используются системы команд современных машин. Современные процессоры фирмы поддерживают архитектуру IA64 и имеют высокую производительность. Так процессор Pentium 4 (Xeon) работает на частоте 3,06 ГГц. Аналогичный эволюционный процесс происходил с фирмой конкурентом – с компанией AMD. Процессоры AMD Opteron с архитектурой IA64 также выходят на частоту в 3 ГГц.

Далее рассматриваются только две архитектуры из типовых архитектур RISC микропроцессоров (МП): SPARC, MIPS, PowerPC, PA-RISC, Alpha.

Фирма IBM разрабатывает семейство микропроцессоров Power PC (Performance Optimization With Enhanced Risc). МП семейства RS/600 использовались для комплектации супер-ЭВМ семейства PS. Суперкомпьютер Blue Gene/L, которой в ноябре 2006 года занимал в TOP 500 первую позицию, был укомплектован двухъядерными процессорами PowerPC 440. Данная супер-ЭВМ после модернизации остается самой мощной в мире, по крайней мере, до 2008 г.

Современный двухъядерный процессор POWER5 реализуется на кристалле площадью 389 кв. мм, он содержит 276 млн транзисторов. На кристалле размещается также общие для обоих ядер кэши 2 и 3 уровней объемом 1,874 и 36 Мбайт.

О проекте Alpha фирма DEC (Digital Equipment Corporation) объявила в 1989 г. Ранее фирма разработала и выпускала ЭВМ серии PDP (модель PDP-11 копировалась в СССР под именем СМ-1600), затем VAX. Первый процессор Alpha-21064 (21 – XXI век, 0 – процессорное поколение, 64 - разрядность в битах) имел также кодовое имя EV-4 (EV - Extended VAX, 4 – поколение техпроцесса). Он состоял из 1.58 млн транзисторов, размещенных на 3-слойных платах толщиной 0.75 мкм. Тактовая частота – 200 МГц. Далее были реализации EV5, EV56, EV6, EV67, EV7, EV79, выполненные фирмой совместно, (а затем в роли филиала) с фирмами Mitsubishi, Samsung, Compad, Hewlett-Packard (HP). Соответствующие архитектуры Alpha-21164, Alpha-21264, Alpha-21364 имели наилучшие характеристики среди аналогов по времени выпуска и технологического уровня производства. Процессоры Alpha считаются наиболее мощными. Так процессор Alpha 21164 (1994 г.) с тактовой частотой 300 МГц обеспечивал рекордную в то время производительность в 1200 MIPS, в 1999 г. процессор первым достиг частоты 1 ГГц. Фирма HP, последний приобретатель фирмы DEC и ее чипа Alpha, заявила, что EV7z (1.3 ГГц) будет последним в этом семействе. Однако фирма Compad собирается продолжить линию Alpha, в долгосрочных планах выпуск в 2012 г. кристалла EV12 по 0.03 мкм технологии, состоящего из 15 миллиардов транзисторов. Процессоры серии Alpha используются в качестве процессорных узлов фирмой Cray Research при создании своих супер-ЭВМ.

6. Глава 6
Производительность ЭВМ

6.1. Оценка вычислительных систем

При оценке вычислительных систем характеристики их производительности принято считать параметрами первого порядка. Одним из двух характеристических параметров первого порядка оценки производительности служит Rpeak – пиковая, предельная производительность вычислительной системы. Далее, по Р.Хокни, следует параметр N/2 (Nhalf), определяемый как длина вектора, при вычислении которого достигается половина предельной производительности. Однако, в настоящее время, в качестве второго характеристического параметра первого порядка принято считать Rmax – максимальная производительность вычислителя при решении задачи Linpack, более того, по этому параметру ранжируются рейтинги производительности высокопроизводительных вычислительных систем.

Данная оценка вычислительных систем показательна для использования ЭВМ в научных, инженерных расчетах и она имеет исторические корни. В те времена, когда ЭВМ были “большими” (БЭСМ-6), стационарными,

они тиражировались малыми сериями и размещались в специальных помещениях, называемых Вычислительными Центрами (ВЦ). А в каждом ВЦ имелось подразделение, которое занималось распределением “машинного времени” между пользователями ЭВМ. Одним из стимулов развития методов дистанционного доступа в ЭВМ в США была необходимость оптимального использования вычислительных ресурсов с учетом часовых поясов страны. Главным, а часто и единственным, параметром оценки ЭВМ в те времена было число операций, команд, выполняемых в секунду машинного времени. Но и эти времена счета на оплату услуг ВЦ, учитывали, кроме ресурсов вычислителя (времени счета задачи в часах, минутах), другие услуги ВЦ: предоставление прикладного программного обеспечения, архивное хранение программ и данных и т.д. Некоторые задачи нуждались в гарантиях абсолютной надежности вычислений, другие – в выполнении в реальном времени. Производительность была главным, но не единственным, параметром оценки ЭВМ.

С появлением Персональных Компьютеров (ПК) области применения ЭВМ стали такими, что потребовалось изменить критерии оценки вычислительных систем. В самом деле, так ли необходимо пользователю (покупателю) ПК знать производительность оборудования при арифметических вычислениях или время счета одной из задач линейной алгебры. Ему сообщается в качестве главных параметров ПК тактовую частоту процессора, объем оперативной памяти, емкость диска, характеристики сопроцессоров. При детализации запросов учитывается возможность апгрейда, масштабирования, повышения производительности ПК путем замены отдельных блоков системы. Учитывается состав программного обеспечения, версия ОС, и т.д.

Поэтому, для оценки вычислительных систем, кроме оценки по производительности, разрабатываются методы оценки, интегрирующие также другие критерии оценки. Одним из таких подходов является метод экспертных оценок. Для класса вычислительного оборудования составляется список критериев оценки с их весовыми коэффициентами. Критериями оценки могут быть производительность, масштабируемость, цена, качество и состав комплектации, системное и прикладное программное обеспечение, сервисное обслуживание, надежность (отказоустойчивость) и т.д. Затем эксперты дают свою оценку по каждому критерию каждому из исследуемых вычислителей. Сумма экспертных баллов, умноженных на весовые коэффициенты, дает итоговую оценку вычислителя, по которой можно сравнивать оборудование данного класса. Данный метод используется в коммерческих проектах, однако широкому распространению таких методик препятствуют следующие обстоятельства. Список критериев и весовые коэффициенты существенно зависят от предметной области и пристрастия составителей методики. Трудно обеспечить также подбор объективных экспертов. Методы оценки производительности вычислительных систем разработаны детально и они дают такие объективные оценки, что общая оценка вычислителей только по

этому критерию может быть достаточно презентативной для широких областей применения вычислителей.

6.2. Производительность вычислительных систем

Оценка производительности вычислительных систем имеет два аспекта: оценка пиковой производительности – номинального быстродействия системы, и получение оценок “реальной” производительности. Если номинальная (пиковая, предельная) оценка однозначно определяется техническими параметрами оборудования, то вторая характеристика указывает производительность системы в реальной среде применения. Такими средами могут быть конкретная задача (задачи), набор типовых задач, предметная область и т.д. Так, номинальные характеристики грузовика: максимальная скорость и грузоподъемность должны корректироваться при использовании грузовика в среде песчаных карьеров. Обычно рассматриваются три подхода к оценке производительности вычислительных систем: методами имитационного моделирования; на базе исследований аналитических моделей; эталонными измерениями.

Имитационное моделирование используется для оценки производительности систем, когда невозможно проведение натурных испытаний оборудования, а погрешности аналитических моделей при учете архитектурных особенностей оборудования не устраивают исследователей. Применяется для оценки производительности проектируемых систем. Чаще всего, имитационная модель с нужной степенью детализации создается для проверки архитектурных решений и логических схем проектируемых систем и оценка производительности при этом может быть не главной целью создания модели.

Особым случаем имитационного моделирования является “эмуляция”, это процесс обеспечения точного воспроизведения работы одного вычислителя на другом вычислительном оборудовании. Служит для обеспечения эксплуатации накопленного фонда алгоритмов и программ ЭВМ при смене поколений вычислителей.

Аналитические методы исследования производительности ЭВМ рассматриваются на примере измерения производительности микропроцессоров. На основании данных о технических параметрах вычислителя, и его подсистем можно составить модель виртуального вычислителя и по ней оценить производительность исследуемого вычислителя. Пусть известна только тактовая частота ЭВМ. Тогда, учитывая тот факт, что ни одно вычислительное действие не может быть выполнено быстрее, чем за время тактового цикла, то по значению тактовой частоты можно оценивать предельную производительность этого вычислителя. Если ЭВМ - скалярный микропроцессор и о нем известно, что каждый конвейерный этап УУ и АЛУ выполняется за один такт, а тактовая частота составляет 100 МГц, то можно предположить, что результаты выполнения последовательности арифметических операций будут вырабатываться каждые 10 нс, а скорость

выполнения команд будет равна миллиону команд в секунду. Эти параметры будут грубо определять пиковую производительность ЭВМ. Итак, наиболее абстрактной единицей измерения производительности процессоров является тактовая частота, частота тактового генератора (clock rate). Частота генератора задает цикл (cycle), наименьшую единицу времени, распознаваемая устройством. Минимальное время исполнения любой логической операции (переключение транзистора) - один такт, никакая операция в процессоре не может быть выполнена быстрее, чем за один период генератора. Тактовая частота измеряется в “Герцах” – число тактов в секунду. Один МГц - это миллион тактов в секунду, ГГц – миллиард тактов в секунду и т.д. С частотой тактового генератора работают элементы процессора, интегрированные на чипе – АЛУ, кэш-память, дешифратор команд, регистры. Тактовая частота системной шины меньше частоты процессора (в 2,5 – 10 раз, Alpha 21064), поэтому другие параметры шины также влияют на производительность процессора. Поскольку любая операция, в том числе и пересылка данных, не может происходить быстрее, чем за такт, то желательно передавать как можно больше информации за один такт. Обычно, системная шина для данных обеспечивает полосу пропускания в 32, 64, 128 бит. Поэтому, тактовая частота и разрядность шины является существенным параметром процессора. Так, для Intel Pentium 64-разрядная шина данных обеспечивает обмен с ОЗУ со скоростью 528 Мбайт/с. Для оценки характеристики системной шины часто используется только интегральный показатель – пропускная способность шины. Процессоры Xeon имеет шины с производительностью в 6,2 Гбайт/с, процессоры AMD Opteron – в 5,3 Гбайт/с.

Архитектура процессора, также как и тактовая частота, влияет на работу процессора, поэтому два процессора с одинаковой тактовой частотой не обязательно будут иметь одинаковую производительность. Например, микропроцессоры с суперскалярной архитектурой могут выполнять более одной операции за тактовый цикл. Данных параметр оценки производительности процессора удобен для сравнительной оценки ЭВМ одного семейства, для процессоров с одинаковой архитектурой. Итак, тактовая частота вычислителя, даже с уточнениями частоты шины, может быть оценкой лишь номинальной производительности процессора.

Другой обобщенной мерой производительности процессоров может служить число команд, выполняемые в единицу времени. Для вычислителей фон-неймановской архитектуры скорость выполнения команд может быть параметром, который может быть использован для оценки общего времени выполнения программы. Этот параметр - одна операция (команда, инструкция ЭВМ) в секунду. Обычно эта единица используется в номинации: “Мипс” - MIPS (Million Instruction Per Second) - миллион операций (команд, инструкций ЭВМ) в секунду. Так как время выполнения различных команд может различаться, то данный параметр при использовании со-

провождается разного вида уточнениями: время выполнения только логических команд ЭВМ, производительность на заданной смеси команд и т.д. Например, тестовая смесь для определения производительности ЭВМ в MIPSax Dhrystone состоит из 100 команд, 53% которых есть операторы присваивания, 32% – операторы управления и 15% - вызовы функций. Другим способом нормирования данного параметра является сравнение производительность исследуемого процессора с производительностью эталонной ЭВМ. Эталонной ЭВМ принято считать ЭВМ VAX 11/780, производительность которой считается равной одному MIPSy. Фирма IBM предложила использовать в качестве эталона производительности свою ЭВМ - RS/6000. Считается, что 1 RS/6000 MIPS примерно равен 1,6 VAX 11/780 MIPS.

Достоинством данного рейтинга является его наглядность: более быстрые машины имеют более высокий рейтинг MIPS. Однако проведение сравнения ЭВМ с различными системами команд при использовании MIPS в качестве метрики вызывает затруднения. Рейтинг MIPS для процессоров с сопроцессорами плавающей точкой будет явно занижен при выполнении векторных работ. Итак, и второй рейтинг оценки производительности ЭВМ имеет очевидные недостатки.

Корпорацией SPEC (Standard Performance Evaluation Corporation) в 1992 г. для измерения производительности ПК был предложен рейтинг SPEC. Корпорация разработала два базовых набора тестов CIN92 и CFP92, ориентированные на интенсивные расчеты, для измерения производительность процессора и системы памяти, а также эффективности оптимизирующих блоков компиляторов ПК.

Первый набор тестов измеряет производительность процессора при обработке целых чисел. Он состоит из 6 Си-программ: интерпретатор Лиспа, упаковка текстовых файлов, и т.д.

Второй пакет цикла служит для измерения скорости процессора при обработке чисел с плавающей точкой. Он состоит из 14 программ на Си и Фортране из различных прикладных областей: квантовая химия, оптика, робототехника, квантовая физика и т.д. Результаты измерений производительности тестируемой ЭВМ на каждом тесте этих пакетах нормируется с производительностью эталонной ЭВМ (эталонная ЭВМ – все та же ЭВМ VAX 11/780), они называются SPECint92 и SPECfp92. Эти безразмерные рейтинги есть среднее геометрическое 6 (14) результатов отношений производительности для каждого теста наборов, они характеризуют быстродействие ЭВМ при обработке целочисленных и вещественных данных. Более поздние версии этих тестов называются SPECint95 и SPECfp95, SPECint2000 и SPECfp2000. Корпорация SPEC проводит работу по обеспечению тестирования многопроцессорных ЭВМ. Для широкого круга научных и инженерных расчетов данный рейтинг производительности признан наиболее показательным, он является непременным параметром любого процессора.

Внедрение конвейерной обработки данных, позволившее получать результаты вычислений арифметических операций с плавающей точкой за каждый такт, выпуск многоядерных процессоров и мультипроцессорных ЭВМ, привели в настоящее время к использованию в качестве меры производительности ЭВМ скорость вычислений чисел с плавающей точкой. С другой стороны, так как для большинства вычислительных алгоритмов существуют оценки числа арифметических операций, необходимых для выполнения расчетов, данная мера и может служить тем показателем, который интересует пользователей в первую очередь. Такие измерения производятся мерой “Мегафлопс” – MFLPOPS (Million of Floating point Operation Per Second) - миллион операций в секунду на данных с плавающей запятой, единица быстродействия ЭВМ на операциях с плавающей запятой. Соответственно, существуют единицы GFLPOPS (Гигафлопс) и TFLPOPS (Терафлопс) т.д. Данный рейтинг широко используется для оценки пиковой и реальной производительности вычислительных систем.

Примеры оценки производительности некоторых процессоров по трем описанным показателям.

Процессор Itanium 2 с тактовой частотой в 1,3 – 1.5 ГГц может вырабатывать 4 флопа за такт, производительность по SPECint - 1322 и по SPECfp -2119.

Процессор Alpha 21364 с тактовой частотой в 1,15 – 1.77 ГГц может вырабатывать 4 флопа за такт, производительность по SPECint - 877 и по SPECfp -1482.

Процессор AMD Opteron с тактовой частотой в 1,4 – 2 ГГц может вырабатывать 2 флопа за такт, производительность по SPECint - 1335 и по SPECfp -1339.

Существуют и другие подходы к оценке производительности ЭВМ. По мнению некоторых инженеров (фирма Sun) производительность процессоров следует оценивать в единицах количества выполняемых инструкций на один ватт или на одну монтажную единицу (rack unit) стойки сервера.

6.3. Измерение реальной производительности

Измерения для получения реальной производительности вычислительных систем проводятся контрольными (тестовыми) программами при помощи эталона. Эталон - бенч-марк (Benchmark) это: стандарт, по которому могут быть сделаны измерения или сравнения; или процедура, задача или тест, которые могут быть использованы для сравнения систем между собой или со стандартом, как в предыдущем пункте.

Для повышения общности и представительности оценки производительности вычислительных систем контрольные, тестовые программы можно разделить на следующие группы.

- Программы нижнего уровня.

Эти программы тестируют основные машинные операции - +,/,* с учетом времени доступа к памяти, работы кэша, характеристик ввода/вывода.

- Ядра программ.

Ядра программ - короткие характерные участки программ, например, Эймсовские ядра НАСА, синтетический тест Ветстоун (Whetstone), Ливерморские фортрановские циклы.

Тексты Ливерморских циклов представляют собой 14 (малый пакет) или 24 (большой пакет) фрагментов реальных программ, эксплуатируемых в Ливерморской Национальной Лаборатории им. Лоуренса. Каждый фрагмент является явно заданным итерационном циклом Фортрана. Примеры тел двух циклов.

X(K) = Q + Y(K) * (R * Z(K+10) + T * Z(K+11)) (1 цикл)

X(K) = X(K+1) – Y(K) (12 цикл)

- Основные подпрограммы и типовые процедуры.

Примером основных подпрограмм могут быть программы быстрого преобразования Фурье и программа Линпак (Linpack). Программа Линпак - это процедура решения системы линейных уравнений методом исключения Гаусса. В этой схеме вычислений точно известно число операций с плавающей точкой, которое зависит от размерности массива – параметра процедуры. Стандартные значения размерностей: 100 или 1000. Для параллельных ЭВМ имеется соответствующая версия теста.

- Полные основные прикладные программы.

В качестве примеров программ этого уровня могут служить Лос-Аламоские тестовые программы моделирования поведения плазмы и программы гидродинамики.

Компания AIM Technolgy разработала систему генерации различных рабочих нагрузок, соответствующих уровню тестируемой системы и требованиям по её использованию. Тестовые пакеты AIM разделены на две категории: стандартные и заказные. Заказные пакеты создаются для точного моделирования предметной области пользователя или конкретной конфигурации вычислителя. Стандартные пакеты моделируют среду типовых прикладных задач: среду автоматизации проектирования в механике, среду СУБД и т.д.

Для оценки производительности средств вычислительной техники, используемых в сфере бизнеса, предложен пакет тестов TPC.

Однако, универсальной системы измерений средств вычислительной техники не существует.

6.4. Рейтинг TOP500

Международными организациями систематически производится сбор данных о самых высокопроизводительных вычислительных системах (супер-ЭВМ) в мире и публикация данных о 500 самых мощных машинах. Эта таблица рейтингов - таблица TOP500 - периодически (два раза в год)

обновляется, 30 редакция списка опубликована в ноябре 2007 года. Все машины списка – многопроцессорные вычислительные системы. Среди параметров вычислителей указывается главные: Rmax – максимальная производительность при решении тестовой задачи Linpack и Rpeak – пиковая, предельная производительность ЭВМ. Содержимое таблиц упорядочено по Rmax. Приводится параметр - Nhalf – длина вектора входного параметра тестовой задачи, на котором вычислитель достигает только половину предельной производительности. Указывается также место и год установки ЭВМ, число и тип процессоров.

30 редакцию рейтинга по-прежнему возглавляет система Server Blue Solution IBM, состоящая из 212992 процессоров, производительность которой достигла: Rmax = 478200 Gflops, Rpeak = 596378 Gflops. В этой редакции Россия представлена 7 системами.

По аналогии, публикуется список самых мощных суперкомпьютеров стран СНГ TOP50, на май 2007 г опубликована 6 редакция списка. В нем на первом месте размещены данные о кластере, разработанном фирмой “Т-платформы” и установленном в 2007 г. в Томском Университете. Его параметры: 564 процессора, 1128 ядра, 282 узла (2*Xeon 5150, 2.667 GHz), производительность по тестовой задаче - 9013, пиковая – 12002 Gflops.

Список литературы

1. Степанов А.Н. Архитектура вычислительных систем и компьютерных сетей. - СПб.: Питер, 2007. – 509 с.
2. Цилькер Б.Я., Орлов С.Д. Организация ЭВМ и систем. - СПб.: Питер, 2006. – 668 с.
3. Королев Л.Н. Архитектура ЭВМ. - М.: Научный мир. 2005.- 272 с.
4. Хорошевский В.Г. Организация вычислительных систем. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2005. -512 с.
5. Брайант Р., О’Халларон Д. Компьютерные системы: архитектура и про граммирование. Пер с англ.-СПб.: БХВ-Петербург, 2005. -1104 с.
6. Столлингс В. Структурная организация и архитектура компьютерных систем. Пер. с англ.. - М.: Изд. дом Вильямс, 2002. – 896 с.
7. Смирнов А.Д. Архитектура вычислительных систем: - М.: Наука, 1990.- 320 с.
8. Аверьянов Г.П., Рошаль А.С. Введение в информатику. Учебное пособие. -М.: МИФИ, 2002. -252 с.
9. Барановская Т.П., Лойко В.И., Семенов М.И., Трубилин А.И. Архитектура компьютерных систем и сетей. - М.: Финансы и статистика. 2003.- 256 с.
10. Корнеев В.В., А.В. Киселев А. В. Современные микропроцессоры. - М.: Нолидж, 1998. -240 с.
11. Амамия М., Танака Ю. Архитектура ЭВМ и искусственный интеллект. Пер с японск. – М.: Мир,1993. - 400 с.

12. Кун С. Матричные процессоры на СБИС. Пер. с англ. - М.: Мир, 1991. - 672 с.
13. Касперски К. Техника оптимизации программ. Эффективное использование памяти. - СПб.: БХВ-Петербург, 2003. - 464 с.
14. Цимбал Л.А. Синергетика информационных процессов. Закон информативности и его следствия. - М.: Наука,1995. -119 с.
15. Задыхайло И.Б., Котов Е.И., Мямлин А.Н., Поздняков Л.А., Смирнов В.К. Вычислительная система с внутренним языком повышенного уровня. - М.: ИПМ АН СССР, Препринт N41, 1975. – 43 с.
16. Цикритзис Д., Бернстайн Ф. Операционные системы. Пер. с анг.- М.: Мир, 1977. –336 с.
17. Брой М. Информатика. Основополагающее введение. - М.:Диалог-МИФИ, 1996-299с.
18. Юзвишин И.И. Информациология. - М.: Радио и связь, 1996. -215 с.
19. Корогодин В.И., Корогодина В.Л. Информация как основа жизни. – Дубна: Феникс, 2003. -208 с.
20. Генис А. Культурология. – М.:Подкова, 2002.- 504 с.
21. Лацис А. Как построить и использовать суперкомпьютер. –М: Бестселлер, 2003. – 240 с.
22. Шура-Бура М.Р., Штаркман В.С. Вычислительная машина М-20. Инструкция по математической эксплуатации. – М.: ЦБТИ, 1962. – 90 с.