3.11 Кооперативная и вытесняющая многозадачность. Процессы, потоки выполнения. Скалярные, супер-скалярные, векторные, SMT процессоры. Шины доступа к памяти и NUMA.

Кооперативная и вытесняющая многозадачность

http://ru.wikipedia.org/wiki/Многозадачность

Многозада́чность (англ. multitasking) — свойство операционной системы или среды программирования, обеспечивать возможность параллельной (или псевдопараллельной) обработки нескольких процессов. Истинная многозадачность операционной системы возможна только в распределенных вычислительных системах.

Примитивные многозадачные среды обеспечивают чистое «разделение ресурсов», когда за каждой задачей закрепляется определённый участок памяти, и задача активизируется в строго определённые интервалы времени.

Более развитые многозадачные системы проводят распределение ресурсов динамически, когда задача стартует в памяти или покидает память в зависимости от её приоритета и от стратегии системы. Такая многозадачная среда обладает следующими особенностями:

- Каждая задача имеет свой приоритет, в соответствии с которым получает время и память
- Система организует очереди задач так, чтобы все задачи получили ресурсы, в зависимости от приоритетов и стратегии системы
- Система организует обработку прерываний, по которым задачи могут активироваться, деактивироваться и удаляться
- По окончании положенного кванта времени задача может временно выбрасываться из памяти, отдавая ресурсы другим задачам, а потом через определённое системой время, восстанавливаться в памяти (свопинг)
- Система обеспечивает защиту памяти от несанкционированного вмешательства других задач
- Система распознаёт сбои и зависания отдельных задач и прекращает их
- Система решает конфликты доступа к ресурсам и устройствам, не допуская тупиковых ситуаций общего зависания от ожидания заблокированных ресурсов
- Система гарантирует каждой задаче, что рано или поздно она будет активирована
- Система обрабатывает запросы реального времени
- Система обеспечивает коммуникацию между процессами

Типы псевдопараллельной многозадачности

1. Невытесняющая многозадачность
   Тип многозадачности, при котором операционная система одновременно загружает в память два или более приложений, но процессорное время предоставляется только основному приложению. Для выполнения фонового приложения оно должно быть активизировано.

2. Совместная или кооперативная многозадачность
   Тип многозадачности, при котором фоновые задачи выполняются только во время простоя основного процесса и только в том случае, если на это получено разрешение основного процесса.

3.11

Кооперативную многозадачность можно назвать многозадачностью “второй ступени” поскольку она использует более передовые методы, чем простое переключение задач, реализованное многими известными программами (например, MS-DOS shell из MS-DOS 5.0 при простом переключении активная программа получает все процессорное время, а фоновые приложения полностью замораживаются. При кооперативной многозадачности приложение может захватить фактически столько процессорного времени, сколько оно считает нужным. Все приложения делят процессорное время, периодически передавая управление следующей задаче.

3. Вытесняющая или приоритетная многозадачность (режим реального времени)
   Вид многозадачности, в котором операционная система сама передает управление от одной выполняемой программы другой. Распределение процессорного времени осуществляется планировщиком процессов. Этот вид многозадачности обеспечивает более быстрый отклик на действия пользователя.

   http://ru.wikipedia.org/wiki/Вытесняющая_многозадачность
   Вытесняющая многозадачность — это вид многозадачности при котором планирование процессов основывается на абсолютных приоритетах. Процесс с меньшим приоритетом (например пользовательская программа) может быть вытеснен при его выполнении более приоритетным процессом (например системной или диагностической программой). Иногда этот вид многозадачности называют приоритетным.

   Каждая работающая программа имеет свое защищенное адресное пространство. Многопоточное (англ. multithread) выполнение отдельных задач позволяет при задержке в выполнении одного потока не останавливать задачу полностью, а работать со следующим потоком.

Процессы, потоки выполнения
http://www.realcoding.net/teach/visualc7/Glava12/Index6.htm
Многозадачная (multi-process) система позволяет двум или более программам выполняться одновременно. Многопотоковая (multi-threaded) система позволяет одной программе выполнять сразу несколько потоков одновременно. Современные операционные системы сочетают в себе оба эти свойства.

Процесс — это понятие, относящееся к операционной системе. Каждый раз, как вы запускаете приложение, система создает и запускает новый процесс. С каждым процессом система связывает такие ресурсы, как:

- виртуальное адресное пространство;
- исполнимый код и данные;
- базовый приоритет;
- описатели объектов;
- переменные окружения.

Поток (thread) — это основной элемент системы, которому ОС выделяет машинное время. Поток может выполнять какую-то часть общего кода процесса, в том числе и ту часть, которая в это время уже выполняется другим потоком.

http://www.ipm.kstu.ru/os/lec/2.php
http://seregaborzov.wordpress.com/2006/11/08/chem_process_otlichaetca_ot_potoka/

- Поток (thread) определяет последовательность исполнения кода в процессе.

3.11

- Процесс ничего не исполняет, он просто служит контейнером потоков.
- Потоки всегда создаются в контексте какого-либо процесса, и вся их жизнь проходит только в его границах.
- Потоки могут исполнять один и тот же код и манипулировать одними и теми же данными, а также совместно использовать описатели объектов ядра, поскольку таблица описателей создается не в отдельных потоках, а в процессах.
- Так как потоки расходуют существенно меньше ресурсов, чем процессы, старайтесь решать свои задачи за счет использования дополнительных потоков и избегайте создания новых процессов(но подходите к этому с умом).

Скалярные, супер-скалярные, векторные, SMT процессоры

Классификация параллельных архитектур по Флинну (M. Flynn)
Вычислительная система с одним потоком команд и данных (однопроцессорная ЭВМ — SISD, Single Instruction stream over a Single Data stream).
Вычислительная система с общим потоком команд (SIMD, Single Instruction, Multiple Data — одиночный поток команд и множественный поток данных).
Вычислительная система со множественным потоком команд и одиночным потоком данных (MISD, Multiple Instruction Single Data — конвейерная ЭВМ).
Вычислительная система со множественным потоком команд и данных (MIMD, Multiple Instruction Multiple Data).

Скалярный процессор — это простейший класс микропроцессоров. Скалярный процессор обрабатывает один элемент данных за одну инструкцию (SISD процессор, типичными элементами данных могут быть целые или числа с плавающей запятой).

http://www.hpc.nw.ru/KOI/COURSES/HPC/g2.2.2.html

Суперскалярный процессор представляет собой нечто большее, чем обычный последовательный (скалярный) процессор. В отличие от последнего, он может выполнять несколько операций за один такт. Основными компонентами суперскалярного процессора являются устройства для интерпретации команд, снабженные логикой, позволяющей определить, являются ли команды независимыми, и достаточное число исполняющих устройств. В исполняющих устройствах могут быть конвейеры. Суперскалярные процессоры реализуют параллелизм на уровне команд.

Термин "конвейер" в компьютерной архитектуре обозначает ... конвейер. Идея конвейера состоит в том, чтобы сложную операцию разбить на множество более простых, которые могут выполняться одновременно. При движении объектов по конвейеру на разных его участках выполняются разные операции, а при достижении каждым объектом конца конвейера он окажется полностью обработанным. Конвейеры применяются как при обработке команд, так и в арифметических операциях. Для эффективной реализации конвейера должны выполняться следующие условия:

- система выполняет повторяющуюся операцию;
- эта операция может быть разделена на независимые части, степень перекрытия которых невелика;
- трудоемкость подопераций примерно одинакова.

Количество подопераций называют глубиной конвейера. Важным условием нормальной работы конвейера является отсутствие конфликтов, то есть данные, подаваемые в конвейер, должны быть независимыми. В том случае, когда очередной операнд зависит от

результата предыдущей операции, возникают такие периоды работы конвейера ("пузыри"), когда он пуст. Это еще одна проблема в работе конвейерных систем.

Увеличение быстродействия, которое можно получить с помощью конвейера приблизительно дается следующей формулой:

$$\frac{n \times d}{n + d}$$

где n - количество операндов, загружаемых в конвейер, d - глубина конвейера. Пусть требуется выполнить операцию сложения над двумя одномерными массивами по 200 элементов, причем выполнение сложения требует пять операций. В этом случае ускорение составит (200*5)/(200+5)=4.88. Разумеется, это идеальная ситуация, недостижимая в реальной жизни, в частности, считается, что нет "пузырей" и т.д. Это предполагает, как уже упоминалось, взаимную независимость данных.
В конвейерах команд также могут возникать простои, источником которых является зависимость между командами. Такие ситуации возникают при наличии в циклах ветвлений, то есть условных операторов.

Примером компьютера с суперскалярным процессором является IBM RISC/6000. Тактовая частота процессора у ЭВМ была 62.5 МГц, а быстродействие системы на вычислительных тестах достигало 104 Mflop (Mflop - единица измерения быстродействия процессора - миллион операций с плавающей точкой в секунду). Суперскалярный процессор не требует специальных векторизующих компиляторов, хотя компилятор должен в этом случае учитывать особенности архитектуры.

Векторный процессор — это процессор, в котором операндами некоторых команд могут выступать упорядоченные массивы данных — векторы (SIMD процессор). Отличается от скалярных процессоров, которые могут работать только с одним операндом в единицу времени. Абсолютное большинсто процессоров являются скалярными или близкими к ним. Векторные процессоры были распространены в сфере научных вычислений, где они являлись основой большинства суперкомпьютеров начиная с 1980-х до 1990-х. Но резкое увеличение производительности и активная разработка новых процессоров привели к вытеснению векторных процессоров со сферы повседневных процессоров.

Одновременная многопоточность (SMT — simultaneous multithreading) — следующий шаг в расширении возможностей процессора, ориентированный на приложения, требовательные к производительности, и поддерживающий метод распараллеливания задач на уровне инструкций на несколько каналов обработки процессора.

Одновременная многопоточная обработка на уровне приложений позволяет «видеть» два процессора вместо одного задействованного. В случае когда режим SMT отключен (однопоточная обработка), каждый физический процессор использует один логический. Когда же режим SMT активирован, каждый процессор, исполняющий в данный момент команды приложения, использует два логических процессора вместо одного. Технология многопоточности реализуется за счет того, что при выполнении разных команд приложения используются разные устройства процессора, что позволяет распределять по ним потоки.

Шины доступа к памяти и NUMA

Шины микрокомпьютера образует группа линий передачи сигналов с адресной информацией, данных, а также управляющих сигналов. Фактически ее можно разделить

на три части: адресную шину, шину данных и шину управляющих сигналов (здесь рассматривается 80286 процессор).

Уровни этих сигналов в данный момент времени определяют состояние системы в этот момент.

На адресную шину, состоящую из 24 линий, микропроцессор выставляет адрес байта или слова, который будет пересылаться по шине данных в процессор или из него. Кроме того, шина адреса используется микропроцессором для указания адресов периферийных портов, с которыми производится обмен данными.

Шина данных состоит из 16 линий. по которым возможна передача как отдельных байтов. так и двухбайтовых слов. При пересылке байтов возможна передача и по старшим 8 линиям, и по младшим. Шина данных двунаправлена, так как передача байтов и слов может производится как в микропроцессор, так и из него.

Шина управления формируется сигналами, поступающими непосредственно от микропроцессора, сигналами от шинного контроллера, а также сигналами, идущими к микропроцессору от других микросхем и периферийных адаптеров.

Микропроцессор использует шинный контроллер для формирования управляющих сигналов, определяющих перенос данных по шине. Он выставляет три сигнала -SO, -SI, M/-IO, которые определяют тип цикла шины (подтверждение прерывания, чтение порта ввода/вывода, останов, чтение памяти, запись в память). На основании значений этих сигналов шинный контроллер формирует управляющие сигналы, контролирующие динамику данного типа шины.

Для того, чтобы понять динамику работы, разберем, каким образом осуществляется процессором чтение слов из оперативной памяти. Это происходит в течение 4 тактов CLK, или 2 состояний процессора (т.е. каждое состояние процессора длится 2 такта синхросигнала CLK). Во время первого состояния, обозначаемого, как T 4s 0, процессор выставляет на адресную шину значение адреса, по которому будет читаться слово. Кроме того, он формирует на шине совместно с шинным контроллером соответствующие значения управляющих сигналов. Эти сигналы и адрес обрабатываются схемой управления памятью, в результате чего, начиная с середины второго состояния процессора
T 4c 0 (т.е. в начале четвертого такта CLK), на шине данных появляется значение содержимого соответствующего слова из оперативной памяти. И наконец, процессор считывает значение этого слова с шины данных. На этом перенос (копирование) значения слова из памяти в процессор заканчивается.

Таким образом, если частота кварцевого генератора, определяющая частоту CLK, равна 20 МГц, то максимальная пропускная способность шины данных равна (20/4) миллионов слов в секунду, или 10 В/сек. Реальная пропускная способность существенно ниже.

http://ru.wikipedia.org/wiki/Симметричная_мультипроцессорность

Симметричное мультипроцессирование (англ. Symmetric Multiprocessing, или SMP) это архитектура многопроцессорных компьютеров, в которой два или более одинаковых процессоров подключаются к общей памяти. Большинство многопроцессорных систем сегодня используют архитектуру SMP. SMP системы позволяют любому процессору работать над любой задачей независимо от того, где в памяти хранятся данные для этой

задачи; с должной поддержкой операционной системы, SMP системы могут легко перемещать задачи между процессорами эффективно распределяя нагрузку. С другой стороны, память гораздо медленнее процессоров, которые к ней обращаются, даже однопроцессорным машинам приходится тратить значительное время на получение данных из памяти. В SMP ситуация ещё более усугубляется, так как только один процессор может обращаться к памяти в данный момент времени.

SMP это лишь один подход к построению многопроцессорной машины; другим подходом является NUMA, которая предоставляет процессорам отдельные банки памяти. Это позволяет процессорам работать с памятью параллельно, и это может значительно повысить пропускную способность памяти, в случае когда данные привязаны к конкретному процессу (а следовательно и процессору). С другой стороны, NUMA повышает стоимость перемещения данных между процессорами, значит и балансирование загрузки обходится дороже. Преимущества NUMA ограничены специфическим кругом задач, в основном серверами, где данные часто прочно привязаны к конкретным задачам или пользователям.