МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (СГАУ)

Вычислительные системы

Электронный учебно-методический комплекс
по дисциплине в LMS Moodle

Работа выполнена по мероприятию блока 1 «Совершенствование
образовательной деятельности» Программы развития СГАУ
на 2009 – 2018 годы по проекту «Модернизация учебного процесса на факультете экономики и управления
на основе развития системы электронного и дистанционного обучения»
Соглашение № 1/21 от 3 июня 2013 г.

САМАРА
2013

УДК 004.272 (075) ББК 32.97я7
В 949

Автор-составитель: **Востокин Сергей Владимирович**

Вычислительные системы [Электронный ресурс] : электрон. учеб.-метод. комплекс по дисциплине в LMS Moodle / Мин-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т); авт.-сост. С. В. Востокин. - Электрон. текстовые и граф. дан. - Самара, 2013. – 1 эл. опт. диск (CD-ROM).

В состав учебно-методического комплекса входят:

1. Курс лекций.
2. Учебное пособие.
3. Задания на лабораторные работы.
4. Задания на практические работы.
5. Темы для подготовки к экзамену.
6. Тесты для итогового контроля знаний.
7. Рабочая программа.

УМКД «Вычислительные системы» предназначен для студентов факультета информатики, обучающихся по направлению подготовки магистров 230100.68 «Информатика и вычислительная техника» в В семестре.

УМКД разработан на кафедре Информационные системы и технологии.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)

Курс лекций по дисциплине «Вычислительные системы»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Лекция 1. Введение в вычислительные системы

План лекции

Причины появления параллельных вычислительных архитектур
Уровни параллелизма

Обзор лекции

Скорость работы компьютеров становится все выше, но и предъявляемые к ним требования постоянно растут. Астрономы пытаются воспроизвести всю историю Вселенной с момента большого взрыва и до сегодняшнего дня. Фармацевты хотели бы разрабатывать новые лекарственные препараты с помощью компьютеров, не принося в жертву легионы крыс. Разработчики летательных аппаратов могли бы получать лучшие результаты, если бы вместо строительства огромных аэродинамических труб моделировали свои конструкции на компьютере. Если говорить коротко, какими бы мощными ни были компьютеры, их возможностей никогда не хватит для решения многих нетривиальных задач (особенно научных, технических и промышленных).

Хотя тактовая частота постоянно растет, скорость коммутации нельзя увеличивать бесконечно. Главной проблемой остается скорость света — невозможно заставить протоны и электроны двигаться быстрее. Из-за высокой теплоотдачи компьютеры превратились в кондиционеры. Наконец, поскольку размеры транзисторов постоянно уменьшаются, в конце концов наступит время, когда каждый транзистор будет состоять из нескольких атомов, поэтому основной проблемой могут стать законы квантовой механики (например, принцип неопределенности Гейзенберга).

В результате, чтобы иметь возможность решать более сложные задачи, разработчики обратились к компьютерам параллельного действия (далее — параллельные компьютеры). Невозможно построить компьютер с одним процессором и временем цикла в 0,001 нс, но зато можно построить компьютер с 1000 процессорами, время цикла каждого из которых составляет 1 нс. И хотя быстродействия каждого процессора во втором случае очевидно мало, теоретически мы должны получить требуемую производительность.

Параллелизм можно вводить на разных уровнях. На самом низком уровне он может быть реализован в процессоре за счет конвейеризации и суперскалярной архитектуры с несколькими функциональными блоками. Скрытого параллелизма можно добиться путем значительного удлинения слов в командах. Посредством дополнительных функций можно «научить» процессор одновременно обрабатывать несколько программных потоков. Наконец, можно установить на одной микросхеме несколько процессоров.

Однако все эти приемы, вместе взятые, способны повысить производительность максимум в 10 раз по сравнению с классическими последовательными решениями.

На следующем уровне возможно внедрение в систему внешних плат ЦП с улучшенными вычислительными возможностями. Как правило, в подключаемых процессорах реализуются специальные функции, такие как обработка сетевых пакетов, обработка мультимедийных данных, криптография и т.д. Производительность специализированных приложений за счет этих функций может быть повышена в 5-10 раз.

Чтобы повысить производительность в сто, тысячу или миллион раз, необходимо свести воедино многочисленные процессоры и обеспечить их эффективное взаимодействие. Этот принцип реализуется в виде больших мультипроцессорных систем и мультикомпьютеров (кластерных компьютеров). Естественно, объединение тысяч процессоров в единую систему порождает новые проблемы, которые нужно решать.

Наконец, в последнее время появилась возможность интеграции через Интернет целых организаций. В результате формируются слабо связанные распределенные вычислительные сетки, или решетки. Такие системы только начинают развиваться, но их потенциал весьма высок.

Когда два процессора или обрабатывающих элемента находятся рядом и обмениваются большими объемами данных с небольшими задержками, они называются сильно связанными (tightly coupled). Соответственно, когда два процессора или обрабатывающих элемента располагаются далеко друг от друга и обмениваются небольшими объемами данных с большими задержками, они называются слабо связанными (loosely coupled). Мы обсудим принципы разработки систем этих форм параллелизма и рассмотрим ряд примеров. Начав с сильно связанных систем, для которых характерен внутрипроцессорный параллелизм, мы постепенно перейдем к слабо связанным системам и в завершающей части поговорим о распределенных вычислительных системах.

Лекция 2. Внутрипроцессорный параллелизм

План лекции

Параллелизм на уровне команд

Внутрипроцессорная многопоточность

Однокристальные мультипроцессоры

Обзор лекции

Один из очевидных способов увеличить производительность микросхемы состоит в том, чтобы заставить ее выполнять больше операций в единицу времени. В этой лекции мы рассмотрим некоторые приемы повышения быстродействия за счет параллелизма на уровне микросхемы, в частности, параллелизм на уровне команд, многопоточность и размещение на микросхеме нескольких процессоров. Все эти методики хоть и отличаются друг от друга, но в той или иной степени помогают решить задачу. Их роднит базовый принцип — «уплотнение» операций во времени.

Низкоуровневый параллелизм достигается, в частности, вызовом нескольких команд за один тактовый цикл. Процессоры, в которых реализуется этот принцип, делятся на две категории: суперскалярные и VLIW.

Рассмотрим схему суперскалярного процессора. В наиболее распространенных конфигурациях команда должна быть готова к исполнению определенной точке конвейера. Суперскалярные процессоры способны за один тактовый цикл вызывать несколько команд. Число фактически вызываемых команд зависит как от конструкции процессора, так и от текущей ситуации. Аппаратные ограничения диктуют максимальное число одновременно вызываемых команд — обычно от двух до шести. К тому же, если для выполнения команды нужен недоступный функциональный блок или еще не полученный результат выполнения другой команды, такая команда не будет вызвана даже при наличии физической возможности.

Другой вид параллелизма на уровне команд реализуется в процессорах сверхдлинным командным словом (Very Long Instruction Word, VLIW). В своем первоначальном исполнении VLIW-системы, действительно, отличались длинными словами с командами, обращавшимися к нескольким функциональным блокам. Для примера рассмотрим конвейер. Он включает в себя пять функциональных блоков и способен одновременно выполнять две целочисленные операции, одну операцию с плавающей точкой, одну команду загрузки и одну команду сохранения. В одной команде этой VLIW-системы содержится пять кодов операций и пять пар операндов — по одному коду и одной паре на каждый функциональный блок. С учетом того, что код операции занимает 6 бит, регистр — 5 бит, а ячейка памяти — 32 бита, общая длина команды может достигать 134 бита, что, согласитесь, немало.

Однако такое решение было признано неудачным. Дело в том, что не все команды могли обращаться к соответствующим функциональным блокам, результате в изобилии появлялись бессмысленные пустые операции. В

современных VLIW-системах должен быть предусмотрен какой-либо механизм маркировки связок команд, например, для этого может использоваться бит «завершения связки». Процессор может выбрать и запустить связку целиком. Задача по подготовке связок команд, которые могут выполняться совместно, решается компилятором.

Фактически в VLIW-системах решение проблемы совместимости команд переносится с периода исполнения на стадию компиляции. В результате аппаратное обеспечение упрощается и удешевляется. Кроме того, поскольку в работе компилятора нет жестких временных ограничений, связки команд формируются более осмысленно, нежели если бы это происходило в период исполнения. К сожалению, ввести в практику столь радикальные преобразования архитектуры процессора очень сложно, что подтверждается более чем умеренными темпами распространения процессора Itanium.

Нелишне заметить, что параллелизм на уровне команд не является единственно возможной формой низкоуровневого параллелизма. Существует также параллелизм на уровне памяти, предусматривающий одновременное исполнение в памяти множества операций.

Для всех современных конвейеризованных процессоров характерна одна и та же проблема — если при запросе к памяти слово не обнаруживается в кэшах первого и второго уровней, на загрузку этого слова в кэш уходит длительное время, в течение которого конвейер простаивает. Одна из методик решения этой проблемы называется внутрипроцессорной многопоточностью (on-chip multithreading). Она позволяет процессору одновременно управлять несколькими программными потоками и тем самым маскировать простои. Вкратце принцип многопоточности можно изложить так: если программный поток 1 блокируется, процессор может обеспечить полную загрузку аппаратуры, запустив программный поток 2. Основополагающая идея проста, реализуется она разными способами, которые мы и рассмотрим. Первый из них, называемый мелкомодульной многопоточностью (fine-grained multithreading), применительно к процессору, способному вызывать одну команду за такт. При мелкомодульной многопоточности простой маскируется путем исполнения потоков «по кругу», то есть в смежных циклах запускаются разные потоки.

Поскольку разные программные потоки никак друг с другом не связаны, каждому из них нужен свой набор регистров. Он должен быть указан для каждой вызываемой команды, и тогда аппаратное обеспечение будет знать, к какому набору регистров при необходимости нужно обращаться. Следовательно, максимальное число одновременно исполняемых программных потоков определяется в период разработки микросхемы.

Обращениями к памяти причины простоя не ограничиваются. Иногда для исполнения следующей команды требуется результат предыдущей команды, который еще не вычислен. В других случаях команда вызвана быть не может, так как она следует за условным переходом, направление которого еще неизвестно. Общее правило формулируется так: если в конвейере k ступеней, но по кругу можно запустить, по меньшей мере, k программных потоков, то в одном потоке любой отдельно взятый момент не может выполняться более одной команды, поэтому конфликты между ними исключены.

В такой ситуации процессор может работать на полной скорости, без простоя. Естественно, далеко не всегда число доступных потоков равно числу ступеней конвейера, поэтому некоторые разработчики предпочитают методику, называемую крупномодульной многопоточностью (coarse-grained multithreading). В данном случае программный поток A продолжает выполняться последовательно, вплоть до простоя. При этом теряется один цикл. Далее происходит переключение на первую команду программного потока B. Так как эта команда сразу переходит в состояние простоя, в цикле выполняется уже команда C.

Так как каждый раз при простое команды теряется один цикл, по своей эффективности крупномодульная многопоточность, казалось бы, уступает мелкомодульной, однако у нее есть одно существенное преимущество — за счет меньшего числа программных потоков значительно сокращается расход ресурсов процессора. При недостаточном количестве активных потоков эта методика оптимальна.

Судя по нашему описанию, при крупномодульной многопоточности просто выполняется переключение между потоками, однако это — не единственный предусматриваемый данной методикой вариант действий. Есть возможность немедленного переключения с команд, которые потенциально способны вызвать простой (например, загрузка, сохранение и переходы), без выяснения, действительно ли намечается простой. Эта стратегия позволяет переключаться раньше обычного (сразу после декодирования команды) и исключает бесконечные циклы. Иными словами, исполнение продолжается до того момента, пока не обнаружится возможность возникновения проблемы, после чего следует переключение. Такие частые переключения роднят крупномодульную многопоточность с мелкомодульной.

Вне зависимости от используемого варианта многопоточности, необходимо как-то отслеживать принадлежность каждой операции к тому или иному программному потоку. В рамках мелкомодульной многопоточности каждой операции присваивается идентификатор потока, поэтому при

перемещениях по конвейеру ее принадлежность не вызывает сомнений. Крупномодульная многопоточность предусматривает возможность очистки конвейера перед запуском каждого последующего потока. В таком случае четко определяется идентичность потока, исполняемого в данный момент. Естественно, данная методика эффективна только в том случае, если паузы между переключениями значительно больше времени, необходимого для очистки конвейера.

Лекция 3. Сопроцессоры

План лекции

Сетевые процессоры
Мультимедиа-процессоры
Криптопроцессоры

Обзор лекции

Большинство современных компьютеров подключены к локальной сети или Итернету. В результате технологического прогресса в области сетевого оборудования передача данных в сетях сегодня происходит так быстро, что программно обрабатывать все входящие и исходящие данные становится все сложнее. В связи с этим разрабатываются специальные сетевые процессоры, предназначенные для обработки трафика, и ими уже сегодня оснащаются многие профессиональные вычислительные системы. Мы вкратце напомним основы организации сетей и обсудим принципы работы сетевых процессоров.

Одним из путей аппаратного решения проблемы быстрой обработки пакетов является использование специализированных интегральных схем (Application-Specific Integrated Circuit, ASIC). Такая микросхема подобна аппаратно реализованной программе, которая может выполнять любое из заранее предусмотренных действий. Основой многих современных маршрутизаторов являются схемы ASIC. Впрочем, и со специализированными интегральными схемами связаны некоторые проблемы. Прежде всего, их долго проектировать и не менее долго производить. Кроме того, это жестко запрограммированные устройства, то есть чтобы внести новую функциональность, приходится разрабатывать и изготавливать новую микросхему. Хуже того, настоящим кошмаром являются ошибки, так как единственным способом их исправления является разработка, изготовление и установка новой (исправленной) микросхемы.

Наконец, этот подход является весьма затратным, если только большой объем производства не позволяет компенсировать расходы на разработку.

Второй подход основан на использовании программируемых вентильных матриц (Field Programmable Gate Array, FPGA). Такая матрица представляет собой набор вентилей, из которых путем перекоммутации строится требуемая схема. Сроки выхода программируемых вентильных матриц на рынок гораздо короче, чем у специализированных интегральных схем, к тому же их можно перепрограммировать в «полевых условиях» при помощи специального программатора. Но в то же время они очень сложные, дорогие и более медленные, чем схемы ASIC, поэтому программируемые вентильные матрицы не получили широкого распространения, исключая некоторые узкоспециальные области. Наконец, перейдем к сетевым процессорам — устройствам, способным обрабатывать входящие и исходящие пакеты со скоростью их передачи, то есть в реальном времени. Обычно они реализуются в виде съемной платы, содержащей, помимо кристалла сетевого процессора, память и вспомогательную логику. К плате подключается одна или несколько сетевых линий. Процессор получает из линии пакеты, обрабатывает их, после чего передает по другой линии, если это маршрутизатор, или отправляет в главную системную шину (то есть в шину PCI), если это оконечное устройство, которым может быть, например, персональный компьютер.

Сетевые процессоры оптимизированы для быстрой обработки большого количества входящих и исходящих пакетов. Это означает, что по каждой из сетевых линий проходят миллионы пакетов в секунду, а маршрутизатор должен поддерживать десятки таких линий. Столь серьезных показателей можно достигнуть только на процессорах с высокой степенью внутреннего параллелизма. Кроме того, в процессор обязательно входят несколько PPE-кошроллеров (Protocol/ Programmable/Packet Processing Engine — программируемая система обработки пакетов и протоколов), каждая из которых состоит из RISC-ядра (возможно, модифицированного) и внутренней памяти небольшого объема для хранения программы и нескольких переменных.

Есть два подхода к организации PPE-контроллеров. В простейшем случае все PPE-контроллеры делаются идентичными. Когда в сетевой процессор приходит новый пакет, он передается для обработки тому PPE-контроллеру, который в данный момент бездействует. Если свободных PPE-контроллеров нет, пакет ставится в очередь в расположенной на плате памяти SDRAM, ожидая освобождения одного из PPE-контроллеров. При такой

организации горизонтальные связи, показанные на рис. 8.13, отсутствуют, так как у разных PPE-контроллеров нет необходимости общаться друг с другом.

Другой подход к организации PPE-контроллеров — конвейер, где каждый из PPE-контроллеров выполняет один этап обработки, после чего передает указатель на полученный пакет следующему PPE-контроллеру в конвейере. В обоих вариантах организации PPE-контроллеры являются полностью программируемыми.

В более совершенных сетевых процессорах PPE-контроллеры поддерживают многопоточность, то есть каждый из них имеет несколько наборов регистров и аппаратный регистр, показывающий, какой из наборов используется. Это позволяет одновременно выполнять несколько программ (то есть программных потоков) и переключаться между ними, просто изменяя переменную «текущего рабочего набора регистров». Когда один из программных потоков вынужден ждать (например, при обращении к SDRAM, на которое требуется несколько циклов), PPE-контроллер может быть мгновенно переключен на поток, способный продолжать работу. Это позволяет добиваться высокой загрузки PPE-контроллеров, даже несмотря на необходимость часто ожидать завершения обмена данными с SDRAM или других медленных внешних операций.

Помимо PPE-контроллеров, у всех сетевых процессоров имеется управляющий процессор для выполнения всех действий, не относящихся напрямую к обработке пакетов (например, обновление таблиц маршрутизации). Обычно он представляет собой RISC-процессор общего назначения, память для данных и команд которого находится на одном кристалле с процессором. Более того, в сетевом процессоре может быть несколько специализированных процессоров, предназначенных для выполнения критически важных операций. Они представляют собой очень маленькие специализированные интегральные схемы (ASIC), способные выполнять только одно несложное действие, такое как поиск целевого адреса в таблице маршрутизации. Все компоненты сетевого процессора взаимодействуют друг с другом на мультигигабитных скоростях по одной или нескольким расположенным на кристалле параллельным шинам.

Еще одна область применения сопроцессоров — обработка фотографических изображений высокого разрешения, а также аудио- и видеопотоков. Обычно центральный процессор недостаточно хорош, когда в этих приложениях приходится выполнять сложные вычисления над большими объемами данных. По этой причине некоторые современные персональные компьютеры и большинство разрабатываемых моделей

оборудуются специальными сопроцессорами для обработки мультимедийной информации, на которые можно переложить значительную часть работы.

Безопасность, а особенно сетевая безопасность, является еще одной (уже третьей) областью, в которой широко используются сопроцессоры. Когда между клиентом и сервером устанавливается соединение, обычно требуется их взаимная аутентификация. По установленному таким образом безопасному (шифруемому) соединению можно безопасно передавать данные и не думать о злоумышленниках, прослушивающих линию. Проблема здесь в том, что безопасность обеспечивается средствами криптографии, а эта область требует весьма объемных вычислений. В криптографии сейчас распространены два основных подхода к защите данных: шифрование с симметричным ключом и шифрование с открытым ключом. Первый основан на очень тщательном перемешивании битов (как будто сообщение помещают в некий электронный миксер). В основе второго подхода лежит умножение и возведение в степень больших чисел 1024-разрядных), что требует исключительно больших временных затрат. Многими компаниями выпущены криптографические сопроцессоры, позволяющие шифровать данные для их безопасной передачи и потом расшифровывать их. Зачастую они представляют собой карты расширения, вставляемые в PCI-разъем. Благодаря специальному аппаратному обеспечению, эти процессоры могут выполнять необходимые криптографические вычисления намного быстрее, чем центральный процессор.

Лекция 5. Мультипроцессоры

План лекции

Мультипроцессоры и мультикомпьютеры
Семантика памяти
UMA-мультипроцессоры в симметричных мультипроцессорных архитектурах
NUMA-мультипроцессоры
COMA-мультипроцессоры

Обзор лекции

В любой параллельной компьютерной системе процессоры, выполняющие разные части единого задания, должны как-то взаимодействовать друг с другом, чтобы обмениваться информацией. Как именно должен происходить обмен? Для этого было предложено и реализовано две стратегии: мультипроцессоры и мультикомпьютеры.

Ключевое различие между стратегиями состоит в наличии или отсутствии общей памяти. Это различие сказывается как на конструкции, устройстве и программировании таких систем, так и на их стоимости и размерах.

Параллельный компьютер, в котором все процессоры совместно используют общую физическую память, называется мультипроцессором, или системой с общей памятью. Все процессы, работающие в мультипроцессоре совместно, могут иметь единое виртуальное адресное пространство, отображенное на общую память. Любой процесс с помощью команд LOAD и STORE может считать слово из памяти или записать слово в память. Больше ничего не требуется. Два процесса имеют возможность легко обмениваться информацией — для этого один из них просто записывает данные в общую память, а другой их считывает.

Во втором варианте параллельной архитектуры каждый процессор имеет собственную память, доступную только этому процессору. Такая схема называется мультикомпьютером, или системой с распределенной памятью.

Ключевое отличие мультикомпьютера от мультипроцессора состоит в том, что каждый процессор в мультикомпьютере имеет собственную локальную память, к которой этот процессор может обращаться, выполняя команды LOAD и STORE, но никакой другой процессор с помощью этих команд не может получить доступ к локальной памяти данного процессора. Таким образом, мультипроцессоры имеют одно физическое адресное пространство, разделяемое всеми процессорами, а мультикомпьютеры содержат отдельные физические адресные пространства для каждого процессора.

Хотя во всех мультипроцессорах процессорам предоставляется образ общего единого адресного пространства, часто наряду с ним имеется множество модулей памяти, в каждом из которых хранится какой-то фрагмент физической памяти. Процессоры и модули памяти соединяются сложной коммуникационной сетью (мы поговорим об этом в подразделе «Коммуникационные сети» раздела «Мультикомпьютеры»). Несколько процессоров могут попытаться считать слово из памяти в то же время, когда другие процессоры будут пытаться его записать; сообщения могут доставляться не в том порядке, в котором они были отправлены. Добавим к этим проблемам существование многочисленных копий некоторых фрагментов памяти (например, в кэш-памяти), и в результате мы придем к хаосу, если не принять серьезные контрмеры. В этом подразделе мы выясним, что в действительности представляет собой общая память и как при таких обстоятельствах можно разумно использовать модули памяти.

Самые простые мультипроцессоры имеют всего одну шину (рис. 8.21, а). Два или более процессора и один или несколько модулей памяти используют эту шину для взаимодействия. Если процессору нужно считать слово из памяти, он сначала проверяет, свободна ли шина. Если шина свободна, процессор помещает адрес нужного слова на шину, устанавливает несколько управляющих сигналов и ждет, когда память поместит на шину запрошенное слово.

Если шина занята, процессор просто ждет, когда она освободится. С этой схемой связана одна проблема. При наличии двух или трех процессоров доступ к шине регулировать не сложно, трудности возникают, когда процессоров 32 или 64. Производительность системы в этом случае полностью определяется пропускной способностью шины, и многим процессорам большую часть времени приходится простаивать.

Чтобы разрешить проблему, нужно добавить к каждому процессору кэш-память. Кэш-память может находиться внутри микросхемы процессора, рядом с микросхемой процессора, на плате процессора. Допустимы любые комбинации этих вариантов. Поскольку в этом случае считывать многие слова можно будет из кэша, трафик на шине снизится, и система сможет обслуживать большее количество процессоров. Таким образом, кэширование дает в данном случае значительный эффект.

В следующей схеме каждый процессор имеет не только кэш, но и собственную локальную память, к которой он получает доступ через выделенную локальную шину. Чтобы оптимально задействовать эту конфигурацию, компилятор должен поместить в локальные модули памяти весь программный код, строки, константы и другие данные, предназначенные только для чтения, а также стеки и локальные переменные. Общая память потребуется только для хранения совместно используемых переменных. В большинстве случаев такое разумное распределение значительно снижает интенсивность трафика на шине и не требует активного содействия со стороны компилятора.

Количество процессоров в UMA-мультипроцессорах с одной шиной обычно ограничивается несколькими десятками, а для мультипроцессоров с перекрестной или многоступенчатой коммутацией требуется дорогое оборудованияе, к тому же количество процессоров в них не намного больше. Чтобы объединить в одном мультипроцессоре более 100 процессоров, нужно какое-то иное решение. Ранее предполагалось, что все модули памяти имеют одинаковое время доступа. Если не замыкаться на этой концепции, можно прийти к мультипроцессорам с неоднородным доступом к памяти (NonUniform Memory Access, NUMA). Как и UMA-мультипроцессоры, они

предоставляют единое адресное пространство для всех процессоров, но, в отличие от UMA-машин, доступ к локальным модулям памяти происходит быстрее, чем к удаленным. Следовательно, все UMA-программы смогут без изменений работать на NUMA-машинах, но производительность будет хуже, чем на UMA-машине с той же тактовой частотой.

NUMA-машины имеют три ключевые характеристики, которые в совокупности отличают их от других мультипроцессоров: существует единое адресное пространство, видимое всеми процессорами; доступ к удаленной памяти производится командами LOAD и STORE; доступ к удаленной памяти выполняется медленнее, чем доступ к локальной.

NUMA- и CC-NUMA-машины обладают одним серьезным недостатком: обращения к удаленной памяти выполняются гораздо медленнее, чем к локальной. В CC-NUMA-машине эта разница в производительности в какой-то степени нивелируется за счет кэш-памяти. Однако если объем запрашиваемых удаленных данных значительно превышает вместимость кэш-памяти, постоянно будут происходить кэш-промахи, что негативно скажется на производительности.

Мы уже знаем, что достаточно высокую производительность имеют UMA-машины, но число процессоров в них невелико, к тому же они довольно дороги. NC-NUMA-машины хорошо масштабируются, но в них требуется ручное или полуавтоматическое размещение страниц памяти, результаты которого часто плачевны. Дело в том, что очень непросто предсказать, где и какие страницы могут понадобиться, кроме того, страницы трудно перемещать из-за их больших размеров. CC-NUMA-машины, такие как мультипроцессор Sun Fire E25K, начинают работать очень медленно, если большому числу процессоров требуются большие объемы удаленных данных. Так или иначе, каждая из этих схем имеет существенные недостатки.

Однако существует мультипроцессор, в котором все эти проблемы решаются за счет использования основной памяти каждого процессора в качестве кэш-памяти. Такая система называется COMA (Cache Only Memory Access — доступ только к кэш-памяти). В ней страницы не имеют собственных «домашних» машин, как в системах NUMA и CC-NUMA, фактически, страницы в этой системе вообще не имеют «прописки».

Вместо этого физическое адресное пространство делится на строки кэша, которые по запросу свободно перемещаются в системе. Блоки памяти не имеют собственных машин. У них, как у кочевников в некоторых странах третьего мира, дом там, где они оказались. Память, которая привлекает строки по мере необходимости, называется притягивающей. Использование

основной памяти в качестве большого кэша увеличивает процент кэш-попаданий, а, следовательно, и производительность.

Лекция 6. Распределенные вычислительные системы

План лекции

Мультикомпьютеры
Коммуникационные сети
Процессоры с массовым параллелизмом
Кластерные вычисления
Коммуникационное программное обеспечение для мультикомпьютеров
Планирование
Общая память на прикладном уровне
Производительность
Грид-системы

Обзор лекции

В категорию MIMD входят два вида процессоров с параллельной архитектурой: мультипроцессоры и мультикомпьютеры. В предыдущем разделе мы рассматривали мультипроцессоры. Мы выяснили, что мультипроцессоры могут иметь общую память, доступ к которой выполняется обычными командами LOAD и STORE. Для реализации такой памяти может использоваться множество схем, включая шины слежения, сети с перекрестной и многоступенчатой коммутацией, различные схемы на основе каталога. Во всех случаях программы, написанные для мультипроцессора, могут получать доступ к любому месту в памяти, не имея никакой информации о внутренней топологии или схеме реализации. Именно благодаря такой иллюзии мультипроцессоры весьма популярны у пользователей и программистов.

Однако мультипроцессорам свойственны и некоторые недостатки, и это автоматически означает усиление роли мультикомпьютеров. В первую очередь, мультипроцессоры плохо масштабируются. Мы уже знаем, сколько разнообразных устройств потребовалось инженерам компании Sun ввести в систему E25K, чтобы поддерживать работу 72 процессоров. Что касается мультикомпьютеров, то далее мы рассмотрим систему из 65 536 процессоров. Пройдут годы, прежде чем кто-нибудь сумеет построить коммерческий мультипроцессор с 65 536 узлами, да и к тому времени в ходу уже будут мультикомпьютеры с миллионами процессоров.

Далее, на производительности мультипроцессора может серьезно сказываться конкуренция за доступ к памяти. Если 100 процессоров постоянно пытаются считывать и записывать одни и те же переменные, конкуренция за ресурсы модулей памяти, шин и каталогов может сильно ударить по производительности.

Вследствие этих и других факторов разработчики проявляют повышенный интерес к таким параллельным компьютерным архитектурам, в которых каждый процессор имеет собственную память, недоступную напрямую для других процессоров. Это — мультикомпьютеры. Поскольку программы на разных процессорах в мультикомпьютере не могут получить доступ к памяти других процессоров командами LOAD и STORE, они взаимодействуют друг с другом с помощью примитивов send и receive, которые используются для передачи сообщений. Это различие полностью меняет модель программирования.

Мультикомпьютеры связываются друг с другом через коммуникационные сети. Интересно отметить, что мультикомпьютеры и мультипроцессоры в этом отношении очень похожи, поскольку мультипроцессоры часто содержат несколько модулей памяти, которые также должны связываться друг с другом и с процессорами. Следовательно, многое из того, о чем мы будем говорить в этом подразделе, применимо к обоим типам параллельных компьютерных архитектур.

Основная причина сходства коммуникационных связей в мультипроцессоре и мультикомпьютере заключается в том, что в обоих случаях имеет место передача сообщений. Даже в однопроцессорной машине, когда процессору нужно считать или записать слово, он активизирует определенные линии на шине и ждет ответа. Это примерно то же самое, что и обмен сообщениями: инициатор посылает запрос и ждет ответа. В больших мультипроцессорах при взаимодействии между процессорами и удаленной памятью процессор почти всегда посылает в память сообщение, так называемый пакет, в котором запрашиваются те или иные данные, а память посылает процессору ответный пакет.

Мультикомпьютеры имеют столь разнообразные формы и размеры, что выстроить для них сколько-нибудь внятную классификацию очень трудно. Тем не менее, можно выделить два основных «стиля» — это процессоры с массовым параллелизмом и кластеры.

Процессоры с массовым параллелизмом (Massively Parallel Processors, MPP) — это огромные суперкомпьютеры стоимостью в несколько миллионов долларов. Они используются в различных отраслях науки и техники для выполнения сложных вычислений, обработки большого числа транзакций в

секунду, управления большими базами данных, и т. д. Изначально это были суперкомпьютеры, предназначенные в основном для научных расчетов, но сейчас многие из них находят применение в коммерции. В целом, можно говорить, что MPP-мультикомпыотеры вытеснили SIMD-машины, векторные суперкомпьютеры и матричные процессоры.

В большинстве MPP-машин используются стандартные процессоры. Еще одна характеристика MPP — огромные объемы ввода-вывода. С помощью MPP-мультикомпьютеры обычно приходится обрабатывать огромные массивы данных, иногда терабайты.

Другой вариант мультикомпьютера — кластерный компьютер [12, 137]. Как правило, кластер состоит из нескольких сотен или тысяч связанных сетью персональных компьютеров или рабочих станций, причем к сети они подключаются через обычную сетевую плату. Различие между MPP и кластером такое же, как между мэйнфреймом и персональным компьютером. У обоих есть процессор, ОЗУ, диски, операционная система и т.д. Но в мэйнфрейме все это (за исключением, может быть, операционной системы) работает гораздо быстрее, и из-за этого применяются и управляются они совершенно по-разному. То же самое можно сказать о MPP и кластерах.

Еще несколько лет назад взаимодействие между элементами, образующими MPP, происходило гораздо быстрее, чем между машинами, составляющими кластер. Однако с появлением на рынке высокоскоростных сетей этот разрыв стал сходить «на нет». Вероятно, кластеры постепенно вытеснят MMP-машины, подобно тому как персональные компьютеры вытеснили мэйнфреймы, которые применяются теперь только в узкоспециализированных областях. Основной нишей для систем MMP останутся дорогостоящие суперкомпьютеры, в которых главное — производительность, а вопросы стоимости не имеют решающего значения.

Для программирования мультикомпыотера требуется специальное программное обеспечение (обычно это библиотеки), позволяющее обеспечить взаимодействие между процессами и синхронизацию. Отметим, что в большинстве случаев программные пакеты предназначаются и для MPP-машин, и для кластеров, поэтому приложения являются переносимыми между платформами. В системах передачи сообщений два и более процесса работают независимо друг от друга. Например, один из процессов может генерировать данные, а другой (или другие) — их потреблять. Если у отправителя есть еще данные, нет никакой гарантии, что получатель (получатели) готов принять эти данные, поскольку каждый процесс работает по собственной программе.

В большинстве систем передачи сообщений используются два примитива send и receive, но возможны и другие варианты семантики. Тремя основными вариантами являются: синхронная передача сообщений; буферизуемая передача сообщений; неблокирующая передача сообщений.

Несколько лет назад пакет PVM (Parallel Virtual Machine — параллельная виртуальная машина) считался самым популярным пакетом для обмена информацией между мультикомпьютерами. Однако в настоящее время он почти повсеместно вытеснен пакетом MPI (Message-Passing Interface — интерфейс передачи сообщений). Пакет MPI гораздо сложнее, чем PVM; он поддерживает намного больше библиотечных вызовов и намного больше параметров для каждого вызова.

Программистам систем MPI несложно разрабатывать задания, в которых запросы делаются сразу к нескольким процессорам и которые выполняются достаточно долго. При наличии нескольких запросов от нескольких пользователей для обработки каждого запроса задействуется разное количество процессоров на разное время. Поэтому кластеру необходим планировщик, определяющий, когда запускать каждое из заданий.

Из наших примеров видно, что мультикомпьютеры поддаются масштабированию гораздо лучше, чем мультипроцессоры. Этот факт привел к возникновению систем передачи сообщений, таких как MPI. Многим программистам эта модель не нравится, и они предпочли бы иметь иллюзию общей памяти, даже если ее на самом деле не существует. Если бы удалось достичь этой цели, это было бы прекрасно во всех отношениях: и в отношении удешевления аппаратуры (по крайней мере, на уровне каждого узла), и в отношении удобства программирования. Многие исследователи пришли к выводу, что общую память не обязательно строить на архитектурном уровне — существуют другие пути. Далее рассматривается, как ввести общую память в программную модель мультикомпьютера, если аппаратно она не поддерживается.

Цель создания параллельного компьютера — добиться, чтобы он работал быстрее, чем однопроцессорная машина. Если эта цель не достигнута, никакого смысла в разработке параллельного компьютера нет. Более того, эта цель должна быть достигнута при минимальных затратах. Машина, которая работает в два раза быстрее, чем однопроцессорная, но стоит в 50 раз дороже последней, вряд ли будет пользоваться спросом. В этом подразделе мы рассмотрим некоторые аспекты производительности параллельных компьютерных архитектур: аппаратные и программные метрики производительности и приёмы повышения производительности.

До недавнего времени было очень сложно обеспечить совместную работу разных организаций, в которых используются разные операционные системы, разные базы данных и протоколы. Однако рост потребности в крупномасштабном сотрудничестве между организациями привел к развитию систем и технологий объединения разрозненных компьютеров в то, что получило название распределенных вычислений (grid computing). Систему распределенных вычислений можно рассматривать как очень большой, интернациональный слабо связанный гетерогенный кластер.

Лекция 7. Многоуровневая компьютерная организация

План лекции

Языки, уровни и виртуальные машины
Современные многоуровневые машины
Развитие многоуровневых машин

Обзор лекции

Существует огромная разница между тем, что удобно людям, и тем, что могут компьютеры. Люди хотят сделать X, но компьютеры могут сделать только Y. Из-за этого возникает проблема. Вышеупомянутую проблему можно решить двумя способами. Оба способа подразумевают разработку новых команд, более удобных для человека, чем встроенные машинные команды. Эти новые команды в совокупности формируют язык, который мы будем называть Я 1. Встроенные машинные команды тоже формируют язык, и мы будем называть его Я 0. Компьютер может выполнять только программы, написанные на его машинном языке Я 0. Два способа решения проблемы различаются тем, каким образом компьютер будет выполнять программы, написанные на языке Я 1, — ведь в конечном итоге компьютеру доступен только машинный язык Я 0.

Первый способ выполнения программы, написанной на языке Я 1, подразумевает замену каждой команды эквивалентным набором команд на языке Я 0. В этом случае компьютер выполняет новую программу, написанную на языке Я 0, вместо старой программы, написанной на Я 1. Эта технология называется трансляцией.

Второй способ означает создание программы на языке Я 0, получающей в качестве входных данных программы, написанные на языке Я 1. При этом каждая команда языка Я 1 обрабатывается поочередно, после чего сразу выполняется эквивалентный ей набор команд языка Я 0. Эта технология не требует составления новой программы на Я 0. Она называется

интерпретацией, а программа, которая осуществляет интерпретацию, называется интерпретатором.

Между трансляцией и интерпретацией много общего. В обоих подходах компьютер в конечном итоге выполняет набор команд на языке Я 0, эквивалентных командам Я 1. Различие лишь в том, что при трансляции вся программа Я 1 переделывается в программу Я 0, программа Я 1 отбрасывается, а новая программа на Я 0 загружается в память компьютера и затем выполняется.

При интерпретации каждая команда программы на Я 1 перекодируется в Я 0 и сразу же выполняется. В отличие от трансляции, здесь не создается новая программа на Я 0, а происходит последовательная перекодировка и выполнение команд. С точки зрения интерпретатора, программа на Я 1 есть не что иное, как «сырые» входные данные. Оба подхода широко используются как вместе, так и по отдельности.

Впрочем, чем мыслить категориями трансляции и интерпретации, гораздо проще представить себе существование гипотетического компьютера или виртуальной машины, для которой машинным языком является язык Я 1. Назовем такую виртуальную машину М 1, а виртуальную машину для работы с языком Я 0 — М 0. Если бы такую машину М 1 можно было бы сконструировать без больших денежных затрат, язык Я 0, да и машина, которая выполняет программы на языке Я 0, были бы не нужны. Можно было бы просто писать программы на языке Я 1, а компьютер сразу бы их выполнял. Даже с учетом того, что создать виртуальную машину, возможно, не удастся (из-за чрезмерной дороговизны или трудностей разработки), люди вполне могут писать ориентированные на нее программы. Эти программы будут транслироваться или интерпретироваться программой, написанной на языке Я 0, а сама она могла бы выполняться существующим компьютером. Другими словами, можно писать программы для виртуальных машин так, как будто эти машины реально существуют.

Трансляция и интерпретация целесообразны лишь в том случае, если языки Я 0 и Я 1 не слишком отличаются друг от друга. Это значит, что язык Я 1 хотя и лучше, чем Я 0, но все же далек от идеала. Возможно, это несколько обескураживает в свете первоначальной цели создания языка Я 1 — освободить программиста от бремени написания программ на языке, понятном компьютеру, но малоприспособленном для человека. Однако ситуация не так безнадежна. Очевидное решение проблемы — создание еще одного набора команд, которые в большей степени, чем Я 1 ориентированы на человека и в меньшей степени на компьютер. Этот третий набор команд также формирует язык, который мы будем называть Я 2, а соответствующую

виртуальную машину — М 2. Человек может писать программы на языке Я 2, как будто виртуальная машина для работы с машинным языком Я 2 действительно существует. Такие программы могут либо транслироваться на язык Я 1, либо выполняться интерпретатором, написанным на языке Я 1. Язык, находящийся в самом низу иерархической структуры, — самый примитивный, а тот, что расположен на ее вершине — самый сложный.

Между языком и виртуальной машиной существует важная зависимость. Каждая машина поддерживает какой-то определенный машинный язык, состоящий из всех команд, которые эта машина может выполнять. В сущности, машина определяет язык. Сходным образом язык определяет машину, которая может выполнять все программы, написанные на этом языке. Машину, определяемую тем или иным языком, очень сложно и дорого конструировать из электронных схем, однако представить себе такую машину мы можем. Компьютер для работы с машинным языком С или С++ был бы слишком сложным, но в принципе его можно разработать, учитывая высокий уровень современных технологий. Однако существуют веские причины не создавать такой компьютер — это крайне неэффективное, по сравнению с другими, решение. Действительно, технология должна быть не только осуществимой, но и рациональной.

Компьютер с n уровнями можно рассматривать как n разных виртуальных машин, у каждой из которых есть свой машинный язык. Термины «уровень» и «виртуальная машина» мы будем использовать как синонимы. Только программы, написанные на Я 0, могут выполняться компьютером без трансляции или интерпретации. Программы, написанные на Я 1, Я 2, ..., Я n, должны проходить через интерпретатор более низкого уровня или транслироваться на язык, соответствующий более низкому уровню.

Большинство современных компьютеров состоит из двух и более уровней. Существуют машины даже с шестью уровнями (рис. 1.2). Уровень 0 — это аппаратное обеспечение машины. Электронные схемы на уровне 1 выполняют машинно-зависимые программы. Ради полноты нужно упомянуть о существовании еще одного уровня, который расположен ниже нулевого. Этот уровень попадает в сферу электронной техники и здесь не рассматривается . Он называется уровнем физических устройств. На этом уровне находятся транзисторы, которые для разработчиков компьютеров являются примитивами. Объяснить, как работают транзисторы, — задача физики.

На самом нижнем уровне из тех, что мы будем изучать, а именно, на цифровом логическом уровне, объекты называются вентилями. Хотя вентили

состоят из аналоговых компонентов, таких как транзисторы, они могут быть точно смоделированы как цифровые устройства. У каждого вентиля есть один или несколько цифровых вхоэдс! (сигналов, представляющих 0 или 1). Вентиль вычисляет простые функции этих сигналов, такие как И или ИЛИ. Каждый вентиль формируется из нескольких транзисторов. Несколько вентилей формируют 1 бит памяти, который может содержать 0 или 1. Биты памяти, объединенные в группы, например, по 16, 32 или 64, формируют регистры. Каждый регистр может содержать одно двоичное число до определенного предела. Из вентилей также может состоять сам компьютер.

Следующий уровень называется уровнем микроархитектуры. Уровень 2 мы будем называть уровнем архитектуры набора команд. Следующий уровень обычно является гибридным. Большинство команд в его языке есть также и на уровне архитектуры набора команд (команды, имеющиеся на одном из уровней, вполне могут быть представлены и на других уровнях). У этого уровня есть некоторые дополнительные особенности: новый набор команд, другая организация памяти, способность выполнять две и более программы одновременно и некоторые другие. При построении уровня 3 возможно больше вариантов, чем при построении уровней 1 и 2.

Новые средства, появившиеся на уровне 3, выполняются интерпретатором, который работает на втором уровне. Этот интерпретатор был когда-то назван операционной системой. Команды уровня 3, идентичные командам уровня 2, выполняются микропрограммой или аппаратным обеспечением, но не операционной системой. Другими словами, одна часть команд уровня 3 интерпретируется операционной системой, а другая часть — микропрограммой. Вот почему этот уровень считается гибридным. Мы будем называть этот уровень уровнем операционной системы. Уровни с четвертого и выше предназначены для прикладных программистов, решающих конкретные задачи.

У первых цифровых компьютеров в 40-х годах было только два уровня: уровень архитектуры набора команд, на котором осуществлялось программирование, и цифровой логический уровень, выполнявший программы. Схемы цифрового логического уровня были ненадежны, сложны для производства и понимания.

В 1951 году Морис Уилкс (Maurice Wilkes), исследователь Кембриджского Университета, предложил идею разработки трехуровневого компьютера, призванную упростить аппаратное обеспечение [219]. Эта машина должна была иметь встроенный неизменяемый интерпретатор (микропрограмму), функция которого заключалась в выполнении программ уровня ISA посредством интерпретации. Так как аппаратное обеспечение

должно было теперь вместо программ уровня ISA выполнять только микропрограммы с ограниченным набором команд, требовалось меньшее количество электронных схем. Поскольку электронные схемы тогда делались из электронных ламп, данное упрощение призвано было сократить количество ламп и, следовательно, повысить надежность (которая в то время выражалась числом поломок за день).

В 60-е годы человек попытался ускорить дело, автоматизировав работу оператора. Программа под названием операционная система загружалась в компьютер на все время его работы. Программист приносил пачку перфокарт со специализированной программой, которая выполнялась операционной системой.

С 1970 года, когда получило развитие микропрограммирование, у производителей появилась возможность вводить новые машинные команды расширением микропрограммы, то есть путем программирования. Это открытие привело к виртуальному взрыву в производстве программ машинных команд, поскольку производители начали конкурировать друг с другом, стараясь выпустить лучшие программы. Эти команды не представляли особой ценности, поскольку те же задачи можно было легко решить, используя уже существующие программы, но обычно они работали немного быстрее. Например, во многих компьютерах использовалась команда INC (INCrement), которая прибавляла к числу единицу. Тогда уже существовала общая команда сложения ADD, и не было необходимости вводить новую команду, прибавляющую к числу единицу. Тем не менее команда INC работала немного быстрее, чем ADD, поэтому ее также включили в набор команд.

В 60-70-х годах количество микропрограмм значительно увеличилось. Однако они работали все медленнее и медленнее, поскольку требовали большого объема памяти. В конце концов исследователи осознали, что отказ от микропрограмм резко сократит количество команд, и компьютеры станут работать быстрее. Таким образом, компьютеры вернулись к тому состоянию, в котором они находились до изобретения микропрограммирования.

Мы рассмотрели развитие компьютеров, чтобы показать, что граница между аппаратным и программным обеспечением постоянно смещается. Сегодняшнее программное обеспечение может быть завтрашним аппаратным обеспечением и наоборот. Также обстоит дело и с уровнями — между ними нет четких границ. Для программиста не важно, как на самом деле выполняется команда (за исключением, может быть, скорости выполнения). Программист, работающий на уровне архитектуры системы, может использовать команду умножения, как будто это команда аппаратного

обеспечения, и даже не задумываться об этом. То, что для одного человека — программное обеспечение, для другого — аппаратное. Позже мы еще вернемся к этим вопросам.

Лекция 8. Развитие компьютерной архитектуры

План лекции

Нулевое поколение — механические компьютеры
Первое поколение — электронные лампы
Второе поколение — транзисторы
Третье поколение — интегральные схемы
Четвертое поколение — сверхбольшие интегральные схемы
Пятое поколение — невидимые компьютеры

Обзор лекции

В ходе эволюции компьютерных технологий были разработаны сотни разных компьютеров. Многие из них давно забыты, в то время как влияние других на современные идеи оказалось весьма значительным. В этом разделе мы дадим краткий обзор некоторых ключевых исторических моментов, чтобы лучше понять, каким образом разработчики дошли до концепции современных компьютеров. Мы рассмотрим только основные моменты развития, оставив многие подробности за скобками.

Первым человеком, создавшим счетную машину, был французский ученый Блез Паскаль A623-1662), в честь которого назван один из языков программирования. Паскаль сконструировал эту машину в 1642 году, когда ему было всего 19 лет, для своего отца, сборщика налогов. Это была механическая конструкция с шестеренками и ручным приводом. Счетная машина Паскаля могла выполнять только операции сложения и вычитания.

Тридцать лет спустя великий немецкий математик Готфрид Вильгельм Лейбниц 1646-1716) построил другую механическую машину, которая помимо сложения и вычитания могла выполнять операции умножения и деления. В сущности, Лейбниц три века назад создал подобие карманного калькулятора с четырьмя функциями.

Еще через 150 лет профессор математики Кембриджского Университета, Чарльз Бэббидж 1792-1871), изобретатель спидометра, разработал и сконструировал разностную машину. Эта механическая машина, которая, как и машина Паскаля, могла лишь складывать и вычитать, подсчитывала таблицы чисел для морской навигации. В машину был заложен только один алгоритм — метод конечных разностей с использованием полиномов. У этой

машины был довольно интересный способ вывода информации: результаты выдавливались стальным штампом на медной дощечке, что предвосхитило более поздние средства ввода-вывода — перфокарты и компакт-диски.

Хотя его устройство работало довольно неплохо, Бэббиджу вскоре наскучила машина, выполнявшая только один алгоритм. Он потратил очень много времени, большую часть своего семейного состояния и еще 17 000 фунтов, выделенных правительством, на разработку аналитической машины. У аналитической машины было 4 компонента: запоминающее устройство (память), вычислительное устройство, устройство ввода (для считывания перфокарт), устройство вывода (перфоратор и печатающее устройство). Память состояла из 1000 слов по 50 десятичных разрядов; каждое из слов содержало переменные и результаты. Вычислительное устройство принимало операнды из памяти, затем выполняло операции сложения, вычитания, умножения или деления и возвращало полученный результат обратно в память. Как и разностная машина, это устройство было механическим.

В конце 30-х годов немец Конрад Зус (Konrad Zuse) сконструировал несколько автоматических счетных машин с использованием электромагнитных реле. Ему не удалось получить денежные средства от правительства на свои разработки, потому что началась война. Зус ничего не знал о работе Бэббиджа, его машины были уничтожены во время бомбежки Берлина в 1944 году, поэтому его работа никак не повлияла на будущее развитие компьютерной техники. Однако он был одним из пионеров в этой области.

Немного позже счетные машины были сконструированы в Америке. Машина Джона Атанасова (John Atanasoff) была чрезвычайно развитой для того времени. В ней использовалась бинарная арифметика и информационные емкости, которые периодически обновлялись, чтобы избежать уничтожения данных. Современная динамическая память (ОЗУ) работает по точно такому же принципу. К несчастью, эта машина так и не стала действующей. В каком-то смысле Атанасов был похож на Бэббиджа — мечтатель, которого не устраивали технологии своего времени.

Компьютер Джорджа Стибитса (George Stibbitz) действительно работал, хотя и был примитивнее, чем машина Атанасова. Стибитс продемонстрировал свою машину на конференции в Дартмутском колледже в 1940 году. На этой конференции присутствовал Джон Моушли (John Mauchley), ничем не примечательный на тот момент профессор физики из университета Пенсильвании. Позднее он стал очень известным в области компьютерных разработок.

Пока Зус, Стибитс и Атанасов разрабатывали автоматические счетные машины, молодой Говард Айкен (Howard Aiken) в Гарварде упорно проектировал ручные счетные машины в рамках докторской диссертации. После окончания исследования Айкен осознал важность автоматических вычислений. Он пошел в библиотеку, прочитал о работе Бэббиджа и решил создать из реле такой же компьютер, который Бэббиджу не удалось создать из зубчатых колес. Работа над первым компьютером Айкена «Mark I» была закончена в 1944 году. Компьютер имел 72 слова по 23 десятичных разряда каждое и мог выполнить любую команду за 6 секунд. В устройствах ввода-вывода использовалась перфолента. К тому времени, как Айкен закончил работу над компьютером «Mark II», релейные компьютеры уже устарели. Началась эра электроники.

В начале войны англичанам удалось приобрести ENIGMA у поляков, которые, в свою очередь, украли ее у немцев. Однако, чтобы расшифровать закодированное послание, требовалось огромное количество вычислений, и их нужно было произвести сразу после перехвата радиограммы. Поэтому британское правительство основало секретную лабораторию для создания электронного компьютера под названием COLOSSUS. В создании этой машины принимал участие знаменитый британский математик Алан Тьюринг. COLOSSUS работал уже в 1943 году, но, так как британское правительство полностью контролировало этот проект и рассматривало его как военную тайну на протяжении 30 лет, COLOSSUS нестал базой для дальнейшего развития компьютеров. Мы упомянули о нем только потому, что это был первый в мире электронный цифровой компьютер.

Джон Моушли, который был знаком с работами Атанасова и Стибблитса, понимал, что армия заинтересована в счетных машинах. Он потребовал от армии финансирования работ по созданию электронного компьютера. Требование было удовлетворено в 1943 году, и Моушли со своим студентом Дж. Преспером Экертом (J. Presper Eckert) начали конструировать электронный компьютер, который они назвали ENIAC (Electronic Numerical Integrator and Computer — электронный цифровой интегратор и калькулятор). ENIAC состоял из 18 000 электровакуумных ламп и 1500 реле, весил 30 тонн и потреблял 140 киловатт электроэнергии. У машины было 20 регистров, каждый из которых мог содержать 10-разрядное десятичное число. (Десятичный регистр — это память очень маленького объема, которая может вмещать число до какого-либо определенного максимального количества разрядов, что-то вроде одометра, запоминающего километраж пройденного автомобилем пути.) В ENIAC было установлено 6000 многоканальных переключателей и имелось множество кабелей, протянутых к разъемам.

Транзистор был изобретен сотрудниками лаборатории Bell Laboratories Джоном Бардином Oohn Bardeen), Уолтером Браттейном (Walter Brattain) и Уильямом Шокли (William Shockley), за что в 1956 году они получили Нобелевскую премию в области физики. В течение десяти лет транзисторы совершили революцию в производстве компьютеров, и к концу 50-х годов компьютеры на вакуумных лампах уже безнадежно устарели. Первый компьютер на транзисторах был построен в лаборатории МТИ. Он содержал слова из 16 бит, как и Whirlwind I. Компьютер назывался TX-0 (Transistorized experimental computer 0 — экспериментальная транзисторная вычислительная машина 0) и предназначался только для тестирования будущей машины TX-2.

Машина TX-2 не имела большого значения, но один из инженеров этой лаборатории, Кеннет Ольсен (Kenneth Olsen), в 1957 году основал компанию DEC (Digital Equipment Corporation — корпорация по производству цифровой аппаратуры), чтобы производить серийную машину, сходную с TX-0. Эта машина, PDP-1, появилась только через четыре года главным образом потому, что те, кто финансировал DEC, считали производство компьютеров невыгодным. Поэтому компания DEC продавала в основном небольшие электронные платы. Компьютер PDP-1 появился только в 1961 году. Он имел 4096 слов по 18 бит и быстродействие 200 000 команд в секунду. Этот параметр был в два раза меньше, чем у 7090, транзисторного аналога 709. PDP-1 был самым быстрым компьютером в мире в то время. PDP-1 стоил 120 000 долларов, в то время как 7090 стоил миллионы. Компания DEC продала десятки компьютеров PDP-1, и так появилась компьютерная промышленность.

Одну из первых машин модели PDP-1 отдали в МТИ, где она сразу привлекла внимание некоторых молодых исследователей, подающих большие надежды. Одним из нововведений PDP-1 был дисплей размером 512 x 512 пикселов, на котором можно было рисовать точки. Вскоре студенты МТИ составили специальную программу для PDP-1, чтобы играть в «Войну миров» — первую в мире компьютерную игру.

Через несколько лет компания DEC разработала модель PDP-8, 12-разрядный компьютер. PDP-8 стоил гораздо дешевле, чем PDP-1 A6 000 долларов). Главное нововведение — единственная шина (omnibus). Шина — это набор параллельно соединенных проводов для связи компонентов компьютера. Это нововведение радикально отличало PDP-8 от IAS. Такая структура с тех пор стала использоваться во всех компьютерах. Компания DEC продала 50 000 компьютеров модели PDP-8 и стала лидером на рынке мини-компьютеров.

Изобретение в 1958 году Робертом Нойсом (Robert Noyce) кремниевой интегральной схемы означало возможность размещения на одной небольшой микросхеме десятков транзисторов. Компьютеры на интегральных схемах были меньшего размера, работали быстрее и стоили дешевле, чем их предшественники на транзисторах.

К 1964 году компания IBM лидировала на компьютерном рынке, но существовала одна большая проблема: компьютеры 7094 и 1401, которые она выпускала, были несовместимы друг с другом. Один из них предназначался для сложных расчетов, в нем использовалась двоичная арифметика на регистрах по 36 бит, во втором применялась десятичная система счисления и слова разной длины. У многих покупателей были оба этих компьютера, и им не нравилось, что они совершенно несовместимы.

Когда пришло время заменить эти две серии компьютеров, компания IBM сделала решительный шаг. Она выпустила линейку транзисторных компьютеров System/360, которые были предназначены как для научных, так и для коммерческих расчетов. Линейка System/360 имела много нововведений. Это было целое семейство компьютеров для работы с одним языком (ассемблером).

Появление сверхбольших интегральных схем (СБИС) в 80-х годах позволило

помещать на одну плату сначала десятки тысяч, затем сотни тысяч и, наконец, миллионы транзисторов. Это привело к созданию компьютеров меньшего размера и более быстродействующих. До появления PDP-1 компьютеры были настолько велики и дороги, что компаниям и университетам приходилось иметь специальные отделы (вычислительные центры). К 80-м годам цены упали так сильно, что возможность приобретать компьютеры появилась не только у организаций, но и у отдельных людей. Началась эра персональных компьютеров.

Персональные компьютеры требовались совсем для других целей, чем их предшественники. Они применялись для обработки слов, электронных таблиц, а также для выполнения приложений с высоким уровнем интерактивности (например, игр), с которыми большие компьютеры не справлялись.

Первые персональные компьютеры продавались в виде комплектов. Каждый комплект содержал печатную плату, набор интегральных схем, обычно включающий схему Intel 8080, несколько кабелей, источник питания и иногда 8-дюймовый дисковод. Сложить из этих частей компьютер покупатель должен был сам. Программное обеспечение к компьютеру не прилагалось. Покупателю приходилось писать программное обеспечение

самому. Позднее появилась операционная система CP/M, написанная Гари Килдаллом (Gary Kildall) для Intel 8080. Эта действующая операционная система помещалась на дискету, она включала в себя систему управления файлами и интерпретатор для выполнения пользовательских команд, которые набирались с клавиатуры.

Итак, к первому поколению причисляются компьютеры на электронных лампах (такие, как ENIAC), ко второму — транзисторные машины (IBM 7094), к третьему — первые компьютеры на интегральных схемах (IBM 360), к четвертому — персональные компьютеры (линейки ЦП Intel). Что же касается пятого поколения, то оно больше ассоциируется не с конкретной архитектурой, а со сменой парадигмы. Компьютеры будущего будут встраиваться во все мыслимые и немыслимые устройства и за счет этого действительно станут невидимыми. Они прочно войдут в повседневную жизнь — будут открывать двери, включать лампы, распределять деньги и выполнять тысячи других обязанностей. Эта модель, разработанная Марком Вайзером (Mark Weiser) в поздний период его деятельности, первоначально получила название повсеместной компьютеризации, но в настоящее время не менее распространен термин «всепроникающая компьютеризация». Это явление обещает изменить мир не менее радикально, чем промышленная революция.

Лекция 9. Типы компьютеров

План лекции

Технологические и экономические аспекты
Широкий спектр компьютеров
Одноразовые компьютеры
Микроконтроллеры
Игровые компьютеры
Персональные компьютеры
Серверы
Комплексы рабочих станций
Мэйнфреймы

Обзор лекции

В предыдущем разделе мы кратко изложили историю компьютерных систем. В этом разделе мы расскажем о положении дел в настоящий момент и сделаем некоторые предположения на будущее. Хотя персональные компьютеры — наиболее известные типы «умных» машин, в наши дни существуют и другие типы машин, поэтому стоит кратко рассказать о них.

Компьютерная промышленность двигается вперед как никакая другая. Главная движущая сила — способность производителей помещать с каждым годом все больше и больше транзисторов на микросхему. Чем больше транзисторов (крошечных электронных переключателей), тем больше объем памяти и мощнее процессоры. Закон Мура гласит, что количество транзисторов на одной микросхеме удваивается каждые 18 месяцев, то есть увеличивается на 60 % каждый год. Размеры микросхем и даты их производства подтверждают, что закон Мура действует до сих пор.

Развивать компьютерные технологии исходя из закона Мура можно двумя путями: создавать компьютеры все большей и большей мощности при постоянной цене или выпускать одну и ту же модель с каждым годом за меньшие деньги. Компьютерная промышленность идет по обоим этим путям, создавая широкий спектр разнообразных компьютеров.

В самой верхней строчке табл. 1.3 находятся микросхемы, которые приклеиваются на внутреннюю сторону поздравительных открыток для проигрывания мелодий типа «Happy Birthday», свадебного марша или чего-нибудь подобного. Тот, кто воспитывался на компьютерах стоимостью в миллионы долларов, воспринимает такие доступные всем компьютеры примерно так же, как доступный всем самолет.

Как бы то ни было, одноразовые компьютеры окружают нас. Вероятно, наиболее значимым достижением в этой области стало появление микросхем RFID (Radio Frequency Identification — радиочастотная идентификация). Теперь на безбатарейных микросхемах этого типа толщиной меньше 0,5 мм и себестоимстью в несколько центов устанавливаются крошечные приемопередатчики радиосигналов; кроме того, им присваивается уникальный 128-разрядный идентификатор. При получении импульса с внешней антенны они за счет достаточно длинного радиосигнала отправляют ответный импульс со своим номером. В отличие от размера микросхем, спектр их практического применения весьма значителен.

Вторая категория в таблице отведена под компьютеры, которыми оснащаются разного рода бытовые устройства. Такого рода встроенные компьютеры, называемые также микроконтроллерами, выполняют функцию управления устройствами и организации их пользовательских интерфейсов. Диапазон устройств, работающих с помощью микрокомпьютеров, крайне широк.

Следующая категория — игровые компьютеры. Это, по существу, обычные компьютеры, в которых расширенные возможности графических и звуковых контроллеров сочетаются с ограничениями по объему ПО и пониженной расширяемостью. Первоначально в эту категорию входили

компьютеры с процессорами низших моделей для простых игр типа пинг-понга, которые предусматривали вывод изображения на экран телевизора. С годами игровые компьютеры превратились в достаточно мощные системы, которые по некоторым параметрам производительности ничем не хуже, а иногда даже лучше персональных компьютеров.

Чтобы получить представление о том, чем комплектуются игровые компьютеры, рассмотрим конфигурации трех популярных моделей этой категории. Первая из них — Sony PlayStation 2. В ней установлен 128-разрядный специализированный RISC-процессор с тактовой частотой 296 МГц (он называется Emotion Engine) на базе архитектуры MIPS IV. Кроме того, PlayStation 2 оснащается модулем памяти емкостью 32 Мбайт, специальной графической микросхемой на 160 МГц, 48-канальной звуковой платой и DVD-приводом. Вторая рассматриваемая модель — Microsoft XBOX. В ней устанавливается процессор Intel Pentium III на 733 МГц, 64 Мбайт памяти, графическая микросхема 300 МГц, 256-канальная звуковая микросхема, DVD-привод и жесткий диск емкостью 8 Гбайт. Наконец, третья модель называется Nintendo GameCube. Она комплектуется 32-разрядным процессором Gekko с тактовой частотой 485 МГц (он представляет собой модификацию RISC-процессора IBM PowerPC), памятью объемом 24 Мбайт, графической микросхемой на 200 МГц, 64-канальной звуковой микросхемой и фирменным оптическим диском емкостью 1,5 Гбайт.

В следующую категорию входят персональные компьютеры. Именно они ассоциируются у большинства людей со словом «компьютер». Персональные компьютеры бывают двух видов: настольные и портативные (ноутбуки). Как правило, те и другие комплектуются модулями памяти общей емкостью в сотни мегабайтов, жестким диском с данными на несколько десятков гигабайтов, приводом CD-ROM/DVD, модемом, звуковой картой, сетевым интерфейсом, монитором с высоким разрешением и рядом других периферийных устройств. На них устанавливаются сложные операционные системы, они расширяемы, при работе с ними используется широкий спектр программного обеспечения. Некоторые специалисты называют «персональными» компьютеры с процессорами Intel, отделяя их тем самым от машин, оснащенных высокопроизводительными RISC-микросхемами (такими как Sun UltraSPARC), которые в таком случае именуются «рабочими станциями». На самом деле, особой разницы между этими двумя типами нет.

Мощные персональные компьютеры и рабочие станции часто используются в качестве сетевых серверов — как в локальных сетях (обычно в пределах одной организации), так и в Интернете. Серверы, как правило, поставляются в однопроцессорной и мультипроцессорной конфигурациях. В

системах из этой категории обычно устанавливаются модули памяти общим объемом в несколько гигабайтов, жесткие диски емкостью в сотни гигабайтов и высокоскоростные сетевые интерфейсы. Некоторые серверы способны обрабатывать тысячи транзакций в секунду.

С точки зрения архитектуры однопроцессорный сервер не слишком отличается от персонального компьютера. Он просто работает быстрее, занимает больше места, содержит больше дискового пространства и устанавливает более скоростные сетевые соединения. Серверы работают под управлением тех же операционных систем, что и персональные компьютеры, — как правило, это различные версии UNIX и Windows.

В связи с тем, что по соотношению «цена/производительность» позиции рабочих станций и персональных компьютеров постоянно улучшаются, в последние годы появилась практика их объединения в рамках кластеров рабочих станций (Clusters Of Workstations, COW), которые иногда называют просто «кластерами». Они состоят из нескольких персональных компьютеров или рабочих станций, подключенных друг к другу по высокоскоростной сети и снабженных специальным программным обеспечением, которое позволяет направлять их ресурсы на решение единых задач (как правило, научных и инженерных).

Наконец мы дошли до больших компьютеров размером с комнату, напоминающих компьютеры 60-х годов и традиционно называемых мэйнфреймами. В большинстве случаев эти системы — прямые потомки больших компьютеров серии 360. Обычно они работают не намного быстрее, чем мощные серверы, но у них выше скорость процессов ввода-вывода и обладают они довольно большим пространством на диске — 1 Тбайт и более 1 терабайт = 10^{12} байт). Такие системы стоят очень дорого и требуют крупных вложений в программное обеспечение, данные и персонал, обслуживающий эти компьютеры. Многие компании считают, что дешевле заплатить несколько миллионов долларов один раз за такую систему, чем даже думать о необходимости заново программировать все прикладные программы для маленьких компьютеров.

До последнего времени существовала еще одна крупная категория вычислительных машин — суперкомпьютеры. Их процессоры работали с очень высокой скоростью, в них устанавливались модули памяти общей емкостью в несколько десятков гигабайтов, высокоскоростные диски и сетевые интерфейсы. Суперкомпьютеры используются для решения различных научных и технических задач, которые требуют сложных вычислений, например таких, как моделирование сталкивающихся галактик, синтез новых лекарственных препаратов, моделирование потока воздуха

вокруг крыла аэроплана. Сейчас, когда вычислительные возможности, аналогичные тем, что предлагают суперкомпьютеры, реализуются в виде кластеров, эта категория компьютеров постепенно отмирает.

Лекция 10. Семейства компьютеров

План лекции

Pentium 4

Знакомство с микросхемой UltraSPARC III

Знакомство с микросхемой 8051

Обзор лекции

Персональные компьютеры представляют немалый интерес хотя бы по той причине, что все читатели ими, несомненно, пользуются. На серверных системах выполняются все службы в Интернете. Наконец, встроенные компьютеры, хотя и незаметны для пользователей, контролируют работу многих агрегатов в автомобилях, телевизорах, микроволновых печах, стиральных машинах, да и вообще практически во всех мыслимых электронных устройствах стоимостью выше 50 долларов.

В этом разделе мы вкратце рассмотрим принципы действия трех моделей компьютеров, которые далее по ходу изложения материала будем привлекать в качестве примеров. Это — Pentium 4, UltraSPARC III и 8051.

Рассматриваются характеристики линейки процессоров Intel на примере процессоров 4004, 8008, 8080, 8086, 8088, 80286, 80386, 80486, Pentium, Pentium Pro, Pentium II, Pentium III, Pentium 4.

Первый компьютер SPARC был 32-разрядным и работал на частоте 36 МГц. Центральный процессор назывался IU (Integer Unit — блок целочисленной арифметики) и был весьма посредственным. У него имелось только три основных формата команд и в общей сложности всего 55 команд. С появлением процессора с плавающей точкой добавилось еще 14 команд. Отметим, что компания Intel начала с 8- и 16-разрядных микросхем (модели 8088, 8086, 80286), а уже потом перешла на 32-разрядные (модель 80386), а компания Sun, в отличие от Intel, сразу начала с 32-разрядных.

Грандиозный перелом в развитии SPARC произошел в 1995 году, когда была разработана 64-разрядная версия (версия 9) с адресами и регистрами по 64 бит. Первой рабочей станцией с такой архитектурой стал процессор UltraSPARC I, вышедший в свет в 1995 году. Он был полностью совместим с 32-разрядными версиями SPARC, хотя сам был 64-разрядным.

В то время как предыдущие машины работали с символьными и числовыми данными и были приспособлены для выполнения программ уровня текстовых процессоров и редакторов электронных таблиц, UltraSPARC с самого начала был предназначен для работы с изображениями, аудио, видео и вообще мультимедиа.

Среди нововведений, помимо 64-разрядной архитектуры, появились 23 новые команды, в том числе команды для упаковки и распаковки пикселов из 64-разрядных слов, масштабирования и вращения изображений, перемещения блоков, а также для компрессии и декомпрессии видео в реальном времени. Эти команды назывались VIS (Visual Instruction Set — набор команд для работы с визуальными данными) и предназначались для поддержки мультимедиа. Они были аналогичны MMX-командам. Процессор UltraSPARC предназначался для веб-серверов с десятками процессоров и физической памятью до 2 Тбайт 1 терабайт = 10^{12} байт). Тем не менее некоторые версии UltraSPARC могут использоваться и в ноутбуках.

За UltraSPARC I последовали UltraSPARC II, UltraSPARC III и UltraSPARC IV. Эти модели отличались друг от друга по скорости, и у каждой из них появлялись какие-то новые особенности.

Третий пример разительно отличается и от первого (Pentium 4 для персональных компьютеров), и от второго (UltraSPARC III для серверов). Микросхема 8051 применяется во встроенных системах. Ее история началась в 1976 году — к тому моменту уже в течение двух лет на рынке доминировала 8-разрядная модель 8080. Производители бытовых устройств к тому времени практиковали разработку приборов под управлением 8080, однако для этого требовался процессор 8080, один или несколько модулей памяти и, опять же, одна или несколько микросхем ввода-вывода. Совокупная стоимость трех (как минимум) микросхем и затраты, связанные с их соединением, были весьма велики, в связи с чем эти решения применялись только в сложных и дорогих устройствах. Таким образом, со стороны производителей бытовых приборов сформировался спрос на микросхемы, совмещающие блоки процессора памяти и ввода-вывода, который компания Intel не замедлила удовлетворить.

В результате появилась модель 8748 — микроконтроллер на основе 17 000 транзисторов, состоящий из процессора наподобие 8080, постоянной памяти емкостью 1 Кбайт для размещения программы, оперативной памяти на 64 байт для размещения переменных, 8-разрядного таймера и 27 шин ввода-вывода, управляющих переключателями, кнопками и световыми индикаторами. Несмотря на вопиющую простоту выполнения, микросхема пользовалась коммерческим успехом, что побудило Intel выпустить в 1980

году новую модель — 8051. В ней были предусмотрены 60 000 транзисторов, значительно более быстрый, чем в 8748, процессор, 4 Кбайт постоянной и 128 байт оперативной памяти, 32 шины ввода-вывода, последовательный порт и два 16-разрядных таймера. Вскоре вышли и другие модификации микросхемы, сформировавшие семейство микроконтроллеров MSC-51.

Лекция 11. Архитектура IA-64

План лекции

Проблема Pentium 4
Вычисления с явным параллелизмом команд
Сокращение числа обращений к памяти
Планирование команд
Предикация
Спекулятивная загрузка

Обзор лекции

IA-32 — это старая архитектура команд с совершенно неподходящими для современной техники свойствами. Это типичная CISC-архитектура с командами разной длины и огромным количеством различных форматов, которые трудно декодировать быстро и «на лету». Современная техника лучше всего работает с RISC-архитектурами, в которых команды одинаковы по размеру, а код операции имеет фиксированную длину, поэтому его легко декодировать. Хотя команды архитектуры IA-32 во время выполнения программы можно разделить на микрооперации наподобие RISC-команд, но для этого требуется дополнительное аппаратное обеспечение (пространство на микросхеме), к тому же это занимает время и усложняет разработку. Это первый недостаток.

IA-32 — это архитектура, ориентированная на двухадресные команды. В настоящее время популярны архитектуры команд типа загрузки/сохранения, в которых обращения к памяти выполняются только для того, чтобы поместить операнды в регистры, а все вычисления делаются с использованием трехадресных регистровых команд. Поскольку скорость работы процессора растет гораздо быстрее, чем памяти, положение дел с IA-32 со временем все больше ухудшается. Это второй недостаток.

Архитектура IA-32 содержит небольшой и нерегулярный набор регистров. Из-за малого числа регистров общего назначения (четыре или шесть, в зависимости от того, куда отнести регистры ESI и EDI) постоянно приходится записывать в память промежуточные результаты, что ведет к

дополнительным обращениям к памяти, даже когда они по логике вещей не нужны. Это третий недостаток.

Из-за недостаточного числа регистров возникает множество ситуаций зависимостей, особенно WAR-зависимостей, поскольку промежуточные результаты нужно куда-то помещать, а дополнительных регистров нет. Из-за недостатка регистров постоянно требуется их подмена (то есть использование скрытых регистров). Во избежание слишком частых кэш-промахов команды приходится выполнять не по порядку. Однако семантика архитектуры IA-32 определяет точные прерывания, поэтому команды, выполняемые не по порядку, должны записывать результаты в выходные регистры в строгом порядке. Все это очень сложно реализовать аппаратно. Это четвертый недостаток.

Чтобы скорость работы была высокой, система должна быть в значительной степени конвейеризирована. Однако это значит, что для выполнения любой команды требуется множество циклов. Следовательно, становится существенным точное предсказание переходов, поскольку в конвейер должны попадать только нужные команды. Однако даже если процент неправильных прогнозов невысок, существенно снижается производительность. Это пятый недостаток.

Чтобы избежать проблем с неправильным прогнозированием переходов, процессору приходится осуществлять спекулятивное выполнение команд со всеми вытекающими отсюда последствиями. Это шестой недостаток.

Основной принцип организации архитектуры IA-64 сводится к тому, чтобы перенести нагрузку с периода выполнения в период компиляции. Процессор Pentium 4 в ходе выполнения переупорядочивает команды, подменяет регистры, распределяет функциональные блоки и выполняет множество других функций, что ведет к максимальной загрузке всех аппаратных ресурсов. В модели IA-64 эти задачи заранее решает компилятор. В результате он генерирует программу, которую можно выполнять без излишних манипуляций аппаратными средствами. К примеру, в Pentium 4 компилятор получает информацию о том, что в машине всего 8 регистров, хотя на самом деле их 128, в результате во время выполнения программы приходится как-то выкручиваться, чтобы избежать взаимозависимостей. Согласно архитектуре IA-64, компилятор получает достоверную информацию о количестве регистров в машине, а затем генерирует программу, в которой нет никаких конфликтов между регистрами. Кроме того, компилятор следит за загрузкой функциональных блоков и не запускает команды, в которых предполагается обращение к занятым функциональным блокам. Модель, в которой аппаратный параллелизм является видимым для компилятора,

называется EPIC (Explicitly Parallel Instruction Computing — вычисления с явным параллелизмом команд). В определенной степени модель EPIC можно считать развитием RISC-технологии.

Лучший способ ускорить обращения к памяти — выполнять эту операцию в фоновом режиме. В процессоре Itanium 2 предусмотрено 128 64-разрядных регистров общего назначения. Первые 32 из них являются статическими, а оставшиеся 96 группируются в стек регистров, напоминающий регистровое окно UltraSPARC III. В отличие от UltraSPARC, количество доступных программе регистров меняется от одной процедуры к другой. В итоге каждая процедура получает доступ к 32 статическим регистрам и некоторому (переменному) количеству регистров, распределяемых динамически.

При вызове процедуры указатель стека регистров смещается таким образом, чтобы входные параметры оказались видимыми в регистрах, но сами регистры не были распределены между локальными переменными. Процедура сама определяет количество необходимых ей регистров и соответствующим образом перемещает указатель стека. Сохранять содержимое этих регистров при входе и восстанавливать при выходе не требуется, хотя, если процедуре нужно изменить статический регистр, она должна сначала явно сохранить его прежнее значение, а впоследствии восстановить его. Поскольку количество регистров выражено доступной переменной и обуславливается требованиями каждой конкретной процедуры, неэффективное применение регистров исключается. Кроме того, увеличивается максимальная глубина вызова процедур, при которой регистрыне требуется «сбрасывать» в память.

Один из наиболее серьезных недостатков Pentium 4 — сложность планирования команд для обработки в различных функциональных блоках без взаимозависимостей. Для решения этой проблемы в период выполнения задействуются очень сложные механизмы, аппаратная поддержка которых требует много места на микросхеме. В архитектуре IA-64 и ее реализации — Itanium 2 — эти проблемы решены путем передачи соответствующих функций компилятору. Каждая программа теперь состоит из последовательности групп команд. Команды в рамках одной группы не конфликтуют друг с другом, не потребляют больше ресурсов и не обращаются к большему количеству функциональных блоков, чем предусмотрено системой, не образуют RAW- и WAW-взаимозависимости (WAR-взаимозависимости допускаются в ограниченном объеме). Создается впечатление, что последовательные группы команд выполняются строго по

порядку, хотя на самом деле, если это безопасно, процессор может запустить часть команд следующей группы до завершения предыдущей.

Таким образом, процессор может планировать выполнение команд внутри каждой отдельно взятой группы в любом порядке, по возможности, — в параллельном режиме. При этом вероятность возникновения конфликтов между командами равна нулю. Если группа команд нарушает вышеуказанные правила, поведение программы становится неопределенным. В такой ситуации обязанность по обеспечению соответствия полученного из исходной программы кода на ассемблере всем требованиям ложится на компилятор. Для того чтобы быстро провести компиляцию одновременно с отладкой программы, компилятор может поместить каждую команду в отдельную группу; сделать это просто, но производительность в результате этой операции снижается, а время оптимизации выходного кода значительно увеличивается.

Еще одна особенность архитектуры IA-64 — новый способ обработки условных переходов. Если бы была возможность избавиться от большинства из них, центральный процессор стал бы гораздо проще и работал бы гораздо быстрее. На первый взгляд может показаться, что устранить условные переходы невозможно, поскольку в программах всегда полно операторов if. Однако в архитектуре IA-64 используется специальная технология, названная предикацией (predication), которая позволяет сильно сократить их число [14, 98].

В нынешних машинах все команды являются безусловными в том смысле, что когда центральный процессор встречает команду, он просто ее выполняет. Здесь никогда не решается вопрос: «Выполнять или не выполнять?» И напротив, в предикатной архитектуре команды содержат условия, которые сообщают, в каком случае нужно выполнять команду, а в каком — нет. Именно этот переход от безусловных команд к предикатным позволяет избавиться от многих условных переходов. Вместо того чтобы выбирать ту или иную последовательность безусловных команд, все команды сливаются в одну последовательность предикатных команд, в которой у разных команд разные предикаты.

Еще одна особенность IA-64, повышающая быстродействие, — поддержка спекулятивной загрузки. Если команда LOAD спекулятивна и не срабатывает, то вместо того, чтобы вызвать исключение, она просто прекращает выполняться и сообщает, что регистр, в который она должна была загрузить значение, недействителен. Для этого используется тот самый бит отравления. Исключение будет вызвано только в том случае, если затем попытаться использовать этот регистр.

Обычно при спекулятивной загрузке компилятор помещает команды LOAD перед другими командами. Поскольку выполнение этих команд начинается раньше, чем нужно, они могут завершиться еще до того, как потребуются результаты. В том месте, где ему нужно получить значение определенного регистра, компилятор вставляет команду CHECK. Если значение там уже есть, команда CHECK работает так же, как NOP, и выполнение программы просто сразу продолжается дальше. Если значения в регистре еще нет, следующая команда вынуждена простаивать.

Суммируя, можно сказать, что в машинах с архитектурой IA-64 реализовано несколько механизмов повышения быстродействия. Во-первых, это современная конвейеризированная трехадресная RISC-машина, поддерживающая механизм загрузки/сохранения. Во-вторых, компилятор определяет, какие команды могут выполняться одновременно, и, не вступая в конфликт, группирует эти команды в пучки. Таким образом, процессор может просто планировать обработку пучков, не думая ни о каких проверках. В-третьих, предикация позволяет объединить команды обоих переходов в операторе if, устраняя при этом как условный переход, так и необходимость прогнозирования этого перехода. Наконец, спекулятивная загрузка позволяет вызывать операнды заранее, и даже если позднее окажется, что эти операнды не нужны, ничего страшного не произойдет.

Лекция 12. Вычислительная инфраструктура СГАУ

План лекции

Кластер «Сергей Королев»
Кластер HP
КС-ЭВМ
Виртуализация
Системы хранения данных
Программное обеспечение

Обзор лекции

Суперкомпьютер «Сергей Королев», создан в рамках программы развития национального исследовательского университета при поддержке правительства Самарской области по мероприятию «Развитие среды генерации знаний на базе межвузовского медиацентра, путем создания суперкомпьютерного центра, ориентированного, в том числе, на исследования в сфере нанотехнологий, и наращивания телекоммуникационной инфраструктуры», а также по программе

«Академические инициативы» компании IBM. Кластер установлен в суперкомпьютерном центре СГАУ. Система построена на базе линейки оборудования IBM BladeCenter с использованием блейд-серверов HS22 и обеспечивает пиковую производительность 15 триллионов операций с плавающей точкой в секунду.

В рамках программы «Университетский кластер» компания Hewlett Packard поставила в СГАУ кластерную систему в конструктиве HP BLc3000 Twr CTO Enclosure. Система построена на базе управляющего сервера HP ProLiant BL260c и семи вычислительных блейд серверов HP ProLiant 2xBL220c. Пиковая производительность кластера около 1.5 ТФлопс.

Кластерная система «Университетский кластер» построена на базе операционной системы Linux и программного обеспечения промежуточного слоя OSCAR. На базе поставляемых кластерных систем, в нескольких университетах России построен вычислительный Грид-сегмент с использованием технологии Globus Toolkit 4.2. Все программное обеспечение является свободно распространяемым и может быть использовано на правах лицензионных соглашений: GNU GPL License, MIT License и др.

Кластер используется для решения задач моделирования и проектирования наноструктур, нелинейного оптического моделирования, распределенной обработки и хранения файлов изображений сверхвысокого разрешения.

В рамках президентской программы «Развитие суперкомпьютеров и ГРИД-технологий» в 2010 г. федеральный ядерный центр в Сарове (РФЯЦ-ВНИИЭФ), входящий в госкорпорацию «Росатом», разработал две модели персональных суперкомпьютеров – универсального и специального назначения, а также программные пакеты для математического моделирования. В 2010 году в СГАУ была поставлена универсальная модель КС-ЭВМ 1.

Универсальный суперкомпьютер предназначен для широкого спектра инженерных задач. Его пиковая производительность составляет 1.1 Тфлопс на арифметических операциях двойной точности. Суперкомпьютер построен на базе двенадцатиядерных процессоров AMD и включает три четырехсокетные материнских платы. Таким образом, система содержит 144 процессорных ядра. На каждой материнской плате установлено по 128 Гб оперативной памяти, всего общей памяти 384 Гб. В качестве высокоскоростной коммуникационной среды для передачи межпроцессных сообщений используется безкоммутаторное соединение InfiniBand адаптеров.

В универсальном суперкомпьютере используется смешанный тип охлаждения – тепло от процессоров отводится трубками с водой на радиатор,

расположенный на задней стенке корпуса, и рассеивается с помощью вентиляторов. Работает система под управлением ОС Linux.

Технология виртуализации заняла прочное место в ИТ инфраструктуре различных коммерческих и государственных организаций. В СГАУ данная технология используется довольно давно в основном для упрощения администрирования различных сетевых сервисов (DNS, электронная почта, управление лицензиями на программное обеспечение и т.д.).

На данный момент виртуализации все больше и больше внедряется в различные задачи, решаемые управлением ИТ СГАУ. Это и поддержка функционирования компьютерной инфраструктуры межвузовского медиацентра, корпоративный веб-хостинг, организация видеотрансляций в сети Интернет и др.

Технология реализуется на современном оборудовании IBM, установленном в суперкомпьютерном центре и программном обеспечении VMware vSphere.

Надежность всей виртуальной инфраструктуры обеспечивается за счет многоуровневого резервирования всех основных компонентов среды, от источников бесперебойного электропитания, кондиционирования и серверного оборудования до использования максимальных возможностей программного обеспечения Vmware vSphere, таких как High Availability и Fault Tolerance.

Системы хранения данных (СХД) СКЦ СГАУ состоят из серверов, дисковых хранилищ и сетевой инфраструктуры для высокопроизводительных систем. Системы хранения СКЦ СГАУ предоставляют сервис общей домашней директории на всех вычислительных системах центра. Каждый пользователь имеет квоту в 10Гб данных и 100000 файлов в своей домашней директории. Для пользователей, которым требуется более 10Гб и/или 100000 файлов, возможно выделение дополнительного объема дискового пространства.

Суперкомпьютерный центр (СКЦ) СГАУ имеет широкий набор программного обеспечения.

ANSYS - многоцелевой конечно-элементный пакет для проведения анализа в широком круге инженерных дисциплин (прочность, теплофизика, динамика жидкостей и газов).

LSTC LS-DYNA - многоцелевой конечно-элементный комплекс разработки Livermore Software Technology Corp. для анализа высоконелинейных и быстротекущих процессов в задачах механики твердого и жидкого тела.

NUMECA FINE™/Turbo - специализированная CFD-среда для расчета и анализа течений в элементах лопаточных машин. Применяется для решения практических задач гидрогазодинамики при проектировании и оптимизации всех типов лопаточных машин (турбомашин): многоступенчатых осевых, радиальных и смешанных компрессоров, турбин, насосов, вентиляторов и пропеллеров, а также улиток, диффузоров, теплообменников и выхлопных систем.

OpenMPI - открытая библиотека реализации стандарта MPI 2, разработанная и поддерживаемая академическим сообществом.

CUDA - архитектура параллельных вычислений от NVIDIA, позволяющая существенно увеличить вычислительную производительность благодаря использованию GPU (графических процессоров).

TORQUE - система пакетной обработки заданий. Семейство компиляторов GCC.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Учебное пособие по дисциплине «Вычислительные системы»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Основная литература по дисциплине

1. Таненбаум, Э. Архитектура компьютера [Текст] / Э. Таненбаум, Т. Остин - СПб. [и др.] : Питер , 2013. - 816 с. - (Классика computer science). – ISBN 978-5-496-00337-7 (0 экз.)

2. Цилькер, Б.Я. Организация ЭВМ и систем [Текст] : [учеб. для вузов по направлению "Информатика и вычисл. техника"] / Б.Я. Цилькер, С.А. Орлов. - СПб. и др. : Питер : Питер принт, 2006. - 667 с. - (Учебник для вузов). – ISBN 5-94723-759-8 (1 экз.)

3. Цилькер, Б.Я. Организация ЭВМ и систем [Текст] : [учеб. для вузов по направлению "Информатика и вычисл. техника"] / Б.Я. Цилькер, С.А. Орлов. - СПб. и др. : Питер : Питер принт, 2004. - 667 с. - (Учебник для вузов). – ISBN 5-94723-759-8 (112 экз.)

Дополнительная литература по дисциплине

1. Востокин, С.В. Вопросы, задания и упражнения по курсу "Операционные системы" [Текст] : [лаб. практикум] / М-во образования и науки Рос. Федерации, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т) ; [сост. С. В. Востокин]. - Самара : Изд-во СГАУ, 2012. - 29 с. (20 экз.)

2. Востокин, С.В. Операционные системы [Электронный ресурс] : [учеб. для вузов по направления подгот. бакалавров "Инф. и выч. техн.", "Фундам. информатика и информ. технологии", "Прикладная математика и информатика", "Прикладная математика и физика"] /С.В. Востокин ; М-во образования и науки РФ, Самар. гос.аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т). - Электрон. текстовые дан. - Самара : Изд-во СГАУ, 2012. – ISBN 978-5-7883-0916-3 (2 эл. опт. диска)

3. Таненбаум, Э.Современные операционные системы [Текст] / Эндрю Таненбаум. - 2-е изд. - СПб. [и др.] : Питер : Питер принт, 2005. - 1037 с. - (Классика computer science). - ISBN 5-318-00299-4 (5 экз.)

4. Таненбаум, Э. Современные операционные системы [Текст] / Эндрю Таненбаум. - 2-е изд., [перераб. и испр.]. - СПб. [и др.] : Питер : Питер Пресс, 2007. - 1037 с. - (Классика computer science). - ISBN 978-5-318-00299-1 (2 экз.)

5. Солдатова, О. П. Системное программирование [Текст] : Курс лекций / О. П. Солдатова, С. В. Востокин ; Самар. гос. аэрокосм. ун-т им. С. П. Королева. - Самара : [б. и.], 2002. - 123 с. - ISBN 5-7883-0226-9 (94 экз.)

Электронные источники и интернет ресурсы

1. Microsoft Developer Network http://msdn.microsoft.com

2. www.prenhall.com/tanenbaum

3. Проект автоматизации параллельных и распределенных вычислений «Темплет» http://templet.ssau.ru

4. Интуит национальный открытый университет.http://www.intuit.ru/

Методические указания и рекомендации

Текущий контроль знаний студентов завершается на отчетном занятии, результатом которого является допуск или недопуск студента к экзамену по дисциплине. Основанием для допуска к экзамену является выполнение и отчет студента по всем лабораторным работам, а также прохождение теста на понимание терминологии по предмету. Тест включает 4 варианта по 12 вопросов.

Экзамен проводится согласно положению о текущем и промежуточном контроле знаний студентов, утвержденному ректором университета. Экзаменационная оценка ставится на основании письменного и устного ответов студента по экзаменационному билету, а также, при необходимости, ответов на дополнительные вопросы. Экзаменационный билет включает 3 вопроса. Дополнительно может быть предложен как теоретический вопрос, так и вопрос на понимание программ по листингам.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Задания на лабораторные работы по дисциплине «Вычислительные системы»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Лабораторная работа №1. Конфигурирование средств управления доступом на кластер СГАУ

Цель работы: настройка клиентов удаленного доступа PuTTY и WinSCP

Теоретические сведения:

Настройка доступа по открытому ключу в PuTTY. Запускаем программу PuTTYgen из набора программ PuTTY. Нажимаем "Generate" для генерации пары открытого/закрытого ключей. Сохраняем закрытый ключ без ввода пароля. Копируем из окна программы открытый ключ и сохраняем его в Блокноте. Текст ключа должен быть в одной строке без переносов (ssh-rsa ...текст ключа... rsa-key...). Входим на кластер через PuTTY с помощью логина/пароля. Для установки сгенерированного открытого ключа нужно отредактировать файл ~/.ssh/authorized_keys. В этом файле в первой строке записан открытый ключ, сгенерированный системой во время первого входа пользователя. Его не нужно удалять или заменять! Свой ключ, необходимо вставить после имеющегося ключа (в следующей строке).

[user@mgt1 ~]$ nano ~/.ssh/authorized_keys

Запускается текстовый редактор nano. Далее можно вставить текст просто нажав правую кнопку мыши в окне PuTTY. Скопированный текст из буфера вставится в окно. Важно чтобы тескт располагался в одной строке, если не так, поправить. F3 - сохранить файл. F2 - выйти из nano.

Далее, во вкладке "Data" вводим свое имя пользователя для автоматического входа. Во вкладке "Auth" указываем путь к закрытому ключу.

Настройка доступа по открытому ключу в PuTTY описана на сайте суперкомпьютерного центра СГАУ по адресу (http://hpc.ssau.ru/node/25).

Настройка соединения в WinSCP. Запускаем WinSCP. Создаем новый профиль соединения и сохраняем его. Заполняем поля Host name: sk.ssau.ru, User name: Ваше имя пользователя. При первом подключении к кластеру, появится окно предупреждения что открытый ключ сервера не найден в кэше

программы. Добавляем его, нажав ОК (Да). Подключившись, вводите пароль. Настройка соединения в WinSCP описана по адресу (http://hpc.ssau.ru/node/24).

Лабораторная работа №2. Настройка SSH доступа к удаленной вычислительной системе

Цель работы: знакомство с утилитами удаленного доступа к вычислительной системе по протоколу SSH, запуск заданий в системе Torque

Теоретические сведения:

Удаленный доступ на кластеры, возможен только с ip адресов корпоративной сети СГАУ и ip-адресов, которые были указаны в заявке на регистрацию пользователя. Сетевое имя головного узла суперкомпьютера «Сергей Королев» sk.ssau.ru. Для входа на кластер необходимо использовать программу эмулятор терминала с поддержкой протокола SSH версии 2, например PuTTY, SecureCRT. Настройка соединения в PuTTY описана по адресу (http://hpc.ssau.ru/node/23).

Удаленный доступ на кластеры по методу открытого и закрытого ключа. Классический метод доступа к UNIX системам с вводом имени пользователя и пароля не всегда бывает удобен. Пользователи, чтобы запомнить пароль, делают его простым, что ведет к проблемам в безопасности. Существует метод доступа по методу закрытого и открытого ключа, не требующий ввода пароля. Настройка доступа по открытому ключу в PuTTY описана на сайте суперкомпьютерного центра СГАУ по адресу (http://hpc.ssau.ru/node/25).

Передача файлов на кластер. Для передачи файлов на кластер используйте программы работающие по протоколу SFTP, например WinSCP. Настройка соединения в WinSCP описана по адресу (http://hpc.ssau.ru/node/24).

В операционной системе Линукс можно использовать программу scp. В приведенном ниже примере копируется файл file.dat из домашней директории пользователя в домашнюю директорию на кластере "Сергей Королев"

[user@host ~]$ scp file.dat user@sk.ssau.ru:

Домашняя директория. Домашняя директория пользователя физически находится на системе хранения данных подключенная, с помощью сетевой файловой системы ко всем кластерам и вычислительным нодам СКЦ СГАУ. Скопированные данные становятся доступны на всех вычислительных системах СКЦ. По-умолчанию квота на дисковое пространство пользователя равна 10Гб. Ее можно увеличить при необходимости.

Dot файлы. В домашней директории пользователей находится несколько специальных шаблонных файлов. Эти файлы часто называются dot файлы, потому что имя файла начинается с точки. По умолчанию эти файлы скрыты, их можно увидеть, набрав команду ls -a. .forward - файл содержащий Ваш почтовый адрес. На этот адрес будут перенаправляться сообщения посылаемые Вам системой или СПО. bash_profile - в этом файле можно задавать свои переменные окружения и прочие настройки рабочей среды.

Пакетная система. В качестве системы пакетной обработки заданий (СПО) на кластерах используется менеджер ресурсов Torque и планировщик Moab. Подробнее о системе Torque см. в разделе "Программное обеспечение"

Команды на кластерах СКЦ СГАУ можно выполнять только через систему пакетной обработки заданий, как в пакетном, так и в интерактивном режиме. Удаленный вход пользователей на вычислительные ноды кластеров запрещен.

Основные команды работы с СПО:

qsub - постановка заданий в очередь;

qstat - просмотр статуса выполнения задания;

qdel - удаление задания из очереди.

Основные опции команды qstat

qstat -q - список очередей и их параметры;

qstat -a - список задач с расширенной информацией;

qstat -f <номер задачи> - полная информация о задаче;

qstat -n - информация на каких узлах запущена задача.

Основные опции команды qdel

qdel <номер задачи> - удаление задачи из очереди;

qdel all - удаление всех своих задач из очереди.

Для получения списка узлов с параметрами и состоянием рекомендуется использовать команду pestat.

Лабораторная работа №3. Настройка доступа к вычислительной системе через сервис Templet Web

Цель работы: получение навыков работы в система контроля версий и запуска приложений через сервис Templet Web

Теоретические сведения:

Для работы в системе Templet Web вам неоходимо установить клиент системы контроля версий Subversion на рабочий компьютер и настроить репозиторий проекта.

TortoiseSVN - это бесплатный Windows-клиент с открытыми исходным кодом для системы управления версиями **Apache™ Subversion®** . То есть **TortoiseSVN** управляет файлами и директориями во времени. Файлы хранятся в центральном хранилище. Хранилище больше похоже на обычный файловый сервер, кроме того он запоминает каждое изменение когда-либо сделанное в ваших файлах и директориях. Это позволяет вам восстановить старые версии ваших файлов и проверить историю изменений — как, когда и кто изменял ваши данные.

Установка TortoiseSVN

Системные требования. TortoiseSVN работает под операционной системой Windows XP (с Service Pack 3) или выше, и доступен как в 32-битной, так и в

64-битной версии. Установщик для 64-битной Windows также включает 32-битную часть. Что означает что вы не должны устанавливать 32-битную версию отдельно чтобы контекстное меню и оверлей TortoiseSVN работало в 32-битных приложениях.

Установка. TortoiseSVN поставляется в виде простого в использовании установочного файла. Сделайте на нем двойной клик и следуйте инструкциям - остальное он сделает за вас. Не забудьте перезагрузить компьютер после установки.

Если вы используете Windows XP, у вас по крайней мере, должен быть установлен Service Pack 3. Он не будет работать, если у вас все еще не установлен этот SP! У вас должны быть права администратора системы для установки TortoiseSVN. Поддержка Windows 98, Windows ME и Windows NT4 была прекращена начиная с версии 1.2.0 и Windows 2000 и XP до SP2 начиная с 1.7.0. Вы всё ещё можете загрузить и установить старые версии, если они вам понадобятся

Создать репозиторий проекта SVN вы можете на бесплатных хостингах проектов с открытым исходным кодом SourceForge (sourceforge.net), GoogleCode (code.google.com) или других. Для настройки репозитория следуйте инструкциям, приведенным на сайтах хостингов проектов.

После настройки репозитория проекта, проект необходимо зарегистрировать в сервисе Templet Web. Для этого зарегистрируйтесь на сервисе по адресу templet.ssau.ru/templet. Перейдите на вкладку «Репозиторий», далее «Добавить репозиторий». После добавления проверьте соединение с репозиторием, выбрав «Проверить соединение». Выберите вкладку «Проект» (для возврата в главный экран нажмите «Домой»), далее «Создать проект». В диалоге создания проекта укажите репозиторий и (если требуется) шаблон проекта. Последняя настройка связана с подключением окружения (компьютера или кластерной системы, на которой будет выполняться запуск проекта).

Настройка выполняется на вкладке «Окружения». Теперь вы можете запускать задачи проекта в заданном окружении.

Варианты заданий

Выполнить запуск программ, реализующих перечисленные алгоритмы на высокопроизводительных кластерах СГАУ:

1. Параллельное умножение матриц
    а) итеративный параллелизм (вариант 1)
    б) метод портфеля задач (вариант 2)
    в) круговой конвейер (вариант 3)
2. Вычисление интеграла функции по методу адаптивной квадратуры
    а) рекурсивный параллелизм (вариант 4)
    б) метод портфеля задач (вариант 5)
3. Решение сеточного уравнения методом Якоби
    а) с разделяемыми переменными (вариант 6)
    б) с передачей сообщений (вариант 7)
    в) красное-черное (вариант 8)
4. Гравитационная задача N тел (вариант 9)
5. LU-разложение (вариант 10)

Возможен выбор другого алгоритма, по согласованию с преподавателем.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Задания на практические работы по дисциплине «Вычислительные системы»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Задания на практические работы по лекции 1

1. В Pentium длина команды может достигать 17 байт. Является ли Pentium VLIW-процессором?
2. Пусть диапазон машинного представления числа составляет 0-255. Что получится в результате усечения чисел 96, -9, 300 и 256?
3. Совместимы ли следующие TriMedia-команды и, если нет, то почему?

1) целочисленное сложение, целочисленное вычитание, загрузка, сложение с плавающей точкой, непосредственная загрузка;
2) целочисленное вычитание, целочисленное умножение, непосредственная загрузка, сдвиг, сдвиг;
3) непосредственная загрузка, сложение с плавающей точкой, умножение с плавающей точкой, ветвление, непосредственная загрузка.

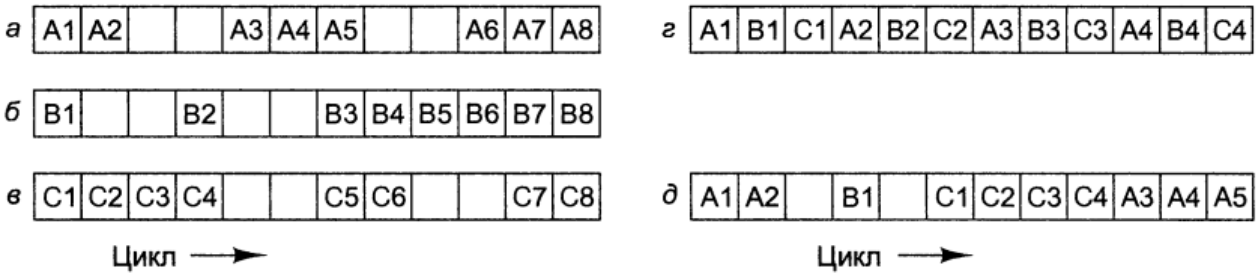


Рисунок 1

4. На рис.1 показаны первые 12 циклов машинных команд. Для каждого варианта опишите, что происходит в следующих трех циклах.
5. На конкретном процессоре при кэш промахе на уровне 1 и кэш-попадании на уровне 2 команда выполняется k машинных циклов. Пусть для маскировки кэш-промахов в кэше уровня 1 используется мелкомодульная многопоточность. Сколько программных потоков должны работать одновременно, чтобы полностью избежать холостых циклов?
6. Утром пчелиная матка созывает рабочих пчел и сообщает им, что сегодня им нужно собрать нектар ноготков. Рабочие пчелы вылетают из улья и летят в разных направлениях в поисках ноготков. Что это за система, SIMD или MIMD?

7. Обсуждая модели состоятельности памяти, мы упомянули, что такая модель представляет собой контракт между программным обеспечением и памятью. Почему необходим такой контракт?
8. Рассмотрим мультипроцессор с общей шиной. Что произойдет, если два процессора попытаются получить доступ к глобальной памяти в один и тот же момент?
9. Предположим, что по техническим причинам следящий кэш может следить только за адресными линиями, а за информационными — нет. Повлияет ли это изменение на протокол сквозной записи?
10. Рассмотрим простую модель мультипроцессорной системы с шиной и без кэширования. Предположим, что одна из каждых четырех команд обращается к памяти, причем при каждом обращении к памяти шина занимается на все время выполнения команды. Если шина занята, то запрашивающий процессор ставится в очередь FIFO. Насколько быстрее будет работать система с 64 процессорами по сравнению с однопроцессорной системой?
11. Протокол MESI имеет четыре состояния. Другой протокол согласования кэшей при отложенной записи имеет три состояния. Каким из состояний протокола MESI можно пожертвовать, и каковы будут последствия каждого из четырех вариантов? Если бы вам пришлось выбрать только три состояния, какие бы вы выбрали?
12. Бывают ли в протоколе MESI такие ситуации, когда строка кэша присутствует в локальной кэш-памяти, но при этом все равно требуется транзакция шины? Если да, то опишите такую ситуацию.
13. Предположим, что к общей шине подсоединено n процессоров. Вероятность того, что один из процессоров пытается использовать шину в данном цикле, равна p. Какова вероятность, что:

1) шина свободна (0 запросов);

2) совершается один запрос;

3) совершается более одного запроса.

14. Сколько схем перекрестной коммутации в полноценном процессоре Fire E25K компании Sun?
15. Предположим, что провод между коммутатором 2A и коммутатором 3B в сети omega поврежден. Какие именно элементы будут отрезаны друг от друга?
16. «Горячие» точки (области памяти, к которым часто происходят обращения) в сетях с многоступенчатой коммутацией представляют собой серьезную проблему. А являются ли они проблемой в системах с шинной организацией?
17. Сеть omega соединяет 4096 RISC-процессоров, время цикла каждого из которых составляет 60 нс, с 4096 бесконечно быстрыми модулями памяти. Каждый коммутирующий элемент дает задержку 5 нс. Сколько слотов отсрочки требуется для команды LOAD?

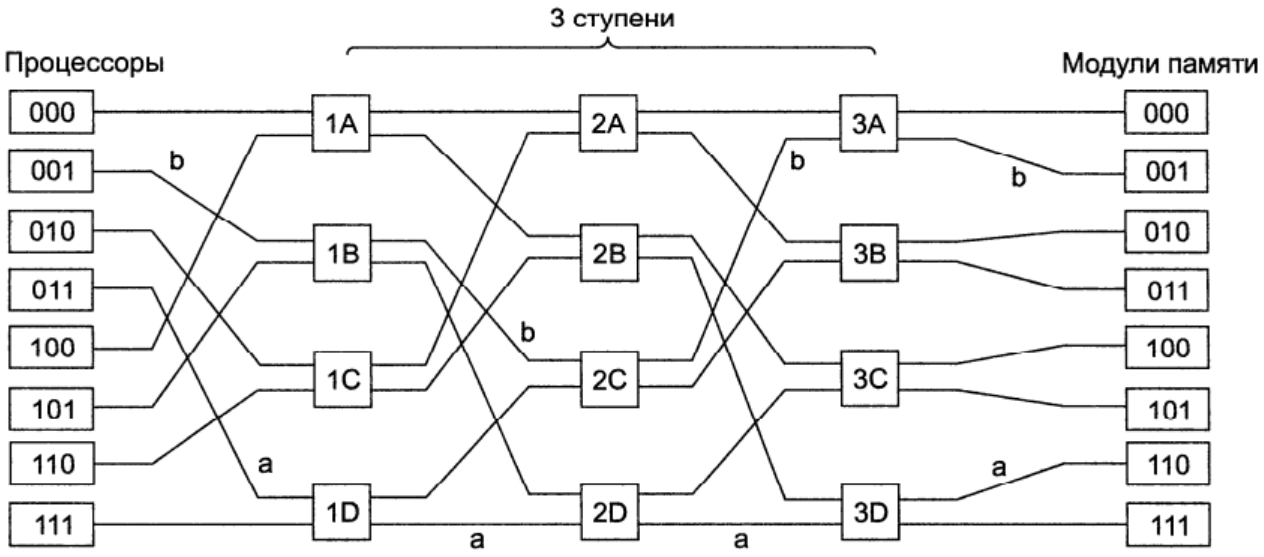


Рисунок 2

18. Рассмотрим машину, использующую сеть omega. Предположим, что программа и стек i хранятся в модуле памяти i. Какое незначительное изменение топологии может значительно повлиять на производительность? (Эта модифицированная топология используется в IBM RP3 и BBN Butterfly.) Какой недостаток имеет новая топология по сравнению со старой?
19. В NUMA-мультипроцессоре обращение к локальной памяти занимает 20 нс,

а к памяти другого процессора — 120 нс. Программа во время выполнения совершает N обращений к памяти, 1% из которых — обращения к странице P. Изначально эта страница находится в удаленной памяти, а на копирование ее из локальной памяти требуется C нс. При каких обстоятельствах эту страницу следует копировать локально, если ее не используют другие процессоры?

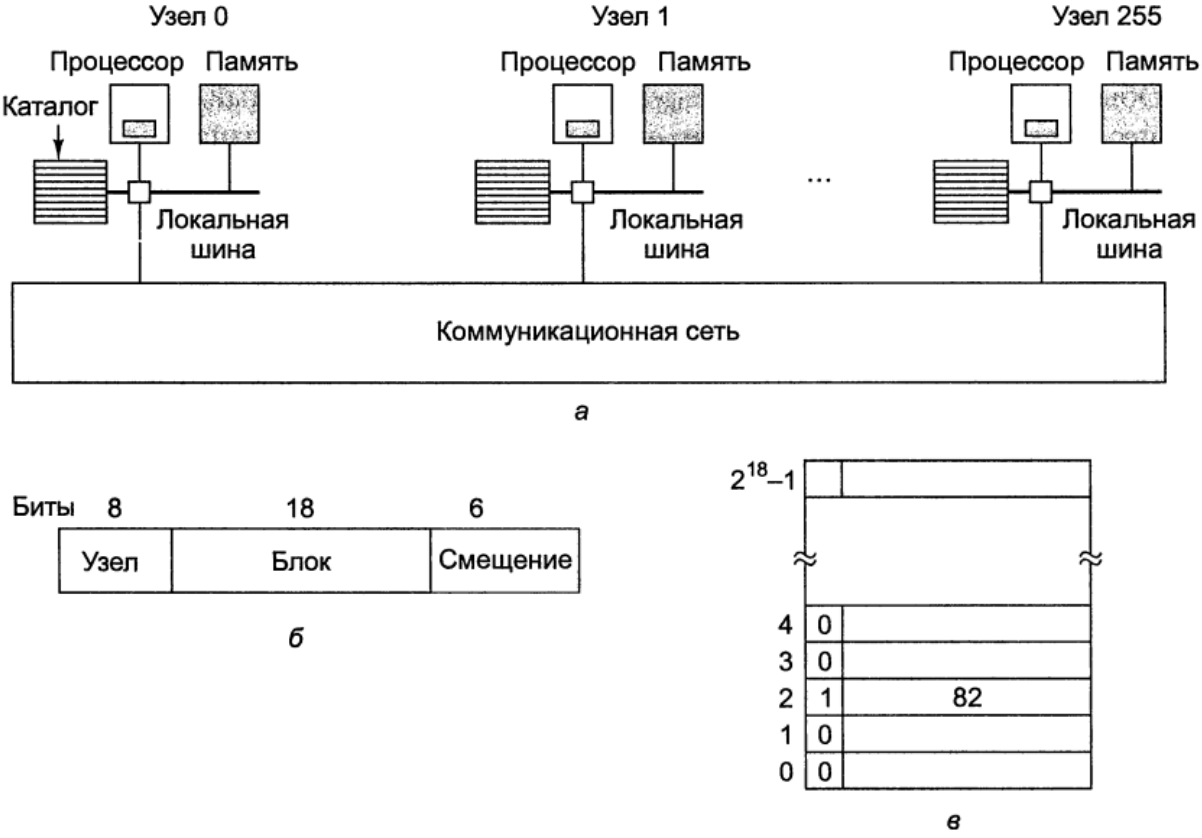


Рисунок 3

20. Рассмотрим CC-NUMA-мультипроцессор, такой, как на рис., но содержащий 512 узлов по 8 Мбайт каждый. Если длина строки кэша составляет 64 байта, каков процент непроизводительных затрат для каталогов? Как повлияет увеличение числа узлов на непроизводительные затраты (они увеличатся, уменьшатся или останутся без изменений)?

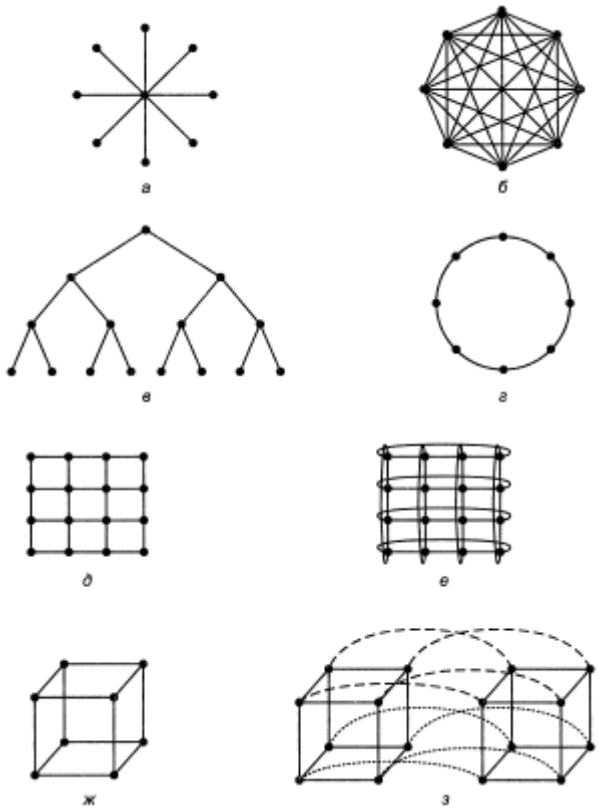


Рисунок 4

21. Вычислите диаметр сети для каждой из топологий, представленных на рис.4
22. Для каждой из топологий, представленных на рис., определите коэффициент отказоустойчивости (максимальное число линий связи, после утраты которых сеть не окажется разделена на две части).
23. Рассмотрим топологию двойной тор (е), расширенную до размера k x k. Каков диаметр такой сети? (Подсказка: четное и нечетное значение k нужно рассматривать отдельно.)
24. Представим сеть в форме куба 8x8x8. Каждая линия связи имеет дуплексную пропускную способность 1 Гбайт/с. Какова пропускная способность сечения в этой сети?
25. Закон Амдала ограничивает потенциальное ускорение, достижимое в параллельном компьютере. Вычислите как функцию от f максимально возможное ускорение, если число процессоров стремится к бесконечности. Каково значение этого предела для f = 0,1?

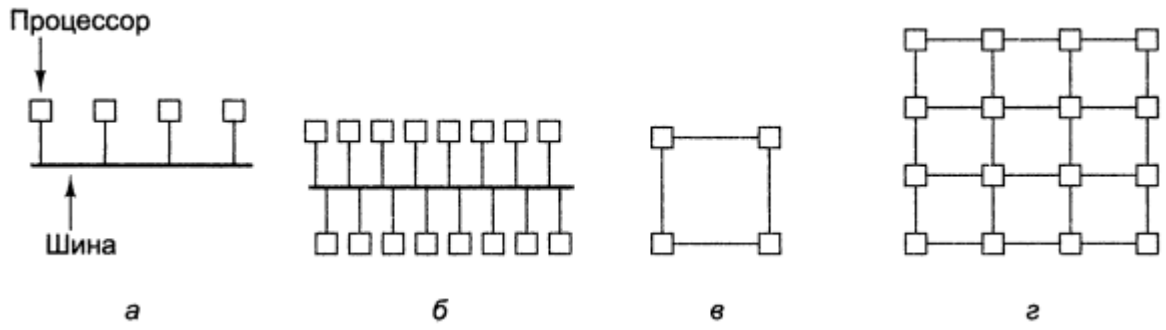


Рисунок 5

26. Рисунок 5 призван показать, что масштабирование в случае с шиной невозможно, а в случае с решеткой возможно и желательно. Предположим, каждая шина или линия связи имеет пропускную способность b. Вычислите среднюю пропускную способность на каждый процессор для каждого из четырех случаев. Затем масштабируйте каждую систему до 64 процессоров и выполните те же вычисления. Чему равен предел, если число процессоров стремится к бесконечности?
27. Мы обсуждали три варианта примитива send — синхронный, блокирующий и неблокирующий. Предложите четвертый вариант, напоминающий блокирующий, но немного отличающийся по свойствам. Какое преимущество и каков недостаток имеет новый примитив по сравнению с обычной блокирующей операцией send?
28. Рассмотрим компьютер, который работает в сети с аппаратным широковещанием (например, Ethernet). Почему важно соотношение операций чтения (которые не изменяют внутреннее состояние переменных) и записи (которые изменяют внутреннее состояние переменных)?

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Темы для подготовки к экзамену (зачету) по дисциплине «Вычислительные системы»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Список вопросов

1. Причины появления параллельных вычислительных архитектур. Уровни параллелизма.
2. Параллелизм на уровне команд.
3. Внутрипроцессорная многопоточность.
4. Однокристальные мультипроцессоры.
5. Сетевые процессоры.
6. Мультимедиа-процессоры.
7. Криптопроцессоры.
8. Мультипроцессоры и мультикомпьютеры.
9. Семантика памяти.
10. UMA-мультипроцессоры в симметричных мультипроцессорных архитектурах.
11. NUMA-мультипроцессоры.
12. COMA-мультипроцессоры.
13. Мультикомпьютеры.
14. Коммуникационные сети.
15. Процессоры с массовым параллелизмом.
16. Кластерные вычисления.
17. Коммуникационное программное обеспечение для мультикомпьютеров.
18. Планирование.
19. Общая память на прикладном уровне.
20. Грид-системы.
21. Языки, уровни и виртуальные машины.
22. Обзор архитектуры IA-64.
23. Вычислительная инфраструктура СГАУ.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)**

Тесты для итогового контроля знаний по дисциплине «Вычислительные системы»

направление 230100.68 – «Информатика и вычислительная техника»
(магистратура)

Факультет информатики
Кафедра информационных систем и технологий

Разработал:
Востокин С.В.,
профессор кафедры ИСТ

Самара 2013 г.

Дисциплина «Вычислительные системы»

Студент _____ группа_____

Вариант 1

Вопрос 1	**Основана на решении проблемы совместимости команд на стадии компиляции**	**Балл**
1	**VLIW архитектура**	
2	**Суперскалярная архитектура**	
3	**Внутрипроцессорная многопоточная архитектура**	
4	**Архитектура на базе сопроцессоров**	

Вопрос 2	**Архитектура с однородным доступом к памяти**	**Балл**
1	**UMA**	
2	**COMA**	
3	**NUMA**	
4	**CC-NUMA**	

Вопрос 3	**В одной микросхеме находится несколько автономных процессоров**	**Балл**
1	**MPP**	
2	**COW**	
3	**Грид-система**	
4	**Многоядерная система**	

Вопрос 4	**Использует базу данных с информацией о том, где находится каждая строка кэша и каково её состояние**	**Балл**
1	**UMA с шинной организацией**	
2	**UMA с коммутацией**	
3	**CC-NUMA**	
4	**NC-NUMA**	

Вопрос 5	**Такой архитектуры не существует**	**Балл**
1	**SISD**	
2	**SIMD**	
3	**MISD**	
4	**MIMD**	

Вопрос 6	**В данных процессорах операнды некоторых команд могут выступать упорядоченные массивы данных**	**Балл**
1	**Векторный процессор**	
2	**Матричный процессор**	
3	**Мультипроцессор**	
4	**Мультикомпьютер**	

Вопрос 7	Какая из приведенных архитектур не является ISA	Балл
1	**Архитектура x86**	
2	**Архитектура ARM**	
3	**Архитектура AVR**	
4	**Архитектура SMP**	

Вопрос 8	SSE команды появились впервые в процессорах	Балл
1	**Pentium**	
2	**Pentium Pro**	
3	**Pentium II**	
4	**Pentium III**	

Вопрос 9	1 петафлопс это	Балл
1	**10^9 операций с плавающей точной в секунду**	
2	**10^{12} операций с плавающей точной в секунду**	
3	**10^{15} операций с плавающей точной в секунду**	
4	**10^{18} операций с плавающей точной в секунду**	

Вопрос 10	Фон Нейман реализовал свою архитектуру в компьютере	Балл
1	**IAS**	
2	**PDP-11**	
3	**CRAY-1**	
4	**Z1**	

Вопрос 11	Метод сокращения числа условных переходов называется	Балл
1	**Предсказание переходов**	
2	**Ассемблирование**	
3	**Предикация**	
4	**Деприкация**	

Вопрос 12	При любом считывании памяти возвращается значение самой последней записи	Балл
1	**Строгая состоятельность**	
2	**Секвинциальная состоятельность**	
3	**Процессорная состоятельность**	
4	**Слабая состоятельность**	

Дисциплина «Вычислительные системы»

Студент _____ группа_____

Вариант 2

Вопрос 1	**Подход использования одного конвейера с большим количеством функциональных блоков реализует**	**Балл**
1	**VLIW архитектура**	
2	**Суперскалярная архитектура**	
3	**Внутрипроцессорная многопоточная архитектура**	
4	**Архитектура на базе сопроцессоров**	

Вопрос 2	**Популярная реализация данной архитектуры основана на хранении информации о том, где находится каждая строка кэша и каково её состояние**	**Балл**
1	**UMA**	
2	**COMA**	
3	**NUMA**	
4	**CC-NUMA**	

Вопрос 3	**Объединение территориально удаленных компьютеров в единую вычислительную систему**	**Балл**
1	**MPP**	
2	**COW**	
3	**Грид-система**	
4	**Многоядерная система**	

Вопрос 4	**Самая простая мультипроцессорная архитектура, применяемая для нескольких процессоров**	**Балл**
1	**UMA с шинной организацией**	
2	**UMA с коммутацией**	
3	**CC-NUMA**	
4	**NC-NUMA**	

Вопрос 5	**Мультипроцессорная и мультикомпьютерная архитектура**	**Балл**
1	**SISD**	
2	**SIMD**	
3	**MISD**	
4	**MIMD**	

Вопрос 6	**MIMD с общей памятью**	**Балл**
1	**Векторный процессор**	
2	**Матричный процессор**	
3	**Мультипроцессор**	
4	**Мультикомпьютер**	

Вопрос 7	Процессоры смартфонов в основном основаны на данной архитектуре	Балл
1	**Архитектура x86**	
2	**Архитектура ARM**	
3	**Архитектура AVR**	
4	**Архитектура SMP**	

Вопрос 8	Первый 16-разрядный процессор на микросхеме	Балл
1	**8008**	
2	**8080**	
3	**8086**	
4	**8088**	

Вопрос 9	1 экзафлопс это	Балл
1	**10^9 операций с плавающей точной в секунду**	
2	**10^{12} операций с плавающей точной в секунду**	
3	**10^{15} операций с плавающей точной в секунду**	
4	**10^{18} операций с плавающей точной в секунду**	

Вопрос 10	Первая релейная вычислительная машина	Балл
1	**IAS**	
2	**PDP-11**	
3	**CRAY-1**	
4	**Z1**	

Вопрос 11	Прерывания обрабатывает	Балл
1	**DSM**	
2	**ISR**	
3	**MPP**	
4	**NOW**	

Вопрос 12	Все процессоры воспринимают один и тот же порядок чтения-записи	Балл
1	**Строгая состоятельность**	
2	**Секвинциальная состоятельность**	
3	**Процессорная состоятельность**	
4	**Слабая состоятельность**	

Дисциплина «Вычислительные системы»

Студент _____ группа_____

Вариант 3

Вопрос 1	**Методика минимизации промахов в кэш, обеспечивающая повышение производительности**	**Балл**
1	**VLIW архитектура**	
2	**Суперскалярная архитектура**	
3	**Внутрипроцессорная многопоточная архитектура**	
4	**Архитектура на базе сопроцессоров**	

Вопрос 2	**Реализует использование основной памяти каждого процессора в качестве кэш-памяти**	**Балл**
1	**UMA**	
2	**COMA**	
3	**NUMA**	
4	**CC-NUMA**	

Вопрос 3	**Компьютеры данной архитектуры состоят из большого числа процессоров, связанных скоростной внутренней коммуникационной сетью**	**Балл**
1	**MPP**	
2	**COW**	
3	**Грид-система**	
4	**Многоядерная система**	

Вопрос 4	**В данной архитектуре отсутствует кэш-память**	**Балл**
1	**UMA с шинной организацией**	
2	**UMA с коммутацией**	
3	**CC-NUMA**	
4	**NC-NUMA**	

Вопрос 5	**Технологии SSE реализуют данную архитектуру в процессорах Intel**	**Балл**
1	**SISD**	
2	**SIMD**	
3	**MISD**	
4	**MIMD**	

Вопрос 6	**Этот процессор реализует архитектуру SIMD**	**Балл**
1	**Векторный процессор**	
2	**Матричный процессор**	
3	**Мультипроцессор**	
4	**Мультикомпьютер**	

Вопрос 7	Используется в низко производительных встроенных системах	Балл
1	**Архитектура x86**	
2	**Архитектура ARM**	
3	**Архитектура AVR**	
4	**Архитектура SMP**	

Вопрос 8	Защита памяти впервые появилась в этом процессоре семейства x86	Балл
1	**80186**	
2	**80286**	
3	**80386**	
4	**80486**	

Вопрос 9	1 терафлопс это	Балл
1	**10^9 операций с плавающей точной в секунду**	
2	**10^{12} операций с плавающей точной в секунду**	
3	**10^{15} операций с плавающей точной в секунду**	
4	**10^{18} операций с плавающей точной в секунду**	

Вопрос 10	Миникомпьютеры 70-х годов	Балл
1	**IAS**	
2	**PDP-11**	
3	**CRAY-1**	
4	**Z1**	

Вопрос 11	Линковщиком не является	Балл
1	**Компоновщик**	
2	**Компонующий загрузчик**	
3	**Редактор связей**	
4	**Транслятор**	

Вопрос 12	Все процессоры видят операции записи любого процессора в том порядке, в котором они выполняются и все процессоры видят операции записи в любое слово памяти в одном и том же порядке	Балл
1	**Строгая состоятельность**	
2	**Секвинциальная состоятельность**	
3	**Процессорная состоятельность**	
4	**Слабая состоятельность**	

Дисциплина «Вычислительные системы»

Студент _____ группа_____

Вариант 4

Вопрос 1	**Обычно реализует обработку графики и вычисления с плавающей точкой**	**Балл**
1	**VLIW архитектура**	
2	**Суперскалярная архитектура**	
3	**Внутрипроцессорная многопоточная архитектура**	
4	**Архитектура на базе сопроцессоров**	

Вопрос 2	**Доступ к локальным модулям памяти происходит быстрее чем к удаленным**	**Балл**
1	**UMA**	
2	**COMA**	
3	**NUMA**	
4	**CC-NUMA**	

Вопрос 3	**Вычислительная сеть, объединяющая несколько рабочих станций**	**Балл**
1	**MPP**	
2	**COW**	
3	**Грид-система**	
4	**Многоядерная система**	

Вопрос 4	**Использует неблокирующую сеть с числом коммутационных узлов N^2**	**Балл**
1	**UMA с шинной организацией**	
2	**UMA с перекрестной коммутацией**	
3	**CC-NUMA**	
4	**NC-NUMA**	

Вопрос 5	**Классическая архитектура фон Неймана**	**Балл**
1	**SISD**	
2	**SIMD**	
3	**MISD**	
4	**MIMD**	

Вопрос 6	**Имеет распределенную память**	**Балл**
1	**Векторный процессор**	
2	**Матричный процессор**	
3	**Мультипроцессор**	
4	**Мультикомпьютер**	

Вопрос 7	Архитектура, разрабатываемая Intel	Балл
1	**Архитектура x86**	
2	**Архитектура ARM**	
3	**Архитектура AVR**	
4	**Архитектура SMP**	

Вопрос 8	Первый 32 разрядный процессор семейства x86	Балл
1	**80186**	
2	**80286**	
3	**80386**	
4	**80486**	

Вопрос 9	Первый векторный суперкомпьютер	Балл
1	**CRAY-1**	
2	**PDP-11**	
3	**IAS**	
4	**Z1**	

Вопрос 10	1 гигафлопс это	Балл
1	**10^9 операций с плавающей точной в секунду**	
2	**10^{12} операций с плавающей точной в секунду**	
3	**10^{15} операций с плавающей точной в секунду**	
4	**10^{18} операций с плавающей точной в секунду**	

Вопрос 11	Для языка процессора не верно утверждение	Балл
1	**Доступны все объекты целевой машины**	
2	**Программа работает только на компьютерах одного семейства**	
3	**Нет символических имен для переменных**	
4	**Оператор соответствует ровно одной машинной команде**	

Вопрос 12	Не гарантируется, что операции записи, произведенные одним процессором, будут восприниматься другими в том же порядке	Балл
1	**Строгая состоятельность**	
2	**Секвинциальная состоятельность**	
3	**Процессорная состоятельность**	
4	**Слабая состоятельность**	

Ответы к тестам

Вариант № 1

№ вопроса	1	2	3	4	5	6	7	8	9	10	11	12
№ ответа	1	1	4	3	3	1	4	4	3	1	3	1

Вариант № 2

№ вопроса	1	2	3	4	5	6	7	8	9	10	11	12
№ ответа	2	4	3	1	4	3	2	3	4	4	2	2

Вариант № 3

№ вопроса	1	2	3	4	5	6	7	8	9	10	11	12
№ ответа	3	2	1	4	2	1	3	2	2	2	4	3

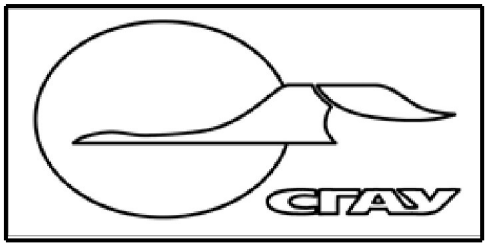
Вариант № 4

№ вопроса	1	2	3	4	5	6	7	8	9	10	11	12
№ ответа	4	3	2	2	1	4	1	3	1	1	3	4

Критерии оценивания тестов

Оценка	Количество верных ответов	
	не менее	не более
неудовлетворительно	0	4
удовлетворительно	4	7
хорошо	7	10
отлично	10	12

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)



СОГЛАСОВАНО

Управление образовательных программ

_____/ А.В. Дорошин /

"_____"_____ 20___ г.

УТВЕРЖДАЮ

Проректор по учебной работе

_____/ Ф.В. Гречников /

"____" _____ 20___ г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Наименование модуля (дисциплины)

Вычислительные системы

Цикл, в рамках которого происходит освоение модуля (дисциплины)	М2. Профессиональный цикл
Часть цикла	М2.Б Обязательные дисциплины

Код учебного плана

230100_68_2-13-56-3022

Факультет	6
Кафедра	Информационные системы и технологии

Курс	6		
Семестр	В		
Лекции (СЛ)	18		
Семинарские и практические занятия (СП)	18		
Лабораторные занятия (СЛР)	36	Экзамен	В
Контроль самостоятельной работы / Индивидуальные занятия (КСР / ИЗ)	0	Зачет	
Самостоятельная работа (СРС)	72		
Всего (Всего с экзаменами)	180		

Наименование стандарта, на основании которого составлена рабочая программа:

Магистерская программа "Программное обеспечение мобильных устройств" 230100.68 "Информатика и вычислительная техника"

Соответствие содержания рабочей программы, условий ее реализации, материально-технической и учебно-методической обеспеченности учебного процесса по дисциплине всем требованиям государственных стандартов подтверждаем.

Составители:

Востокин С.В., д.т.н., доц. _____
(подпись)

Заведующий кафедрой:

Прохоров С.А., д.т.н., проф. _____
(подпись)

Рабочая программа обсуждена на заседании кафедры

Информационные системы и технологии

Протокол №___ от "____" _____ 20___ г.

Наличие основной литературы в фондах научно-технической библиотеки (НТБ) подтверждаем:

Директор НТБ _____ /_____/
(подпись) (расшифровка подписи)

Согласовано:

Декан _____ /_____/
(подпись) (расшифровка подписи)

1 Цели и задачи модуля (дисциплины), требования к уровню освоения содержания

1.1 Перечень развиваемых компетенций

Коды компетенций из ФГОС-3 " Информатика и вычислительная техника" 230100: ОК-1, ОК-2, ОК-3, ОК-4, ОК-5, ОК-7, ПК-4, ПК-7.

1.2 Цели и задачи изучения модуля (дисциплины)

Целью данного курса является получения общих сведений об архитектурах современных вычислительных систем и принципах их функционирования, а также практических навыков их использования.

1.3 Требования к уровню подготовки студента, завершившего изучение данного модуля (дисциплины)

Студенты, завершившие изучение данной дисциплины, должны знать: функциональное назначение, архитектуру, принципы управления ресурсами в современных вычислительных системах; уметь: разрабатывать прикладные программы и библиотеки с использованием интерфейсов прикладного программирования, решать конкретные инженерные и исследовательские задачи, требующие применения навыков системного программирования.

1.4 Связь с предшествующими модулями (дисциплинами)

Курс базируется на сведениях из курсов:
- Архитектура современных операционных систем
- Архитектура современных распределенных систем.

1.5 Связь с последующими модулями (дисциплинами)

Курс «Архитектура современных распределённых систем» дает необходимые сведение для выполнения проектов и работ, требующих навыков системного программирования для направления подготовки 230100 «Информатика и вычислительная техника», используется в курсах: «Методы проектирования и поддержки требований к программному обеспечению», а также при выполнении выпускной квалификационной работы магистра.

2 Содержание рабочей программы (модуля)

Семестр 1		
СЛ 0,1667 18 часов 0,5 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 1	1. Введение в вычислительные системы.
		2. Внутрипроцессорный параллелизм.

		3. Сопроцессоры.
		4. Мультипроцессоры.
		5. Мультикомпьютеры.
		6. Распределенные вычислительные системы.
		7. Многоуровневая компьютерная организация.
		8. Развитие компьютерной архитектуры.
		9. Типы компьютеров.
		10. Семейства компьютеров.
		11. Архитектура IA-64
		12. Вычислительная инфраструктура СГАУ.
СП 0,1667 18 часов 0,5 ЗЕТ	Активные 0	
	Интерактивные 1	1. Система программирования Templet
		2. Типовой вычислительный процесс "портфель задач"
		3. Типовой вычислительный процесс "конвейер"
		4. Распределенные вычисления в системе Templet Web
		5. Вычисления в системах с общей памятью в Templet Web
	Традиционные 0	
СЛР 0,3333 36 часов 1 ЗЕТ	Активные 0	
	Интерактивные 1	Лаб.раб.№1. Конфигурирование средств управления доступом на кластер СГАУ.
		Лаб.раб.№2. Настройка SSH доступа к удаленной вычислительной системе.
		Лаб.раб.№3. Настройка доступа к вычислительной системе через сервис Templet Web
	Традиционные 0	
КСР 0 0 часов 0 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	

СРС 0,6667 72 часов 2 ЗЕТ	Активные 1	1.Самостоятельная работа с литературой по предмету.
		2.Изучение примеров приложений по теме лаб.работ.
		3.Подготовка итогового письменного отчета по лаб.работам.
		4.Подготовка к экзамену по теоретическим вопросам.
	Интерактивные 0	
	Традиционные 0	

3 Инновационные методы обучения

Для развития профессиональных навыков, необходимых обучающимся, Программа предполагает широкое использование активных и интерактивных форм проведения занятий: дискуссий, презентаций, конференций, проектной работы. При подаче лекционного материала исп
ользуются мультимедиа материалы. Программа предполагает выполнение дополнительных заданий с элементами исследования (разработка подсистем распределённой системы автоматизации параллельного программирования). В лабораторном практикуме ведется работа с электронной технической документацией через сеть Интернет.

4 Технические средства и материальное обеспечение учебного процесса

Компьютерный класс с компьютерами, имеющими подключение к сети Интернет. Программное обеспечение: ОС MS Windows XP/Vista/7 (лицензии СГАУ), MS Visual Studio 2005 Academic Edition или MS Visual Studio 2008 Express (лицензии СГАУ, бесплатно для учебного использования).

5 Учебно-методическое обеспечение

5.1 Основная литература

1. Таненбаум, Э. Архитектура компьютера [Текст] / Э. Таненбаум, Т. Остин - СПб. [и др.] : Питер , 2013. - 816 с. - (Классика computer science). - ISBN 978-5-496-00337-7 (0 экз.)
2. Цилькер, Б.Я. Организация ЭВМ и систем [Текст] : [учеб. для вузов по направлению "Информатика и вычисл. техника"] / Б.Я. Цилькер, С.А. Орлов. - СПб. и др. : Питер : Питер принт, 2006. - 667 с. - (Учебник для вузов). - ISBN 5-94723-759-8 (1 экз.)
3. Цилькер, Б.Я. Организация ЭВМ и систем [Текст] : [учеб. для вузов по направлению "Информатика и вычисл. техника"] / Б.Я. Цилькер, С.А. Орлов. - СПб. и др. : Питер : Питер принт, 2004. - 667 с. - (Учебник для вузов). - ISBN 5-94723-759-8 (112 экз.)

5.2 Дополнительная литература

1. Востокин, С.В. Вопросы, задания и упражнения по курсу "Операционные системы" [Текст] : [лаб. практикум] / М-во образования и науки Рос. Федерации, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т) ; [сост. С. В. Востокин]. - Самара : Изд-во СГАУ, 2012. - 29 с. (20 экз.)
2. Востокин, С.В. Операционные системы [Электронный ресурс] : [учеб. для вузов по направления подгот. бакалавров "Инф. и выч. техн.", "Фундам. информатика и информ. технологии", "Прикладная математика и информатика", "Прикладная математика и физика"] /С.В. Востокин ; М-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т). - Электрон. текстовые дан. - Самара : Изд-во СГАУ, 2012. – ISBN 978-5-7883-0916-3 (2 эл. опт. диска)
3. Таненбаум, Э.Современные операционные системы [Текст] / Эндрю Таненбаум. - 2-е изд. - СПб. [и др.] : Питер : Питер принт, 2005. - 1037 с. - (Классика computer science). - ISBN 5-318-00299-4 (5 экз.)
4. Таненбаум, Э. Современные операционные системы [Текст] / Эндрю Таненбаум. - 2-е изд., [перераб. и испр.]. - СПб. [и др.] : Питер : Питер Пресс, 2007. - 1037 с. - (Классика computer science). - ISBN 978-5-318-00299-1 (2 экз.)
5. Солдатова, О. П. Системное программирование [Текст] : Курс лекций / О. П. Солдатова, С. В. Востокин ; Самар. гос. аэрокосм. ун-т им. С. П. Королева. - Самара : [б. и.], 2002. - 123 с. - ISBN 5-7883-0226-9 (94 экз.)

5.3 Электронные источники и интернет ресурсы

1. Microsoft Developer Network http://msdn.microsoft.com
2. www.prenhall.com/tanenbaum
3. Проект автоматизации параллельных и распределенных вычислений «Темплет» http://templet.ssau.ru
4. Интуит национальный открытый университет.http://www.intuit.ru/

5.4 Методические указания и рекомендации

Текущий контроль знаний студентов завершается на отчетном занятии, результатом которого является допуск или недопуск студента к экзамену по дисциплине. Основанием для допуска к зачёту является выполнение и отчет студента по всем лабораторным работам.

Экзамен проводится согласно положению о текущем и промежуточном контроле знаний студентов, утвержденному ректором университета. Экзаменационная оценка ставится на основании письменного и устного ответов студента по экзаменационному билету, а также, при необходимости, ответов на дополнительные вопросы. Экзаменационный билет включает 3 вопроса. Дополнительно может быть предложен как теоретический вопрос, так и вопрос на понимание программ по листингам.