**Лекция 7** **Принципы организации ЭВМ**

**Цель: рассмотреть основные принципы организации ЭВМ**

**1.1 Основные понятия и определения.**

В основе архитектуры современных ВМ лежит представление алгоритма реше­ния задачи в виде программы последовательных вычислений. Согласно стандарту ISO 2382/1-84, программа для ВМ — это «упорядоченная последовательность ко­манд, подлежащая обработке».

ВМ, где определенным образом закодированные команды программы хранят­ся в памяти, известна под названием вычислительной машины с хранимой в памя­ти программой. Идея принадлежит создателям вычислителя ENIАС Эккерту, Моч-ли и фон Нейману.

Сущность фон-неймановской концепции вычислительной машины можно све­сти к четырем принципам:

* двоичного кодирования;
* программного управления;
* однородности памяти;
* адресности.

**Принцип двоичного кодирования**

Согласно этому принципу, вся информация, как данные, так и команды, кодиру­ются двоичными цифрами 0 и 1. Каждый тип информации представляется двоичной последовательностью и имеет свой формат. Последовательность битов в фор­мате, имеющая определенный смысл, называется полем. В числовой информации обычно выделяют поле знака и поле значащих разрядов. В формате команды можно выделить два поля:

* поле кода операции;
* поле адресов (адресную часть).

Код операции представляет собой указание, какая операция должна быть вы­полнена, и задается с помощью двоичной комбинации.

Вид адресной части и число составляющих ее адресов зависят от типа команды: в командах преобразования данных АЧ содержит адреса объектов обработки (опе­рандов) и результата; в командах изменения порядка вычислений — адрес следую­щей команды программы; в командах ввода/вывода — номер устройства ввода/ вывода. Адресная часть также представляется двоичной последовательностью. Таким образом, команда в вычислительной ма­шине имеет вид (длина кода операции + длина поля адресов)-разрядной двоичной комбинации.

**Принцип программного управления**

Все вычисления, предусмотренные алгоритмом решения задачи, должны быть пред­ставлены в виде программы, состоящей из последовательности управляющих слов — команд. Каждая команда предписывает некоторую операцию из набора опе­раций, реализуемых вычислительной машиной. Команды программы хранятся в последовательных ячейках памяти вычислительной машины и выполняются в естественной последовательности, то есть в порядке их положения в программе. При необходимости, с помощью специальных команд, эта последовательность мо­жет быть изменена. Решение об изменении порядка выполнения команд програм­мы принимается либо на основании анализа результатов предшествующих вычис­лений, либо безусловно.

**Принцип однородности памяти**

Команды и данные хранятся в одной и той же памяти и внешне в памяти неразли­чимы. Распознать их можно только по способу использования. Это позволяет про­изводить над командами те же операции, что и над числами, и, соответственно, открывает ряд возможностей. Так, циклически изменяя адресную часть команды, можно обеспечить обращение к последовательным элементам массива данных. Более полезным является другое следствие принципа однородности, когда команды одной программы могут быть получены как результат исполнения другой программы.

**Принцип адресности**

Структурно основная память состоит из пронумерованных ячеек, причем процес­сору в произвольный момент доступна любая ячейка. Двоичные коды команд и данных разделяются на единицы информации, называемые словами, и хранятся в ячейках памяти, а для доступа к ним используются номера соответствующих яче­ек — адреса.

**Фон-неймановская архитектура**

Большинство совре­менных ВМ по своей структуре отвечают принципу программного управления. Типичная фон-неймановская ВМ содержит: память, устройство управ­ления, арифметико-логическое устройство и устройство ввода/вывода.

Вторичная память

Периферийные устройства ввода

Периферийные устройства вывода

Порты ввода

Основная память

Порты вывода

Центральный процессор

Арифметико-логическое устройство

Устройство управления

В любой ВМ имеются средства для ввода программ и данных к ним. Информация поступает из подсоединенных к ЭВМ периферийных устройств ввода. Результаты вычислений выводятся на периферийные устройства вывода. Связь и взаимодействие вычислительной машины и периферийных устройств обеспечивают порты ввода и порты вывода. Термином порт обозначают аппаратуру сопряжения периферийного устройства с ВМ и управления им. Совокупность портов ввода и вывода называют устройством вво­да/вывода (УВВ) или модулем ввода/вывода ВМ (МВБ).

Введенная информация сначала запоминается в основной памяти, а затем пе­реносится во вторичную память, для длительного хранения. Чтобы программа могла выполняться, команды и данные должны располагаться в основной памяти (ОП), организованной таким образом, что каждое двоичное слово хранится в от­дельной ячейке, идентифицируемой адресом, причем соседние ячейки памяти имеют следующие по порядку адреса. Доступ к любым ячейкам запоминающего устройства (ЗУ) основной памяти может производиться в произвольной последо­вательности. Такой вид памяти известен как память с произвольным доступом. ОП современных ВМ в основном состоит из полупроводниковых оперативных запоминающих устройств (ОЗУ), обеспечивающих как считывание, так и запись информации. Для таких ЗУ характерна энергозависимость — хранимая информа­ция теряется при отключении электропитания. Если необходимо, чтобы часть ос­новной памяти была энергонезависимой, в состав ОП включают постоянные за­поминающие устройства (ПЗУ), также обеспечивающие произвольный доступ. Хранящаяся в ПЗУ информация может только считываться (но не записываться).

Размер ячейки основной памяти обычно принимается равным 8 двоичным раз­рядам — байту. Для храпения больших чисел используются 2,4 или 8 байтов, раз­мещаемых в ячейках с последовательными адресами. В этом случае за адрес числа часто принимается адрес его младшего байта. Так, при хранении 32-разрядного числа в ячейках с адресами 200, 201, 202 и 203 адресом числа будет 200.

Для долговременного хранения больших программ и массивов данных в ВМ обыч­но имеется дополнительная память, известная как вторичная. Вторичная память энер­гонезависима и чаще всего реализуется на базе магнитных дисков. Информация в ней хранится в виде специальных программно поддерживаемых объектов — файлов (согласно стандарту ISO, файл — это «идентифицированная совокупность экземп­ляров полностью описанного в конкретной программе типа данных, находящихся вне программы во внешней памяти и доступных программе посредством специ­альных операций»).

Устройство управления (УУ) — важнейшая часть ВМ, организующая автома­тическое выполнение программ (путем реализации функций управления) и обес­печивающая функционирование ВМ как единой системы. Для пояснения функ­ций УУ ВМ следует рассматривать как совокупность элементов, между которыми происходит пересылка информации, в ходе которой эта информация может под­вергаться определенным видам обработки. Пересылка информации между любы­ми элементами ВМ инициируется своим сигналом управления (СУ), то есть управ­ление вычислительным процессом сводится к выдаче нужного набора СУ в нужной временной последовательности. Основной функцией УУ является формирование управляющих сигналов, отвечающих за извлечение команд из памяти в порядке, определяемом програм­мой, и последующее исполнение этих команд. Кроме того, УУ формирует СУ для синхронизации и координации внутренних и внешних устройств ВМ.

Еще одной неотъемлемой частью ВМ является арифметико-логическое устрой­ство (АЛУ). АЛУ обеспечивает арифметическую и логическую обработку двух входных переменных, в результате которой формируется выходная переменная. Функции АЛУ обычно сводятся к простым арифметическим и логическим опера­циям, а также операциям сдвига. Помимо результата операции АЛУ формирует ряд признаков результата (флагов), характеризующих полученный результат и со­бытия, произошедшие в процессе его получения (равенство нулю, знак, четность, перенос, переполнение и т. д.). Флаги могут анализироваться в УУ с целью приня­тия решения о дальнейшей последовательности выполнения команд программы.

УУ и АЛУ тесно взаимосвязаны и их обычно рассматривают как единое уст­ройство, известное как центральный процессор (ЦП) или просто процессор. Поми­мо УУ и АЛУ в процессор входит также набор регистров общего назначения (РОН), служащих для промежуточного хранения информации в процессе ее обработки.

**Типы структур вычислительных машин и систем**

Достоинства и недостатки архитектуры вычислительных машин и систем изна­чально зависят от способа соединения компонентов. При самом общем подходе можно говорить о двух основных типах структур вычислительных машин и двух типах структур вычислительных систем.

**Структуры вычислительных машин**

В настоящее время примерно одинаковое распространение получили два способа построения вычислительных машин: с непосредственными связями и на основе шины.

Типичным представителем первого способа может служить классическая фон-неймановская ВМ. В ней между взаимодействующими устройствами (процессор, память, устройство ввода/вывода) имеются непосредственные связи. Особенности связей (число линий в шинах, пропускная способность и т. п.) опре­деляются видом информации, характером и интенсивностью обмена. Достоинством архитектуры с непосредственными связями можно считать возможность развязки «узких мест» путем улучшения структуры и характеристик только определенных связей, что экономически может быть наиболее выгодным решением. У фон-ней­мановских ВМ таким «узким местом» является канал пересылки данных между ЦП и памятью, и «развязать» его достаточно непросто. Кроме того, ВМ с не­посредственными связями плохо поддаются реконфигурации.

В варианте с общей шиной все устройства вычислительной машины подключе­ны к магистральной шине, служащей единственным трактом для потоков команд, данных и управления. Наличие общей шины существенно упрощает реа­лизацию ВМ, позволяет легко менять состав и конфигурацию машины. Благодаря этим свойствам шинная архитектура получила широкое распространение в мини-и микроЭВМ. Вместе с тем, именно с шиной связан и основной недостаток архи­тектуры: в каждый момент передавать информацию по шине может только одно устройство. Основную нагрузку на шину создают обмены между процессором и памятью, связанные с извлечением из памяти команд и данных и записью в па­мять результатов вычислений. На операции ввода/вывода остается лишь часть пропускной способности шины. Практика показывает, что даже при достаточно быстрой шине для 90% приложений этих остаточных ресурсов обычно не хватает, особенно в случае ввода или вывода больших массивов данных.



В целом следует признать, что при сохранении фон-неймановской концепции последовательного выполнения команд программы шинная архитектура в чистом ее виде оказывается недостаточно эффективной. Более распространена архитек­тура с иерархией шин, где помимо магистральной шины имеется еще несколько дополнительных шин. Они могут обеспечивать непосредственную связь между устройствами с наиболее интенсивным обменом, например процессором и кэш­-памятью. Другой вариант использования дополнительных шин — объединение однотипных устройств ввода/вывода с последующим выходом с дополнительной шины на магистральную. Все эти меры позволяют снизить нагрузку на общую шину и более эффективно расходовать ее пропускную способность.

**Структуры вычислительных систем**

Понятие «вычислительная система» предполагает наличие множества процессо­ров или законченных вычислительных машин, при объединении которых исполь­зуется один из двух подходов.

В вычислительных системах с общей памятью имеется общая основ­ная память, совместно используемая всеми процессорами системы. Связь процессоров с памятью обеспечивается с помощью коммуникационной сети, чаще всего вырождающейся в общую шину. Таким образом, структура ВС с общей памятью аналогична рассмотренной выше архитектуре с общей шиной, в силу чего ей свой­ственны те же недостатки. Применительно к вычислительным системам данная схема имеет дополнительное достоинство: обмен информацией между процессо­рами не связан с дополнительными операциями и обеспечивается за счет доступа к общим областям памяти.

Альтернативный вариант организации — распределенная система, где общая память вообще отсутствует, а каждый процессор обладает собственной локальной памятью. Часто такие системы объединяют отдельные ВМ. Обмен информацией между составляющими системы обеспечивается с помощью коммуни­кационной сети посредством обмена сообщениями.





Подобное построение ВС снимает ограничения, свойственныe для общей шины, но приводит к дополнительным издержкам на пересылку сообщений между про­цессорами или машинами.

**2. Классификация архитектур системы команд**

Системой команд вычислительной машины называют полный перечень команд, которые способна выполнять данная ВМ. В свою очередь, под архитектурой системы команд (АСК) принято определять те средства вычислительной машины, которые видны и доступны программисту. АСК можно рассматривать как линию согласования нужд разработчиков программного обеспечения с возможностями создателей аппаратуры вычислительной машины.

В истории развития вычислительной техники как в зеркале отражаются изменения, происходившие во взглядах разработчиков на перспективность той или иной архитектуры системы команд. Сложившуюся на настоящий момент ситуацию в области АСК иллюстрирует рис. 2.3.

Среди мотивов, чаще всего предопределяющих переход к новому типу АСК, остановимся на двух наиболее существенных. Первый — это состав операций, выполняемых вычислительной машиной, и их сложность. Второй — место хранения операндов, что влияет на количество и длину адресов, указываемых в адресной части команд обработки данных. Именно эти моменты взяты в качестве критериев излагаемых ниже вариантов классификации архитектур системы команд.



Рис. Хронология развития архитектур системы команд

**Классификация по составу и сложности команд**

Современная технология программирования ориентирована на языки высокого уровня (ЯВУ), главная цель которых — облегчить процесс программирования. Переход к ЯВУ, однако, породил серьезную проблему: сложные операторы, характерные для ЯВУ, существенно отличаются от простых машинных операций, реализуемых в большинстве вычислительных машин. Проблема получила название *семантического разрыва,* а ее следствием становится недостаточно эффективное выполнение программ на ВМ. Пытаясь преодолеть семантический разрыв, разработчики вычислительных машин в настоящее время выбирают один из трех подходов и, соответственно, один из трех типов АСК:

* архитектуру с полным набором команд: **CISC** (Complex Instruction Set Computer);
* архитектуру с сокращенным набором команд: **RISC** (Reduced Instruction Set Computer);
* архитектуру с командными словами сверхбольшой длины: **VLIW** (Very Long Instruction Word).

В вычислительных машинах типа CISC проблема семантического разрыва решается за счет расширения системы команд, дополнения ее сложными командами, семантически аналогичными операторам ЯВУ. Основоположником CISC-архитектуры считается компания IBM, которая начала применять данный подход с семейства машин IBM 360 и продолжает его в своих мощных современных универсальных ВМ, таких как IBM ES/9000. Аналогичный подход характерен и для компании Intel в ее микропроцессорах серии 8086 и Pentium.

Для CISC-архитектуры типичны:

* наличие в процессоре сравнительно небольшого числа регистров общего назначения;
* большое количество машинных команд, некоторые из них аппаратно реализуют сложные операторы ЯВУ;
* разнообразие способов адресации операндов;
* множество форматов команд различной разрядности;
* наличие команд, где обработка совмещается с обращением к памяти.

К типу CISC можно отнести практически все ВМ, выпускавшиеся до середины 1980-х годов, и значительную часть производящихся в настоящее время. Рассмотренный способ решения проблемы семантического разрыва вместе с тем ведет к усложнению аппаратуры ВМ, главным образом устройства управления, что, в свою очередь, негативно сказывается на производительности ВМ в целом. Это заставило более внимательно проанализировать программы, получаемые после компиляции с ЯВУ. Был предпринят комплекс исследований, в результате которых обнаружилось, что доля дополнительных команд, эквивалентных операторам ЯВУ, в общем объеме программ не превышает 10-20%, а для некоторых наиболее сложных команд даже 0,2%. В то же время объем аппаратных средств, требуемых для реализации дополнительных команд, возрастает весьма существенно. Так, емкость микропрограммной памяти при поддержании сложных команд может увеличиваться на 60%. Детальный анализ результатов упомянутых исследований привел к серьезному пересмотру традиционных решений, следствием чего стало появление *RISC-архитектуры.* Термин RISC впервые был использован Д. Паттерсоном и Д. Дитцелем в 1980 году. Идея заключается в ограничении списка команд ВМ наиболее часто используемыми простейшими командами, оперирующими данными, размещенными только в регистрах процессорах. Обращение к памяти допускается лишь с помощью специальных команд чтения и записи. Резко уменьшено количество форматов команд и способов указания адресов операндов. Сокращение числа форматов команд и их простота, использование ограниченного количества способов адресации, отделение операций обработки данных от операций обращения к памяти позволяет существенно упростить аппаратные средства ВМ и повысить их быстродействие. RISC-архитектура разрабатывалась таким образом, чтобы уменьшить TВИЧ за счет сокращения CPI и I/. Как следствие, реализация сложных команд за счет последовательности из простых, но быстрых RISC-команд оказывается не менее эффективной, чем аппаратный вариант сложных команд в CISC-архитектуре. Отметим, что в последних микропроцессорах фирмы Intel и AMD широко используются идеи, свойственные RISC-архитектуре, так что многие различия между CISC и RISC постепенно стираются.

Помимо CISC- и RISC-архитектур в общей классификации был упомянут еще один тип АСК — архитектура с командными словами сверхбольшой длины (VLIW). Концепция VLIW базируется на RISC-архитектуре, где несколько простых RISC-команд объединяются в одну сверхдлинную команду и выполняются параллельно. В плане АСК архитектура VLIW сравнительно мало отличается от RISC. Появился лишь дополнительный уровень параллелизма вычислений, в силу чего архитектуру VLIW логичнее адресовать не к вычислительным машинам, а к вычислительным системам. Таблица 2.1 позволяет оценить наиболее существенные различия в архитектурах типа CISC, RISC и VLIW.

Таблица 2.1. Сравнительная оценка CISC-, RISC- и VLIW-архитектур

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристика** | **CISC** | **RISC** | **VLIW** |
| Длина команды | Варьируется  | Единая  | Единая  |
| Расположение нолей и команде | Варьируется  | Неизменное  | Неизменное  |
| Количество регистров | Несколько (часто специализированных) | Много регистров общего назначения | Много регистров общего назначения |
| Доступ к памяти | Может выполняться как часть команд различных типов | Выполняется только специальными командами | Выполняется только специальными командами |

**Классификация по месту хранения операндов**

Количество команд и их сложность, безусловно, являются важнейшими факторами, однако не меньшую роль при выборе АСК играет ответ на вопрос о том, где могут храниться операнды и каким образом к ним осуществляется доступ. С этих позиций различают следующие виды архитектур системы команд:

* стековую;
* аккумуляторную;
* регистровую;
* с выделенным доступом к памяти.

Выбор той или иной архитектуры влияет на принципиальные моменты: сколько адресов будет содержать адресная часть команд, какова будет длина этих адресов, насколько просто будет происходить доступ к операндам и какой, в конечном итоге, будет общая длина команд.

**Стековая архитектура**

*Стеком* называется память, по своей структурной организации отличная от основной памяти ВМ. Принципы построения стековой памяти детально рассматриваются позже, здесь же выделим только те аспекты, которые требуются для пояснения особенностей АСК на базе стека. Стек образует множество логически взаимосвязанных ячеек (рис. 2.4), взаимодействующих по принципу «последним вошел, первым вышел» (L1FO, Last In First Out).



Рис. 2.4. Принцип действия стековой памяти

Верхнюю ячейку называют *вершиной стека*. Для работы со стеком предусмотрены две операции: *push* (проталкивание данных в стек) и *pop* (вытаякивание данных из стека). Запись возможна только в верхнюю ячейку стека, при этом вся хранящаяся в стеке информация предварительно проталкивается на одну позицию вниз. Чтение допустимо также только из вершины стека. Извлеченная информация удаляется из стека, а оставшееся его содержимое продвигается вверх. В вычислительных машинах, где реализована АСК на базе стека (их обычно называют стековыми), операнды перед обработкой помещаются в две верхних ячейки стековой памяти.

**Аккумуляторная архитектура**

Архитектура на базе аккумулятора исторически возникла одной из первых. В ней для хранения одного из операндов арифметической или логической операции в процессоре имеется выделенный регистр — *аккумулятор.* В этот же регистр заносится и результат операции. Поскольку адрес одного из операндов предопределен, в командах обработки достаточно явно указать местоположение только второго операнда. Изначально оба операнда хранятся в основной памяти, и до выполнения операции один из них нужно загрузить в аккумулятор. После выполнения команды обработки результат находится в аккумуляторе и, если он не является операндом для последующей команды, его требуется сохранить в ячейке памяти.

**Регистровая архитектура**

В машинах данного типа процессор включает в себя массив регистров (регистровый файл), известных как регистры общего назначения (РОН). Эти регистры, в каком-то смысле, можно рассматривать как явно управляемый кэш для хранения недавно использовавшихся данных. Размер регистров обычно фиксирован и совпадает с размером машинного слова. К любому регистру можно обратиться, указав его номер. Количество РОН в архитектурах типа CISC обычно невелико (от 8 до 32), и для представления номера конкретного регистра необходимо не более пяти разрядов, благодаря чему в адресной части команд обработки допустимо одновременно указать номера двух, а зачастую и трех регистров (двух регистров операндов и регистра результата). RISC-архитектура предполагает использование существенно большего числа РОН (до нескольких сотен), однако типичная для таких ВМ длина команды (обычно 32 разряда) позволяет определить в команде до трех регистров. Регистровая архитектура допускает расположение операндов в одной из двух запоминающих сред: основной памяти или регистрах. С учетом возможного размещения операндов в рамках регистровых АСК выделяют три подвида команд обработки: регистр-регистр; регистр-память; память-память.

В варианте «регистр-регистр» операнды могут находиться только в регистрах. В них же засылается и результат. Подтип «регистр-память» предполагает, что од и и из операндов размещается в регистре, а второй в основной памяти. Результат обычно замещает один из операндов. В командах типа «память-память» оба операнда хранятся в основной памяти. Результат заносится в память. Каждому из вариантов свойственны свои достоинства и недостатки (табл. 2.4).

В выражениях вида *(т, п),* приведенных в первом столбце таблицы, *т* означает количество операндов, хранящихся в основной памяти, а n — общее число операндов в команде арифметической или логической обработки.

Вариант «регистр-регистр» является основным в вычислительных машинах типа RISC. Команды типа «регистр-память» характерны для CISC-машин. Наконец, вариант «память-память» считается неэффективным, хотя и остается в наиболее сложных моделях машин класса CISC.

|  |  |  |
| --- | --- | --- |
| **Вариант** | **Достоинства** | **Недостатки** |
| Регистр-регистр (0,3) | Простота реализации, фиксированная длина команд, простая модель формирования объектного кода при компиляции программ, возможность выполнения всех команд за одинаковое количество тактов | Большая длина объектного кода, из-за фиксированной длины команд часть разрядов в коротких командах не используется |
| Регистр-память (1.2) | Данные могут быть доступны без загрузки в регистры процессора, простота кодирования команд, объектный код получается достаточно компактным | Потеря одного из операндов при записи результата, длинное поле адреса памяти в коде команды сокращает место под помер регистра, что ограничивает общее число РОН. СР1 зависит от места размещения операнда |
| Память-память (3,3) | Компактность объектного кода, малая потребность в регистрах для хранения промежуточных данных | Разнообразие форматов команд и времени их исполнения, низкое быстродействие из-за обращения к памяти |

**Архитектура с выделенным доступом к памяти**

В архитектуре с выделенным доступом к памяти обращение к основной памяти возможно только с помощью двух специальных команд: *load* и *store.* В английской транскрипции данную архитектуру называют Load/Store architecture. Команда *load* (загрузка) обеспечивает считывание значения из основной памяти и занесение его в регистр процессора (в команде обычно указывается адрес ячейки памяти и номер регистра). Пересылка информации в противоположном направлении производится командой *store* (сохранение). Операнды во всех командах обработки информации могут находиться только в регистрах процессора (чаще всего в регистрах общего назначения). Результат операции также заносится в регистр. В архитектуре отсутствуют команды обработки, допускающие прямое обращение к основной памяти. Допускается наличие в АСК ограниченного числа команд, где операнд является частью кода команды.

АСК с выделенным доступом к памяти характерна для всех вычислительных машин с RISC-архитектурой. Команды в таких ВМ, как правило, имеют длину 32 бита и трехадресный формат. В качестве примеров вычислительных машин с выделенным доступом к памяти можно отметить HP PA-RISC, IBM RS/6000, Sun SPARC, MIPS R4000, DEC Alpha и т. д. К достоинствам АСК следует отнести простоту декодирования и исполнения команды.

**Контрольные вопросы**

1. Какие бывают типы архитектур ЭВМ?
2. В чем заключается принцип открытой архитектуры ЭВМ?
3. Назовите основные принципы фон Неймана?
4. В чем заключается принцип двоичного кодирования?
5. В чем заключается принцип программного управления?
6. В чем заключается принцип однородности памяти?
7. В чем заключается принцип адресности?
8. Назовите основные типы структур вычислительных машин и систем