Архитектуры ВС

27 сентября 2014 г.

10:11

**Архитектура** — это описание вычислительной системы на некотором общем уровне, включающем систему команд, средств пользовательского интерфейса, организацию памяти, операции ввода/вывод и управление.

Реализация архитектуры может отличаться на уровне организации подсистем. Те или иные организации архитектуры могут отличаться, как по сложности, так и по стоимости. Детали реализации, невидимые для пользователя, например кэш-память, не оказывают влияние на архитектуру.

Общность архитектур разных вычислительных систем обеспечивает их совместимость с точки зрения пользователя. В контексте разработки вычислительных систем (ВС) и проектировании их аппаратных средств термин «архитектура» используют для описания действия, организации и конфигурации основных функциональных блоков. Описание архитектуры должно включать разъяснение принципа действия и диапазон возможностей любого узла ВС.

Современное состояние и тенденции развития вычислительной техники

1. Создание унифицированных модульных архитектур, позволяющих поэтапно расширять вычислительные мощности, при этом акцент делается на магистрально-модульном построении, когда к общей магистрали наряду с общими системами подключаются подсистемы.
2. Ускорение развития систем параллельной обработки, путем создания новых многопроцессорных систем и машинных комплексов.
3. Постепенный переход на ВС распределённой обработки и децентрализованное управление ими.
4. Широкое внедрение архитектуры ВС типа RISC (архитектура с сокращённым набором команд). Данная архитектура обеспечивает более простые машинные команды, выполняемые за один такт.
5. Создание однокристальных компьютеров, ориентированных на один язык программирования.
6. Создание технологий программирования, отличающихся производительностью и надёжностью.

Сначала предполагалось, что все устройства работают последовательно, позднее допускалась одновременная работа двух устройств. Структурный принцип предполагает, что ВС состоит из вышеперечисленных устройств (рис. 1), а программы и данные располагаются в памяти вместе. В ЦП устройство управления определяет какие действия необходимо выполнить АЛУ. Такое определение действий происходит путем определения адреса и считывания команд из выбранной ячейки оперативной памяти. ЦП имеет память в виде накопителей и регистров. Программа фон-Неймановской модели состоит из набора команд, которые проверяются одна за другой и выполняются. В архитектуре современных ПК сохранены и детализированы основные принципы фон-Неймановской модели вычисления. При этом современные ПК имеет следующие системные особенности:

* процессор, совместимый с x86-архитектурой;
* единую систему распределения пространства памяти;
* унифицированное распределение адресов пространства ввода-вывода с фиксированным положением портов;
* систему аппаратных прерываний, позволяющую периферийным устройства формировать сигнал, по которому процессор определённым образом реагирует на события связанные с этим устройством;
* набор системных устройств и интерфейсов ввода-вывода;
* унифицированные по конструктивно-механическим и электрическим параметрам типы расширений (ISA, PCI, PCI-Express, AGP, USB).

Виды ВС и способы их реализации

Любая ВС включает в себя технические средства и ПО, ориентированные на решение определенных задач. Такая ориентация может осуществляться как за счет программных средств и возможностей операционной системы, так и за счет специализированных ВС:

1. ВС, ориентированные на числовую обработку:
   1. персональные ЭВМ (30-500 гигафлопс (операций в сек. с плавающей точкой));
   2. специализированный ЭВМ для числовой обработки (ВС управления радиолокационными станциями, суперкомпьютеры (Tianhe-2, 33.86 петафлопс; Ломоносов, 408 терафлопс), матричные процессоры).
2. Ориентированные на обработку сигналов:
   1. процессоры быстрого преобразования Фурье;
   2. универсальные (сигнальные);
   3. специализированные БИС для цифровой обработки.
3. ВС ориентированные на обработку символов (RISC- и PROLOG- машины).
4. Ориентированные на обработку больших объемов данных (машины БД, баз знаний).
5. Ориентированные на обработку в сетях связи.

Внедрение специализированных процессоров в различных предметных областях привело к появлению распределенных ВС.

Классификация ВС по Флинту

Различные способы реализаций параллельных вычислений можно представить как способы организации одновременного воздействия одного или нескольких потоков команд на один или несколько потоков данных.

Все ВС могут быть разбиты на четыре класса:

1. Система с одиночным потоком команд и данных (SISD).
2. Система с множественным потоком команд и одиночным потоком данных (MISD).
3. Одиночный поток команд, множественный поток данных (SIMD). В этой системе одиночный поток команд, но они векторные, т. е. инициируют множество операций. Каждый элемент вектора рассматривается как компонент отдельного потока данных. Т. о. одновременно обрабатывается множество потоков данных. Одна команда может выполняться над массивом данных. Широкого распространения не получила.
4. Множественные потоки команд и данных (MIMD). Данная система предполагает наличие в ВС нескольких процессорных устройств и, следовательно, несколько потоков данных. Примерами таких устройств являются мультипроцессорные матрицы.

Принципы функционирования ЦП

**Машинная команда** — это описание операции, которую должна выполнять ЭВМ. Содержит выполняемые операции, указание по определению операндов и их адресов, типы размещения результатов выполненной операции.

Команды часто распределяются на арифметические, логические, команды пересылки данных, команды ввода-вывода в порты и команды вызова прерывания.

Совокупность выполняемых машиной операций называется **системой команд**.

**Машинное слово** — это вектор битов, используемых аппаратной частью ЭВМ как единое целое.

**Операнды** — это отчасти машинные команды, которые определяют объекты, над которыми выполняются операции.

**Регистр** — это совокупность элементов, принимающих значение 1 или 0, используемых для хранения информации. Часто регистры имеют тот же размер, что и машинные слова.

**Счётчик команд (IP)** — регистр процессора, содержащий адрес текущей выполняемой команды.

Основные регистры:

* AX — аккумулятор;
* BX — базовый;
* CX — счётчик;
* DX — данные.

Сегментные регистры:

* CS — указатель на кодовый сегмент. Пара CS:IP однозначно определяет реальный физический адрес следующей выполняемой инструкции.
* DS — данные;
* SS — стек;
* ES — дополнительный.

EAX (аккумулятор), BAX (базовый), ECX (счётчик), EDX (данных), EBP (указателя базы), ESP (указатель на вершину стека), ESI (индексный), EDI (индексный) — регистры общего назначения, которые можно использовать. При выполнении текущей инструкции процессор автоматически изменяет значение в регистре IP, в результате чего регистровая память CS:IP всегда указывает на следующую, подлежащую исполнению память.

**Регистр флага** — это бит, принимающий значение единицы, если выполнено некоторое условие, и значение ноль в противном случае. Процессор имеет регистр флагов, содержащий набор флагов, отражающий текущее состояние процессора.

**Сегментом** называется область, которая начинается с адреса кратного 16.

При выполнении программы определяется три главных сегмента:

1. **Сегмент кода** — содержит машинные команды, из которых обычно первая исполняемая команда находится в начале этого сегмента и операционная система передаёт управление по адресу данного сегмента для выполнения программы.
2. **Сегмент данных** — содержит данные, переменные, массивы, константы, необходимые для работы программы. Начальный адрес сегмента расположен в регистре DS.
3. **Сегмент стека** — содержит адреса возврата, как для программы при возврате в ОС, так и для вызовов подпрограмм при возврате в главную программу. Адрес данного сегмента в регистре SS.
4. **Дополнительный сегмент** — используется в специальных случаях.

Стек так же можно использовать для временного хранения данных. Регистр указателя стека (SP) постоянно указывает на вершину стека. Данные помещаются в стек и извлекаются из него с помощью PUSH и POP. Увеличение стека происходит по направлению оставшихся ячеек памяти, к младшим.

Таким образом при помещении данных в стек, содержимое регистра SP уменьшается, а при извлечении данных увеличивается.

Основные команды

1. Команды пересылки данных:
   1. MOV <операнд-назначение> <операнд-источник>
   2. XCHG <регистры, которые нужно поменять местами>
   3. MUL — умножает регистр AX
   4. DIV — делит регистр AX
   5. ADD <первый операнд> <второй операнд>
   6. SUB <первый операнд> <второй операнд>

Цикл выполнения программы

Это время выполнения для одной машинной команды. Цикл выполнения команды реализуется за несколько машинных циклов, точное число которых зависит от сложности команды и, как правило, равно числу обращений процессора к памяти и устройствам ввода/вывода.

Выбор и выполнение программы:

1. Выбрать команду из оперативной памяти и поместить её в регистр.
2. Выбрать операнды из памяти, если в команде используется операнд расположенный в оперативной памяти, то блок управления начинает её выборку из памяти.
3. Увеличение IP.
4. Выполнение команды.

Различают следующие основные **частоты системной шины:**

1. Системная (Host Bus Clock) — внешняя частота шины процессора, это частота является опорной для всех остальных компонентов.
2. Внутренняя частота процессора (CPU Clock) — на котором работает его вычислительное ядро.

Отношение внутренней частоты к внешней называют **коэффициентом умножения.** Частота шины PCI обеспечивает соединение внешней частоты процессора на различные коэффициенты 2 или 3.

Частота шины ISA Bus Clock должна быть близка к 8 MHz, на ней работают все приёмно-передающие элементы.

Главной отличительной способность многопроцессорной системы является её производительность, т. е. количество операций, выполняемых в единицу времени. Различают пиковую и реальную производительности. Под пиковой понимают величину равную произведению пиковой произвольности одного процессора на число процессоров в данной машине. Это характеристика является базовой, по которой производят сравнение вычислительных систем. Чем больше пиковая производительность, тем быстрее пользователь может решить свою задачу. Пиковая производительность — величина теоретическая и недостижимая при запуске конкретного приложения.

Существует два способа её оценки:

1. Опирается на число команд выполняемых за единицу времени. Единицей измерения является MIPS (Millions instructions per second). Данная характеристика даёт самое общее представление о производительности.
2. Заключается в определении числа вещественных операций (FLOPS). Такой способ является более приемлемым для пользователя. Пользуясь этой характеристикой пользователь может получить оценку времени, однако пиковая производительность получается только в идеальных условиях, то есть при отсутствии конфликтов и при равномерной загрузке всех устройств. В реальных условиях на выполнение конкретной программы влияют такие аппаратно-программные особенности как:
   * особенности структуры процессора;
   * системы команд;
   * состав устройств;
   * реализация ввода-вывода;
   * эффективность работы компилятора;
   * время взаимодействия с памятью.

В большинстве ПК используются многоуровневая иерархическая память. В качестве уровней используется регистровая память, основная оперативная память, кэш-память, жёсткие диски, твердотельные накопители и пр.

Выдерживается следующий принцип: при повышении уровня памяти скорость обработки, должна увеличиваться, а объём уровня уменьшаться. Эффективность использования такой иерархии достигается за счёт хранения часто использованных данных памяти верхнего уровня, время доступа к которой минимально.

Тесты используемые при оценки производительности

Для того, чтобы оценить эффективность работы вычислительной системы на реальных задачах был разработан фиксированный набор тестов. Наиболее известным из них является LINPACK. Тест состоит в решении системы линейных уравнений с помощью LU-факторизации. Основное время затрачивается на решение векторных операций (сложение и умножение). Производительность определяется количеством «полезных» вычислительных операций и выражается в гигафлопсах. Автор данного теста Джек Донгарра. Результаты теста используются при составлении списка топ-500 самых мощных компьютеров мира.

В настоящее время большое распространение получили тестовые программы, взятые из разных предметных областей, и представляющие собой реальные промышленные приложения. Такие тесты позволяют оценить производительность действительно в реальных задачах. И получить наиболее полное представление об эффективности работы компьютера с конкретными приложениями.

Пакеты тестов:

1. NBP. Они состоят из восьми различных программ для определения производительности параллельных компьютеров.
2. PERFECT. Он представляет собой комплект из 13 прикладных FORTRAN-программ, представляющих четыре типа вычислительных задач: аэро- и гидродинамики, моделирование химических и физических процессов, инженерное проектирование и обработка сигналов.
3. Matrix Multiply (MM) — этот тест содержит 9 различных программ умножения матриц размером 500 × 500. Им оценивается работа кэш-памяти и уровень оптимизации компилятора.
4. SLALOM — тест производительности для суперкомпьютеров. Оценивает объем вычислений, которые может произвести компьютер за одну минуту.
5. Stanford — состоит из 8 целочисленных тестов (умножение матриц, сортировка тремя методами, перестановки, ханойские башни, расстановка восьми ферзей, головоломка, быстрое преобразование Фурье).
6. STREAM -- тест для работы с памятью. Основан на измерении времени выполнения больших векторных операций: копирование в памяти, умножение на константу, умножение и сложение.

Адресация данных и команд в ассемблере

Любая машинная команда — это машинный код, который определяет первую операцию, указывает на данные. В адресной части команды хранится адресный код. В большинстве случаев фактическое обращение к данным происходит по физическому адресу. Обычно физической адрес не совпадает с адресным полем команды, но зависит от него. В общем случае происходит преобразование из адресного кода в физический. Способы адресации являются одним из основных архитектурных признаков. В настоящее время известно более двух десятков различных способов адресации и их модификации. Все способы делятся на две большие группы:

1. Прямые — либо накопительный адрес операнда, либо сам операнд находится непосредственно по адресному коду без всякого преобразования.
2. Непрямые — требуют выполнения процедур формирования физического адреса по адресному коду для этого ЭВМ.

**Понятие прерывания**

Прерывание -- это реакция вычислительной системы на некоторое асинхронное событие, которое заключается в том, что выполнение текущей программы временно прекращается и выполняется некоторая другая подпрограмма (обработка прерывания), после чего чаще всего продолжается выполнение прерванной программы.

Прерывания:

1. Внутренние
   1. Программные
   2. От схем контроля
2. Внешние
   1. От клавиатуры
   2. От таймера
   3. От HDD
   4. ...

Программные прерывания вызываются командой INT и используется для передачи управления из прикладных программ в ОС или БИОС. Прерывание от схем контроля возникает в тех случаях, когда невозможна корректная работа процессора, например прерывание при нарушении защиты памяти.

Внешние прерывания возникают по требованию внешних устройств. В тех случаях, когда им требуется обслуживание. Исключением из этих правил является системный таймер. Он используется для отслеживания системного времени и интервалов времени при работе с внешними устройствами. Вся ОС основана на прерываниях.

Для обеспечении возможности работы с прерываниями аппаратная часть компьютера обязательно содержит соответствующие средства поддержки прерываний. В реальном режиме работы такими средствами являются:

1. Опрос процессором входной линии запроса прерывания перед исполнением очередной команды.
2. Вектора прерываний.
3. Команды INT, INTO, IRET.
4. Флаг разрешения прерываний -- IF.
5. Контроллер прерываний.
6. Схемы выработки сигналов прерываний во внешних устройствах.

**Цикл работы процессора:**

1. Опрос входного сигнала запроса на прерывание.
2. Выборка команды:
   1. Чтение кода команды.
   2. Получение адреса следующей команды.
3. Выполнение команды. Выборка операндов, выполнение операции. Запись результатов.

Векторы прерываний в реальном режиме работы располагается в самом начале оперативной памяти и занимает первые 2^10 байт. Всего векторов 256, от 0 до 255. За каждым вектором закреплены свои прерывания. По своему содержанию вектор прерывания -- это два слова содержащий адрес обработчика прерывания. В старшем слове располагается сегментная часть адреса, в младшем -- смещение относительно сегментной части адреса. Команда INT вызывает прерывание с вектором, указанным в качестве аргумента команды, например INT 21h передает управление в DOS. INT 13h, INT 17h в БИОС. Команда IRET используется для возврата управления из обработчика прерываний назад в прерванную программу. Флаг IF в регистре флагов процессора определяет, будет ли процессор воспринимать запрос на прерывание от внешних устройств. IF == 0 ? не воспринимает.

**Состав и размещение обработчиков прерываний**

ПО системы прерываний образует совокупность всех обработчиков прерываний. Обработчиком программных прерываний является ОС, точнее та её часть, которая занимается выполнением запросов прикладных программ.

Прерывание от схем контроля так же обрабатываются ОС. Типичным их действием является принудительное аварийное завершение программы, во время исполнения которой возникло прерывание, сопровождающееся сообщением об ошибке.

Прерывание от внешних устройств обрабатываются процедурами, физически расположенными в драйверах внешних устройств. Упрощенный алгоритм функционирования драйвера устройства ввода-вывода, работающего по прерываниям выглядит следующим образом:

1. Вход в драйвер.

Когда прикладная программа передает запрос на ввод-вывод операционной системе, та, в свою очередь, формирует запрос соответствующему драйверу: заполняет структуру данных, описывающую запрос и передает управление драйверу.

1. Инициализация выполнения запроса на внешние устройства.

Получив управление драйвер анализирует запрос и подает команду внешнему устройству (помещает соответствующие коды в регистры этого устройства). С этого момента внешнее устройство начинает выполнять операцию в соответствии с полученной командой.

1. Разрешение прерывания от внешнего устройства.

В общем случае требует разрешения прерывания на внешнем устройстве (записи соответствующего кода в регистр управления) и разрешения прохождения запроса.

1. Выход из драйвера.

После разрешения прерывания драйвер передает управление в операционную систему, чтобы та могла загрузить процессор полезной работой. Например выполнением одной из прикладных программ. Пока внешнее устройство выполняет запущенную операцию. Далее, когда внешние устройства заканчивают выполнение операции, выполняющаяся программа прерывается и управление попадает в обработчик прерывания драйвера. Он проверяет успешность выполнения операции внешним устройством, и если операция завершилась неуспешно передаёт управление в ОС с признаком ошибки. В случае успешного завершения операции выполняется передача данных между внешним устройством и областью памяти прикладной программы, выдавшей запрос. После этого драйвер проверяет объем переданных данных. Если он не совпадает с запрошенным объёмом, подает следующую команду внешнему устройству, разрешает прерывание и возвращает управление назад в прерванную программу. Когда очередная команда будет выполнена, тогда произойдет прерывания и описанный процесс повторится. Однажды обработчик прерываний обнаружит, что запрос выполнен (переданы все данные, предписанные запросом). После чего он запретит прерывания от внешнего устройства и передаст управление в ОС с признаком успешного завершения процесса выполнения запроса.

...В языках высокого уровня стиль вызова задается в описании процедур и функций. В ассемблерных программах стиль вызова задается директивой:

A proc pascal, near

A endp

B proc c, far

B endp

В Сях стек очищает вызывающая процедура, а в паскале вызываемая.

Вопрос согласования типов аргументов в Ассемблерных программах решается путем указания их размерностей при перечисление в директиве proc

A proc c, far, b: byte, c: word, d: dword.

Также можно использовать директиву arg, которая следует за заголовком процедуры.

Директива arg позволяет не только описывать аргументы, но и присваивать значение их длины некоторому имени:

arg g: dword, f: word, = h

h получит значение равное 6.

При передаче параметров по ссылке в стеке размещаются их адреса, размерность которых зависит от используемой модели памяти. Если модель памяти предполагает один сегмент данных, то в стек помещается только смешение до аргумента. Если имеется множество сегментов данных -- в стеке для каждого аргумента, передаваемого по ссылке, будет размещен адрес, состоящий из двух частей: смещение и сегментные части.

Поскольку модульное программирование предполагает раздельную компиляцию модулей, неизбежно появляются неопределённые имена, приводящие к ошибкам компиляции.

# Взаимодействие программ с операционной системой Виндоус

На сегодняшний день наиболее распространенными являются консольные и оконные Виндоус-приложения. Консольные Виндоус-приложения построены так же, как и 16-разрядные приложения МС-ДОС. При их запуске внешне происходит то же самое: открывается окно и пр.

Следующим отличием является то, что консольные приложения используют сплошную (flat) модель памяти, в которой отсутствует понятие сегмента. Третьим отличием являются запросы к ОС, которые выполняются не с помощью программных прерываний, как в ДОС, а в виде вызова подпрограмм (АПИ-функций).

Оконные приложения существенно отличаются от консольных. Эти отличия связаны с использованием средств ОС для работы с окнами. В оконных приложениях окно является самостоятельным объектом ОС и отделено от самой программы.

Воздействие пользователя на окно является событиями, которые обрабатываются операционной системой и для каждого события формируется сообщение, описывающее это событие. Все сообщения выстраиваются в очередь сообщений, находящуюся в ведении ОС.

Задачей прикладной программы является получение сообщений из системной очереди с помощью функции GetMessage и передачи из через ОС оконной процедуре … (DispatchMessage). Оконная процедура выполняет все действия необходимые для обработки действий пользователя (события).

Для того чтобы приложение получило своё окно нужно последовательно вызвать функции CreateWindow и ShowWindow.

# МАКРО-средства ассемблера

МАКРО-средства — это совокупность конструкций языка, заставляющих транслятор генерировать или модифицировать исходный текст программы.

В Си и Си++ существует понятие препроцессора.

С учётом возможности использования макро-средств ассемблерный транслятор можно рассматривать как совокупность двух взаимосвязанных модулей: макро-процессора и транслятора. Макро-процессор — это модуль, который выискивает в тексте программы макро-средства и соответствующим образом модифицирует исходный текст программы. Модифицированный текст далее обрабатывается транслятором и в итоге получается объектный код программы.

Примеры макро-средств: директивы proc и local.

Все макро-средства ассемблера можно поделить на четыре группы:

1. Макро-команды
2. Блоки повторений
3. Блоки условной трансляции
4. Вспомогательные директивы

Механизм макрокоманд является наиболее используемым из макро-средств. Он позволяет при разработке программы использовать заранее написанные фрагменты текста, имеющие собственные имена. Макрокоманды можно рассматривать как аналогию процедур, но это не совсем корректно. Отличие заключается в том, что выполнение макрокоманд не в процессе выполнения программы, а во время трансляции и сводится в подстановке поименованного фрагмента текста в исходный текст программы. Определения макрокоманды (макроопределение) -- это описание фрагмента текста и присваивание ему некоторого имени.

Используются директивы macro, endm.

Синтаксис macro:

-> имя -> macro -> [список аргументов] ->

Структура macro:

name macro a, b, c

... тело макроопределения ...

endm

name -- имя,

a, b, c -- аргументы

Тело макроопределения -- это любой синтаксически правильный с точки зрения ассемблера текст, в котором используются формальные аргументы.

Определение макрокоманды в тексте должно быть размещено до первого вызова этой макрокоманды.

Примерчик для DOS:

exit macro returncode

MOV AH, 4Ch

MOV AL, returncode

INT 21h

endm

Вызов: exit 0

Вызов макрокоманды -- это указание имени макрокоманды с последующим перечислением фактических аргументов.

Расширение макрокоманды (или макрорасширение) -- это текст подставляемый макропроцессором вместо макрокоманды. По сути макрорасширение -- это тело макроопределения в котором формальные аргументы заменены на фактические, указанные в макрокоманде.

Запись в аргументах во a = bx позволяет задать значение формального аргумента по умолчанию, если соответствующий фактический аргумент в вызове опущен.

При использовании макрокоманд часто возникает вопрос дублирования меток, имеющихся в теле макроопределения и появляющихся в расширениях макрокоманд. Для решения этого вопроса нужно выносить имена меток в список директив...

Директивы REPT и WHILE

При использовании директивы WHILE макрогенератор транслятора будет повторять последовательность строк, пока выражение истинно. Директива REPT будет повторять строки столько раз, сколько определенно в выражении. Отличие REPT от WHILE в том, что REPT уменьшает значение..

**Директива IRP**

IRP формальный\_аргумент, строка\_символов\_1, ..., строка\_символов\_N

последовательность\_строк

Выполняет столько раз, сколько смогут подставится строки символов вместо аргумента.

Директива IRPC

IRPC формальный\_аргумент, строка\_символов\_1, ..., строка\_символов\_N

последовательность\_строк