Архитектура ЭВМ

1 Введение

- Жёсткая архитектура (данные и программные код прошиты в железе и защищены от записи)
- Архитектура с программным управлением. Впервые описал в 1943 году Тьюринг в виде машины Тьюринга.

1.1 Понятие операционной системы

- Расширение архитектуры ЭВМ
- Разделение ресурсов (во времени, например принтер; в пространстве, например RAM, жесткий диск)

1.2 Принципы фон-Неймана

1.2.1 Двоичное кодирование

Все данные представлены в виде двоичных чисел.

Система счисление реализуется регистрами, состоящие из набора триггеров. Если в регистре n-разрядов, то $m = nx$, где x - основание системы счисления, а m - общее число состояний (или выходов).
Из $m = nx \Rightarrow n = m/x$, с помощью n разрядов можно закодировать алфавит мощностью $N = X^n = X^(m/x)$. Если представить N, как функцию $f(X), m = const$, то наиболее оптимальная система счисление X будет являться точкой максимума этой функции, которая равна ближайшему целому числу от $e = 2,71\ldots$, т.е. система счисления с основанием 3. Но, так как реализация троичной с.с. довольно затруднительна, было принято решение кодировать данные в двоичной системе счисления.

Алсо, существует троичная ЭВМ - Сетунь, разработанная в МГУ.

1 байт = 8 битам. Причины выбора именно такого количества битов:

- Использование в IBM кодирования BCD (binary-coded decimal)
- Все основные символы можно закодировать с помощью 8 битов (Всего 255 символов)
- Ближайшая удобная степень числа 2

Минимально адресуемая область памяти - машинное слово (word), которое состоит из двух байтов. В архитектуре intel байты машинного слова при передаче и хранении идут в обратном порядке (т.е. сначала старший байт, потом младший). При передаче данных через сеть машинного слова, байты расположены в прямом порядке.
Передача по сети:
$$\left| \overbrace{0000001}^{1} \middle| \overbrace{0^7 0^6 0^5 0^4 0^3 0^2 0^1 0^0}^{0} \right|$$
Расположение байтов машинного слова в архитектуре Intel:
$$\left| \overbrace{0000001}^{0} \middle| \overbrace{0000000}^{1} \right|$$

1.2.2 Программное управление

Central processor unit (CPU) состоит из арифметико-логического устройства (АЛУ) и управляющего устройства (УУ), который содержит специальный регистр $Instructionposition(IP)$, в котором хранится адрес команды, подлежащей выборке из оперативной памяти (Random access memory (RAM). CPU соединен с оперативной памятью с помощью *системной шины*.
Цикл CPU:

1. Извлечение команды по адресу в IP
2. Декодирование команды

3. Выполнение

4. Переход к следующей команде

Системная шина (bus) - физически, провода, соединяющие CPU, RAM и прочие устройства ЭВМ. Существует отдельно шина адреса, шина данных и шина управления. Когда говорят о разрядности архитектуры, то имеют ввиду разрядность шины данных, т.е. число битов, используемых для кодирования данных, которые можно передать по системной шине.

1.2.3 Однородность памяти

Программы и данные хранятся в одной и той же памяти. Поэтому ЭВМ не различает, что хранится в данной ячейке памяти — число, текст или команда. Над командами можно выполнять такие же действия, как и над данными.

1.2.4 Адресуемость памяти

Структурно основная память состоит из пронумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка. Отсюда следует возможность давать имена областям памяти, так, чтобы к хранящимся в них значениям можно было бы впоследствии обращаться или менять их в процессе выполнения программы с использованием присвоенных имен.

Часть I
Операционные системы

2 Этапы развития ОС

1945 - 1953 г.г. ЭВМ I поколения.
1995 - 1965 г.г. Появление транзисторов, перфокарт, операционных систем пакетной обработки.
1965 - 1980 г.г. Появление интегральных схем, среди О.С. появляется многозадачность (аппаратное разделение памяти), первые попытки создать универсальный компьютер (IBM разработала совместимую между собой серию компьютеров разной мощности: IBM System 360/OS 360). Разделение времени.

В MIT разработана OS TSS, поддерживающая разделение времени. Совместно с MIT, Bell labs, General Electric была разработна ОС Multics (?), поддерживающая безотказную работу и предоставляющая каждому процессу независимую память.

Кенни Томпсон и Денис Ритчи в 1969 году разрабатывают ОС UNICS, в 1971 году выходит версия UNICS с открытым исходным кодом - UNIX. UNIX v.3 содержит компилятор C, в UNIX v.4 ядро переписано на язык C, в UNIX v.5 все программы переписаны на C. Далее UNIX разделился на AT&T Unix System 5 и BSD (Berkley System Distribution).

1980 год. IBM PC, PC/T, PC/AT, OS CP/M (Digital Research). Билл Гейтс создает DOS. Дуглас Элельбарт изобретает манипулятор "мышь графический интерфейс пользователя. Стив Джобс Appale Macintosh, Mac OS - модификация BSD. В 1991 году Линукс Торвардс создает на базе Unix операционную систему Linux (только ядро), распространяемую бесплатно.

POSIX Для стандартизации взаимодействий с аппаратурой операционных систем IEEE был разработан стандарт POSIX.

3 Классификация ОС

1. Однозадачные

2. Разделение времени (многозадачность)

3. О.С. реального времени

Структура ядра

1. Монолитное

2. Микроядро (MacOS)

3. Экзоядро

4 Процессы

4.1 Основные понятия

Понятие задачи Задача - это совокупность программ, данных и инструкций по запуску.

Понятие программы Именованная область на диске, обладающая собственным состоянием.

Процесс Это область памяти RAM, в который находится все необходимое для выполнения задачи.
В ядре ОС существует менеджер процессов, выполняющий следующие задачи:

1. Загрузка
2. Выделение памяти
3. Настройка адресов (relocation)
4. Выгрузка
5. Освобождение памяти
6. Освобождение ресурсов

Структура памяти процесса: 1) код, 2) константы + переменные, 3) Свободное место, 4) стек, 5) куча - свободная не распределенная память (heap)

Представление строк в памяти

1. С-строка, завершающаяся символом с кодом 0.
2. В первом байте строки указана длина строки (ограничение в 255 символов)

Глобальные переменные

1. Инициализированные
2. Под них не выделена память на жестком диске

Локальные переменные

1. Размещаются в стеке.

Стек Структура данных, поддерживающая две команды: pop (взять с вершины стека), push (положить на вершину стека). В памяти растет в сторону области памяти с константами и переменными. Стек реализуется регистром процессора, который указывает на вершину стека.

Соглашение вызова Соглашение о размещение параметров функции в стеке перед вызовом этой функции.

4.2 Работа с кучей

В языке С функции *malloc*() и *free*(), в C++ *new* и *delete*. Все операции выделения и освобождения памяти работают с ссылками - переменные содержащие адрес памяти.

Алгоритмы выделения памяти

1. Первый подходящий
2. Наилучший подходящий

Информация о свободных и занятых блоках кучи хранится в виде таблицы: адрес (начальный) | размер | статус
При освобождении памяти необходимо объединить смежные свободные блоки.

Типичные проблемы

1. Утечки памяти
2. Использование указателей на освобожденную область памяти

Одним из вариантов решения является использование сборщиков мусора или умных указателей.

5 Компиляция программ

Файлы исходного кода передается препроцессору, который возвращает единый файл с исходным кодом (code.cpp, без ограничения общности). Файл code.cpp передается на вход компилятору, возвращающий текст на ассемблере, который преобразуется в объектный файл (program.obj - Windows, program.o - Unix), содержащий машинный код, константы и переменные, символьная информация и ссылки на другие библиотеки. Объектный файл и библиотеки передаются сборщику (линковщику), результатом работы которого является исполняющий файл.

Структура исполняющего файла (.exe - windows)

1. Информация для настройки программы в RAM: таблица релокации (список смещений от начала файла команд, требующих настройки адресов)

6 Поток

Потоки могут быть реализованы, только если операционной системой поддерживается многозадачность.

Понятие потока Поток - это процесс + состояние процесса + IP + использованные ресурсы. Процесс может находиться в одном из трех состояний: выполнение $\leftarrow$ ожидание $\leftarrow$ готовность $\leftrightarrow$ выполнение.

Прерывания В RAM есть таблица векторов прерываний (адресов процедур обработки прерываний). Источником регулярных внешних прерываний в ОС является таймер.

6.1 Менеджер потоков

Задачи

1. Разблокировка потока (из состояние блокирован в состояние готов)
2. Переключение потоков (из состояние готов в состояние работает)

6.2 Типы потоков

1. Пакетные
2. Интерактивные
3. Потоки реального времени

6.3 Алгоритмы планирования интерактивных потоков

Кооперативная многозадачность (потоки переключаются только после того как работающий поток вернет управление системе).
Вытесняющая многозадачность. Реализуется с помощью циклических очередей потоков или очередей с приоритетами.

Квант времени Количество времени выделяемое каждому потоку, пока его не переведут из состояния выполнения в состояние готов. Приоритет = f(остаток кванта). Чем ниже приоритет, тем реже переключаются на этот поток.

Амортизация приоритетов Необходимо время от времени восстанавливать приоритет на начальный уровень.

7 Задача синхронизации. Взаимодействие процессов.

Thread - нить Поток в рамках одной программы, работающий параллельно с другими нитями.

Транзакция Набор операций, выполняемых как единое целое, называют транзакцией.

```
int n = 0; // Share variable
f1() {
        while(true) { ++n; ++n; }
}

f2() {
        while(true) { if(n % 2 != 0) print(n); }
}

int main() {
        new thread(f1).start();
        new thread(f2).start();
}
```

Раздел программы, работающей с общими данными или зависящей от них называют критической секцией.

Проблемы Два потока не должны одновременной находится в своих критических секциях. Поток должен выходить из критической секции. Поток вне критической секции не должен мешать другим потокам войти в свою критическую секцию.

Решения

1. Запретить прерывания (возможно только внутри ядра).
2. Блокировка общих данных на время выполнения критической секции.

8 Управление памятью

С точки зрения программы работать на прямую с памятью невозможно. Для программы существует некоторая *модель памяти* (абстракция), выделяют следующие виды моделей: отсутствие модели, виртуальная память.

8.1 Отсутствие модели

Работа ведется на прямую с физической памятью.

Возможные проблемы:

1. Разное расположение в памяти (решение: релокация при загрузке).
2. Средства защиты

   Решение: использование базового регистра (адрес начала секции программного кода, данных, стека) и ограничительного регистра (адрес конца секции программного кода, данных, стека) (в intel не используется).
3. Нехватка памяти

   Решение, если не хватает памяти для запуска нескольких программ одновременно: программу, которая не выполняется - выгрузить (свопинг)

   Решение, если не хватает памяти для запуска одной программы: загружать программу по частям - оверлайн.

8.2 Виртуальная память

Виртуальная память на самом деле не существует, а является абстракцией в ОС для организации доступа к физической памяти программам. Ее делят на страничную и сегментную (каждой программе соответствует свой сегмент).

Организация страничной виртуальной памяти

1. Любой программе выделено некоторое адресное пространство, размер которого, обычно, соответствует разрядности шины данных для физической памяти, но может быть любым.ы
2. Память делится на блоки (страницы) одинакового размера, равного степени двойки.
3. Каждый адрес состоит из двух частей: номер страницы и смещение внутри этой страницы.
4. У каждого процесса есть своя таблица страниц, у ОС есть общая таблица всех страниц. Таблица состоит из полей: бит отображения (показывает, отображена ли страница в физическую память), номер страницы (4 бита), смещение (12 бит), т.е. каждый адрес представляет из себя два байта.

Хранение таблицы страниц

1. В процессоре (самый быстрый способ, но самый дорогой)
2. Целиком в памяти (на одно обращение к памяти, требуется одно обращение к таблице)
3. Гибридное решение (TLB): принцип локальности, буфер часто используемых ссылок на страницы, ассоциативная память (реализован на аппаратном уровне), в процессоре есть два регистра: LDTR (Local Descriptor Table Registr) - хранит адрес расположения таблиц в памяти для текущего процесса, GDTR (Global DTR).

8.3 Сегментная виртуальная память

В центральном процессоре есть несколько базовых регистров, в сочетании со смещением определяется адрес в этом сегменте.

1. *CS* сегмент кода.
2. *SS* сегмент стека.
3. *DS* сегмент данных.

Существует таблица сегмента в которой определяются: адрес начала и конца сегмента, уровень привилегий, права доступа (Read (r), write(w), execute(x)). Например, для CS разрешено только читать и исполнять, а SS только читать и писать.

Сегменты позволяют защитить данные в оперативной памяти.

9 Ввод и вывод в ОС

9.1 Классификация устройств

По типу

1. Устройства ввода (сканер, мышь, клавиатура)
2. Устройства вывода (монитор, принтер, плоттер)
3. Коммуникация (модем)
4. Запоминающие (SSD, HHD, CD, DVD, магнитная лента, магнитно-оптические накопители)

По адресации

1. Символьные (Побайтовая адресация)
2. Блочная адресация (Способны адресовать только некоторую совокупность байтов - блок. Например, на ЖД блок равен 512 байтам)

По способу доступа

1. Произвольный
2. Последовательный

9.2 Порт I/O

Современные устройства состоят из двух частей: механической (для ОС неважна ее реализация) и контроллер, который представляет из себя программно-аппаратное устройство, передающие в ОС информацию о состоянии устройства и обеспечивающий обмен данными между ОС и устройством.

Порт I/O представляет из себя число, обозначающие некоторое адресное пространство, через которое происходит обмен данными между ОС и устройством. Существует шина I/O через которую передают данные, команды и статус.

9.2.1 Организация обмена данными

1. Отдельное адресное пространство памяти и I/O. В процессоре реализуются специальные команды:

```
in [register] [port]
out [register] [port]
```

2. Совместное адресное пространство.

3. Гибридное адресное пространство, когда часть памяти выделяют для портов I/O.

9.3 Синхронный и асинхронный ввод/вывод

Синхронный ввод вывод заключается в активном ожидании, т.е. ОС ждет команды пока устройство не готово.

При асинхронном I/O управление в программу возвращается сразу. Пример: Веб-технология AJAX, которая позволяет делать скрытые запросы на сервер без обновления всего веб-документа.

```
var x = XMLHTTPRequest();
// Handler. It run when control return in program
x.onReadyStateChange = function { /* some code */ };
x.open("e1.ru", true);
x.send()
```

В операционных системах устройство, завершив операцию, сигнализирует об этом (вызывает прерывание).

9.4 Драйверы

С точки зрения ОС, драйвер реализует API этой системы для взаимодействия с устройствами (скрывает детали реализации взаимодействия с устройствами). Обычно, драйвер работает в адресном пространстве ядра ОС.

9.5 Подключение драйверов к программе

1. *Статическая сборка.* Собрать ядро ОС, указав какие драйверы (библиотеки) подключать, а какие нет. При добавлении новых устройств требуется перекомпиляция ядра. Преимущество в том, что используется только тот код, который используется, что в свою очередь позволяет снизить потребляемую память.

2. *Динамическое подключение.* Технология Plug and play - динамическая загрузка в некоторый момент времени (при старте и процессе работе ОС). В частности, в ОС Linux, можно просмотреть список текущих загруженных модулей ядра (драйверов), с помощью команд: *lsmod* и *insmod*.

3. *Горячая загрузка.* Подключение устройств через шину USB без перезагрузки компьютера. устройство, поддерживающие USB реализует некоторые функции по стандарту, что позволяет взаимодействовать с ОС.

9.6 Жесткий диск

Работа жестких дисков с точки зрения операционной системы.

9.6.1 Механика

Цилиндр с несколькими тысячами дорожек, на которых храниться информация. Каждая дорожка состоит из нескольких секторов (с программной точки зрения 512 байт, на твердотельных 4 Кбайт), который представляет из себя блок с данными перед которыми идет номер сектора, а перед ним преамбула, после данных идет контрольная сумма (позволяет узнать верность считанных данных, и восстановить их при необходимости):

| Преамбула | Номер сектора | Данные | Контрольная сумма |

Изначально на жестком диске присутствовала адресация: CHS (Cylinder Head Sector). В современных жестких дисках используется адресация: LBA (Logical Block Address), преобразует некоторый понятный ОС адрес (#1023) во внутренние представление (Дорожка, сектор).

9.6.2 Надежность

Надежность жестких дисков обеспечивается за счет избыточности хранимой информации: решение RAID (Rendindand Array ...).

1. *RAID-0* (Зеркало). Есть некоторая последовательность исходных данных, которые равномерно распределяются по двум жестким дискам. Зеркало позволяет повысить скорость работы с жестким диском, но не надежность хранения данных! С точки зрения ОС, для программ не существует двух жестких, физическая реализация скрывается.

2. *RAID-1* Некоторая последовательность исходных данных полностью дублируются на другом жестком диске.

3. *RAID-5* Данные с одного жесткого диска равномерно распределяется на трех или четырех жестких дисках, и на не котором другом диске храниться контроль (например, логическая операция исключающего или)

4. *RAID-6* Позволяет выжить, если выйдет из строя два жестких диска.

Решение EVA: в стандартом телекоммуникационном шкафу храниться очень много жестких дисков.

9.6.3 Логическая структура

Жесткий диск разделен на логические составляющие - разделы, которые могут использоваться различными ОС и иметь разный тип файловой системы: | Раздел #1 | ... | Раздел #N |

В самом первом секторе жесткого диска находиться *структура разделов* и *программа загрузки (boot)*. Первый сектор имеет название MBR (Master Boot Record).

BIOS Базовая система ввода и вывода, которая выполняет подготовку компьютера при старте. При старте BIOS загружается в память и начинает выполнять некоторые процедуры, например штатное тестирование системы (POST - Power On Self Test).

CMOS - энергонезависимая память BIOS, в которой хранится различные параметры конфигурации (место, откуда загружать ОС), текущее время.

Процедура BOOT STRAP начальной загрузки.

Таблица разделов Каждый раздел характеризуется следующей записью:

1. Начало раздела

2. Конец раздела

3. Флаги (активность, является ли раздел с ОС)

4. Тип файловой системы

Один из разделов может иметь тип расширенный раздел который может иметь структуру из четырех разделов.

ОС DOS позволяет разбить жесткий диск на основной и расширенный разделы, в последнем можно сделать до 4 логических дисков. Windows позволяет делать сколько угодно основных и расширенных (которые могут включать в себя другие расширенные) дисков.

10 Файловая система

Задача: какое место занято/свободно Файловая система: каталог (имя файла/размер/дата/флаги/ссылка на таблицу (связанный список) с данными о размещении файлов в кластерах)

Операции над файлами: создание (занесение записи в каталог), открытие (загрузка файлов в память), удаление (состоит из двух частей: 1. удаление из каталога (необязательно физическое удаление записи с носителя) флаг "удален"(в FAT меняется первый символ из имени файла), 2. очистка цепочки кластеров)

Проблемы: потерянные кластеры (завершили очистку файлов в каталоге, но не очистили кластеры), две пересекающиеся цепочки кластеров.

Решение проблем

1. Устранить пост-фактум (при перезагрузке)

2. Журналирование в файловой системе гарантирует, что некоторая последовательность будет выполнена целиком или не будет выполнена вообще (транзакция). Запись журнала в конечном итоге приводит к выполнению следующих действий: чтение, выполнение, контроль. Если операция была выполнена целиком, то где-то у этой записи ставиться флаг (выполнен?). Если произошел сбой, то во время исправления ошибок запись будет выполняться еще раз. Все операции в журнале должны быть повторяемыми.

   Примеры ФС: ext3 в Linux, NTFS

Дополнительные возможности файловой системы Кроме каталога файлов есть таблица с дополнительными произвольными атрибутами файла. Например, в NTFS таблица MFT (Master File Table). В этом случае каталог содержит ссылку на эту таблицу (как замена цепочки кластеров, например). Атрибуты, например, DATA (хранится цепочка кластеров, либо сам файл, если его размер очень мал), разрешение доступа, и любые другие атрибуты.

Жесткие ссылки в ФС (hard link), мягкая ссылка (специальная запись в каталоге).

11 Разрешение доступа

Идентификация - это присваивание имени чему-то или кому-то без подтверждения. *Аутентификация* - проверка подлинности, осуществляется различными способами: общее знание (разделяемый секрет, лол), например, пароль; биометрическая аутентификация (отпечаток носа); многофакторная аутентификация (special for профессиональный параноик); одноразовая аутентификация (одноразовые пароли) *Авторизация* - разрешение или не разрешение выполнения некоторых действий с некоторым объектом (например с файлом или каталогом). Разрешение или не разрешение некоторых действий субъекту.

Хранение таблицы разрешений доступа в ОС Для субъектов (список пользователей): таблица (SID/UID | Name | First Name - Last Name | Password (информация для аутентификации, храниться в виде хэша)) Объединение пользователей в кучи (говна) - группы.

Второй вариант хранения: хранить только реально заполненные значения.

Оптимизация: пользователи с однотипными разрешениями объединяются. Создание ролей. Вторая оптимизация: объединение одинаковых списков контроля доступа. Таблица уникальных ACL.