Министерство образования республики Беларусь
Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Институт информационных технологий

Специальность ПОИТ

КОНТРОЛЬНАЯ РАБОТА

По курсу Сетевые технологии и администрирование операционных систем

Студент-заочник 3 курса
Группы № 581072
Богданова Кристина Евгеньевна
Тел. +375 (25) 929-99-11

Проверил
Калитеня Иван Леонидович

Минск, 2018

Содержание

Аббревиатуры и сокращения

ОС – операционная система.
API – Application programming interface.
DLL – Dynamic Link Library.
GDI – Graphics Device Interface.
SMARTDRV – SmartDrive.
MSCDEX – Microsoft CD-ROM Extensions.
ЦП – центральный процессор.
ЦПУ – центральное процессорное устройство.
URL – Uniform Resource Locator.
DMA – Direct memory access.
PCI – Peripheral component interconnect.
LRU – Least Recently Used.
MRU – Most Recently Used.
LFU – Least Frequently Used.
ARC – Adaptive Replacement Cache.
HTTP – HyperText Transfer Protocol.
CMS – Content Management System.

1. Операционная система – основные функции и задачи

ОС – это комплекс взаимосвязанных системных программ для организации взаимодействия пользователя с компьютером и выполнения всех других программ. ОС относятся к составу системного программного обеспечения и являются основной его частью. Семейства операционных систем: Windows, UNIX, Mac OS.

Основные функции ОС:

- управление устройствами компьютера (ресурсами), т.е. согласованная работа всех аппаратных средств ПК: стандартизованный доступ к периферийным устройствам, управление оперативной памятью и др.
- управление процессами, т.е. выполнение программ и их взаимодействие с устройствами компьютера.
- управление доступом к данным на энергонезависимых носителях (таких как жесткий диск, компакт-диск и т.д.), как правило, с помощью файловой системы.
- ведение файловой структуры.
- пользовательский интерфейс, т.е. взаимодействие с пользователем.

Дополнительные функции:

- параллельное или псевдопараллельное выполнение задач (многозадачность).
- взаимодействие между процессами: обмен данными, взаимная синхронизация.
- защита самой системы, а также пользовательских данных и программ от действий пользователей с недостаточной квалификацией, злонамеренных действий пользователей или приложений.
- разграничение прав доступа и многопользовательский режим работы (аутентификация, авторизация).

Состав операционной системы. В общем случае в состав ОС входят следующие модули:

- Программный модуль, управляющий файловой системой.
- Командный процессор, выполняющий команды пользователя.
- Драйверы устройств.
- Программные модули, обеспечивающие графический пользовательский интерфейс.
- Сервисные программы.
- Справочная система.

Драйвер устройства (device driver) – специальная программа, обеспечивающая управление работой устройств и согласование информационного обмена с другими устройствами.

Командный процессор (command processor) – специальная программа, которая запрашивает у пользователя команды и выполняет их (интерпретатор программ).

Интерпретатор команд отвечает за загрузку приложений и управление информационным потоком между приложениями.

Для упрощения работы пользователя в состав современных ОС входят программные модули, обеспечивающие графический пользовательский интерфейс. Процесс работы компьютера в определенном смысле сводится к обмену файлами между устройствами. В ОС имеется программный модуль, управляющий файловой системой.

Сервисные программы позволяют обслуживать диски (проверять, сжимать, дефрагментировать и др.), выполнять операции с файлами (копирование, переименование и др.), работать в компьютерных сетях.

Для удобства пользователя в состав ОС входит справочная система, позволяющая оперативно получить необходимую информацию о функционировании как ОС в целом, так и о работе ее отдельных модулей.

Состав модулей ОС, а также их количество зависит от семейства и вида ОС. Так, например, в ОС MS DOS отсутствует модуль, обеспечивающий графический пользовательский интерфейс.

Наиболее общим подходом к структуризации операционной системы является разделение всех ее модулей на две группы:

1. Ядро – это модули, выполняющие основные функции ОС.

2. Вспомогательные модули, выполняющие вспомогательные функции ОС.

Одним из определяющих свойств ядра является работа в привилегированном режиме.

Модули ядра выполняют следующие базовые функции ОС: Управление процессами, Управление системой прерываний, Управление памятью, управление устройствами ввода-вывода, Функции, решающие внутрисистемные задачи организации вычислительного процесса: переключение контекстов, загрузка/выгрузка страниц, обработка прерываний. Эти функции недоступны для приложений. Функции, служащие для поддержки приложений, создавая для них так называемую прикладную программную среду.

Приложения могут обращаться к ядру с запросами – системными вызовами – для выполнения тех или иных действий: для открытия и чтения файла, вывода графической информации на дисплей, получения системного времени

и т.д. Функции ядра, которые могут вызываться приложениями, образуют интерфейс прикладного программирования – API.

Пример. Базовый код API Win32 содержится в трех библиотеках динамической загрузки (DLL): USER32, GDI32 и KERNEL32.

Kernel - модуль Windows, который поддерживает низкоуровневые функции по работе с файлами и управлению памятью и процессами. Этот модуль обеспечивает сервис для 16- и 32-разрядных приложений. GDI - модуль Windows, обеспечивающий реализацию графических функций по работе с цветом, шрифтами и графическими примитивами для дисплея и принтеров.

User - модуль Windows, который является диспетчером окон и занимается созданием и управлением отображаемыми на экране окнами, диалоговыми окнами, кнопками и другими элементами пользовательского интерфейса.

Критическая ошибка в ядре операционной системы, например, попытка обращения к ошибочному или запрещённому адресу в памяти, вызывает отказ ОС, после которого она не может продолжать свою дальнейшую работу. Поэтому разработчики операционной системы уделяют особое внимание надежности кодов ядра, в результате процесс их отладки может растягиваться на многие месяцы.

Обычно ядро оформляется в виде программного модуля некоторого специального формата, отличающегося от формата пользовательских приложений.

Вспомогательные модули ОС выполняют вспомогательные функции ОС (полезные, но менее обязательные чем функции ядра).

Примеры вспомогательных модулей:

- Программа архивирования данных.
- Программа дефрагментации диска.
- Текстовый редактор.

Вспомогательные модули ОС оформляются либо в виде приложений, либо в виде библиотек процедур. Вспомогательные модули ОС подразделяются на следующие группы:

– утилиты – программы, решающие задачи управления и сопровождения компьютерной системы: обслуживание дисков и файлов.

– системные обрабатывающие программы – текстовые или графические редакторы, компиляторы, компоновщики, отладчики.

– программы предоставления пользователю дополнительных услуг пользовательского интерфейса (калькулятор, игры).

– библиотеки процедур различного назначения, упрощающие разработку приложений (библиотека математических функций, функций ввода-вывода).

Как и обычные приложения, для выполнения своих задач утилиты, обрабатывающие программы и библиотеки ОС, обращаются к функциям ядра посредством системных вызовов.

Функции, выполняемые модулями ядра, являются наиболее часто используемыми функциями операционной системы, поэтому скорость их выполнения определяет производительность всей системы в целом. Для обеспечения высокой скорости работы ОС все модули ядра или большая их часть постоянно находятся в оперативной памяти, то есть являются резидентными.

Вспомогательные модули обычно загружаются в оперативную память только на время выполнения своих функций, то есть являются транзитными. Такая организация ОС экономит оперативную память компьютера.

Разделение операционной системы на ядро и вспомогательные модули обеспечивает легкую расширяемость ОС. Чтобы добавить новую высокоуровневую функцию, достаточно разработать новое приложение, и при этом не требуется модифицировать основные функции, образующие ядро системы.

Объектами ядра ОС являются:

- Процессы.
- Файлы.
- События.
- Потоки.
- Семафоры – объекты, позволяющие войти в заданный участок кода не более чем n потокам.
- Мьютексы – одноместные семафоры, служащие в программировании для синхронизации одновременно выполняющихся потоков.
- Файлы, проецируемые в память.

2. Файлы настройки MS DOS, отличия, последовательность выполнения

К файлам конфигурации MS DOS относятся находящиеся в корневом каталоге AUTOEXEC.BAT, CONFIG.SYS.

AUTOEXEC.BAT (от англ. automatic execution — автоматическое исполнение и англ. batch — пакет, группа) — системный пакетный файл (файл, содержащий последовательность команд на языке интерпретатора командной строки — поставляемого в составе MS-DOS COMMAND.COM или его клонов вроде 4DOS (англ.)), расположенный в корневом каталоге загрузочного устройства (дискеты или диска). Впервые этот файл появился в операционной системе MS-DOS, а его имя является аббревиатурой и описывает его функцию — автоматическое исполнение команд при загрузке системы. Аббревиатура была необходима из-за ограничения 8.3 на длину имени файла (8 знаков собственно имени и 3 знака расширения) в файловой системе FAT.

AUTOEXEC.BAT ничем не отличается от прочих пакетных файлов (в отличие от другого файла конфигурирования DOS, Windows 9x и OS/2 — CONFIG.SYS). Как правило, он используется в DOS для установки ключевых переменных окружения (таких как PATH), загрузки драйверов и резидентных программ (SMARTDRV.EXE, MSCDEX, драйвера мыши, программ русификации клавиатуры и экрана и т. п.), а также запуска утилит конфигурации (например, для настройки параметров звуковых карт) и проверки системы (например, антивирусных сканеров).

В MS-DOS AUTOEXEC.BAT исполняется после старта командного интерпретатора, который загружается после обработки команд из файла конфигурации CONFIG.SYS. С помощью директивы shell= в CONFIG.SYS можно задействовать командный интерпретатор, отличный от COMMAND.COM, и/или указать имя для стартового пакетного файла, отличное от AUTOEXEC.BAT. При отсутствии файла AUTOEXEC.BAT (или при отмене его исполнения) в большинстве версий DOS командный интерпретатор запрашивает у пользователя текущие дату и время.

Начиная с MS-DOS 6.0, появилась возможность пропустить исполнение AUTOEXEC.BAT с помощью функциональных клавиш F5 и F8 (для этого в COMMAND.COM ввели поддержку опций /Y и /D), но с помощью директивы switches= в файле CONFIG.SYS эти клавиши можно заблокировать.

Начиная с MS-DOS 6.0 появилась возможность в CONFIG.SYS описывать меню для выбора одной из нескольких конфигураций. При выборе пункта меню имя секции с описанием требуемой конфигурации, указанное в этом пункте, сохраняется в переменной окружения CONFIG. Это позволяет с по-

мощью команд goto %CONFIG% и if "%CONFIG%"== в AUTOEXEC.BAT (и в прочих пакетных файлах) выполнять разные действия в зависимости от выбора конфигурации при загрузке.

Пример файла AUTOEXEC.BAT

```
@ECHO OFF
REM C:\WINDOWS\SMARTDRV.EXE
C:\WINDOWS\SMARTDRV.EXE 2038 512
PROMPT $p$g
PATH C:\DOS;C:\WINDOWS;C:\LWORKS;C:\EXPLORER.4LC
SET TEMP=C:\DOS
MODE LPT1:,,P >nul
C:\DOS\SHARE.EXE /F:150 /L:1500
C:\WINDOWS\mouse.COM /Y
cd windows
WIN
```

В этом примере можно видеть отключение дублирования обрабатываемых строк на экран, запуск драйвера SMARTDRV (одна строка закомментирована, в другой SMARTDRV запускается с аргументами), установку переменных окружения (PROMPT, PATH, TEMP), запуск резидентной программы (MODE) и других драйверов (SHARE и MOUSE), и, наконец, переход в каталог windows (CD) и запуск собственно Windows (WIN).

CONFIG.SYS — файл конфигурирования операционных систем семейств DOS, Windows 9x и OS/2. Это текстовый файл, содержащий директивы настройки системы и команды загрузки драйверов, он должен располагаться в корневом каталоге загрузочного устройства (англ. Boot disk) (дискеты или диска).

Под DOS директивы в этом файле задают некоторые аппаратные (такие, как состояние индикатора Num Lock) и системные параметры (например, количество и вложенность стеков для обработки аппаратных прерываний, количество дисковых буферов и т. п.), а также загружают драйверы для управления дополнительной и расширенной памятью (HIMEM.SYS, EMM386.EXE), экраном (ANSI.SYS, DISPLAY.SYS), дисководом CD-ROM и т. п.

Под DOS CONFIG.SYS обрабатывается ядром системы IO.SYS. После обработки CONFIG.SYS загружается файл MSDOS.SYS и указанный директивой shell= в CONFIG.SYS интерпретатор командной строки или, в случае отсутствия этой директивы, COMMAND.COM. Уже командный интерпретатор отвечает за исполнение файла AUTOEXEC.BAT.

Это происходит во всех версиях DOS вплоть до MS-DOS 7.x. Также, начиная с MS-DOS 6.0 появилась возможность пропустить обработку CONFIG.SYS с помощью функциональных клавиш F5 и F8, но эту возмож-

ность можно запретить с помощью директивы switches= в этом же файле. В предыдущих версиях MS-DOS (до версии 6.0) не существовало возможности обойти обработку CONFIG.SYS при загрузке, в результате ошибка в написании (например, указание обычного исполнимого файла вместо драйвера) могла приводить к фатальным сбоям и невозможности загрузиться и для восстановления требовалась загрузка со сменного носителя (дискеты).

Клоны DOS кроме файла CONFIG.SYS могут использовать файлы и с другими именами, что облегчает сосуществование разных версий DOS на одном диске. Например, в свободной операционной системе FreeDOS CONFIG.SYS ищется только если не найден файл FDCONFIG.SYS, а в некоторых версиях DR-DOS ищется файл DCONFIG.SYS.

CONFIG.SYS имеет свой специальный синтаксис. В основном, он состоит из директив вида команда=значение (или то же самое, но без знака равенства — например, numlock off). Также, сразу после команды можно поставить знак вопроса ('?', например dos?=high) — в этом случае перед исполнением директивы запрашивается подтверждение исполнения. Необходимо отметить, что синтаксис CONFIG.SYS в FreeDOS отличается от синтаксиса в MS-DOS —в FreeDOS иной синтаксис для организации меню загрузки.

Начиная с MS-DOS 6.0 появилась возможность в CONFIG.SYS группировать директивы в секции и описывать меню. Секции позволяют задать несколько конфигураций, а меню позволяет выбрать одну из них при загрузке системы. Секция начинается с имени секции в квадратных скобках [<имя секции>] и заканчивается с началом следующей секции (или с концом файла). При этом секция [menu] используется для описания меню, а секция [common] обрабатывается перед обработкой любой выбранной в меню конфигурации.

Пример файла CONFIG.SYS:

```
numlock = off
break = on
dos = high,umb
country = 7,,c:\dos\country.sys
files = 40
device = c:\dos\himem.sys
device = c:\dos\emm386.exe ram i=b000-b7ff
shell = command.com /p /e:512
```

В этом примере можно видеть отключение переключателя Num Lock, включение комбинации Ctrl+C во время работы программ, установку параметров загрузки DOS, региональных настроек, допустимое количество одновременно открытых файлов, загрузку драйверов, и, наконец, указание опций интерпретатора командной строки command.com.

3. Кэширование информации, условия установки кэш памяти, виды кэш памяти, влияние объема кэш на производительность ОС

Кэш или кеш — промежуточный буфер с быстрым доступом, содержащий информацию, которая может быть запрошена с наибольшей вероятностью. Доступ к данным в кэше осуществляется быстрее, чем выборка исходных данных из более медленной памяти или удаленного источника, однако ее объем существенно ограничен по сравнению с хранилищем исходных данных.

Кэш — это память с большей скоростью доступа, предназначенная для ускорения обращения к данным, содержащимся постоянно в памяти с меньшей скоростью доступа (далее «основная память»). Кэширование применяется ЦПУ, жесткими дисками, браузерами, веб-серверами, службами DNS и WINS.

Кэш состоит из набора записей. Каждая запись ассоциирована с элементом данных или блоком данных (небольшой части данных), которая является копией элемента данных в основной памяти. Каждая запись имеет идентификатор, часто называемый тегом, определяющий соответствие между элементами данных в кэше и их копиями в основной памяти.

Когда клиент кэша (ЦПУ, веб-браузер, операционная система) обращается к данным, прежде всего исследуется кэш. Если в кэше найдена запись с идентификатором, совпадающим с идентификатором затребованного элемента данных, то используются элементы данных в кэше. Такой случай называется попаданием кэша. Если в кэше не найдена запись, содержащая затребованный элемент данных, то он читается из основной памяти в кэш, и становится доступным для последующих обращений. Такой случай называется промахом кэша. Процент обращений к кэшу, когда в нем найден результат, называется уровнем попаданий, или коэффициентом попаданий в кэш.

Например, веб-браузер проверяет локальный кэш на диске на наличие локальной копии веб-страницы, соответствующей запрошенному URL. В этом примере URL — это идентификатор, а содержимое веб-страницы — это элементы данных.

Если кэш ограничен в объеме, то при промахе может быть принято решение отбросить некоторую запись для освобождения пространства. Для выбора отбрасываемой записи используются разные алгоритмы вытеснения.

При модификации элементов данных в кэше выполняется их обновление в основной памяти. Задержка во времени между модификацией данных в кэше и обновлением основной памяти управляется так называемой политикой записи.

В кэше с немедленной записью каждое изменение вызывает синхронное обновление данных в основной памяти.

В кэше с отложенной записью (или обратной записью) обновление происходит в случае вытеснения элемента данных, периодически или по запросу клиента. Для отслеживания модифицированных элементов данных записи кэша хранят признак модификации (при изменении устанавливается в 1). Промах в кэше с отложенной записью может потребовать два обращения к основной памяти: первое для записи заменяемых данных из кэша, второе для чтения необходимого элемента данных.

В случае, если данные в основной памяти могут быть изменены независимо от кэша, то запись кэша может стать неактуальной. Протоколы взаимодействия между кэшами, которые сохраняют согласованность данных, называют протоколами когерентности кэша.

Кэш центрального процессора разделен на несколько уровней. Максимальное количество кэшей — четыре. В универсальном процессоре в настоящее время число уровней может достигать трех. Кэш-память уровня N+1, как правило, больше по размеру и медленнее по скорости доступа и передаче данных, чем кэш-память уровня N.

Самым быстрым является кэш первого уровня — L1 cache (level 1 cache). По сути, он является неотъемлемой частью процессора, поскольку расположен на одном с ним кристалле и входит в состав функциональных блоков. В современных процессорах обычно L1 разделен на два кэша — кэш команд (инструкций) и кэш данных (Гарвардская архитектура). Большинство процессоров без L1 не могут функционировать. L1 работает на частоте процессора, и, в общем случае, обращение к нему может производиться каждый такт. Зачастую является возможным выполнять несколько операций чтения/записи одновременно.

Вторым по быстродействию является кэш второго уровня — L2 cache, который обычно, как и L1, расположен на одном кристалле с процессором. В ранних версиях процессоров L2 реализован в виде отдельного набора микросхем памяти на материнской плате. Объем L2 от 128 кбайт до 1−12 Мбайт. В современных многоядерных процессорах кэш второго уровня, находясь на том же кристалле, является памятью раздельного пользования — при общем объеме кэша в n Мбайт на каждое ядро приходится по n/c Мбайта, где c — количество ядер процессора.

Кэш третьего уровня наименее быстродействующий, но он может быть очень большим — более 24 Мбайт. L3 медленнее предыдущих кэшей, но все равно значительно быстрее, чем оперативная память. В многопроцессорных

системах находится в общем пользовании и предназначен для синхронизации данных различных L2.

Существует четвертый уровень кэша, применение которого оправдано только для многопроцессорных высокопроизводительных серверов и мейнфреймов. Обычно он реализован отдельной микросхемой.

При чтении данных кэш-память позволяет увеличить производительность. При записи данных увеличение производительности можно получить только ценой снижения надежности. Поэтому в различных приложениях может быть выбрана та или иная политика записи кэш-памяти.

Существуют две основные политики записи кэш-памяти — сквозная запись и отложенная запись:

– Сквозная запись — запись производится непосредственно в основную память (и дублируется в кэш), то есть запись не кэшируется.

– Отложенная запись — запись данных производится в кэш. Запись же в основную память производится позже (при вытеснении или по истечении времени), группируя в одной операции несколько операций записи в соседние ячейки. Технология обратной записи на некоторое время делает данные в основной памяти неактуальными, для самого ЦП эти неактуальности не заметны, но перед обращением к памяти другого ведущего системной шины (контроллера DMA, bus-master-устройства шины PCI) кэш должен быть записан в память принудительно. При использовании обратной записи в многопроцессорной системе кэши различных ЦП должны быть согласованы (или процессоры должны использовать одну кэш-память).

Изначально все заголовки буферов помещаются в список свободных буферов. Если процесс намеревается прочитать или модифицировать блок, то он выполняет следующий алгоритм:

– пытается найти в хеш-таблице заголовок буфера с заданным номером;

– в случае, если полученный буфер занят, ждет его освобождения;

– в случае, если буфер не найден в хеш-таблице, берет первый буфер из хвоста списка свободных;

– в случае, если список свободных буферов пуст, то выполняется алгоритм вытеснения;

– в случае, если признак модификации полученного буфера установлен в 1, выполняет асинхронную запись содержимого буфера во внешнюю память.

– удаляет буфер из хеш-таблицы, если он был помещен в нее;

– помещает буфер в хеш-таблицу с новым номером.

Процесс читает данные в полученный буфер и освобождает его. В случае модификации процесс перед освобождением устанавливает признак модификации буфера в 1. При освобождении буфер помещается в голову списка свободных буферов.

Таким образом:

– если процесс прочитал некоторый блок в буфер, то велика вероятность, что другой процесс при чтении этого блока найдет буфер в оперативной памяти;

– запись данных во внешнюю память выполняется только тогда, когда не хватает буферов с установленным в 0 признаком модификации, либо по запросу.

Если список свободных буферов пуст, то выполняется алгоритм вытеснения буфера. Алгоритм вытеснения существенно влияет на производительность кэша. Существуют следующие алгоритмы:

– LRU — вытесняется буфер, неиспользованный дольше всех;

– MRU — вытесняется последний использованный буфер;

– LFU — вытесняется буфер, использованный реже всех;

– ARC — алгоритм вытеснения, комбинирующий LRU и LFU, запатентованный IBM.

Применение того или иного алгоритма зависит от стратегии кэширования данных. LRU наиболее эффективен, если данные гарантированно будут повторно использованы в ближайшее время. MRU наиболее эффективен, если данные гарантированно не будут повторно использованы в ближайшее время. В случае, если приложение явно указывает стратегию кэширования для некоторого набора данных, то кэш будет функционировать наиболее эффективно.

Кэш оперативной памяти состоит из следующих элементов:

– набор страниц оперативной памяти, разделенных на буферы, равные по длине блоку данных соответствующего устройства внешней памяти;

– набор заголовков буферов, описывающих состояние соответствующего буфера;

– хеш-таблицы, содержащей соответствие номера блока заголовку;

– списки свободных буферов.

В процессе передачи информации по сети может использоваться кэширование интернет-страниц — процесс сохранения часто запрашиваемых документов на (промежуточных) прокси-серверах или машине пользователя, с целью предотвращения их постоянной загрузки с сервера-источника и уменьшения трафика. Таким образом, информация перемещается ближе к

пользователю. Управление кэшированием осуществляется при помощи HTTP-заголовков.

Как вариант, кэширование веб-страниц может осуществляться с помощью CMS конкретного сайта для снижения нагрузки на сервер при большой посещаемости. Кэширование может производиться как в память, так и в файловый кэш. Недостаток кэширования заключается в том, что изменения, внесенные на одном браузере, могут не сразу отражаться в другом браузере, в котором данные берутся из кэш-памяти.

Многие программы записывают куда-либо промежуточные или вспомогательные результаты работы, чтобы не вычислять их каждый раз, когда они понадобятся. Это ускоряет работу, но требует дополнительной памяти (оперативной или дисковой). Примером такого кэширования является индексирование баз данных.

Влияние кэш-памяти процессора на быстродействие компьютера. При выполнении запроса на предоставление данных ядру, контроллер памяти ищет их сначала в кэше первого уровня, затем - в кэше второго и третьего уровней.

По статистике, кэш-память первого уровня любого современного процессора обеспечивает до 90 % кэш-попаданий. Второй и третий уровни - еще 90% от того, что осталось. И только около 1 % всех запросов процессора заканчиваются кэш-промахами.

Указанные показатели касаются простых задач. С повышением нагрузки на процессор число кэш-промахов увеличивается.

Эффективность кэш-памяти процессора сводит к минимуму влияние скорости оперативной памяти на быстродействие компьютера. Например, компьютер одинаково хорошо будет работать с оперативной памятью 1066 МГц и 2400 МГц. При прочих равных условиях разница производительности в большинстве приложений не превысит 5%.

При оценке эффективности кэш-памяти возникают следующие вопросы:

Какая структура кэш-памяти лучше: двух- или трехуровневая?

Трехуровневая кэш-память более эффективна.

Чтобы определить, как сильно L3 влияет на работу процессора, сайтом Tom’s Hardware был проведен эксперимент. Заключался он в замере производительности процессоров Athlon II X4 и Phenom II X4. Оба процессора оснащены одинаковыми ядрами. Первый отличается от второго лишь отсутствием кэш-памяти L3 и более низкой тактовой частотой.

Приведя частоты обеих процессоров к одинаковому показателю, было установлено, что наличие кэш-памяти L3 повышает производительность

процессора Phenom на 5,8 %. Но это средний показатель. В одних приложениях он был почти равен нулю (офисные программы), в других – достигал 8% и даже больше (компьютерные 3D игры, архиваторы и др.).

Как влияет размер кэша на производительность процессора?

Оценивая размер кэш-памяти, нужно учитывать характеристики процессора и круг решаемых им задач.

Кэш-память двуядерного процессора редко превышает 3 MB. Тем более, если его тактовая частота ниже 3 Ггц. Производителям известно, что дальнейшее увеличение размера кэша такого процессора не принесет прироста производительности, зато существенно повысит его стоимость.

Другое дело высокочастотные 4-, 6- или даже 8-миядерные процессоры. Некоторые из них (например, Intel Core i7) поддерживают технологию Hyper Threading, обеспечивающую одновременное выполнение каждым ядром двух задач. Естественно, что потенциал таких процессоров не может быть раскрыт с малым кэшем. Поэтому его увеличение до 15 или даже 20 MB вполне оправдано.

В процессорах Intel алгоритм наполнения кэш-памяти построен по так называемой инклюзивной схеме, когда содержимое кэшей верхнего уровня (L1, L2) полностью или частично дублируется в кэше нижнего уровня (L3). Это в определенной степени уменьшает полезный объем его пространства. С другой стороны, инклюзивная схема позитивно сказывается на взаимодействии ядер процессора между собой.

В целом же, эксперименты свидетельствуют, что в среднестатистическом «домашнем» процессоре влияние размера кэша на производительность находится в пределах 10 %, и его вполне можно компенсировать, например, высокой частотой.

Эффект от большого кэша наиболее ощутим при использовании архиваторов, в 3D играх, во время кодирования видео. В не требовательных же приложениях разница стремится к нулю (офисные программы, интернет-серфинг, работа с фотографиями, прослушивание музыки и др.).

Многоядерные процессоры с большим кэшем необходимы на компьютерах, предназначенных для выполнения многопоточных приложений, одновременного решения нескольких сложных задач.

Особенно актуально это для серверов с высокой посещаемостью. В некоторых высоконагружаемых серверах и суперкомпьютерах предусмотрена даже установка кэш-памяти четвертого уровня (L4). Изготавливается она в виде отдельных микросхем, подключаемых к материнской плате.

Как узнать размер кэш-памяти процессора?

Существуют специальные программы, предоставляющие подробную информацию о процессоре компьютера, в том числе и о его кэш-памяти. Одной из них является программа CPU-Z.

Можно ли как-то увеличить кэш-память процессора?

Возможность увеличения кэш-памяти процессора предусмотрена в некоторых серверах и суперкомпьютерах, путем ее подключения к материнской плате.

В домашних же или офисных компьютерах такая возможность отсутствует. Кэш-память является внутренней неотъемлемой частью процессора, имеет очень маленькие физические размеры и не подлежит замене. А на обычных материнских платах нет разъемов для подключения дополнительной кэш-памяти.

Список использованных источников

1. Гордеев А. В. Операционные системы: Учебник для вузов. — 2-е изд. — СПб.: Питер, 2007. — 416 с.

2. Иртегов Д. В. Введение в операционные системы. — 2-е изд. — СПб.: BHV-СПб, 2007.

3. Таненбаум Э. С., Вудхалл А. С. Операционные системы. Разработка и реализация = Operating Systems: Design and Implementation. — 3-е изд. — СПб.: Питер, 2007. — 704 с.

4. Орлов С. А. Организация ЭВМ и систем. Фундаментальный курс по архитектуре и структуре современных компьютерных средств. — 2-е изд. — СПб.: Питер, 2007. — 686 с.