**Оптимизация использования памяти. Оптимизация использования жесткого диска. Оптимизация использования сети. Инструменты повышения производительности программного обеспечения.**

**Оптимизация использования оперативной памяти**

Оптимизация использования памяти может преследовать две цели – увеличение объема памяти, доступной приложениям, и повышение быстродействия обращений к памяти. На старых машинах, когда объем установленной физической памяти не превышал единиц мегабайт, зачастую приходилось чем-то жертвовать; в современных системах достижение обеих целей уже не противоречит друг другу. Нет сомнений в том, что, чем больше установленный объем ОЗУ, тем лучше – компьютер будет мощнее: он позволит загружать приложения, особо «жадные» до памяти; увеличить число одновременно работающих приложений (в многозадачной ОС); ускорить работу. Однако для обеспечения возможности использования памяти в ряде случаев требуются некоторые дополнительные действия по конфигурированию компьютера.   
В системах с размером установленной памяти более 640 Кбайт возможны различные варианты использования последних 384 Кбайт из первого мегабайта физической памяти:

* память не используется;
* область (или часть ее) перемещается в конец дополнительной памяти;
* область (или часть ее) используется в качестве теневой *(Shadow)*памяти адаптеров и ROM BIOS.

Перемещение неиспользуемого остатка первого мегабайта в конец дополнительной памяти (разрешается установкой параметра Memory Relocation в CMOS Setup) возможно не всегда. Обычно такое перемещение становится невозможным, если хоть часть из этого кусочка используется в качестве теневой памяти.   
Также перемещение может предлагаться лишь при небольших объемах установленной памяти, и на современных системных платах эта возможность почти не встречается. Поэтому не стоит удивляться сообщению об объеме памяти, обнаруженном тестом POST, в котором относительно установленного «не хватает» 384 Кбайт.   
Иногда BIOS предлагает такое распределение памяти, при котором под стандартную память выделяются 512 Кбайт, а остальная память идет как расширенная. Пользу такого распределения оценить трудно. При этом вышесказанное про верхние 384 Кбайт становится справедливым для оставшихся 512 Кбайт, правда, на возможность их перемещения в конец дополнительной памяти ограничения будут мягче.   
Острее всего проблема обеспечения доступности памяти стоит для приложений MS-DOS, исполняемых в среде этой системы. Обычно «битва» идет за килобайты стандартной памяти (conventional memory), доступной приложениям. Из 640 Кбайт после загрузки ОС и необходимых резидентных драйверов на долю приложений может остаться около 500 Кбайт, а то и меньше, что для ряда приложений неприемлемо. Несмотря на повсеместное внедрения ОС типа Windows 9х, интерес к запуску больших приложений MS-DOS (например, бухгалтерских программ, в том числе и принудительно распространяемых налоговыми органами, а также игр) сохраняется и поныне. Сообщение о нехватке памяти на компьютере с ОЗУ размером, например, 32 Мбайт для приложения, скромно просящего всего 590 Кбайт, неприятно удивляет неискушенных пользователей. Однако эта нехватка не фатальна, если правильно выбрать версию и способ загрузки ОС. Все программные настройки, влияющие на объем доступной памяти, кроются в файлах CONFIG.SYS и AUTOEXEC.BAT, размещенных в корневом каталоге загрузочного диска.   
Понимая проблемы, возникающие в связи с использованием памяти различными процессорами, для DOS-приложений можно рекомендовать следующие версии MS-DOS, оптимальные по размеру свободной стандартной памяти:

* для компьютеров класса XT на процессорах 8086/88 – MS-DOS 3.30;
* для компьютеров класса AT на процессорах 80286 – MS-DOS 5.x;
* для компьютеров класса AT на процессорах 80386 и выше – MS-DOS 6.2x (правда, одно время встречались некачественные системные платы для 80386, на которых MS-DOS 6.2x отказывалась загружаться в НМА, хотя MS-DOS 5.x работала нормально).

Кроме MS-DOS, существуют и другие операционные системы реального режима, совместимые с MS-DOS (например, PC DOS, DR DOS, COMPAQ DOS). Каждая из них имеет свои преимущества и недостатки по сравнению с соответствующим поколением MS-DOS, но их обсуждение выходит за рамки данного пособия. Операционные системы (и оболочки) защищенного режима на компьютеры с процессором класса ниже 386 пытаться устанавливать почти бессмысленно. Система MS-DOS 3.3 (более ранние рассматривать не будем) загружалась целиком в стандартную память, но была довольно компактной и оставляла приемлемое место для приложений своего времени. Когда вышла MS-DOS версии 4 с более развитыми возможностями, появившимися ценой ее разрастания, многие приложения отказывались работать в ее среде именно из-за нехватки памяти, и эта версия ОС широкого распространения не получила. Более удачной стала версия 5, которая «научилась» использовать «высокую память» (НМЛ) на компьютерах с процессором 286 и выше, если таковая присутствовала. Для этого был введен специальный драйвер высокой памяти, и в файле CONFIG.SYS должны присутствовать строки:   
DEVICE=[<path>]HIMEM.SYS (загрузка драйвера высокой памяти)   
DOS=HIGH (указание на загрузку ОС в высокую память)   
Конечно, в верхнюю память загружается не вся ОС – часть все-таки попадает и в стандартную память. Кроме того, в стандартную память загружаются и резидентные драйверы – например, русификаторы клавиатуры и экрана, драйвер мыши и т. п. Все они отрывают свой кусок от памяти, которую могли бы использовать приложения. На компьютерах с 32-разрядными процессорами (386 и выше), имеющих механизм страничной переадресации, появилась возможность использования «верхней памяти» (UMA) с помощью драйвера EMM386.EXE. Этот драйвер отыскивает в области UMA (A0000-FFFFFh) регионы, не занятые памятью устройств, и отображает их на области доступной дополнительной памяти. В эти регионы, нормально адресуемые процессором в реальном режиме, можно помещать модули операционной системы и загружаемые драйверы; их же могут использовать и приложения.   
Для наиболее компактной загрузки MS-DOS версий 5 и выше (а также Windows 9х, которая может представляться как MS-DOS 7) в файле CONFIG.SYS должны быть следующие директивы:   
DEVICE=[<path>]HIMEM.SYS   (загрузка драйвера  высокой  памяти);   
DEVICE=[<path>]EMM386.EXE   (загрузка диспетчера  расширенной  памяти);   
DOS=HIGH.   UMB   (указание  на  загрузку ОС в высокую и  верхнюю память).   
Резидентные драйверы (русификаторы клавиатуры, экрана и принтера, драйвер мыши) по умолчанию обычно загружаются в стандартную память. Если приложениям MS-DOS не хватает свободной памяти, то, по крайней мере, часть резидентных драйверов можно загрузить в верхнюю память UMA. Для драйверов, загружаемых из файла CONFIG.SYS, вместо команды DEVICE» следует использовать команду DEVICEHIGH [/L:n[,m]]=, которая попытается загрузить драйвер в n-регион UMB. Необязательный параметр m задает требуемый объем памяти (он может отличаться от размера файла с драйвером). Ключ загрузки /L вместе с параметрами n, m используется для ручной оптимизации памяти. Если драйверу требуется выделить несколько областей памяти, то они перечисляются в списке вида /L:nl[,ml]:n2[,m2][;...]. Для резидентных программ, загружаемых из файла AUTOEXEC.BAT, тем же целям служит команда LOADHIGH (LH) с аналогичными необязательными параметрами, задающими размер одной или нескольких областей UMA. Строка запуска будет иметь вид:   
LH[/L:n[,m]]   <путь\><файл>[<параметры>]   
Посмотреть текущее положение свободных блоков памяти можно командой MEM/F из командной строки DOS/Windows. Чтобы не заниматься ручной оптимизацией памяти, в состав DOS/Windows введена диалоговая утилита MEMMAKER.EXE, которая за несколько этапов расставит требуемые команды в файлах CONFIG.SYS и AUTOEXEC.BAT. До ее запуска в этих файлах должны присутствовать ссылки на все необходимые драйверы и программы. Файл CONFIG.SYS должен начинаться с вышеприведенных трех строк (если не указать DOS=HIGH, утилита может «постесняться» использовать высокую память для загрузки DOS). Запуск нерезидентных приложений, а также оболочки типа Norton Commander из файла AUTOEXEC.BAT на время работы MEMMAKER следует отменить (MEMMAKER несколько раз перезагружает компьютер и автоматически запускается после отработки файла AUTOEXEC.BAT). При начальном запуске MEMMAKER спрашивает, требуется ли память EMS для приложений, и, если нет, то установит ключ NOEMS в строке запуска EMM386.   
Для того чтобы в верхнюю память поместилось как можно больше модулей, ее следует оптимизировать. Оптимизация UMA подразумевает такое конфигурирование базовых адресов буферной и постоянной памяти адаптеров, при котором свободные области UMA получались бы максимально возможного размера. Нужно стремиться к плотному соединению областей UMA, занимаемых адаптерами. Тогда максимальный размер блоков UMB увеличится, и драйвер типа EMM386 сможет разместить в нем более крупные модули, что, в свою очередь, позволит освободить дополнительное место в остродефицитной стандартной памяти. Если на конфигурирование адаптеров не обращать внимания, то может оказаться, что два компьютера с одинаковым составом аппаратных средств и программным обеспечением будут после загрузки ОС иметь значительно различающиеся размеры стандартной памяти.   
Многие адаптеры (контроллеры SCSI, адаптеры локальных сетей и др.) позволяют задавать адреса областей встроенных RAM и ROM, отображаемых в пространство памяти компьютера. Это позволяет разрешать (или, наоборот, создавать) конфликты использования UMA, а также оптимизировать использование ее блоков. При конфигурировании устанавливаемых адаптеров необходимо исключить перекрытия занимаемых адресов, поскольку из-за этого, скорее всего, ни одно из конфликтующих устройств работать не будет. Самый неприятный (трудно устранимый) конфликт возникает, если при конфигурировании какого-либо адаптера его память перекрывает буфер или BIOS графического адаптера. Если конфигурирование осуществляется джамперами, то вернуть нормальную конфигурацию труда не составит. А если адаптер конфигурируется только с помощью специальной утилиты, изменяющей содержимое его энергонезависимой памяти, то изменить его конфигурацию можно, лишь загрузив и выполнив соответствующую утилиту. Но при конфликте с графическим адаптером это сделать не так-то просто – в лучшем случае удастся загрузить компьютер со «слепым» экраном, а в худшем – POST откажется продолжать тестирование и загрузку, обнаружив ошибку графического адаптера и сообщив об этом попискиванием динамика. Но эта ситуация не так безнадежна: есть еще в природе графический адаптер MDA, у которого видеобуфер не совпадает по адресам сраспространенными адаптерами EGA и VGA, а расширение BIOS у него отсутствует. Вставив неудачно сконфигурированную плату в компьютер с адаптером MDA (и, конечно, соответствующим монитором), можно утилитой задать правильную конфигурацию. Однако этот способ для новых компьютеров, не имеющих слотов шины ISA, уже непригоден.   
Кроме занимаемого пространства, оптимизация использования UMA касается, как ни странно, и быстродействия. Для областей памяти адаптеров часто бывает полезным применение теневой памяти. Управление теневой памятью осуществляется через CMOS Setup для определенных областей. Поэтому, размещая конфигурируемые области памяти, иногда следует учитывать возможности задания границ теневой памяти в CMOS Setup. Необходимо помнить, что Shadow ROM блокирует запись, a Shadow RAM игнорирует возможность изменения со стороны адаптера затеняемой области памяти, что приведет к ошибкам при некорректном использовании теневой памяти. Для процессоров 386 и выше теневую память может организовать и драйвер EMM386, но этой его функцией пользуются редко.   
Работу приложений в среде MS-DOS, в которой активно используется программный код BIOS, значительно ускоряет затенение ROM BIOS, как системной, так и BIOS графического адаптера и дискового контроллера. Для многозадачных ОС защищенного режима (Windows и др.) затенение ROM BIOS ускоряет только начальный процесс загрузки ОС, поскольку в рабочем режиме здесь в основном используются драйверы, загружаемые в ОЗУ.   
Приложения ОС Windows пользуются *виртуальной памятью*,и никакие старые спецификации EMS и XMS им не нужны. Суммарный объем виртуальной памяти, доступной всем приложениям, определяется размером ОЗУ и файлов подкачки (их может быть и несколько). В Windows 9x размер файла подкачки изменяется динамически, по мере потребностей системы. Для того чтобы приложениям хватало памяти, на диске, несущем файл подкачки, должно быть достаточно свободного пространства (десятки и сотни мегабайт). Конечно же, важен и объем установленной физической памяти – ее малый объем может быть принципиальным ограничением на запуск ряда приложений или установку операционных систем. При малом объеме ОЗУ свопинг (подкачка страниц) будет слишком интенсивным, в результате чего скорость работы приложений существенно снижается (обращения к диску выполняются на несколько порядков медленнее, чем к ОЗУ). Приложения реального времени (например, аудио- и видеопроигрыватели и тем более кодеры) могут стать неработоспособными именно из-за малого объема ОЗУ. Поскольку файл подкачки изменяет свой размер в процессе работы, важно следить за фрагментацией диска, несущего этот файл – обращение к фрагментированному файлу выполняется медленнее, чем к нефрагментированному. При выборе диска для размещения файла подкачки следует учитывать его быстродействие – время доступа и скорость передачи данных. При использовании приложений реального времени, интенсивно обменивающихся с дисками (те же проигрыватели и кодеры, а также программы, записывающие компакт-диски), по возможности файл подкачки следует размещать на других дисках.   
Если на компьютере под управлением ОС защищенного режима (Windows, Unix, OS/2...) перестают запускаться приложения с сообщениями о недостаточном объеме оперативной памяти – проверьте наличие свободного места на жестких дисках, используемых для подкачки. Если памяти не хватает приложениям MS-DOS – проверьте файлы AUTOEXEC.BAT и CONFIG.SYS и запустите утилиту MEMMAKER или выполните ручную оптимизацию памяти.   
Увеличение физического объема оперативной памяти в ряде случаев может привести к неожиданному снижению производительности компьютера. Это возможно, когда системная плата (или процессор со вторичным кэшем) не способна кэшировать весь объем ОЗУ. У многих системных плат для процессоров Pentium кэшируются только первые 64 Мбайт ОЗУ; у первых процессоров Pentium II кэшировались только 512 Мбайт. Память, выходящая за размеры кэшируемой области, конечно же, доступна, но ее производительность гораздо ниже кэшируемой. ОС Windows 9x распределяет память, начиная с верхней границы доступной памяти, причем наверх попадает ее ядро, скорость работы которого существенна для работы многих приложений. Если после увеличения ОЗУ ядро попадает в некэшируемую область, можно наблюдать снижение производительности. Для лечения этого недуга можно воспользоваться условно-бесплатной программой W2CACHE.COM, которая запускается в начале загрузки Windows и, оставаясь резидентной, «съедает» верхнюю часть памяти, заставляя ядро Windows загружаться в нижнюю, кэшируемую область. После окончания загрузки Windows программа освобождает занимаемую память, и ОС отдает ее в распоряжение приложений.

**Оптимизация и ускорение работы жесткого диска**

Наверно все пользователи хотят, чтобы их компьютеры могли выполнять требуемые действия намного быстрее, чем в текущий момент. Процесс обработки информации можно ускорить разными способами. К подобным вариантам относится оптимизация работы винчестера. Для подобного процесса в Виндовс имеется огромное количество настроек и инструментов. Давайте рассмотрим, как ускорить работу жесткого диска и в целом работу компьютера.

**Отключение индексирования**

Данный способ подойдет для тех, кто не использует функцию поиска в операционной системе.

Основой метода является закрытие раздела винчестера от индексирования, что обеспечит ускорение его работы. Для этого:

1. Переходим в «Мой компьютер». Там нужно вызвать контекстное меню на главном диске, зачастую это диск С:\. Затем нажимаем «Свойства».
2. В меню «Общие» выбираем пункт «Разрешить индексировать содержимое…». Нужно снять галочку и подтвердить действие.
3. После данных действий появится оповещение, где будет сообщаться про возможность использования новой политики только к определенным файлам. Выбираем «Продолжить», а затем «Пропустить все».

**Отказываемся от ведения журнала**

В стандартных настройках в системе имеется функция ведения журнала. Там отображаются все сбои и произведенные изменения. Если данную систему Вы не используете или даже не понимаете, что это значит, то можете смело ее отключить.

Инструкция по отключению:

1. Запуск командной строки. Наиболее простейший способ: нажатие «Win+R», а в появившемся окне «Выполнить» ввести «cmd».
2. Откроется приложение, где требуется ввести: fsutil usn deletejournal /D С:, где С – это буква необходимого жесткого диска.
3. Для подтверждения своих намерений нужно нажать на «Enter». Затем командную строку необходимо закрыть.
4. Для вступления в силу изменений, компьютер необходимо перезагрузить.

**Очистка от ненужных файлов**

Довольно часто показатели скорости работы жестких дисков заметно улучшаются после удаления ненужных файлов, которые за длительное время могли собраться на накопителе. Чем больше свободного пространства на винчестере, тем быстрее будет происходить поиск файлов, а это ускорит работу.

Если этот процесс рассмотреть более подробно, то получается, что система не будет производить долгий поиск в своих таблицах адреса определенных единиц информации. При удалении не нужных данных количество таких таблиц уменьшится, чем при заполненном пространстве.

Поэтому нужно чаще удалять файлы с компьютера, которые не используются. Если они важные, то можно просто их собрать в одну папку и заархивировать. Подобная процедура освободит место, а в параметрах основной таблицы будет только один адрес.

Чтобы очистить диски в Виндовс 7 и иных версиях, используются специальные инструменты. Они располагаются в меню «Пуск» в категории служебных программ. Точное название – «Очистка диска». Программа производит поиск неиспользуемых файлов в разделе, а затем предлагает пользователю их удалить.

Кроме этого существует большое количество программ, обеспечивающих проведение данной процедуры.

**Дефрагментация**

Когда производится запись новой информации на устройство, она не всегда размещается рядом. Размещение отдельных пакетов производится по свободному месту тома. После проведения многочисленных записей, одни файл может быть расположен по всему носителю. Это приводит к тому, что система постоянно обращается к разным секторам. Используя дефрагментацию, можно искать подобные данные и собирать их вместе. Затем система будет обращаться к соседним местам, что увеличит скорость работы.

Подобный инструмент способен помочь только при использовании HDD дисков.

Функция имеется в различных программах, которые предлагают сервисы по ускорению работы компьютера.

Операционные системы от Microsoft имеют возможность проводить дефрагментацию с использованием встроенных возможностей. Для этого в «Компьютере» необходимо выбрать необходимый диск и произвести запуск контекстного меню. Затем выбирается «Свойства», там пункт «Сервис» и указать «Оптимизировать». Затем ОС предложит выполнить дефрагментацию.

**Переразметка**

В случаях, когда компьютер используется длительное время и работает винчестер на полную мощностью, можно произвести переразметку. Это поможет при наличии двух и более разделов.

Это форматирование диска, а некоторые разработчики указывают, что заново «режется» жесткий диск на сектора и дорожки.

Идеальным является вариант, когда в системе применяется один жесткий диск с единственным томом. Нормальной считается система с двумя разделами. При большем количестве – система будет более медленно работать, тратя время на поиск необходимых файлов. В данном процессе поиск осуществляется по всем жестким дискам поочередно. Можно один жесткий диск использовать для установки ОС и софта, на другой записывать фильмы и прочие мультимедийные файлы.

**Перемещение файлов на съемный носитель**

Неплохим вариантом считается перемещение неиспользуемых данных на отдельные носители. Зачастую для этого применяются внешние накопители. Важно выбирать надежные внешние HDD, которые будут отличаться высокой скоростью обработки информации.

Изделия можно использовать не только для хранения устаревших файлов, но и в виде флешки. Для подобных целей 500 Гб будет достаточно. С компьютером он соединяется по USB интерфейсу и отличается бесшумной работой.

**Отключение автозагрузки программ**

Наличие лишних программ в автозагрузке существенно замедляет работу компьютера и использует его процессор и память. Если их отключить, то можно увеличить скорость работы диска и в целом производительность системы.

В разных версиях Виндовс управление автозагрузкой располагается в разных местах. Наиболее удобно для подобной цели использовать утилиту «CCleaner». Требуется осуществить запуск утилиты и перейти в меню «Сервис/Автозагрузка».

В автозагрузке нужно оставить только те программы, которые нужны для работы компьютера, например, антивирус. Остальные компоненты можно отключить кнопкой «Выключить» в правой панели.

Любую из отключенных программ можно всегда снова запустить вручную. Если пропадет значок в системном трее, то ее запуск можно вернуть с использованием кнопки «Включить».

**Обновление драйвера контроллера диска**

От актуальности драйверов зависит стабильность всей системы. Рекомендуется обновлять драйверы до новейшей версии, особенно это касается SSD дисков.

Для проверки версии установленных драйверов нужно зайти в «Диспетчер устройств». Необходимо вызвать командную строку, а в ней ввести «devmgmt.msc» и нажать «Enter». Затем нажать правой кнопкой мыши по SATA контроллеру и выбрать пункт «Свойства».

Если используется старый драйвер, то новый можно найти на официальном сайте производителя системной платы или ноутбука, скачать и обновить его.

Существуют специальные утилиты, которые позволяют обновить драйверы. Можно выбрать софт «Driver Booster», который при запуске покажет список драйверов, для которых требуется обновление.

Данная информация поможет Вам оптимизировать жесткий диск и повысить его работоспособность.

**Способы увеличения быстродействия программ**

Современные ЭВМ обладают очень большой мощностью. Скорость работы процессора (ЦП) современных ЭВМ измеряется гигагерцами, объём оперативной памяти гигабайтами, а современные интерфейсы устройств обеспечивают скорость обмена данными порядка, как минимум, нескольких сотен мегабайт в секунду. Производительность, которая ещё несколько лет назад казалась «сказочной» в настоящее время стала нормой жизни.

Однако параллельно росту мощности ЭВМ увеличивается и ресурсоёмкость приложений. У приложений совершенствуется функционал, интерфейс, возрастает объём обрабатываемых данных и как следствие системные требования. Поэтому вопрос об увеличении быстродействия приложений не теряет своей актуальности.

**Общие вопросы быстродействия программ.**

Быстродействие программ (ПО) зависит от многих факторов, но основными из них являются два:

* Соотношение между реальными системными требованиями ПО и существующей аппаратной конфигурацией ЭВМ;
* Алгоритмы работы ПО.

Если низкое быстродействие обусловлено первым фактором, то решением является модернизация аппаратной части (hardware). В некоторых случаях проблему можно решить также с помощью тонкой настройки hardware и операционной системы. Однако этот путь имеет ряд недостатков:

1. Увеличивается производительность hardware, а вовсе не быстродействие ПО;
2. Производительность hardware ограничена возможностями существующих в данный момент элементной базы и инженерных решений в данной области;
3. Большие финансовые затраты на модернизацию и настройку по причине высокой стоимости комплектующих ЭВМ и услуг специалистов требуемой квалификации.

По этим причинам при разработке ПО прибегают к увеличению его быстродействия с помощью различных средств программной инженерии. Это позволяет:

1. Обеспечить работу нового ПО на уже существующем оборудовании;
2. Разработать масштабируемое ПО;
3. Значительно уменьшить финансовые и трудовые затраты при внедрении.

Вместе с тем и у этого пути имеется ряд недостатков:

1. Значительно усложняется процесс разработки ПО, так как более «быстрые» алгоритмы сложнее более «медленных» (на пример алгоритм бинарного поиска сложнее, чем алгоритм линейного поиска);
2. Реализация более сложных алгоритмов, как правило, требует привлечения специалистов более высокой квалификации;
3. В случае работы с большими объёмами данных или выполнении задач требующих больших и сложных вычислений, ресурсоёмкость ПО всё равно остаются достаточно высокой. Несмотря, на какие либо способы увеличения быстродействия.

Таким образом, в общем случае обеспечение быстродействия ПО является комплексной задачей.

Однако следует заметить, что среди существующих задач, очень немногие обладают высокой ресурсоёмкостью. Вследствие этого в большинстве случаев не требуется никаких действий относительно hardware и требуемого результата можно достичь, прибегая только к программной инженерии.

Программная инженерия предоставляет несколько способов увеличения быстродействия программ. Рассмотрим их на примере языков программирования Delphi и Assembler.

**Увеличение быстродействия программ.**

Как было показано в предыдущем параграфе, можно увеличить быстродействие ПО соответствующим образом реализовав его алгоритмы. Количественным показателем быстродействия алгоритма (а, следовательно, и ПО) является время его выполнения, измеренное по специальной методике, так называемого профилирования. Таким образом, в общем случае выбор наиболее «быстрых» алгоритмов сводится к измерению времени их выполнения и сравнении полученных результатов между собой. Такой способ анализа быстродействия является наиболее объективным. На протяжении многих лет программистами был накоплен большой опыт профилирования, который позволяет сделать определённые выводы относительно возможности оптимизации быстродействия ПО ещё на стадии написания.

Эти выводы были обобщены и представлены в виде определённых рекомендаций. Если программист будет следовать данным рекомендациям, то написанная программа вероятнее всего будет обладать большим быстродействием, чем в случае их игнорирования. Однако следует ещё раз подчеркнуть, что достоверные сведения о быстродействии может дать только профилирование. Это обусловлено тем, что быстродействие алгоритма определяет в первую очередь его конкретная реализация. Кроме того необходимо ещё раз отметить, что в отношении увеличения быстродействия ПО программная инженерия не всесильна.

В чём же состоят выше упомянутые рекомендации? Их краткое содержание применительно к языку программирования Delphi приведено ниже.

1. При написании кода программ рекомендуется избегать процедур, состоящих из сотен строк. Практически всегда в них можно выделить блоки, которые лучше оформить в виде отдельной процедуры. Возможно, позже вы ей даже воспользуетесь где-то в другом месте. Не говоря уже о том, что это повышает понимание программы и вами, и другими программистами. К тому же так проще искать «узкие» места в программе.
2. Использование оператора case (switch) вместо многократных if… then… else (if… else). Во втором варианте компилятор будет выполнять проверку условия столько раз, сколько у вас вариантов. В первом проверка выполняется лишь однажды.
3. Некоторые действия могут быть довольно продолжительными, поэтому рекомендуется выносить за рамки цикла всё, что можно выполнить вне его, чтобы избежать большого числа повторений внутри цикла.
4. В циклах типа for нужно стараться, чтобы значение счетчика уменьшалось до нуля, а не наоборот — начиналось с нуля. Это связано с особенностями процессора. Сравнение с нулём выполняется гораздо быстрее, чем с другим числом.
5. Пользоваться типом Variant только при необходимости. Операции над этим типом сложнее, чем, например, над Integer или String.
6. Не злоупотреблять «программированием на компонентах». В частности не использовать компонент TTreeView для хранения древовидных структур данных — он работает очень медленно и предназначен только для визуального отображения. В случае работы со структурами данных лучше использовать алгоритмы, созданные самостоятельно на основе фундаментальных.
7. Сохранение и загрузка свойств компонентов с помощью методов ReadComponent и WriteComponent работает довольно медленно, поэтому по возможности рекомендуется сохранять и восстанавливать состояние программы между сеансами при помощи других способов.
8. Заменить простой в реализации алгоритм на более сложный, но с большим быстродействием. Например, если заранее известно, что в списке для поиска будет много элементов, лучше его отсортировать и применять бинарный поиск вместо линейного.
9. В критических с точки зрения быстродействия местах программы делать вставки на ассемблере. Команды ассемблера напрямую транслируются в машинный код. Таким образом, в отличие от высокоуровневых языков при компиляции отсутствует проблема синхронизации и ряд других негативных обстоятельств.

Для других языков программирования вышеприведённый список может несколько отличаться, в частности отсутствием поддержки ассемблера и как следствие возможности оптимизации с его помощью (Java, Visual C# и др.).

Особо следует отметить, что рекомендации 3 и 4 применяются не только для языков высокого уровня, но и для ассемблера. Помимо вышеуказанных для увеличения быстродействия программ написанных на ассемблере, в том числе и вставок, существуют следующие рекомендации:

1. Замещение универсальных инструкций на учитывающие конкретную ситуацию, например, замена умножения на степень двойки на команды сдвига (отказ от универсальности).
2. Уменьшение количества передач управления в программе: за счет преобразования подпрограмм в макрокоманды для непосредственного включения в машинный код; за счет преобразования условных переходов так, чтобы условие перехода оказывалось истинным значительно реже, чем условие для его отсутствия; перемещение условий общего характера к началу разветвленной последовательности переходов; преобразование вызовов, сразу за которыми следует возврат в программу, в переходы («сращивание хвостов» и «устранение рекурсивных хвостов») и т.д.
3. Максимальное использование всех доступных регистров за счет хранения в них рабочих значений всякий раз, когда это возможно, чтобы минимизировать число обращений к памяти, упаковка множественных значений или флагов в регистры и устранение излишних продвижений стека (особенно на входах и выходах подпрограмм).
4. Использование специфических для данного процессора инструкций, например, инструкции засылки в стек непосредственного значения, которая имеется в процессоре 80286 и более поздних. Другие примеры – двухсловные строковые инструкции, команды перемножения 32-разрядных чисел, деление 64-разрядного на 32-разрядное число и умножение на непосредственное значение, которые реализованы в процессорах 80386 и 80486. Программа должна, разумеется, вначале определить, с каким типом процессора она работает!

**Заключение.**

Методы оптимизации быстродействия, рассмотренные в этой статье, были выработаны и проверены не одним поколением программистов и уже стали классическими. В то же время информационные технологии, в частности технологии программирования, постоянно развиваются. Появляются новые технологии, старые – модернизируются или уходят в прошлое. Растёт производительность аппаратной части ЭВМ и параллельно с этим растёт сложность и ресурсоёмкость выполняемых ими задач.