3. Среда исполнения программ. Программа в среде Microsoft Windows

Кому и командная строка –
дружественный интерфейс.
Программистский фольклор

В предыдущих главах мы попытались разобраться с тем, зачем нам нужна вычислительная техника, что такое алгоритм, программа и чем они отличаются друг от друга, какими инструментами мы в настоящий момент располагаем для того, чтобы эти самые программы создавать. Также мы должны были осознать, почему техника без программ представляет собой лишь мертвую "груду железа", а программы без своего воплощения – более или менее строгую (чаще менее) абстракцию. Очевидный следующий шаг – начать изучать, как же собственно правильно превратить алгоритмическое решение конкретной задачи в текст программы на выбранном языке программирования. Однако к этому мы перейдем в следующей главе, а здесь рассмотрим, достаточно кратко, еще один весьма важный вопрос.

Вообще говоря, в контексте обсуждения методов программирования словосочетание "*вычислительная техника*" требует расшифровки. Вроде бы очевидно, что к вычислительной технике относятся компьютеры. А что еще? Можно ли считать "вычислительной" стиральную машину с программным управлением? А цифровой фотоаппарат? А сотовый телефон? Ведь их назначение вовсе не в том, чтобы складывать и умножать числа. Однако это точка зрения потребителя. А для разработчика программного обеспечения, к каковым, мы надеемся, желает присоединиться и Читатель, важно лишь то, способна ли та или иная техника выполнять программы, поскольку, если способна, то кто-то должен для нее эти самые программы создавать. К счастью, при всем неизмеримом многообразии видов и моделей современной техники написание программ для нее основано на тех же базовых принципах, которые используются при работе с классическим "вычислителем", более знакомым всем под именем "компьютер". Итак, *с точки зрения программиста к вычислительной технике относится все, что имеет возможность выполнять программы*.

Что нужно для того, чтобы программа, которая, как мы уже должны были усвоить, есть выраженный на языке программирования алгоритм, могла быть выполнена? Вроде бы ответ очевиден – нужен тот, кто способен шаг за шагом (инструкцию за инструкцией) выполнять сформулированные в алгоритме действия. Поскольку действий много, нам потребуется место для их хранения и последующего считывания. Кроме того, любая программа оперирует данными (входными и результирующими) – их тоже необходимо хранить. Наконец, входные данные программе обычно поставляет человек, он же "забирает" результаты, а, значит, требуются средства ввода/вывода (обмена информацией).

Здесь, кстати, стоит упомянуть о глобальном противоречии, которое до сих пор определяет развитие всей программной индустрии – удобные способы представления информации у человека и у компьютера, мягко говоря, весьма различны. Человек свободно оперирует образами:

это тигр, а это кот, хотя те же “усы, лапы и хвост”; вот эта конструкция о четырех ногах, вон та на колесиках и даже та, что с одной вычурно изукрашенной подставкой – все это стол. Для компьютера же информация, а еще точнее данные, есть всего лишь последовательность (короткая или длинная) нулей и единиц. В самом начале компьютерной эры (по меркам истории человечества буквально вчера – каких-то полвека назад) мощности ЭВМ едва хватало на то, ради чего их создавали – помочь человеку в выполнении численных расчетов, к которым, так или иначе, сводятся большинство реальных задач. Естественно, что как особ королевских кровей, ЭВМ освобождали от всех побочных дел, вроде перевода информации из вида удобного человеку в вид, понятный машине – на их долю оставались чистые вычисления. Однако, подобно тому как на подрастающих детей родители начинают перекладывать обязанности по уходу сначала за собой, а потом и за семьей в целом, так и на долю компьютеров с ростом их мощности падало все больше и больше задач, не связанных напрямую с выполнением расчетов. И если когда-то программирование велось в машинном коде, потом на ассемблере, затем на языках высокого уровня, то сейчас компьютер пытаются научить “понимать” обычную человеческую речь. Вполне возможно, что в будущем основным занятием программиста будет не “стучать по клавишам”, а с не меньшей скоростью “молоть языком”.

Повышение уровня “дружественности” компьютера к человеку ведется, конечно же, не только в области средств разработки программ, а точнее даже не столько, сколько в области прикладного использования компьютера как еще одного инструмента в руках человека. Без этого компьютер никогда не занял бы того места рядом с нами, которое он занимает сейчас, и так и остался бы инструментов “высоколобых” ученых. И в числе прочих эффектов это повышение дружественности привело к появлению целого класса специальных обслуживающих программ, реализующих промежуточный слой между “голым железом” и “полезными” программами, теми, что выполняют конкретные задачи пользователя. В результате возникла настоящая “среда обитания” программ, что привело, в свою очередь, к существенному усложнению самого процесса программирования, который люди снова начали упрощать, видимо, до следующего витка спирали.

Таким образом, цель данной главы – разобрать, в максимально облегченном варианте, из чего в настоящий момент складывается рабочая среда, в которой выполняется любая прикладная программа, с чем ей приходится взаимодействовать, и что, в конечном счете, должен иметь в виду программист, желающий чтобы написанная им программа не просто выполняла то, что от нее требуется, но и делала это по возможности эффективно.

Некоторые из изложенных здесь сведений пригодятся нам в дальнейшем, другие более не будут в этой книге затронуты, однако, хоть для того, чтобы водить автомобиль и не обязательно знать устройство двигателя внутреннего сгорания, но представлять себе его функциональные обязанности, возможности и потенциальные проблемы все же весьма полезно.

Процессор

Люблю отнять я и прибавить,
люблю прибавить и отнять.
Госпожа Беладонна

Еще раз напомним: изначальное предназначение вычислительной техники, в точном соответствии с названием, – выполнение расчетов, следовательно, главный функциональный элемент любого вычислительного устройства должен… что? Считать? Да. Но не только. Вторая не менее важная задача – управление (здесь вполне уместна аналогия с мозгом человека – он тоже одновременно является и центром принятия решений и центром управления). Управлять

приходится самим собой, устройствами хранения инструкций и данных, устройствами ввода/вывода, и всеми остальными “участниками концессии”. Итак, знакомьтесь…

> Процессор, он же центральный процессор, он же ЦП, он же Central Processing Unit, он же CPU, он же “проц”, он же “камень” – основное *действующее* лицо любой вычислительной системы. Точен, исполнителен, трудолюбив, имеет привычку “гореть на работе”. В силу внутреннего устройства понимает только двоичную логику. Должностные обязанности: “координатор” и “вычислитель”. Слабости: при выполнении некорректных программ имеет тенденцию к “зацикливанию”, жить не может без “мамы”[1].

Современный процессор представляет собой очень сложное устройство – даже упрощенная структурная схема содержит десятки элементов. Однако не пугайтесь. Сколько-нибудь подробное рассмотрение этого вопроса выходит за рамки данной книги. Мы посвятим несколько слов лишь тем элементам и функциям процессора, понимание которых будет полезно нам в дальнейшем.

Но прежде небольшое отступление. Вспомним, какие виды чисел известны человечеству. Натуральные, целые, рациональные, иррациональные, вещественные (они же действительные). Возможно, напрягшись, кто-то припомнит еще и замечательные комплексные числа. Из всего, что перечислено, в компьютере с его двоичной системой счисления[2] достаточно просто могут быть представлены лишь натуральные числа, чуть посложнее с целыми отрицательными, а вот с вещественными совсем плохо (про комплексные же и вовсе умолчим – работа с ними это уже высшая математика, причем как в прямом, так и в переносном смысле – ни в одном из известных авторам процессоров работа с комплексными числами “в железе” не реализована). Из математического анализа известно, что количество вещественных чисел бесконечно не только, так сказать, “в длину” (вправо и влево по координатной оси), но и “в глубину”, то есть на любом сколь угодно малом отрезке [a, b] вещественных чисел также бесконечно много[3]. Таким образом, если целые числа “растут” только в одну сторону, слева от десятичной точки, то вещественные еще и вправо от нее. В результате для представления вещественных чисел используется специальный формат, получивший название “*число с плавающей запятой*”.

Из предыдущего нетрудно заключить, что в процессоре *вычислительных* блоков должно быть как минимум два: один для целых чисел, один для вещественных. На самом деле и тех и других (блоков) побольше, чем по одному, но это уже тема другой книги. Итак, запомним: “*Целые и вещественные числа обрабатываются в процессоре по-разному и даже в разных его частях*”. И еще одно: “*Кроме целых и вещественных чисел процессор ничего другого обрабатывать не умеет*”. К этим двум положениям мы не раз будем возвращаться в дальнейшем, разбирая вопросы представления и интерпретации данных.

С расчетами разобрались. Теперь вторая задача процессора – управление. Каждая программа содержит набор команд (инструкций), указывающих процессору, что он должен сделать, откуда взять исходные данные, куда поместить результат. Понятно, что каждая такая команда должна быть процессору известна, то есть содержаться в его *системе команд*. Общее их количество относительно невелико (в последних процессорах компании Intel – пара сотен). Условно все команды можно разделить на два типа: *вычислительные* и *управляющие* (системные). С вычислительными вроде все ясно. А управляющие зачем? Представим, что у нас имеется горячее желание заставить процессор сложить два числа. Мы находим в списке команд ту, которая отвечает за сложение (не забывая, что для целых чисел команда будет одна, а для вещественных совсем другая), и… Возникает законный вопрос: Как же нам объяснить процессору, какие

---

[1] “мама” – материнская плата (motherboard).

[2] кто не в курсе, что это такое, подождите до следующей главы – пока достаточно знать, что компьютер в состоянии оперировать только двумя цифрами: нулем и единицей.

[3] да простят нас математики за столь вольную трактовку.

именно числа складывать? “Поговорить” с ним напрямую не удастся. Мы оперируем словами, процессор электрическими импульсами – не поймем друг друга. Вспоминаем, что вроде бы где-то слышали о том, что компьютер имеет устройство ввода информации. А, так вот же оно – клавиатура! С криком “Эврика!” нажимаем несколько цифровых клавиш и… Ничего не происходит. Даже с помощью клавиатуры передать данные процессору напрямую невозможно. Как именно это сделать – позднее, а пока допустим, что мы все-таки сумели. Следующий вопрос: куда их положить, чтобы процессор сумел их “достать”? К счастью, в самом процессоре существует *блок регистров* – часть процессора, специально предназначенная для хранения данных. Регистров этих немного, да и размер их невелик, но для нашей суперзадачи их хватит. Итак, прежде чем мы сможем выполнить команду сложения двух чисел, их необходимо *загрузить* в регистры, вот для этого, в частности, нам и понадобятся системные инструкции. Естественно есть множество других ситуаций, в которых процессор должен выполнять не вычисления, а функции управляющего, “говоря” остальным устройствам компьютера: “пойди туда”, “сделай то”, “подай это” и т.д.

Следующий важный момент – поступление данных в процессор. Взаимодействие с внешним миром процессор производит через *шину данных* – набор соединений, по которым данные в двоичной системе передаются в виде электрических сигналов. Чем больше соединений, тем большими порциями может передаваться информация. Размер “порции” называется *разрядностью шины*. Например, в процессоре Intel® Pentium® 4 шина данных 64-разрядна, то есть за единицу времени этот процессор может воспринять (или передать) блок из шестидесяти четырех нулей и единиц.

Процессор – устройство с дискретным “восприятием” времени. Моменты времени для него следуют друг за другом в виде *тактов*, в каждый такой *такт* “помещается” одно элементарное действие. Чем мельче такт, тем быстрее “живет” процессор, а значит, тем выше его быстродействие. Такт современных процессоров составляет менее одной миллиардной секунды. Такие числа не слишком удобны для восприятия, поэтому быстродействие процессора принято характеризовать его *тактовой частотой* – числом тактов в секунду (герц). Таким образом, конкретная модель процессора может быть описана, например, так: процессор Intel® Pentium® 4 с тактовой частотой 3,4 Ггц (гигагерц или GHz).

Оперативная память

Где-то далеко в памяти моей…
Слова из песни

В каждый момент времени (такт) процессор работает ровно с одной командой и средствами хранения команд не обладает[1].

Большинство команд представляют собой операции над некоторой порцией данных. Как мы выяснили выше, небольшое количество данных процессор все-таки в состоянии разместить “внутри себя”, в регистрах. Однако, кроме самых простейших случаев, регистров для хранения всех данных, которые должны быть обработаны в конкретной программе, недостаточно. Таким образом, и сами данные и команды для их обработки должны быть куда-то записаны. Из этого “куда-то” процессор будет их извлекать, выполнять указанные действия и в это же “куда-то” сохранять полученные результаты. Итак, знакомьтесь…

[1] На самом деле это не совсем верно, а точнее совсем не верно, однако с точки зрения программиста ситуация именно такова, поскольку никаких средств прямого использования внутренней памяти процессора (кэш-памяти) нет.

Оперативная память, она же ОП, она же Random Access Memory, она же RAM, она же "мозги" – второе по важности *действующее* лицо вычислительной системы. Как всякая женщина, весьма непостоянна – при выключении питания "забывает" все, что содержала. В отличие от процессора представляет собой совокупность физических и программных элементов. Должностные обязанности: "хранитель оперативной информации". Для любой системы справедлив лозунг: "Памяти много не бывает".

Дать точное определение *оперативной памяти* весьма непросто. Если процессор – это конкретное физическое устройство, которое можно подержать в руках, то на вопрос, что такое ОП, разные специалисты ответят Вам совершенно по-разному. Сборщик компьютеров скажет, что оперативная память – это одна, две или три небольших по размеру платы, устанавливаемые в соответствующие слоты на материнской плате. Программист, создающий прикладные программы, сообщит Вам, что для него оперативная память есть место хранения данных, при этом порции данных имеют имена, а размещение данных и связывание с именами осуществляется системой программирования. Программист, разрабатывающий обслуживающие (системные) программы, с уверенностью скажет, что оперативная память – это адресное пространство, то есть непрерывная последовательность пронумерованных ячеек одного и того же размера, в которых размещается код программы и обрабатываемые данные.

Каждое из приведенных определений отражает некоторые аспекты понятия оперативной памяти.

Долговременное хранение информации

Как уже было сказано, оперативная память хранит информацию, только пока на нее подается питание. Очевидно, в компьютере должно присутствовать и устройство, способное "не забыть" информацию, если вдруг (какой ужас!) кто-то выдернет "шнур из розетки". Итак, знакомьтесь…

Жесткий диск, он же винчестер, он же Hard-Disk Drive, он же HDD, он же "винт" – самое ценное *действующее* лицо вычислительной системы. Нетороплив (с точки зрения процессора и даже оперативной памяти), всегда готов принять на хранение любые Ваши секреты. Впрочем, при беспечном отношении с той же легкостью отдаст их кому-нибудь другому.

В отличие от двух предыдущих элементов вычислительной системы устройства жесткого диска мы даже касаться не будем. С точки зрения программиста значение имеет лишь то, *как* осуществляется долговременное хранение информации. Ключевым понятием в этом процессе является *файл*. Дать четкое определение этому понятию также не просто, как и оперативной памяти.

Итак, приближение первое: *файл – это поименованная область на диске*. Любой файл имеет имя. Чаще всего существуют ограничения на длину имени и допустимые символы, из которых оно может быть составлено. Любой файл некоторым образом располагается на диске, имеет начало (в "системе координат" жесткого диска) и длину в байтах (или более крупных единицах).

Приближение второе: нередко файл записывается на диск частями и в разные моменты времени, как следствие физически он может состоять из отдельных фрагментов дискового пространства. Таким образом, более точно можно сказать, что *файл – последовательность областей диска, логически связанных и имеющих общее имя*.

Подавляющее большинство долговременно хранящейся информации представляется в виде файлов, в том числе и сами программы.

На этом мы закончим наш краткий экскурс в мир компьютерного “железа” и перейдем к “софту” – программному обеспечению.

Классификация программных средств

Существенная часть потребностей современного человека связана с восприятием и обработкой информации во всех ее видах: от текстового до мультимедийного, от формул до музыки.

Поскольку компьютер представляет собой универсальное вычислительное средство, то есть способен выполнить любую программу, которая может быть составлена на основе системы команд процессора, естественным является тот факт, что программ, удовлетворяющих наши потребности существует несколько больше, чем одна. А там, где есть множество объектов, человека всегда неодолимо тянет создать их классификацию. Итак, знакомьтесь…

Программное обеспечение, оно же ПО, оно же Computer Software, оно же “софт” – совокупность всех программных средств, имеющихся в данной вычислительной системе. Если “железо” вычислительной системы – это мозг, то программы, хранящиеся на диске, – умения и навыки, а программы выполняющиеся – мысли.

Вернемся к задаче о сложении двух чисел. Есть команда процессора, есть оперативная память, в которую нужно разместить аргументы, есть клавиатура, на которой можно их набрать, есть жесткий диск, куда можно сохранить результат. Как связать все это в единое целое? Программа для выполнения этой простейшей операции должна быть способна обработать электрические импульсы от клавиатуры, “перекодировать” их в числа, разместить эти числа в оперативной памяти, передать адреса в команду сложения, получить адрес результата, считать его, найти место на жестком диске, куда записать файл с результатом, осуществить запись. Это если вкратце. Понравилось? Как Вы думаете, какой процент от размера такой программы составит собственно команда сложения?

Очевидно, что взаимодействие с аппаратурой вычислительной техники, – задача, которую необходимо решать в любой программе, – должно быть запрограммировано отдельно, то есть между программами, выполняющими задачи пользователя системы, и аппаратурой должен располагаться промежуточный слой из специальных обслуживающих программ. Этот промежуточный слой называется *системное программное обеспечение*. Все остальные программы относятся к *прикладным*. Классы программ пересекаются, то есть одна и та же программа может быть в зависимости от точки зрения отнесена либо к системным, либо к прикладным.

В качестве примера: в системном программном обеспечении существуют так называемые *драйвера* – программы, осуществляющие эффективное управление функциональными блоками вычислительной системы, например, жестким диском. Изготовление драйверов обычно берет на себя фирма-производитель.

Операционная система

Задача обеспечения удобного взаимодействия человека и вычислительной системы уже довольно давно ставится во главу угла при разработке программного обеспечения. При этом первые реальные пользователи вычислительной системы – это программисты. А среди программистов первыми “вступают в контакт” с аппаратурой программисты системные – создатели программ системного уровня. Однако при всем нашем уважении к “системщикам”, программистов прикладных все же существенно больше. Прикладные программисты пишут программы на языках высокого уровня, обычно способны путем некоторых манипуляций с набором исходных текстов получить исполняемый модуль, готовый к запуску, и ожидают, что все дальнейшее возьмет на себя кто-то еще. Итак, знакомьтесь…

Операционная система, она же ОС, она же Operating System, она же OS, она же “операционка”, она же “ось” – основное *управляющее* лицо любой современной вычислительной системы. Представляет собой совокупность программных средств системного уровня. Должностные обязанности: обеспечение взаимодействия пользователя и вычислительной системы, обеспечение эффективного использования ресурсов последней, организация надежного функционирования программного обеспечения.

Устройство современных операционных систем не менее, а скорее даже более сложно, чем устройство самих вычислительных систем. Им посвящены книги не одну сотню страниц толщиной. Мы здесь отметим лишь два существенных момента. Пользователю вычислительной системы ОС предоставляет средства управления, программисту – набор функций системного уровня (так называемый API – Application Programming Interface).

Операционные системы семейства Windows

Среди операционных систем семейство Windows занимает особое положение. Подавляющее большинство персональных компьютеров в мире работают под управлением различных версий этой ОС. Не вдаваясь в детали, отметим основные моменты, которые необходимо иметь в виду программисту при работе с Windows.

Прежде всего, Windows – операционная система с графическим интерфейсом пользователя. Это означает, что существенную часть кода Вашей программы будет составлять обеспечение соответствующего внешнего вида. Правда, для любителей, а также для ситуаций, в которых пользовательский интерфейс не имеет существенного значения, существует возможность создания приложений консольных, функционирующих в старом добром текстовом режиме.

Во-вторых, Windows – система многозадачная, а значит, при написании программы Вы не можете рассчитывать на “единоличное” использование ресурсов вычислительной системы: процессорного времени, оперативной памяти, экрана монитора и т.д. Правда, существенную часть необходимой работы для обеспечения разделения ресурсов берет на себя сама операционная система.

В третьих, Windows любой программе при запуске предоставляет “отдельное” линейное адресное пространство, размером в 4 Гб. Поскольку большинство компьютеров обладают меньшим объемом оперативной памяти, в реальности это пространство обеспечивается механизмами поддержки виртуальной памяти (с использованием жесткого диска).

Источники информации

1. Большая советская энциклопедия
2. Юров В., “Assembler”, – СПб.: Питер, 2002
3. Касперски Крис, “Техника оптимизации программ. Эффективное использование памяти”, – СПб.: БХВ-Петербург, 2003
4. Олифер В.Г., Олифер Н.А., “Сетевые операционные системы”, – СПб.: Питер, 2002
5. “Intel Architecture Software Developer’s Manual. Volume 1: Basic Architecture”, – Intel Corporation, 1997
6. Брэдли Д. “Программирование на языке ассемблера для персональной ЭВМ фирмы IBM”, – М.: Радио и связь, 1988
7. http://www.allcompinfo.com