О микроконтроллерах, студентах, и не только о них...

Александр Фрунзе

alex.fru@mtu-net.ru

Некоторое время назад в беседе со мной один из уважаемых мной преподавателей МИФИ высказал мнение, на мой взгляд, весьма спорное. Он сказал, что в области программирования в учебном курсе по микроконтроллерам студентам надо давать только язык С. Аргументация стандартная: во-первых, программы на языке высокого уровня пишутся быстрее, чем на ассемблере; во-вторых, ассемблеры для разных микроконтроллеров разные, а язык С — один, поэтому, освоив его, можно писать программы для любого МК, не теряя времени при переходе с одного МК на другой; в-третьих, научить студента языку высокого уровня проще, чем обучить ассемблеру. И, в-четвертых, в Америке именно так и учат! Что же касается недостатков языка высокого уровня (более медленное выполнение программы, чем в случае, когда программа написана на ассемблере, и гораздо больший объем занимаемой памяти), то это перестало быть проблемой — микроконтроллеры сегодня производительны как никогда, да и объем памяти у них составляет сейчас десятки, а иногда и сотни килобайт.

Подобное мнение довольно распространено и имеет право на существование. Но меня несколько настораживает то, что оно является доминирующим в преподавательской среде. А поскольку наша высшая школа, несмотря на все ее недостатки, все же работает и продолжает готовить специалистов, то высказанное мнение — не лозунг, а реальность, присутствующая в обучении студентов. И, на мой взгляд, реальность не лучшая, а требующая определенной корректировки. Я хочу критически проанализировать вышеупомянутые (и некоторые близкие к ним) аргументы в пользу преимущественного обучения языкам высокого уровня. А что из этого следует, судите сами.

Итак, первый аргумент: программы на языке высокого уровня пишутся быстрее, чем на ассемблере. Так ли это? Действительно ли, что если мы будем программировать на С, а не на ассемблере, мы всегда сэкономим время?

На самом деле ответ вовсе не очевиден и не однозначен. Во-первых, давайте учтем, что написание программы — это не только написание одной под другой ассемблерных команд или операторов того или иного языка. Написанию команд неизбежно предшествует этап продумывания структуры программы, последовательности совершаемых ей действий, алгоритмов пересылки и обработки данных, ввода и вывода информации. И это продумывание — неотъемлемая часть программирования. Программист, который не решает упомянутых задач, а лишь почти бездумно пишет одну за другой команды — фикция. Даже если такие программисты и существуют, они занимаются чем-то иным, но не программированием — кому такие программисты нужны, кто должен думать за них?

Полагаю, нет необходимости говорить, что упомянутое продумывание практически не зависит от того, на каком языке программист собирается писать программу. И это очень основательно подрывает рассматриваемый аргумент. По моему опыту, продумывание программы или ее фрагмента отнимает в 5–10 раз больше времени, чем перевод продуманного в конечные коды. Поэтому, даже если бы действительно перевод продуманного алгоритма в последовательность команд высокого уровня был бы быстрее, чем перевод в ассемблерную программу, речь шла бы лишь о том, что в первом случае лишь 10–20% работы выполняется со скоростью, от силы в полтора раза более высокой, чем во втором. Так что суммарный выигрыш во времени, если он действительно есть, составит единицы процентов от затрат на всю работу. Если к этому добавить, что очень часто в микроконтроллерных программах объем вычислительных операций, где сказываются преимущества языков высокого уровня, относительно невелик, и в то же время может быть довольно много операций формирования различных импульсов и их последовательностей (в обоих классах языков они формируются с одинаковыми затратами), то выигрыш во времени за счет написания программы на языке высокого уровня и вовсе выглядит эфемерным.

Однако, и это не все. Те, кто занимается реальным программированием, хорошо знают, что мало написать и скомпилировать программу, да зашить ее в микроконтроллер. Ее перед этим еще нужно отладить. А занимает процесс отладки программы часто не меньше времени, чем процесс первоначального ее продумывания. Причем вне зависимости от того, на каком языке написана программа. Более того, если отладочные средства, идущие в комплекте с языком высокого уровня, продуманы недостаточно хорошо и вам при отладке приходится разбираться в ассемблерных командах, сформированных вашим компилятором языка высокого уровня, то процесс такой отладки займет даже больше времени, чем если бы вы написали программу на ассемблере и взялись бы отлаживать именно ее. Не говоря уже о том, что и в том, и в другом случае вам все равно нужно знать ассемблер.

В общем, как ни крути, программирование — это довольно сложный многоступенчатый процесс, начинающийся с момента постановки задачи и за-

вершающийся моментом окончания отладки работающего изделия. Процесс этот для микроконтроллеров плохо формализован в силу большого разнообразия доступных МК и относительно малого количества софтверных компаний-разработчиков средств поддержки разработки. Так что обмануть природу обычно не удается — мало того, что использование языков высокого уровня может ускорить процесс разработки только за счет не очень значительного сокращения одного из его этапов, причем не самого трудоемкого, так еще это сокращение иногда имеет следствием увеличение объема работы на этапе отладки разработанной программы. Другими словами, в области программирования микроконтроллеров скорость преобразования поставленной задачи в работающие коды оказывается практически не зависящей от того, на каком из языков решается задача, но в то же время очень сильно зависящей от квалификации разработчика.

Предвижу возражения типа «попробовал бы ты написать на ассемблере Windows или какую-нибудь стрелялку (леталку, ходилку, ездилку) для компьютера». Сразу скажу, сам не буду и другим не посоветую. Но еще раз подчеркну, что упомянутое возражение относится совсем к другому программированию. Написание программ для ПК и для МК сегодня имеет гораздо больше различий, чем общего, и то, что программы для ПК уже практически невозможно писать на ассемблере, вовсе не является основанием для того, чтобы не програмировать на ассемблере микроконтроллеры. Все, о чем я говорю, относится к области программирования микроконтроллеров. Программирование для ПК, хотя и включает в себя те же этапы, но предполагает использование иных средств разработки и отладки программ, с иными, чем в микроконтроллерах, уровнями формализации в тех или других языках. Поэтому и выводы для программирования ПК будут совсем другими. Кстати, так же, как и в случае с цифровыми сигнальными процессорами, — там тоже пишутся программы, но то программирование также заметно отличается от программирования МК в силу особенностей средств поддержки разработки для ЦСП.

Второй аргумент: ассемблеры для разных микроконтроллеров разные, а язык С — один, поэтому, освоив его, можно писать программы для любого МК, не теряя времени при переходе с одного МК на другой. Действительно, диалекты одного языка высокого уровня от разных производителей различаются гораздо меньше, чем ассемблеры для разных микроконтроллеров. Но опыт очень многих разработчиков свидетельствует о том, что, разобравшись хотя бы однажды с аппаратным устройством и ассемблером одного МК, мы без особых проблем осваиваем и другой. Время освоения может быть разным — от полумесяца-месяца (если нас подпирают обстоятельства) до полугода-года (когда не горит, а лишь душа желает чего-то еще). Но, повторяю, проблема освоения второго языка программирования (после того, как освоен первый) — надуманная. В качестве иллюстрации приведу небольшой пример. Году в 1994-м я предложил моему сыну, тогда еще старшекласснику, неплохо знавшему легендарный Spectrum и ассемблер микропроцессора Z80, адаптировать программы арифметики с плавающей запятой (они были написаны на ассемблере 8080) под МК семейства x51. Он справился с этим менее чем за полгода, о чем в свое время была написана статья в журнале «Радио». Как видите, школьник, далеко не профессионал — и ничего, освоил новый для себя ассемблер менее чем за полгода, в промежутках между школьными уроками и прочими увлечениями.

На самом деле проблема даже не в том, легко ли освоить второй ассемблер. Проблема в том, что порочна сама идея, предполагающая, что специалист учится только в вузе, а после него лишь применяет полученные знания на рабочем месте. Логическое следствие этой идеи — нужно учить только тому, что годится на все случаи жизни. А категории «самое универсальное» и «самое востребованное» — это далеко не одно и то же. Если та или иная дисциплина выбирается из соображений конкурентоспособности подготавливаемого специалиста — это трижды правильно. А если в основе учебного плана лежат соображения, что надо учить так, чтобы полученного багажа хватило студенту вплоть до его пенсии — сегодня это в корне не верно. И, увы, может отражать то, что многие преподаватели здорово отстали от современных требований и сами стремятся жить лишь полученным когда-то багажом (к счастью, исключений из этого правила также немало, я лично знаком со многими преподавателями, которые стараются не отставать от времени, и у них это вполне получается).

Третий аргумент: научить студента языку высокого уровня проще, чем обучить ассемблеру, просто надуман. Плохого студента не научишь ничему. Хорошего можно обучить почти всему. Думаю, каждый, кто учился в вузе, легко вспомнит и талантливых преподавателей, которые играючи обучали чрезвычайно непростым вещам, и бездарей, умудрявшихся запутать любой вопрос, любую тему. Так что вопрос не в том, чему можно научить, а чему почти не удастся, а в том, кто будет учить и как серьезно он к этому подойдет. Бездарь запутает не только с ассемблером, но и с любым С — говорю на основании личного опыта, физфаковский спецкурс по программированию я вспоминаю как один из самых страшных снов. Когда абсолютно неподготовленному человеку чуть ли не с первого занятия начинают рассказывать об отличиях процедуры от процедур-функции, сказав перед этим всего лишь одну (влетевшую в одно ухо и вылетевшую через другое) фразу. Зачем это все нужно, чего же он после этого будет уметь?

Еще один не выдерживающий критики аргумент: так учат в Америке. Америка большая, и мест, где чему-то учат, очень много. И учат в этих местах по-разному, в зависимости от того, как называется тот или иной предмет, и что прописано в учебном курсе по этому предмету в плане соответствующего учебного заведения. Сейчас, когда я пишу эти строки, рядом со мной лежит учебник «Fundamentals of DIGITAL LOGIC with VHDL Design», который привезен моим сыном после прохождения им соответствующего спецкурса в Рочестерском университете (я частично использую материалы из этого учебника при подготовке очередного тома нашей книги по микроконтроллерам). Так вот, дай Бог, чтобы хотя бы каждый третий спецкурс в наших вузах опирался на учебники подобного уровня, объема и охвата. Я уже не говорю о том, сколько информации (широко используемой студентами) выложено на сайтах американских учебных заведений. Так что утверждать, что «супостаты», на которых нам нужно равняться, идут по пути наименьшего сопротивления и стараются предельно упростить программы, выкинув из них все, что потребуется в жизни меньше 50 раз — это либо плохое знание заокеанской действительности, либо умышленное передергивание фактов. Но в любом случае не аргумент, определяющий, что вносить в учебную программу, а что — нет.

Теперь о необъятных ресурсах современных микроконтроллеров. Года три назад, когда мы готовили микроконтроллерную тему номера в «КиТ», С. Гаврилюк, у которого я брал интервью, высказал одну мысль, очень мне тогда понравившуюся. Он отметил, что поскольку специалист-разработчик в развитых западных странах стоит довольно дорого — несколько тысяч долларов в месяц, то его работодателю выгодно заставить этого разработчика провести разработку на контроллере с немерянными ресурсами, где можно не задумываться о производительности, объеме ОЗУ, ПЗУ и т. д. Благодаря этому очень быстро написать программу на языке высокого уровня. Стоимость подобного микроконтроллера (до $50–70) с лихвой компенсируется тем, что разработка велась не месяц (к примеру), а полмесяца, и работодатель экономил в итоге $2–3 тысячи.

На первый взгляд все именно так и должно обстоять — двухлетнее вылизывание конструкции на МК с ограниченными возможностями экономит несколько десятков долларов на контроллере, но требует нескольких тысяч для оплаты труда разработчика (даже нашего, стоящего заметно дешевле немца или американца с близким квалификационным уровнем). А кто из нормальных, экономически грамотных хозяев пойдет на такие затраты, если их можно избежать? Так что же, прощай дешевые микроконтроллеры и программы объемом в 2–3 килобайта, и да здравствуют 100-мегагерцовые монстры с мегабайтами памяти, почти под завязку набитые быстро написанными программами сомнительного качества? И да, и нет. Если речь идет об одном изделии, которое нужно сваять как можно быстрее и проще, описанная логика почти идеальна. Даже с учетом того, что создание программы, как я уже говорил, это лишь на 20% размещение одного под другим операторов языка, а на 80% — продумывание того, что же нужно написать с последующими усилиями по отладке написанного. Даже с учетом того, что при большом объеме ресурсов про-

грамму можно продумать плохо — все равно будет работать. Даже с учетом того, что плохо продуманная программа на языке высокого уровня отлаживается гораздо дольше своего хорошо продуманного ассемблерного аналога. Все равно, когда речь идет о единичном экземпляре, основная статья затрат — оплата труда разработчика, и минимизировать нужно именно время его работы. Причем всеми возможными способами — использованием высокоуровневых языков, недостаточной проработкой алгоритма, отказом от «вылизывания» программы, использованием мощных МК, допускающих перечисленные вольности и т. д.

Но есть и другая логика. Если разрабатываемое изделие будет выпущено тиражом 1000 штук, то разница между 2- и 50-долларовыми контроллерами на этом тираже составит годовую зарплату среднего американского разработчика и пятнадцатилетнюю — среднего отечественного. И ни один хозяин не будет запускать в серию пятидесятидолларовые МК, если можно обойтись двухдолларовыми, программу для которых соответствующий специалист разработает за полгода. Причем всего за $1–2 тысячи (речь идет о разработчиках в нашей стране). Так что безоговорочное упование на то, что подготавливаемому специалисту придется разрабатывать лишь единичные экземпляры оборудования, и игнорирование тенденции, заключающейся во все большем и большем превышении объема и количества серийных разработок над одиночными (по мере роста промышленного производства), снижает конкурентоспособность подготавливаемых специалистов. И это важно осознать в вузах, хотя понимание этого сформировалось еще не везде.

Есть еще один аспект, обычно редко принимаемый во внимание. Логика прогресса такова, что, решив одни задачи, человек ставит перед собой более серьезные, требующие для решения больших (ударение на первом слоге) ресурсов. А это приводит к тому, что когда быстрее, а когда медленнее, но неизбежно на каком-то этапе перед вами будет стоять задача, на решение которой у вашего микроконтроллера не хватит ресурсов, сколь бы много их не было. Поэтому, если у вас выработался эдакий небрежный стиль программирования, в минимальной степени учитывающий особенности того, с чем вы работаете, не предполагающий оптимизацию написанного, стиль, который в неявной форме был сформулирован в первом абзаце настоящей статьи, то с проблемой нехватки ресурсов вы будете сталкиваться чаще других. Увы, подобный стиль программирования фактически навязан разработчикам Windows, в результате чего этот продукт является отражением гениальной менеджерской работы основателя Microsoft, и, в то же время, весьма посредственного труда программистов, реализовавших Windows.

Кстати, близкую мысль высказывает ниже в своей статье и автор знакомого всем пакета ГРАФОР. С его слов, работающие у него аспиранты-программисты в состоянии написать подобную программу на C++, но предполагают, что ее объем в кодах будет не менее сотни килобайт, и для работы с требуемой скоростью отображения нужен процессор не менее Pentium 100–133 МГц. Сравните с ресурсами оригинальной программы и быстродействием использовавшихся тогда вычислительных машин. Впечатляет?

Ну, так что в итоге? Забыть про языки высокого уровня и учить только ассемблеру? Или наоборот, только C и нечего отвлекаться на низкоуровневые языки? Ни то, ни другое. Все зависит от задачи соответствующего спецкурса. Наверное, если курс по микроконтроллерам — факультативный или обзорный, для общего развития, и никаких других языков программирования ни в одном из остальных курсов студенты не проходят, больше подошел бы язык высокого уровня — все равно все в общем, без подробностей.

Для тех же, кто изучает микроконтроллеры для того, чтобы работать с ними после вуза, — обязательно и то, и другое, причем хорошо бы со сравнительными примерами, чтобы студенты «пощупали руками» разницу в программировании на языке высокого уровня и на ассемблере. И самое главное, программы студенты должны сами написать, осознать, оттранслировать и отладить — без такого тренинга специалистов не подготовить. Увы, о подобных спецкурсах в вузах информации крайне мало, я даже не знаю, где учат студентов микроконтроллерной технике на таком уровне. Сейчас подобный курс готовится на кафедре МИФИ, надеюсь, он состоится, и уже в следующем году мы будем располагать результатами того, чему удалось научить студентов, пропущенных через этот спецкурс.