Продолжение. Начало №4–6/2001

Микроконтроллеры? Это же просто!

Ассемблер: основные понятия и приемы

Итак, мы ознакомились с некоторыми из команд микроконтроллера, которые были необходимы для написания программы связи МК и АЦП. Команды эти мы записали в соответствии с требованиями, предъявляемыми ассемблером для x51. Давайте теперь более подробно познакомимся с ассемблером и правилами работы с ним. В результате этого знакомства мы должны будем освоить не только то, как писать тексты программ, но и как их транслировать в коды, заносимые впоследствии в память программ МК.

Вообще программ-ассемблеров для x51 известно довольно много. Все они слегка отличаются друг от друга, но основные правила, установленные Intel для ассемблеров x51, у них едины. Кстати, большинство из них DOSовские, ибо они были разработаны еще в те времена, когда DOS безраздельно господствовала в компьютерном мире. Конечно, в последние лет десять стали доступными и несколько Windows-версий ассемблера для x51, но программисты – народ довольно консервативный, и большинство из них с неохотой переходят на новые программные продукты. Я уже много лет использую DOS-ассемблер TASM. Он, возможно, не столь удобен, как свежие Windows-ассемблеры, но про него я точно знаю, что он не содержит ошибок, и поэтому не горю желанием с ним расставаться. К тому же именно под него в 1996 г. моим сыном был разработан хороший программный отладчик EMF-52, который опубликован в одном из первых номеров "Схемотехники" за этот год, и в совокупности эти программы составляют хороший и удобный программный комплекс, работающий без сбоев на любом компьютере, и требующий минимального количества ресурсов.

После такого вступления перейдем собственно к ассемблеру. Начнем с того, что в тексте программы на ассемблере у нас всегда будут присутствовать две группы команд. Одна из них – это команды микроконтроллера, которые ассемблер и будет транслировать в коды. Другая – это так называемые директивы ассемблера, т. е. команды, которые управляют работой самого ассемблера.

Начнем мы со знакомства с тремя основными директивами – ORG, EQU и END. Обращаю ваше внимание на то, что в ассемблере TASM перед всеми директивами всегда стоит точка, т. е. мы должны писать .ORG, .EQU и .END (в других ассемблерах точка необязательна).

Первая из директив. ORG, предписывает ассемблеру транслировать идущий вслед за ней фрагмент программы (или всю программу) с того адреса, который указан после слова .ORG. Например, .ORG 0 означает, что код команды, идущей вслед за директивой, должен располагаться в ячейке памяти программ с нулевым адресом. Как мы уже говорили раньше, МК семейства x51 при включении питания начинают выполнять программу именно с этой команды. Поэтому очевидно, что директива .ORG 0 должна стоять перед самой первой командой нашей программы.

Отметим далее, что для x51 первые 256 байт памяти программ лучше не занимать под основную программу – в них нужно располагать так называемые подпрограммы обработки прерываний (об этом позже). Поэтому обычно начало программы должно выглядеть следующим образом:

```
.ORG 0
LJMP START
.ORG 100H
START:
      MOV   P1,#11111111B
            ...
```

Здесь директива .ORG 0 указывает ассемблеру разместить коды записанной после нее команды LJMP START в ячейках памяти программ, начиная с адреса 0. Сама эта команда предписывает микроконтроллеру перейти к выполнению команды, стоящей после метки START:, т. е. MOV P1,#11111111B. Место расположения в памяти программ кодов последней начинается с адреса 100H (или 256 десятичного), т. к. метка расположена сразу за директивой .ORG 100H. Все это будет хорошо видно, когда мы познакомимся чуть ниже с листингом программы – текстовым файлом, в котором ассемблер проставит рядом с этими командами их коды и адреса ячеек памяти программ, где эти коды будут располагаться.

Еще одно замечание – метки (в данном случае START:) при написании программ на ассемблере должны начинаться с первой позиции в строке, без каких-либо пробелов перед именем метки. Сами команды и начинающиеся с точки директивы ассемблера должны идти после одного или нескольких пробелов. Если при написании программы вы будете использовать редактор типа Turbo Pascal для DOS, то перед написанием команды или директивы нажмите клавишу табуляции Tab – она поставит перед ними 8 пробелов. А когда вы, работая в этом редакторе, после написания строки с командой нажмете клавишу Enter, курсор расположится под первым символом строчки, расположенной выше, т. е. ввод 8 пробелов осуществится автоматически.

Следующая директива – .END. Она информирует ассемблер о том, что команда, стоящая перед ней – последняя в этой программе, и что на этом месте нужно завершить трансляцию написанных команд в понятные микроконтроллеру коды. Перед .END также нужно поставить один или несколько пробелов (или нажать Tab).

Очень полезной является третья из рассматриваемых директив – .EQU . С ней мы уже встречались в предыдущем разделе. Вспомним, например, строку

```
CS    .EQU    P3.7
```

или

```
RD    .EQU    P3.6
```

Одна из особенностей нашего мозга состоит в том, что мы очень плохо запоминаем неинформативные надписи типа P3.6 или 0B2H, в то время как запомнить надпись CS или RD, явно ассоциирующуюся с сигналом CS или RD АЦП, не представляет труда. В самом деле, попробуйте сообразить сразу, какому импульсу соответствуют идущие друг за другом команды CLR P1.5 и SETB P1.5. А вот если вместо P1.5 поставить имя CONVST, то вопросов не возникнет. А для того, чтобы и ассемблер понял, к какой линии порта или ячейке памяти мы обращаемся, до первого упоминания имени CONVST мы должны разместить строку

```
CONVST   .EQU        P3.5
```

Отметим, что в этой директиве имя переменной (в данном случае CONVST) записывается, подобно метке, с самого начала строки без предшествующих ему пробелов.

При написании программ рекомендуется использовать подобные символические имена везде, где это возможно. Помимо легкости понимания, это дает нам ряд дополнительных удобств. В самом деле, если вы, к примеру, по причине упрощения разводки платы вынуждены будете соединить вход АЦП CONVST не с линией P3.5 МК, а с P3.2, вам нужно будет везде в тексте вашей программы заменить P3.5 на P3.2. Конечно, если программа умещается на одной–двух страницах, это несложно. А если в программе около 2000 строк, и она занимает 30 страниц? Поверьте, найти все без исключения места, где упоминается P3.5, не пропустив ни одного из них, крайне затруднительно. А если вы линию P3.5 нарекли именем CONVST, и в начале вашей программы поставили строку CONVST .EQU P3.5, то вам достаточно заменить линию P3.5 на P3.2 только в этой строке – все остальные замены ассемблер осуществит самостоятельно, и при этом, что крайне важно, ничего не пропустит.

Далее нам необходимо ознакомиться с одной особенностью, присущей именно ассемблеру TASM. Дело в том, что этот ассем-

блер весьма универсален – он может быть адаптирован для очень многих типов микроконтроллеров. Для большинства из них принципы построения их ассемблеров идентичны, различия заключаются в адресах линий портов, регистров, ячеек оперативной памяти, в кодах команд и в словах, которыми мы описываем эти команды (упомянутые слова называются мнемониками MOV, например, мнемоника команды пересылки данных, и т. д.). Для каждого семейства микроконтроллеров у TASM'а есть таблица, в которой записаны мнемоники всех их команд и коды, им соответствующие, а для тех семейств, для которых эти таблицы отсутствуют, их при желании можно сделать самостоятельно.

Такая гибкость TASM'а имеет свою оборотную сторону. Он "не знает" адресов регистров и линий портов МК семейства x51. Поэтому в самом начале программы ему необходимо указать упомянутые адреса при помощи уже описанной директивы .EQU. В принципе, это не такая уж проблема – написать этот фрагмент нужно лишь однажды, и затем переносить его из одной программы в другую. Когда-то я эту работу сделал, и все эти строчки вы найдете в предлагаемой вашему вниманию программе (файл par_adc.obj **www.platan.ru/shem**). Пользуйтесь ими на здоровье. А тех, кого раздражает необходимость размещать перед текстом программы строки с директивой .EQU, успокою – в новых МК, например в AduC8xx, появились дополнительные регистры, имен которых не знают не только TASM, но и ни один из остальных ассемблеров, за исключением специально разработанных под эти МК. Так что чем бы вы ни пользовались, без подобных строк вы вряд ли обойдетесь.

После всего сказанного мы уже готовы к тому, чтобы программу работы МК с АЦП, приведенную в предыдущей части статьи, записать с учетом всех требований ассемблера.

```
;     ПРОГРАММА ЧТЕНИЯ АЦП AD7880,
;     РАБОТАЕМ С ПОРТАМИ P1 И P3, CS=P3.7,
;     RD=P3.6, CONVST=P3.5.
;           *****
;
R7    .EQU   7          ;АДРЕСА РЕГИСТРОВ R0-R7
R6    .EQU   6
R5    .EQU   5
R4    .EQU   4
R3    .EQU   3
R2    .EQU   2
R1    .EQU   1
R0    .EQU   0
ACC   .EQU   0E0H       ;АДРЕС АККУМУЛЯТОРА
B     .EQU   0F0H       ;АДРЕС РЕГИСТРА B
PSW   .EQU   0D0H       ;АДРЕС РЕГИСТРА (СЛОВА) СОСТОЯНИЯ
SP    .EQU   81H        ;АДРЕС УКАЗАТЕЛЯ СТЕКА
DPL   .EQU   82H        ;АДРЕС МЛАДШЕЙ ПОЛОВИНЫ DPTR
DPH   .EQU   83H        ;АДРЕС СТАРШЕЙ ПОЛОВИНЫ DPTR
P0    .EQU   80H        ;АДРЕС РЕГИСТРА ПОРТА P0
P1    .EQU   90H        ;АДРЕС РЕГИСТРА ПОРТА P1
P2    .EQU   0A0H       ;АДРЕС РЕГИСТРА ПОРТА P2
P3    .EQU   0B0H       ;АДРЕС РЕГИСТРА ПОРТА P3
B.0   .EQU   0F0H       ;АДРЕСА ОТДЕЛЬНЫХ БИТОВ РЕГИСТРА B
B.1   .EQU   0F1H
B.2   .EQU   0F2H
B.3   .EQU   0F3H
B.4   .EQU   0F4H
B.5   .EQU   0F5H
B.6   .EQU   0F6H
B.7   .EQU   0F7H
ACC.0 .EQU   0E0H       ;АДРЕСА ОТДЕЛЬНЫХ БИТОВ АККУМУЛЯТОРА
ACC.1 .EQU   0E1H
ACC.2 .EQU   0E2H
ACC.3 .EQU   0E3H
ACC.4 .EQU   0E4H
ACC.5 .EQU   0E5H
ACC.6 .EQU   0E6H
ACC.7 .EQU   0E7H
PSW.0 .EQU   0D0H
PSW.1 .EQU   0D1H       ;АДРЕСА ОТДЕЛЬНЫХ БИТОВ РЕГИСТРА PSW
PSW.2 .EQU   0D2H
PSW.3 .EQU   0D3H
PSW.4 .EQU   0D4H
PSW.5 .EQU   0D5H
PSW.6 .EQU   0D6H
PSW.7 .EQU   0D7H
P0.0  .EQU   080H       ;АДРЕСА ОТДЕЛЬНЫХ ЛИНИЙ ПОРТА P0
P0.1  .EQU   081H
P0.2  .EQU   082H
P0.3  .EQU   083H
P0.4  .EQU   084H
P0.5  .EQU   085H
P0.6  .EQU   086H
P0.7  .EQU   087H
P1.0  .EQU   090H       ;АДРЕСА ОТДЕЛЬНЫХ ЛИНИЙ ПОРТА P1
P1.1  .EQU   091H
P1.2  .EQU   092H
P1.3  .EQU   093H
P1.4  .EQU   094H
P1.5  .EQU   095H
P1.6  .EQU   096H
P1.7  .EQU   097H
P2.0  .EQU   0A0H       ;АДРЕСА ОТДЕЛЬНЫХ ЛИНИЙ ПОРТА P2
P2.1  .EQU   0A1H
P2.2  .EQU   0A2H
P2.3  .EQU   0A3H
P2.4  .EQU   0A4H
P2.5  .EQU   0A5H
P2.6  .EQU   0A6H
P2.7  .EQU   0A7H
P3.0  .EQU   0B0H       ;АДРЕСА ОТДЕЛЬНЫХ ЛИНИЙ ПОРТА P3
P3.1  .EQU   0B1H
P3.2  .EQU   0B2H
P3.3  .EQU   0B3H
P3.4  .EQU   0B4H
P3.5  .EQU   0B5H
P3.6  .EQU   0B6H
P3.7  .EQU   0B7H
;
CS    .EQU   P3.7
RD    .EQU   P3.6
CONVST .EQU  P3.5
;
;
      .ORG   0          ;НИЖЕСЛЕДУЮЩАЯ КОМАНДА С АДРЕСА 0
;
      LJMP   START      ;НА КОМАНДУ ПОСЛЕ МЕТКИ START
;
      .ORG   100H
;
START:
      MOV    P0,#11111111B  ;НАЧАЛЬНАЯ УСТАНОВКА
      MOV    P1,#11111111B
      MOV    P2,#11111111B
      MOV    P3,#11111111B
L7880:                ;СОБСТВЕННО ЧТЕНИЕ
;
      CLR    CONVST     ;ИМПУЛЬС СТАРТА ПРЕОБРАЗОВАНИЯ
      SETB   CONVST
;
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
;
      CLR    CS         ;CS=0
      CLR    RD         ;RD=0
;
      MOV    A,P1       ;ЧИТАЕМ ИЗ ПОРТА P1 МЛ.И СР. ТЕТРАДЫ
      MOV    R4,A       ;СОХРАНЯЕМ ИХ В R4
;
      MOV    A,P3       ;ЧИТАЕМ ИЗ ПОРТА P3 СТ. ТЕТРАДЫ
      MOV    R5,A       ;В R5R4 - РЕЗУЛЬТАТ
;
      SETB   RD         ;КОНЕЦ RD
      SETB   CS         ;КОНЕЦ CS
;
      SJMP   L7880      ;ЗАЦИКЛИВАНИЕ
;
      .END
```

Эта программа находится на сайте **www.platan.ru/shem** (файл par_adc.a51). Все, что в ней написано, вам уже знакомо – ком-

ментарии после точки с запятой, фрагмент присвоения директивой .EQU адресов основным регистрам МК, их битам и линиям портов, метки, оканчивающиеся двоеточиями, директивы .ORG, команды пересылок, установки и сброса линий портов и т. д. Эту программу уже можно оттранслировать при помощи ассемблера TASM, и получить как файл для загрузки в память программ МК, так и уже упоминавшийся листинговый файл с кодами команд и адресами расположения этих кодов.

Особенности трансляции

Прежде чем приступить к трансляции приведенной выше программы в коды, нам необходимо кратко познакомится с двумя основными форматами представления результатов трансляции.

Первый формат, именуемый бинарным или объектным, попросту содержит коды программы в том самом виде, в котором они будут записаны в память программ микроконтроллера. Большинство ассемблеров при создании файла в этом формате присваивают ему расширение *.obj. Я предполагаю, что читатели знают, что такое DOS, директории, файлы, их имена и расширения, bat-файлы, умеют смотреть содержимое файла при помощи вьювера оболочки Norton Commander или ей аналогичной, редактировать текст тем или иным DOS-редактором. Тем, кто со всем этим не знаком, придется найти какую-нибудь книжку по DOS, или расспросить пользователей со стажем.

При трансляции файла par_adc.a51 в объектный формат TASM сформирует нам файл par_adc.obj. Просмотреть его можно вьювером Norton Commander'a, установив на него курсорную плашку и нажав привычное пользователям DOS F3, а затем – F4. Мой программатор понимает этот формат файлов, поэтому я транслирую написанные программы в коды таким образом, чтобы получать именно obj-файлы.

Второй формат носит название Intel Hex-Format или по-простому hex-формат. При трансляции файла par_adc.a51 в hex-формат TASM сформирует нам файл par_adc.hex. Просмотреть его можно все тем же вьювером Norton Commander'a, установив на файл курсорную плашку и нажав уже упоминавшуюся F3. Правда, в отличие от предыдущего случая, F4 нажимать не надо, ибо коды, полученные в результате трансляции, в файле par_adc.hex представлены в формате ASC-II. Помимо них, файл par_adc.hex содержит дополнительную служебную информацию, говорить о которой я не буду, но содержание которой становится более-менее понятным при внимательном сравнении файлов par_adc.obj и par_adc.hex или при чтении документации на ассемблер TASM.

Зачем нужен этот формат? Большинство программаторов понимает оба формата, но, тем не менее, встречаются отдельные изделия, которым один из них недоступен. Если вы станете счастливым обладателем программатора, который понимает только hex-формат, вам придется транслировать программы таким образом, чтобы на выходе получать именно такие файлы (о том, как это делается – чуть ниже). В остальных случаях рекомендую транслировать файлы в obj-формат – с ним проще разбираться, если что-то не так.

Теперь перейдем к трансляции. Создайте на винчестере директорию, в которой должны будут располагаться файлы, с коими нам предстоит работать. Пусть она будет размещена на диске C и наречена ASM51 (C:\ASM51). В ней должны находиться файлы ассемблера tasm.exe, tasm51.tab, tasm51b.bat, tasm51h.bat, tasm_rus.doc (файл с документацией), симулятор emf52l.exe и файл с программой par_adc.a51.

Войдите в Norton Commander (можно в сеансе работы в Windows95/98). Далее войдите в упомянутую директорию C:\ASM51. После этого наберите в командной строке tasm51b.bat и через пробел после него имя транслируемого файла (в нашем случае par_adc.a51) и нажмите <Enter>. Трансляция осуществляется быстро, за доли или единицы секунд, в зависимости от быстродействия вашего компьютера. В ходе ее на экране появятся сообщения:

```
tasm: pass 1 complete.
tasm: pass 2 complete.
tasm: Number of errors = 0
```

Последняя строчка очень важна – ассемблер сообщил, что он не обнаружил ошибок и создал obj- или hex-файл (в нашем случае – файл . par_adc.obj). Но бывает (и к сожалению, гораздо чаще, чем хотелось бы), что написанная программа содержит ошибки. Некоторые из них ассемблер находит и информирует нас строкой:

```
tasm: Number of errors = n,
```

где n – число обнаруженных ошибок. Помимо этого, TASM дает некоторую дополнительную информацию относительно этих ошибок – тип ошибки и в какой строке она находится. Более подробно о сообщениях, выдаваемых им при обнаружении ошибок, мы поговорим в следующем разделе. Сейчас же вернемся к процессу трансляции.

Bat-файл tasm51b.bat, как некоторые из вас уже наверное догадались, осуществляет запуск ассемблера TASM с ключами (указаниями), рекомендующими транслировать исходный ассемблерный файл в объектный. Кроме того, ассемблеру предписывается также выполнить некоторые другие действия – сформировать файл листинга, включить в него таблицу меток, и т. д. Подробнее о том, что он может сделать, и как заставить его выполнить те или функции, вы узнаете, прочитав содержимое файла документации (tasm_rus.doc), а также проанализировав командный файл tasm51b.bat. Но на первых порах можете просто ограничиться использованием этого командного файла, подобно тому, как это было указано тремя абзацами выше.

Если ваш программатор требует для программирования файлы в hex-формате, то последовательность действий следующая. Как и в предыдущем случае войдите в Norton Commander, затем в упомянутую директорию C:\ASM51 и наберите в командной строке tasm51h.bat, после чего через пробел наберите имя транслируемого файла (в нашем случае par_adc.a51). Как и ранее, нажмите <Enter>. Трансляция осуществляется также, как и в предыдущем случае, с теми же сообщениями, только в результате транслирования сформируется файл par_adc.hex, в чем я рекомендую убедиться.

Теперь поговорим о файле листинга. Этот файл (par_adc.lst) формируется ассемблером в процессе трансляции и представляет из себя исходный ассемблерный файл (par_adc.a51), дополненный следующей информацией. Перед каждой командой стоит номер ее строки в ассемблерном тексте, адрес ячейки памяти программ, в которой расположен код операции команды, а после этого адреса – один, два или три байта самой команды (откуда у команд берутся второй и третий байт, мы рассмотрим в главе, посвященной системе команд). Далее, в конце программы находится таблица имен и меток, а также содержимое полученного в результате трансляции obj- или hex–файла. И, наконец, файл листинга разбит на страницы. В качестве примера одна из них приведена ниже.

```
TASM 8051 Assembler.          PAR_ADC.A51              page 1
Speech Technology Incorporated.

 0001  0000          ;    ПРОГРАММА ЧТЕНИЯ АЦП AD7880,
 0002  0000          ;    РАБОТАЕМ С ПОРТАМИ P1 И P3, CS=P3.7,
 0003  0000          ;    RD=P3.6, CONVST=P3.5.
 0004  0000          ;          *****
 0005  0000          ;
 0006  0000          R7    .EQU  7          ;АДРЕСА РЕГИСТРОВ R0-R7
 0007  0000          R6    .EQU  6
 0008  0000          R5    .EQU  5
 0009  0000          R4    .EQU  4
 0010  0000          R3    .EQU  3
 0011  0000          R2    .EQU  2
 0012  0000          R1    .EQU  1
 0013  0000          R0    .EQU  0
 0014  0000          ACC   .EQU  0E0H       ;АДРЕС АККУМУЛЯТОРА
 0015  0000          B     .EQU  0F0H       ;АДРЕС РЕГИСТРА B
 0016  0000          PSW   .EQU  0D0H       ;АДРЕС РЕГИСТРА (СЛОВА)
СОСТОЯНИЯ
 0017  0000          SP    .EQU  81H        ;АДРЕС УКАЗАТЕЛЯ СТЕКА
 0018  0000          DPL   .EQU  82H        ;АДРЕС МЛАДШЕЙ ПОЛОВИНЫ
DPTR
 0019  0000          DPH   .EQU  83H        ;АДРЕС СТАРШЕЙ ПОЛОВИНЫ
DPTR
 0020  0000          P0    .EQU  80H        ;АДРЕС РЕГИСТРА ПОРТА P0
 0021  0000          P1    .EQU  90H        ;АДРЕС РЕГИСТРА ПОРТА P1
 0022  0000          P2    .EQU  0A0H       ;АДРЕС РЕГИСТРА ПОРТА P2
 0023  0000          P3    .EQU  0B0H       ;АДРЕС РЕГИСТРА ПОРТА P3
 0024  0000          B.0   .EQU  0F0H       ;АДРЕСА ОТДЕЛЬНЫХ БИТОВ
РЕГИСТРА B
 0025  0000          B.1   .EQU  0F1H
 0026  0000          B.2   .EQU  0F2H
 0027  0000          B.3   .EQU  0F3H
 0028  0000          B.4   .EQU  0F4H
 0029  0000          B.5   .EQU  0F5H
 0030  0000          B.6   .EQU  0F6H
 0031  0000          B.7   .EQU  0F7H
 0032  0000          ACC.0 .EQU  0E0H       ;АДРЕСА ОТДЕЛЬНЫХ БИТОВ
АККУМУЛЯТОРА
 0033  0000          ACC.1 .EQU  0E1H
```

```
0034  0000          ACC.2  .EQU   0E2H
0035  0000          ACC.3  .EQU   0E3H
0036  0000          ACC.4  .EQU   0E4H
0037  0000          ACC.5  .EQU   0E5H
0038  0000          ACC.6  .EQU   0E6H
0039  0000          ACC.7  .EQU   0E7H
0040  0000          PSW.0  .EQU   0D0H
0041  0000          PSW.1  .EQU   0D1H        ;АДРЕСА ОТДЕЛЬНЫХ БИТОВ
РЕГИСТРА PSW
0042  0000          PSW.2  .EQU   0D2H
0043  0000          PSW.3  .EQU   0D3H
0044  0000          PSW.4  .EQU   0D4H
0045  0000          PSW.5  .EQU   0D5H
0046  0000          PSW.6  .EQU   0D6H
0047  0000          PSW.7  .EQU   0D7H
0048  0000          P0.0   .EQU   080H        ;АДРЕСА ОТДЕЛЬНЫХ ЛИНИЙ
ПОРТА P0
0049  0000          P0.1   .EQU   081H
0050  0000          P0.2   .EQU   082H
0051  0000          P0.3   .EQU   083H
0052  0000          P0.4   .EQU   084H
0053  0000          P0.5   .EQU   085H
0054  0000          P0.6   .EQU   086H
0055  0000          P0.7   .EQU   087H
0056  0000          P1.0   .EQU   090H        ;АДРЕСА ОТДЕЛЬНЫХ ЛИНИЙ
ПОРТА P1
0057  0000          P1.1   .EQU   091H
0058  0000          P1.2   .EQU   092H
0059  0000          P1.3   .EQU   093H
TASM 8051 Assembler.          PAR_ADC.A51              page 2
Speech Technology Incorporated.

0060  0000          P1.4   .EQU   094H
0061  0000          P1.5   .EQU   095H
0062  0000          P1.6   .EQU   096H
0063  0000          P1.7   .EQU   097H
0064  0000          P2.0   .EQU   0A0H        ;АДРЕСА ОТДЕЛЬНЫХ ЛИНИЙ
ПОРТА P2
0065  0000          P2.1   .EQU   0A1H
0066  0000          P2.2   .EQU   0A2H
0067  0000          P2.3   .EQU   0A3H
0068  0000          P2.4   .EQU   0A4H
0069  0000          P2.5   .EQU   0A5H
0070  0000          P2.6   .EQU   0A6H
0071  0000          P2.7   .EQU   0A7H
0072  0000          P3.0   .EQU   0B0H        ;АДРЕСА ОТДЕЛЬНЫХ ЛИНИЙ
ПОРТА P3
0073  0000          P3.1   .EQU   0B1H
0074  0000          P3.2   .EQU   0B2H
0075  0000          P3.3   .EQU   0B3H
0076  0000          P3.4   .EQU   0B4H
0077  0000          P3.5   .EQU   0B5H
0078  0000          P3.6   .EQU   0B6H
0079  0000          P3.7   .EQU   0B7H
0080  0000          ;
0081  0000          CS     .EQU   P3.7
0082  0000          RD     .EQU   P3.6
0083  0000          CONVST .EQU   P3.5
0084  0000          ;
0085  0000          ;
0086  0000                .ORG    0         ;НИЖЕСЛЕДУЮЩАЯ КОМАНДА С
АДРЕСА 0
0087  0000          ;
0088  0000 02 01 00       LJMP   START       ;НА КОМАНДУ ПОСЛЕ МЕТКИ
START
0089  0003          ;
0090  0100                .ORG    100H
0091  0100          ;
0092  0100          START:
0093  0100 75 80 FF       MOV    P0,#11111111B  ;НАЧАЛЬНАЯ УСТАНОВКА
0094  0103 75 90 FF       MOV    P1,#11111111B
0095  0106 75 A0 FF       MOV    P2,#11111111B
0096  0109 75 B0 FF       MOV    P3,#11111111B
0097  010C          L7880:                   ;СОБСТВЕННО ЧТЕНИЕ
0098  010C          ;
0099  010C C2 B5              CLR     CONVST          ;ИМПУЛЬС СТАРТА
ПРЕОБРАЗОВАНИЯ
0100  010E D2 B5          SETB   CONVST
0101  0110          ;
0102  0110 00              NOP
0103  0111 00              NOP
0104  0112 00              NOP
0105  0113 00              NOP
0106  0114 00              NOP
0107  0115 00              NOP
0108  0116 00              NOP
0109  0117 00              NOP
0110  0118 00              NOP
0111  0119 00              NOP
0112  011A 00              NOP
0113  011B 00              NOP
0114  011C 00              NOP
0115  011D 00              NOP
0116  011E 00              NOP
0117  011F 00              NOP
0118  0120 00              NOP
TASM 8051 Assembler.          PAR_ADC.A51              page 3
Speech Technology Incorporated.

0119  0121 00              NOP
0120  0122 00              NOP
0121  0123          ;
0122  0123 C2 B7          CLR    CS          ;CS=0
0123  0125 C2 B6          CLR    RD          ;RD=0
0124  0127          ;
0125  0127 E5 90          MOV    A,P1        ;ЧИТАЕМ ИЗ ПОРТА P1 МЛ.И СР.
ТЕТРАДЫ
0126  0129 FC             MOV    R4,A        ;СОХРАНЯЕМ ИХ В R4
0127  012A          ;
0128  012A E5 B0          MOV    A,P3        ;ЧИТАЕМ ИЗ ПОРТА P3 СТ.
ТЕТРАДЫ
0129  012C FD             MOV    R5,A        ;В R5R4 — РЕЗУЛЬТАТ
0130  012D          ;
0131  012D D2 B6          SETB   RD          ;КОНЕЦ RD
0132  012F D2 B7          SETB   CS          ;КОНЕЦ CS
0133  0131          ;
0134  0131 80 D9          SJMP   L7880       ;ЗАЦИКЛИВАНИЕ
0135  0133          ;
0136  0133                .END
```

Label	Value	Label	Value	Label	Value
ACC	00E0	ACC.0	00E0	ACC.1	00E1
ACC.2	00E2	ACC.3	00E3	ACC.4	00E4
ACC.5	00E5	ACC.6	00E6	ACC.7	00E7
B	00F0	B.0	00F0	B.1	00F1
B.2	00F2	B.3	00F3	B.4	00F4
B.5	00F5	B.6	00F6	B.7	00F7
CS	00B7	CONVST	00B5	DPL	0082
DPH	0083	L7880	010C	PSW	00D0
P0	0080	P1	0090	P2	00A0
P3	00B0	PSW.0	00D0	PSW.1	00D1
PSW.2	00D2	PSW.3	00D3	PSW.4	00D4
PSW.5	00D5	PSW.6	00D6	PSW.7	00D7
P0.0	0080	P0.1	0081	P0.2	0082
P0.3	0083	P0.4	0084	P0.5	0085
P0.6	0086	P0.7	0087	P1.0	0090
P1.1	0091	P1.2	0092	P1.3	0093
P1.4	0094	P1.5	0095	P1.6	0096
P1.7	0097	P2.0	00A0	P2.1	00A1
P2.2	00A2	P2.3	00A3	P2.4	00A4
P2.5	00A5	P2.6	00A6	P2.7	00A7
P3.0	00B0	P3.1	00B1	P3.2	00B2
P3.3	00B3	P3.4	00B4	P3.5	00B5
P3.6	00B6	P3.7	00B7	R7	0007
R6	0006	R5	0005	R4	0004
R3	0003	R2	0002	R1	0001
R0	0000	RD	00B6	SP	0081
START	0100				

```
ADDR  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0000  02 01 00 FF FF FF FF FF FF FF FF FF FF FF FF FF
0010  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0020  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0030  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
TASM 8051 Assembler.          PAR_ADC.A51              page 4
Speech Technology Incorporated.

0050  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0060  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0070  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0080  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0090  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

```
00F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0100  75 80 FF 75 90 FF 75 A0 FF 75 B0 FF C2 B5 D2 B5
0110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0120  00 00 00 C2 B7 C2 B6 E5 90 FC E5 B0 FD D2 B6 D2
0130  B7 80 D9 FF FF FF FF FF FF FF FF FF FF FF FF FF

tasm: Number of errors = 0
```

Целиком листинговый файл вы найдете в директории C:\ASM51 после трансляции исходного файла par_adc.a51 при помощи tasm51b.bat или tasm51h.bat.

Ошибки трансляции

Теперь, как я и обещал, рассмотрим некоторые типичные ошибки, допускаемые программистами при написании программ на ассемблере, и ту информацию, которую выводит на экран TASM при их обнаружении.

Наиболее распространенные ошибки – неправильное написание команд и имен переменных, отсутствие метки, на которую должен быть осуществлен переход, дважды (или чаще) повторяющаяся метка с одним и тем же именем, неправильное написание директивы .END. Можно умышленно сделать эти ошибки в нашей программе par_adc.a51, оттранслировать ее с ними и посмотреть на реакцию ассемблера на них. Если вы не поленитесь и проделаете рекомендуемые действия, то приобретете полезный навык реальной трансляции программы и последовательности действий при обнаружении ошибок при трансляции.

1. Неправильное написание команд.

Для примера предположим, что в строке SETB CONVST (сразу после метки L7880:) мы ошибочно поставили SET CONVST (пропустили букву “B” в конце команды). Внесите эту ошибку в текст программы и запустите ее на ассемблирование. В ходе трансляции на экране отобразится отчет о результате ее:

```
tasm: pass 1 complete.
tasm: unrecognized instruction.      Line 0100 in PAR_ADC.A51
tasm: pass 2 complete.
tasm: Number of errors = 1
```

TASM сообщил, что в строке №100 обнаружена неизвестная ему команда (инструкция). Если вы посмотрите файл листинга, то увидите сразу под сотой строкой, где обнаружена ошибка, то же самое сообщение, что и в вышеприведенном отчете (tasm: unrecognized instruction. Line 0100 in PAR_ADC.A51). Таким образом, если в результате трансляции ассемблер нашел ошибку, в данном случае – неверно записанную команду, просмотрите листинговый файл, найдите в нем строку с сообщением об ошибке, и внимательно посмотрите на команду в вышестоящей строке – именно она содержит ошибку. Исправьте ее, снова запустите программу на ассемблирование и убедитесь, что все в порядке.

2. Ошибка в написании имени переменной.

Для примера возьмем случай, когда в блоке присвоения именам переменных численных значений имя переменной CS (CS .EQU P3.7), а в тексте программы мы ее обозначили как CSS, причем сразу в двух местах:

```
(текст программы)

CLR   CSS    ;CS=0

(текст программы)

SETB  CSS    ;УСТАНОВКА CS В 1

(текст программы)
```

Внесите эти ошибки в текст программы и запустите ее на ассемблирование. В ходе трансляции на экране отобразится следующий отчет о ее результате:

```
tasm: pass 1 complete.
tasm: Label not found:    (CSS)   Line 0122
tasm: Label not found:    (CSS)   Line 0132
tasm: pass 2 complete.
tasm: Number of errors = 2
```

TASM сообщил, что в строках №122 и №132 обнаружена неизвестная ему метка. Если вы, как и в предыдущем случае, посмотрите файл листинга, то обнаружите вышеприведенные сообщения об ошибках прямо перед 122-й и 132-й строками, где эти ошибки и находятся. Вам остается только понять, почему TASM не понял этих имен, для чего нужно пролистать листинговый файл назад к началу, найти то имя, которое вами было определено, и сравнить с тем, которое оказалось в 122-й и 132-й строках. После этого обычно сразу все становится ясно, остается лишь исправить ошибки и запустить исправленный файл на ассемблирование. Проделайте сказанное до получения строки tasm: Number of errors = 0.

3. Отсутствие метки, на которую должен быть осуществлен переход.

Допустим, вы забыли поставить метку START: или пропустили в ней букву. В результате ассемблирования вы должны получить отчет следующего содержания:

```
tasm: pass 1 complete.
tasm: Label not found:    (START)   Line 0088
tasm: pass 2 complete.
tasm: Number of errors = 1
```

Здесь ссылка на то, что TASM не нашел метку, более правильна – это действительно так, метки нет. Соответственно, об этом он сообщает и в листинговом файле перед 88-й строкой, в которой размещена команда LJMP START. Опять рекомендую при появлении подобной информации в отчете внимательно посмотреть вначале на строку, где находится ссылка на несуществующую метку, а затем на то место, где она должна бы присутствовать. Внимательный анализ содержимого этих двух строк должен прояснить, в чем вы промахнулись.

4. Дважды (или чаще) повторяющаяся метка с одним и тем же именем.

Опять экспериментируем с меткой START:, но в этот раз давайте воткнем ее еще в одном месте, где она не нужна, например, перед первой командой NOP. Запустим программу на ассемблирование и получим:

```
tasm: pass 1 complete.
tasm: label value misaligned.        (START)   Line 0102 in PAR_ADC.A51
tasm: pass 2 complete.
tasm: Number of errors = 1
```

В файле листинга сообщение об этой ошибке будет стоять под 102-й строкой, где TASM нашел упомянутую метку во второй раз. В этом случае ищите выше по тексту место, где она встречается в первый раз, и удаляйте ту, которая лишняя.

5. Неправильное написание директивы .END.

Предположим, в результате ассемблирования вы получили отчет следующего содержания:

```
tasm: pass 1 complete.
tasm: No END directive before EOF.   Line 0135 in PAR_ADC.A51
tasm: pass 2 complete.
tasm: Number of errors = 1
```

Для тех, кто не знает, EOF – это End Of File (конец файла), символ, стоящий в начале последней строки в файле. Если в этой строке стоит директива .END, то TASM сначала находит символ EOF, а затем – директиву .END. По правилам же должно быть наоборот – вначале .END, затем EOF. Описанная ошибка заключается в том, что в файле par_adc.a51 после набора строчки с директивой .END вы не нажали клавишу “Enter”. Нажмите ее, чтобы появилась следующая строка, и курсор переместился бы на нее. Если после этого осуществить ассемблирование, все встанет на свои места.

Конечно, в настоящем разделе перечислены не все возможные варианты ошибок и сообщения, выдаваемые TASM’ом при их обнаружении. Мы рассмотрели лишь основные из них. С остальными (точнее, с некоторыми из них) мы еще познакомимся в главе, посвященной системе команд микроконтроллера, ибо пока многие команды вам незнакомы, понять смысл некоторых сообщений будет вам крайне затруднительно.

Добавим, что мы рассмотрели сообщения об ошибках, выдаваемых ассемблером TASM. Другие ассемблеры могут выдавать при обнаружении этих ошибок несколько отличающуюся информацию. В целом, когда вы поймете команды МК, для вас не составит труда догадаться, в чем же состоит ошибка. Тем более, все ассемблеры помечают в листинге строки с ошибками, и внимательно посмотрев содержание этих строк, вы быстро поймете, в чем дело.

Александр Фрунзе
alex.fru@mtu-net.ru