ЯЗЫК ФОРТ ДЛЯ «РАДИО-86РК»

Н. ШИХОВ, г. Козьмодемьянск, Республика Марий-Эл

ФОРТ — один из самых молодых языков программирования, однако благодаря таким своим достоинствам, как быстрота освоения (простейшие приемы программирования на нем можно освоить буквально за несколько минут) и высокая производительность, он уже приобрел во всем мире необычайную популярность. Сегодня мы предлагаем читателям версию этого языка для радиолюбительского компьютера "Радио-86РК".

"УМИРАЮТ" ЛИ ВОСЬМИРАЗРЯДНЫЕ МИКРОПРОЦЕССОРЫ?

По миру победно шествуют мощные микропроцессоры i80486, MC68030, MICRO VAX, на смену им идут PENTIUM, MC680XX и новый сверхмощный 64-разрядный "Альфа-чип" фирмы DEC. Сегодня можно приобрести почти все, что только пожелаешь. Иногда кажется, что для простеньких восьмиразрядных компьютеров уже не осталось места под солнцем.

Однако это не так. В микропроцессорной технике есть место и для "слона" PENTIUM, и для "муравья" i8085, и, как это часто бывает, эффективность применения ЭВМ зависит не столько от мощности и совершенства микропроцессора, сколько от скорости и стоимости ее внедрения в производство.

По последним двум показателям восьмиразрядные "персоналки" не имеют конкурентов. Они не только не "вымирают", а даже продолжают совершенствоваться. Например, микропроцессор К580ВМ1, программно и аппаратно совместимый с К580ВМ80, обладает в 2,5 раза более высокой производительностью и расширенной системой команд, в которую входят четырех- и пятибайтные команды.

Восьмиразрядные микропроцессоры прочно обосновались в микроконтроллерах, управляющих несложными технологическими процессами, в измерительной технике. Мощности восьмиразрядных ЭВМ с лихвой хватает для любых бытовых целей — от управления теплицей или инкубатором до автоматического определения номера звонящего абонента. На любительской радиостанции компьютер выполнит любую рутинную работу — от ведения журнала до автоматического управления любым "железом", и даже, если нужно, проведет радиосвязь без участия оператора.

Во всем этом есть, однако, одна хорошо известная тонкость: компьютер выполняет лишь то, что заложено в него программой. А вот оснащенность программным обеспечением, его качество и сервисные характеристики у восьмиразрядных машин неизмеримо ниже. Особенно сказывается дефицит простых, быстродействующих, компактных трансляторов основных языков программирования. Об этом и пойдет речь далее.

Автор имеет опыт программирования на языках БЕЙСИК, ПАСКАЛЬ, ФОРТ и АССЕМБЛЕР для восьми- и шестнадцатиразрядных процессоров. Из экономии времени и места не будем сравнивать эти языки (тот, кто с ними работал, хорошо знает их достоинства и недостатки), а сразу выделим один из них. Он доводит мощность восьмиразрядных ПЭВМ до уровня шестнадцатиразрядных, позволяет выжать из машины все, на что она способна; обеспечивает доступ не только к любому байту, но и биту, находящемуся в любой ячейке памяти или регистре ЭВМ; занимает в памяти наименьший объем и в то же время обеспечивает наиболее быструю компиляцию программы в коды процессора за один проход текста программы; программирование на нем в 5 раз быстрее, чем на БЕЙСИКе, в 15 раз быстрее, чем на ПАСКАЛе и в 50 раз быстрее, чем на АССЕМБЛЕРе.

ЭТОТ ЯЗЫК — ФОРТ

ФОРТ — один из самых молодых языков программирования, однако благодаря быстроте освоения и высокой производительности программирования он уже приобрел во всем мире необычайную популярность среди ценителей красоты в этом сложном, но увлекательном искусстве. В России и за ее пределами имеется немало версий языка, но публикаций на эту тему мало.

Простейшие приемы программирования на ФОРТе можно освоить буквально за несколько минут. В отличие, например, от АССЕМБЛЕРа, программирование на ФОРТе напоминает скорее игру, нежели кропотливый труд алхимика, хотя в результате можно получить программу, не уступающую АССЕМБЛЕРной.

Особенностями языка ФОРТ являются обратная польская (постфиксная) запись арифметических и логических операций, а также широкое использование стеков, привычное для многочисленных владельцев программируемых калькуляторов. Но главная особенность ФОРТа в том, что его каждый пишет сам для себя, а минимальный словарь резидентной части нужно рассматривать лишь как начальный капитал, дающий большой процент годовых, причем ФОРТ, как губка, впитывает и усваивает лексикон и характер своего хозяина, приближаясь к обычному разговорному языку.

Предлагаемая читателям версия языка ФОРТ не единственная, адаптированная для "Радио-86РК". У автора имеется версия ФОРТ-системы, разработанная в НИИСЧЕТМАШ ЛГУ. Она выполнена в строгом соответствии со стандартом FORTH-83, имеет объем 14 Кбайт и словарь примерно из 500 слов. По мнению автора, владельцами "Радио-86РК" эта программа использоваться на практике не может. Ее словарь перегружен промежуточными словами, не используемыми в прикладных программах, есть слова с одинаковыми функциями. Достоинствами этой версии являются ее соответствие стандарту и наличие встроенного ФОРТ-АССЕМБЛЕРа. Стандартный строчный редактор ничего, кроме сожаления, не вызывает.

Дамп предлагаемой версии языка ФОРТ для компьютера "Радио-86РК" приведен в табл. 1, блочные контрольные суммы — в табл. 2. Версия работает с удобным и привычным редактором "МИКРОН". Отсутствие ФОРТ-АССЕМБЛЕРа компенсируется возможностью просто вставлять в программу машинные коды или использовать АССЕМБЛЕР из пакета "МИКРОН".

Словарь включает в себя наиболее употребительные слова из распространенных версий ФОРТа и позволяет писать программы обработки текстов, строк, символов, битов, байтов и шестнадцатиразрядных слов по алгоритмам любой степени сложности. Он наиболее приспособлен для разработки управляющих, игровых и системных программ. Некоторые слова работают несколько иначе, чем в стандартных версиях.

Внутренняя структура словаря нестандартна, но именно благодаря такой структуре удалось достичь компактности и высокого быстродействия предлагаемой реализации, а также простоты генерации и расширения системы модулями, написанными на АССЕМБЛЕРе.

Программирование на ФОРТе можно изучить по литературе [1—3]. После этого рекомендуется разобраться в прилагаемой демонстрационной программе (табл. 3) и тут же провести первые, пусть не всегда удачные, но поучительные эксперименты. Далее будем предполагать, что читателю уже известны "азы" программирования на ФОРТе.

ОСОБЕННОСТИ ПРЕДЛАГАЕМОЙ ВЕРСИИ

Большинство слов резидентной части словаря от COLD до . (точки) работают так же, как и в стандартном FORTH-83, поэтому есть смысл описать только отличия от этого стандарта.

Итак, несколько иначе работают:

Таблица 1

```
0800 C3 1D 08 FF FF FF C3 03 F8 C3 09 F8 B5 0F 00 11  333C
0810 16 08 00 C1 D1 C9 4F 08 43 4F 4C 44 84 01 06 08  8185
0820 21 40 10 1E 0A 0A 77 23 03 1D C2 25 08 36 10 23  94B5
0830 36 00 31 40 10 21 7E 10 CD 06 0A 1F FF CD 82 08  B488
0840 01 00 21 CD D5 0A CD 17 09 C3 18 0A FF FF FF 7C  A319
0850 08 44 52 4F 50 84 F5 4E 23 46 2C FA 67 08 F1 C9  F88C
0860 E1 CD 06 0A 0A 25 FF 21 7E 10 F1 CD 06 0A 0A 73  78E6
0870 74 65 6B 20 70 75 73 74 FF C3 EA 0B 87 08 44 55  C00F
0880 50 83 2B 70 2B 71 C9 9C 08 52 4F 54 83 5E 71 23  C3E1
0890 56 70 23 4E 73 23 46 72 2B 2B 2B C9 B4 08 2D 52  BC0A
08A0 4F 54 84 50 59 4E 23 46 23 7E 73 23 5E 72 2B 2B  BDE4
08B0 73 2B 77 C9 C7 08 4F 56 45 52 84 CD 82 08 23 23  EC0A
08C0 4E 23 46 2B 2B 2B C9 D7 08 53 57 41 50 84 7E 71  228E
08D0 4F 23 7E 70 47 2B C9 E7 08 4B 45 59 83 CD 82 08  4B4D
08E0 CD 40 10 4F 06 00 C9 0F 09 45 4D 49 54 84 3E 0A  484E
08F0 B9 CC 01 09 3E 0D B9 CC 09 09 CD 43 10 CD 56 08  B9BC
0900 C9 CD 82 08 0E 0D C3 FA 08 CD 06 0A 0A FF C9 76  B525
0910 09 57 4F 52 44 53 85 CD 6B 0B CD 82 08 01 10 08  CCD0
0920 C5 CD 82 08 01 48 10 CD 0F 0C CD 36 09 C1 B7 CA  E6AB
0930 4C 09 CD 56 08 C9 7E 23 56 23 B9 C2 48 09 7A B8  AE61
0940 C2 48 09 3E FF 47 4F C9 AF 4F 47 C9 CD 82 08 03  1B17
0950 03 3E 04 F5 CD 60 0B F1 3D C2 53 09 CD F5 09 0A  8F93
0960 F6 F0 C6 04 F5 CD 60 0B F1 3C C2 64 09 CD 56 08  6464
0970 CD 0F 0C C3 20 09 E0 09 27 81 CD 99 09 CD 82 08  292B
0980 01 10 08 C5 CD 82 08 01 48 10 CD 0F 0C CD 36 09  7D82
0990 C1 B7 CA B0 09 01 FF FF C9 EB 2A 4C 10 7E B7 CA  7133
09A0 A7 09 FE 21 FA AC 09 22 4C 10 EB C9 23 C3 9D 09  3A3C
09B0 E5 2A 4C 10 EB 21 02 00 09 1A BE C2 C3 09 23 13  101E
09C0 C3 B9 09 FE 21 FA CF 09 E1 CD 0F 0C C3 83 09 7E  950C
09D0 B7 FA D7 09 C3 C8 09 23 44 4D EB 22 4C 10 E1 C9  2AEC
09E0 EE 09 45 58 45 43 55 54 86 C5 CD 56 08 C9 11 0A  1B1F
09F0 54 59 50 45 84 CD 82 08 CD 1B 0C CD EE 08 03 0A  DCE1
0A00 B7 F8 C8 C3 F5 09 CD 82 08 C1 CD F5 09 03 C3 E9  E9CA
0A10 09 3A 0A 51 55 49 54 84 AF 32 4B 10 31 40 10 CD  D49E
0A20 06 0A 0A 3E FF CD 82 08 01 80 10 CD 42 0A CD 82  2AA7
0A30 08 01 80 10 CD D5 0A C3 1C 0A C9 0A 51 55 45 52  F03E
0A40 59 85 0B C5 E3 22 4E 10 E1 CD 56 08 CD DD 08 3E  D50D
0A50 03 B9 C2 5B 0A CD 56 08 CD 00 F8 CD 82 08 CD 82  FD79
0A60 08 CD EE 08 CD 71 0A 79 CD 56 08 FE 0D C2 4C 0A  D6DA
0A70 C9 3E 0D B9 C2 87 0A CD 82 08 0E 20 CD 87 0A 0E  0911
0A80 00 CD 82 08 CD 87 0A EB 2A 4E 10 23 3E 08 B9 CC  4F16
0A90 A1 0A 3E 18 B9 CC B9 0A 71 22 4E 10 EB CD 56 08  4E50
0AA0 C9 2B 2B EB E5 CD 82 08 13 1A 1B 4F CD EE 08 CD  A66D
0AB0 82 08 CD EE 08 E1 EB 4E C9 EB CD 06 0A 08 FF 1A  0619
0AC0 4F CD 82 08 CD EE 08 EB C9 58 0B 49 4E 54 45 52  B602
0AD0 50 52 45 54 89 C5 E3 22 4C 10 E1 CD 56 08 CD 1C  C9DF
0AE0 0B CD 7A 09 3E FF B9 C2 09 0B B8 C2 09 0B CD 56  88D8
0AF0 08 CD 9C 0B 3A 4B 10 B7 C4 3C 0B C3 DE 0A 0B 0A  8E93
0B00 03 E6 40 CA 29 0B C3 E9 09 3A 4B 10 B7 CA 16 0B  0E13
0B10 CD FE 0A C3 DE 0A CD E9 09 C3 DE 0A E5 2A 48 10  4951
0B20 EB 2A 46 10 73 23 72 E1 C9 E5 2A 48 10 36 CD 23  8DAA
0B30 71 23 70 23 22 48 10 E1 CD 56 08 C9 CD 82 08 01  D2CE
0B40 82 08 CD 29 0B E5 2A 48 10 36 01 23 71 23 70 23  5473
0B50 22 48 10 E1 CD 56 08 C9 66 0B 53 50 41 43 45 85  31B1
0B60 CD 06 0A 20 FF C9 71 0B 43 52 82 CD 06 0A 0A FF  443E
0B70 C9 82 0B 48 45 52 45 84 CD 82 08 E5 2A 48 10 E3  C19F
0B80 C1 C9 93 0B 4E 45 47 41 54 45 86 79 2F 4F 78 2F  D600
0B90 47 03 C9 0B 0C 4E 55 4D 42 45 52 86 CD 82 08 CD  D49D
0BA0 99 09 1A FE 2D C2 B0 0B CC AF 0B CD 8B 0B C9 13  1D29
0BB0 E5 21 00 00 1A FE 21 FA 02 0C FE 30 FA EA 0B FE  6A62
0BC0 3A F2 DB 0B E6 0F 06 00 4F C5 E5 C1 3A 4A 10 3D  6198
0BD0 09 3D C2 D0 0B C1 09 13 C3 B4 0B FE 41 FA EA 0B  6C70
0BE0 FE 47 F2 EA 0B C6 C9 C3 C4 0B 31 40 10 21 7E 10  747D
0BF0 CD 06 0A 0A 3F FF E5 2A 4C 10 E3 C1 CD F5 09 C3  05C2
0C00 46 08 13 EB 22 4C 10 E1 42 4B C9 16 0C 40 81 0A  E8EE
0C10 5F 03 0A 47 4B C9 20 0C 43 40 82 0A 4F 06 00 C9  5A20
0C20 2F 0C 21 81 7E 02 23 03 7E 02 23 CD 56 08 C9 3C  1E56
0C30 0C 43 21 82 7E 23 23 02 CD 56 08 C9 4C 0C 2D 81  35B2
0C40 7E 91 77 23 7E 98 77 2B CD 56 08 C9 59 0C 2B 81  EA66
0C50 7E 23 81 4F 7E 23 88 47 C9 66 0C 32 2A 82 AF 79  AE22
0C60 17 4F 78 17 47 C9 73 0C 32 2F 82 AF 78 1F 47 79  F860
0C70 1F 4F C9 89 0C 3C 81 CD 40 0C 78 E6 80 FA 84 0C  040A
0C80 AF 4F 47 C9 3E FF 4F 47 C9 93 0C 3E 81 CD CE 08  AAAB
0C90 C3 77 0C 9A 0C 3D 81 C3 36 09 AD 0C 30 3D 82 AF  5903
0CA0 B9 C2 AA 0C B8 C2 AA 0C 0B C9 4F 47 C9 C0 0C 30  6790
0CB0 3C 82 3E 80 A0 CA BC 0C 01 FF FF C9 01 00 00 C9  7D40
0CC0 D3 0C 30 3E 82 3E 80 A0 C2 BC 0C 01 FF FF C9 01  8680
0CD0 00 00 C9 E2 0C 41 4E 44 83 7E A1 4F 23 7E A0 47  C103
0CE0 23 C9 F0 0C 4F 52 82 7E B1 4F 23 7E B0 47 23 C9  4A0D
0CF0 04 0D 2C 81 EB 2A 48 10 71 23 70 23 22 48 10 EB  CFB7
0D00 CD 56 08 C9 17 0D 43 2C 82 EB 2A 48 10 71 23 22  0F2C
0D10 48 10 EB CD 56 08 C9 26 0D 58 4F 52 83 7E 23 A9  8C30
0D20 4F 7E 23 A8 47 C9 51 0D 49 46 C2 CD 82 08 01 48  B4F7
0D30 0D CD 29 0B CD 82 08 0E C2 CD 09 0D CD 78 0B CD  6D35
0D40 82 08 03 03 CD F4 0C C9 CD 9F 0C 79 CD 56 08 B7  48F9
0D50 C9 75 0D 45 4C 53 45 C4 CD 82 08 0E C3 CD 09 0D  3C43
0D60 CD 78 0B CD 82 08 03 03 CD F4 0C CD 78 0B CD 8D  9D24
0D70 08 CD 24 0C C9 86 0D 54 48 45 4E C4 CD 78 0B CD  A971
0D80 CE 08 CD 24 0C C9 91 0D 42 45 47 49 4E C5 C3 78  2D9F
0D90 0B AB 0D 52 45 50 45 41 54 C6 CD 82 08 0E C3 CD  773F
0DA0 09 0D CD CE 08 CD F4 0C C3 7C 0D 06 0E 2B 44 4F  5AA4
0DB0 C3 CD 82 08 01 C9 0D CD 29 0B CD 78 0B CD 82 08  9799
0DC0 01 D8 0D CD 29 0B C3 34 0D D1 C5 CD 40 0C C5 CD  652C
0DD0 56 08 C5 CD 56 08 D5 C9 D1 CD 82 08 C1 CD 82 08  2C2C
0DE0 C1 D5 CD BB 08 CD BB 08 CD 36 09 CD 56 08 B7 C2  AB66
0DF0 FD 0D D1 E3 E5 09 44 4D E1 E3 C3 CE 0D C1 D1 C5  3AF6
0E00 23 23 CD 56 08 C9 17 0E 49 81 CD 82 08 EB 21 04  9190
0E10 00 39 4E 23 46 EB C9 28 0E 4A 81 CD 82 08 EB 21  EC08
0E20 0A 00 39 4E 23 46 EB C9 7B 0E 43 52 45 41 54 85  AA2B
0E30 CD 4D 0E CD 78 0B E5 21 07 00 09 E3 C1 CD 3C 0B  4146
0E40 E5 2A 48 10 36 C9 23 22 48 10 C3 20 0B CD 99 09  5C60
0E50 CD 82 08 E5 2A 48 10 22 46 10 36 FF 23 36 FF 23  C8E6
0E60 0E 20 06 7F 1A 77 23 13 04 B9 C2 64 0E 22 48 10  D8E5
0E70 2B 70 EB 22 4C 10 E1 CD 56 08 C9 85 0E 5D C1 3E  90C8
0E80 FF 32 4B 10 C9 8E 0E 5B C1 AF 32 4B 10 C9 A1 0E  B9C1
0E90 22 81 CD 99 09 CD 82 08 E5 2A 48 10 0E 22 C3 62  C825
0EA0 0E B6 0E 41 4C 4C 4F 54 85 E5 2A 48 10 09 22 48  69AD
0EB0 10 E1 CD 56 08 C9 C9 0E 3A C1 CD 30 0E CD 82 08  1819
0EC0 01 F9 FF CD A9 0E C3 7F 0E D3 0E 3B C1 CD 40 0E  BEC5
0ED0 C3 89 0E E4 0E 2E 22 C2 CD 82 08 01 06 0A CD 29  98BC
0EE0 0B C3 92 0E 99 0F 2A 2F 4D 4F 44 85 AF 3E 03 F5  C8B9
0EF0 78 B7 F2 FB 0E CD 8B 0B F1 3F F5 CD 8D 08 F1 3D  0E42
0F00 C2 EF 0E F5 CD 16 0F F1 D0 CD 8B 0B CD CE 08 CD  753A
0F10 8B 0B CD CE 08 C9 C5 CD 56 08 5E 23 56 E5 21 00  D5CF
0F20 00 E5 3E 10 32 7E 10 29 E3 DC 54 0F D4 52 0F E3  7856
0F30 CD 5E 0C D2 3D 0F 19 D2 3D 0F E3 23 E3 3A 7E 10  333D
0F40 3D C2 24 0F C1 EB E1 72 2B 73 00 CD 82 08 C1 C3  EDAA
0F50 5E 0F 29 C9 29 23 37 C9 11 00 00 C3 62 0F 5E 23  5271
0F60 56 23 D5 5E 23 56 23 E3 7D 91 7C 98 D2 60 08 AF  8D36
0F70 29 F5 EB 29 EB D2 79 0F 23 7D 91 6F 7C 98 67 DA  996C
0F80 87 0F F1 13 C3 8C 0F F1 DA 83 0F 09 C6 10 D2 70  0D76
0F90 0F 42 4B E3 CD 82 08 C1 C9 A5 0F 48 45 58 83 3E  82BA
0FA0 10 32 4A 10 C9 B5 0F 44 45 43 49 4D 41 4C 87 3E  A3DD
0FB0 0A 32 4A 10 C9 00 11 2E 81 78 B7 FC F7 0F E5 21  3B56
0FC0 57 10 36 FF 2B 36 20 E3 CD 82 08 01 00 00 3A 4A  96DC
0FD0 10 4F CD 5B 0F 79 CD 56 08 F6 30 FE 3A FA E2 0F  7880
0FE0 C6 07 E3 2B 77 E3 AF B9 C2 C8 0F B8 C2 C8 0F C1  8F48
0FF0 CD F5 09 CD 56 08 C9 CD 06 0A 2D FF CD 8B 0B C9  32F4
```

' TYPE QUERY INTERPRET "

Отсутствуют слова DO LOOP и +LOOP.

Слово ' (апостроф) в отличие от стандартного в FORT-83 не является словом немедленного исполнения, но если это необходимо, его можно сделать таким командой IMMEDIAT '.

Слово ' возвращает значение адреса исполняемой части слова (поля кода, CFA), стоящего после него, или -1, если слово не найдено. Коренное же отличие работы этого слова в том, что оно производит поиск слов в словаре от начала к концу, что связано с особенностями

Таблица 2

```
0800 - 08FF  08B6
0900 - 09FF  A241
0A00 - 0AFF  56F6
0B00 - 0BFF  F14C
0C00 - 0CFF  4EDA
0D00 - 0DFF  CF2B
0E00 - 0EFF  4824
0F00 - 0FFF  F954
----------------
0800 - 0FFF  22B3
================
```

реализации, которая прежде всего оптимизирована по быстродействию. Повторное описание слов с уже имеющимся именем бессмысленно, так как это слово никогда не будет найдено, пока не будет переименовано более раннее определение.

Слово TYPE берет из стека адрес строки, имеющей в конце либо нулевой байт, либо байт с кодом более 7FH, распечатывает эту строку и возвращает адрес байта ограничителя, стоящего в конце строки. Таким образом, это слово может распечатывать текст, порожденный не

Таблица 3

```
HEX

: IMMEDIAT ' 1 - DUP C@ 40 OR SWAP C! ;

: [COMPILE] CD C, ' , ; IMMEDIAT [COMPILE]

: (" HERE " ; IMMEDIAT ("

: )  HERE - ALLOT ; IMMEDIAT )

: ") DUP TYPE DROP [COMPILE] ) ; IMMEDIAT ")

: .. SPACE SPACE SPACE SPACE ; (" ПЕЧАТЬ ЧЕТЫРЕХ ПРОБЕЛОВ")

..  (" ИДЕТ КОМПИЛЯЦИЯ
    ДЕМОНСТРАЦИОННОЙ ПРОГРАММЫ"")

: WHILE [COMPILE] IF ; IMMEDIAT WHILE

: UNTIL [COMPILE] IF
        [COMPILE] ELSE [COMPILE] REPEAT ; IMMEDIAT UNTIL

: DUMP CR
BEGIN DUP DUP
 [ 78 C, CD C, F815 ,
(" ТАК МОЖНО ВСТАВИТЬ КУСОК ПРЯМО В КОДАХ,
   НА АССЕМБЛЕРЕ ЭТО СООТВЕТСТВУЕТ
   MOV A,B
   CALL 0F815H

   ИЛИ В КОДАХ
   78 CD 15 F8 ")

 ] DROP

 [ 79 C, CD C, F815 , ] DROP
 SPACE SPACE (" ПЕЧАТЬ ДВУХ ПРОБЕЛОВ")
 DUP DUP E + SWAP 2
(" СЕЙЧАС В СТЕКЕ  КОНЕЧНОЕ И НАЧАЛЬНОЕ ЗНАЧЕНИЯ ПАРАМЕТРА
   ЦИКЛА СО СЧЕТЧИКОМ, А НА ВЕРШИНЕ - ШАГ ПРИРАЩЕНИЯ ПАРАМЕТРА"
)
 +DO (" ЭТО НАЧАЛО ЦИКЛА СО СЧЕТЧИКОМ ")
   I (" А ЭТО ПАРАМЕТР ЦИКЛА ")
   @ [ 79 C, CD C, F815 , 78 C, CD C, F815 , ] DROP SPACE
  REPEAT (" ЭТО КОНЕЦ ЦИКЛА, ЕСЛИ ДОСТИГНУТО РАВЕНСТВО
  НАЧАЛЬНОГО И КОНЕЧНОГО ЗНАЧЕНИЯ")  SPACE
 DUP DUP F + SWAP 1
 +DO I C@ DUP 20 < OVER 7F > OR IF
   DROP 5F (" ПЕЧАТЬ ПРОЧЕРКА (5FH) ВМЕСТО СИМВОЛА С КОДОМ
            МЕНЬШЕ 20H ИЛИ БОЛЬШЕ 7FH ")
   THEN EMIT
  REPEAT CR
 10 + OVER OVER > IF
REPEAT DROP DROP ;

CR .. (" ТАК РАБОТАЕТ DUMP
 РАСПЕЧАТКА ОТ АДРЕСА 1100 ДО КОНЦА КОДОФАЙЛА, Т. Е. ДО HERE
"") HERE 1100 DUMP
CR ..
("  КОМПИЛЯЦИЯ ЗАМЕДЛЕНА  ИЗ-ЗА МНОЖЕСТВА КОММЕНТАРИЕВ,
    КОТОРЫЕ ТОЖЕ КОМПИЛИРУЮТСЯ, А ЗАТЕМ УНИЧТОЖАЮТСЯ "")
CR ..
("  УБЕРИТЕ ВСЕ КОММЕНТАРИИ, ОГРАНИЧЕННЫЕ СКОБКАМИ
    И ПРОГРАММА СТАНЕТ ПРОСТОЙ И ПОНЯТНОЙ"") CR

: 1+ 03  C, ; IMMEDIAT 1+ (" КОМПИЛИРУЕТ INX B ")

: 2+ 0303 , ; IMMEDIAT 2+ (" ТАК КАК ПАРА BC - ЭТО ВЕРШИНА СТЕКА")

: 1- 0B  C, ; IMMEDIAT 1- (" КОМПИЛИРУЕТ DCX B ")

: 2- 0B0B , ; IMMEDIAT 2-

: R0  1040 ; CR .. R0 .
  (" ЭТО СТЕК ВОЗВРАТОВ, ИМЕННО ЭТО ЗНАЧЕНИЕ
    ЗАПИСЫВАЕТСЯ В SP В СЛОВАХ COLD И QUIT"")

: S0  R0 3E + ; CR CR .. S0 .
(" .., А ЭТО УКАЗАТЕЛЬ АРИФМЕТИЧЕСКОГО СТЕКА,
    ЗАПИСЫВАЕТСЯ В ПАРУ HL
    УСТАНАВЛИВАЕТСЯ ТОЛЬКО СЛОВОМ COLD"")

: TIB S0 2+ ; CR CR ..
(" АДРЕС НАЧАЛА КОМАНДНОЙ СТРОКИ "") TIB . CR


: (KEY) R0 1+ ; CR .. (KEY) @ .
(" МАШИННО-ЗАВИСИМЫЙ АДРЕС ПОДПРОГРАММЫ ВВОДА = F803"") CR

: (EMIT)  R0 2+ 2+ ; CR .. (EMIT) @ .
(" МАШИННО-ЗАВИСИМЫЙ АДРЕС ПОДПРОГРАММЫ ВЫВОДА= F809"") CR

: (LAST) (EMIT) 2+ ;
CR CR .. (" АДРЕС НАЧАЛА ПОСЛЕДНЕГО СЛОВА В СЛОВАРЕ
    НАХОДИТСЯ В ЯЧЕЙКЕ "") (LAST) .

: (H)    (LAST) 2+ ;
CR CR .. (" АДРЕС ПЕРВОЙ СВОБОДНОЙ ЯЧЕЙКИ, Т. Е. КОНЦА КОДОФАЙЛА
    НАХОДИТСЯ В ЯЧЕЙКЕ "") (H) .
CR .. (" (H) @ АНАЛОГИЧНО HERE

 "")
: (BASE)    (H) 2+ ;
CR CR .. (" БАЙТ С АДРЕСОМ (BASE) ХРАНИТ ОСНОВАНИЕ СИСТЕМЫ
    СЧИСЛЕНИЯ
"")
.. (" ПРИМЕР :
    55 2 (BASE) C! . "") 55 2 (BASE) C! .
HEX (" ШЕСТНАДЦАТИРИЧНАЯ СИСТЕМА, А DECIMAL ДЕСЯТИЧНАЯ")

: BINARY 2 (BASE) C! ;
(" BINARY УСТАНОВИТ ДВОИЧНУЮ СИСТЕМУ СЧИСЛЕНИЯ")

: OCTAL 2 (BASE) C! ;
(" OCTAL УСТАНОВИТ ВОСЬМИРИЧНУЮ СИСТЕМУ СЧИСЛЕНИЯ")

: (?STAT) (BASE) 1+ ; CR CR
.. (" В БАЙТЕ С АДРЕСОМ (?STAT) ПРИЗНАК РЕЖИМА
    00 ЗНАЧИТ ИНТЕРПРЕТАЦИЯ
    FF ЗНАЧИТ КОМПИЛЯЦИЯ"")

: (IN)    (BASE) 2+ ;
(" В ЯЧЕЙКЕ (IN) АДРЕС ОБРАБАТЫВАЕМОГО В ДАННЫЙ МОМЕНТ
 СИМВОЛА ПРОГРАММЫ")

: (OUT) (IN) 2+ ;
(" В ЯЧЕЙКЕ (OUT) АДРЕС,
 В КОТОРЫЙ БУДЕТ ЗАПИСАН СЛЕДУЮЩИЙ
 ВВЕДЕННЫЙ С УСТРОЙСТВА ВВОДА СИМВОЛ
 Т. Е С КЛАВИАТУРЫ, ЕСЛИ В (KEY) АДРЕС F803")

  : EDIT (" СЛОВО : КОМПИЛИРУЕТ ЗАГОЛОВОК СЛОВА EDIT")
  0 (" КЛАДЕМ В СТЕК 0 ")
  EXECUT (" ИСПОЛНИТЬ ПРОГРАММУ ПО АДРЕСУ, ЛЕЖАЩЕМУ В СТЕКЕ")
  ; (" ; КОМПИЛИРУЕТ КОМАНДУ RET
  И УСТАНАВЛИВАЕТ РЕЖИМ ИНТЕРПРЕТАЦИИ")

: ?TERM DUP [ CD C, F812 , 4F47 , ] ;
(" ТАК МОЖНО ОБРАТИТСЯ К МОНИТОРУ ПО АДРЕСУ F812 ")


: ?KEY 0 [ CD C, F81B , 4F C, ] ;
(" .., А ТАК ПО АДРЕСУ F81B ")

: CMOVE 1 1 +DO OVER C@ OVER C! 1+ SWAP 1+ SWAP
  REPEAT DROP DROP ;

(" CMOVE И CMOVE> ЭТО СТАНДАРТНЫЕ СЛОВА ЯЗЫКА ФОРТ")

: CMOVE> ROT OVER + -ROT DUP ROT + SWAP
  1 1 +DO 1- SWAP 1- SWAP OVER C@ OVER C!
  REPEAT DROP DROP ;

: PRINT 1 1 +DO DUP C@ EMIT 1+ REPEAT DROP ;
  CR CR (IN) @
        DECIMAL 1048 PRINT
  (" РАСПЕЧАТАТЬ 1048 БАЙТ ИЗ ВХОДНОГО БУФЕРА")

  CREAT BUFF
  HEX 40 ALLOT (" ТАК СОЗДАЮТСЯ МАССИВЫ РАЗМЕРОМ 64 БАЙТА")

: INPUT CR ." #  ЖДУ? "
(" СЛОВО INPUT ВВОДИТ ЧИСЛО ИЛИ СЛОВО БЕЗ ВЫХОДА В QUIT ")
(" ИСПОЛЬЗУЯ СВОЙ БУФЕР РАЗМЕРОМ В 64 БАЙТА")
  BUFF QUERY (IN) @ [ C5 C, ] DROP
  BUFF INTERPRET DUP [ C1 C, ] (IN) ! ;

(" ДОПУСТИМ У НАС ОТДЕЛЬНЫЙ ТЕРМИНАЛ, С КОТОРЫМ ЕСТЬ СВЯЗЬ
   ЧЕРЕЗ 580ВВ55 КАНАЛ А В РЕЖИМЕ 2 ")
: PPAINI -1 A003 C! 0D A003 C! 9 A003 C! A000 C@ DROP ;
  PPAINI (" ИНИЦИАЛИЗАЦИЯ ПОРТА ")

: INT? BEGIN A002 C@ 8 AND IF ELSE REPEAT ;
(" ОЖИДАНИЕ БИТА ПРЕРЫВАНИЯ")

: TRMO INT? BEGIN A002 C@ 80 AND IF A000 C! ELSE REPEAT ;
(" ПОДПРОГРАММА ВЫВОДА БАЙТА В КАНАЛ А")

: TRMI INT? BEGIN A002 C@ 20 AND IF A000 C@
  7F AND ELSE REPEAT ;
(" ПОДПРОГРАММА ВВОДА БАЙТА ИЗ КАНАЛА А")

(" СЕЙЧАС ДОСТАТОЧНО ЗАПИСАТЬ АДРЕСА
   ПОДПРОГРАММ TRMI В (KEY), А TRMO В (EMIT)
   НАПРИМЕР, ТАК
   ' TRMI (KEY) ! ' TRMO (EMIT) !
   И ВАША МАШИНА БУДЕТ РАБОТАТЬ
   БЕЗ МОНИТОРА ДИСПЛЕЯ И КЛАВИАТУРЫ")

  : KONST1 12 ;   (" ТАК СОЗДАЮТСЯ КОНСТАНТЫ ")

  CREAT VAR1 13 , (" А ТАК - ПЕРЕМЕННЫЕ ")
```

```
  CREAT VAR$ " ABRAKADABRA"
  (" А ТАК МАССИВЫ ЗАПОЛНЕННЫЕ ЛИТЕРАМИ ")

  CREAT ARR1 1 , 2 , 3 , (" А ТАК - МАССИВ ИЗ ТРЕХ СЛОВ")

  CREAT ARR2 4 C, 5 C, 6 C, (" ТАК МАССИВ ИЗ ТРЕХ БАЙТ ")

(" ИМЯ МАССИВА ИЛИ ПЕРЕМЕННОЙ ВОЗВРАЩАЕТ
   В АРИФМЕТИЧЕСКОМ СТЕКЕ ЕГО АДРЕС")

 : := ' 2+ 2+ ! ;
(" ЭТО СЛОВО МОЖЕТ ИЗМЕНИТЬ ЗНАЧЕНИЕ КОНСТАНТЫ
  В РЕЖИМЕ ИНТЕРПРЕТАЦИИ
  ПРИМЕР") 18 := KONST1

: TO DUP
   [ C1 C, 3 C, 3 C, 3 C, C5 C, B C, B C, 1 B
   [ 3 C, 3 C, 3 C, 3 C, ] ! ;
(" А ЭТО СЛОВО МОЖЕТ ИЗМЕНИТЬ ЗНАЧЕНИЕ КОНСТАНТЫ,
  ЕСЛИ ЕГО СКОМПИЛИРОВАТЬ ВНУТРИ ДРУГОГО СЛОВА,
  НАПРИМЕР ")
 : PRIMER1 19 TO KONST1 ;
 (" СЛОВО PRIMER1 ЗАПИШЕТ 19 В KONST1")

 : PICK DUP [ C1E5 , ] OVER + SWAP + 2+ @ ;
(" PICK СНИМАЕТ ИЗ СТЕКА НОМЕР ЭЛЕМЕНТА СТЕКА И КЛАДЕТ В
   СТЕК КОПИЮ ЭТОГО ЭЛЕМЕНТА ")

(" НИЖЕ ОПИСАНЫ НЕКОТОРЫЕ КОМАНДЫ АССЕМБЛЕРА, КОТОРЫЕ МОЖНО
   ПРИМЕНЯТЬ ВНУТРИ ОПИСАНИЙ ДРУГИХ СЛОВ ВМЕСТО КОДОВ
   ЭТОТ СПИСОК МОЖНО УВЕЛИЧИТЬ, ОН ЗАМЕНИТ ФОРТ-АССЕМБЛЕР,
   КОГДА НЕОБХОДИМО НАПИСАТЬ ПРОГРАММУ МАКСИМАЛЬНОГО
   БЫСТРОДЕЙСТВИЯ ")
 : OUT 79 C, D3 C, NUMBER C, ; IMMEDIAT OUT
(" ПОЛЕЗНОЕ СЛОВО, ЕСЛИ НЕОБХОДИМО ВЫВЕСТИ БАЙТ
   В ПОРТ НАХОДЯЩИЙСЯ В ПРОСТРАНСТВЕ ВВОДА/ВЫВОДА")

 : IN  DB C, NUMBER C, 4F C, ; IMMEDIAT IN
(" IN И OUT МОЖНО ПРИМЕНЯТЬ ТОЛЬКО ВНУТРИ ДРУГИХ СЛОВ,
   КАК И ВСЕ СЛОВА С ПРИЗНАКОМ IMMEDIAT ")

(" ПРИМЕР ")
  : OUTA0 OUT A0 ; (" ВЫВОД В ПОРТ A0, СТЕК БЕЗ ИЗМЕНЕНИЯ ")
  : INA0  IN  A0 ; (" ВВОД ИЗ ПОРТА A0 В МЛАДШИЙ БАЙТ ЧИСЛА,
  ЛЕЖАЩЕГО В АРИФМЕТИЧЕСКОМ СТЕКЕ ")

: A,# 3E C, NUMBER C, ; (" A,# 11 КОМПИЛИРУЕТ MVI A,11 ")
  IMMEDIAT A,#
 : A,O AF C, ; (" КОМПИЛИРУЕТ XRA A ОЧИСТКА АККУМУЛЯТОРА")
  IMMEDIAT A,O
 : A,C 79 C, ; (" КОМПИЛИРУЕТ MOV A,C ")
  IMMEDIAT A,C
 : C,A 4F C, ; (" КОМПИЛИРУЕТ MOV C,A ")
  IMMEDIAT C,A
 : A,B 78 C, ;
  IMMEDIAT A,B

WORDS HERE (LAST) @ DUMP
CR .. (" A,B СООТВЕТСТВУЕТ ТЕКСТУ НА АССЕМБЛЕРЕ"")
CR
CR .. (" DW @AB"")
CR .. (" DB 'A,B',83H+40H ;40H = IMMEDIAT"")
CR .. (" CALL 0882H ; DUP"")
CR .. (" LXI  B,78H ; ЗАМЕНА КОДА В ВЕРШИНЕ СТЕКА НА 78H"")
CR .. (" CALL 0D09H ; C,"")
CR .. (" RET"")
CR .. (" @AB:"")
CR .. (" И КОМПИЛИРУЕТ КОМАНДУ MOV A,B ИЛИ 78H"")
QUIT
```

только в системе ФОРТ, но и другими компиляторами.

Слово QUERY берет из стека адрес буфера и заносит в него символы с устройства ввода, пока не будет введен код <ВК> или CTRL+C. Так как это слово берет адрес буфера из стека, то входной буфер можно расположить в любом месте ОЗУ. QUERY — это встроенный редактор системы ФОРТ. Весь порожденный этим словом текст является одной большой строкой, в которой допускаются любые печатные символы в том числе <ПС>. Редактирование внутри QUERY возможно с использованием клавиш курсора <— и —> .

Слово INTERPRET берет из стека адрес буфера и интерпретирует текст из этого буфера, пока не встретит слово QUIT или два нулевых байта, которые заносятся QUERY при вводе кода <ВК>. Так как адрес текста берется из стека, то можно интерпретировать программу, расположенную в любом месте ОЗУ или ПЗУ.

Слово " (кавычки) берет из входного буфера символы, расположенные после него, и компилирует их на вершину кодофайла (для начинающих мы несколько позднее дадим определение этого понятия), пока снова не встретит символ ". На этом символе компиляция заканчивается, а на вершину кодофайла компилируется байт-ограничитель, имеющий код более 7FH, в младших семи битах которого хранится информация о длине строки. Символ " в конце строки не компилируется, а используется в качестве разделителя, т. е. очередное слово может быть записано после него без обязательного пробела, разделяющего слова в тексте программы.

И наконец, слова DO , LOOP и +LOOP. Они просто не нужны, так как в предлагаемой версии имеется только одна универсальная форма организации цикла со счетчиком:

: XX <ОПЕРАТОРЫ> KKKK NNNN MMMM +DO <ОПЕРАТОРЫ ЦИКЛА> REPEAT <ОПЕРАТОРЫ> ;

где KKKK — конечное значение параметра цикла, NNNN — его начальное значение, MMMM — приращение параметра цикла (шаг цикла). Шаг может быть как положительным, так и отрицательным. Цикл выполняется последний раз, когда параметр цикла достигает конечного значения.

Если в программе имеются вложенные циклы со счетчиком, то доступ к параметру внутреннего цикла обеспечивает слово I, а к параметру внешнего цикла — слово J.

Все остальные слова выполняют абсолютно те же функции, что и в стандартном FORTH-83, и подробно описаны в [1]. Это короткое, но полное описание языка ФОРТ и первые 18 с. полностью относятся и к предлагаемой реализации.

НЕСКОЛЬКО СЛОВ О ЦИКЛАХ С УСЛОВИЕМ

Вместо
BEGIN <УСЛОВИЕ> WHILE <ОПЕРАТОРЫ ЦИКЛА> REPEAT
рекомендую
BEGIN <УСЛОВИЕ> IF <ОПЕРАТОРЫ ЦИКЛА> REPEAT
Вместо
BEGIN <УСЛОВИЕ> UNTIL
рекомендую
BEGIN <УСЛОВИЕ> IF ELSE REPEAT

НЕМНОГО О НАЧАЛЬНОЙ СТАДИИ РАБОТЫ ПРОГРАММЫ

Первое, что делает ФОРТ, — заполняет некоторые ячейки ОЗУ константами. 10 байт — с адреса 806H по 80FH — в том же порядке переносятся в ОЗУ, начиная с адреса 1040H: первые 3 байта — это команда JMP 0F803H на подпрограмму ввода символа с клавиатуры, следующие 3 байта — команда JMP 0F809H на подпрограмму вывода символа на экран, еще 2 байта — адрес поля связи последнего слова в словаре (фактически это адрес первого байта последнего слова в словаре). И наконец, последние 2 байта — адрес первой свободной ячейки кодофайла. Этот же адрес записывается и в поле связи последнего слова в словаре (обычно он указывает на байт, следующий за последним байтом последнего слова в словаре). Если возникнет необходимость расширить ФОРТ-систему словами, написанными на АССЕМБЛЕРе, эти два адреса необходимо будет скорректировать.

Затем инициируются стек возвратов (занесением в регистровую пару SP адреса 1040H) и стек параметров (занесением в пару HL адреса 103EH), автоматически компилируется содержимое текстового буфера, начиная с адреса 2100H. Если стартовый файл находится по другому адресу, то его следует занести в ячейки с адресами 841H и 842H.

Стартовый файл должен содержать хотя бы одно слово QUIT <ПРОБЕЛ> или два нулевых байта (см. демонстрационную программу в табл. 3).

Для выхода в МОНИТОР достаточно нажать клавишу <F4> или ввести команду F86C EXECUT. Клавишу <F4> можно перепрограммировать, записав по адресам A59H и A5AH другое значение адреса перехода.

(Окончание следует)

ЛИТЕРАТУРА

1. Бураго А. Ю., Кириллин В. А., Романовский И. В. ФОРТ — язык для микропроцессоров. — Л.: Знание, 1989.

2. Семенов Ю.А. Программирование на языке ФОРТ. — М.: Радио и связь, 1991.

3. Библиотека информационной технологии. Вып. 2. Под ред. Г. Р. Громова. — М.: Инфоарт, 1991.

ЯЗЫК ФОРТ ДЛЯ "РАДИО-86РК"

СОВЕТЫ ПО ПРОГРАММИРОВАНИЮ

Н. ШИХОВ, г. Козьмодемьянск, Республика Марий-Эл

РАБОТА С ЯЗЫКОМ В РЕЖИМЕ ИНТЕРПРЕТАЦИИ

Язык ФОРТ может работать без текстового редактора, однако гораздо удобнее использовать его с одним из текстовых редакторов, например, из пакета "МИКРОН". При этом отпадает проблема сохранения текстов программ. Еще лучше, если обе эти программы "зашиты" в ROM-диске и загружаются одновременно. В статье описывается работа с версией, адаптированной для персонального компьютера "Радио-86РК".

Предположим, что читатель уже ввел коды интерпретатора и редактора "МИКРОН" в свой компьютер и запустил редактор командой G0. Теперь следует проверить работоспособность системы. В редакторе необходимо набрать строку:

: EDIT 0 EXECUT; WORDS QUIT

и нажатием клавиши <СТР> выйти в ФОРТ, аналогично выходу в АССЕМБЛЕР. На экране дисплея появится список доступных в данный момент слов (их можно называть командами или операторами, в зависимости от вкуса) и символ >, приглашающий к вводу этих слов. Последним в словаре будет слово EDIT. Набрав его и нажав затем клавишу <ВК>, вы снова перейдете в текстовый редактор "МИКРОН". Если все получилось, как описано, вас можно поздравить с первым успехом: вы написали первую программу на языке ФОРТ. Дело в том, что в словаре не было слова EDIT, и вы только что описали его, используя слово EXECUT.

В языке ФОРТ всего три основных правила:

— использовать только известные системе слова,

— строго следить за состоянием стеков,

— соблюдать парность парных слов.

Изучение ФОРТА лучше всего построить на экспериментальной проверке работы всех слов базового словаря. Кроме них, в языке имеются еще и числа, которые не должны выходить за пределы шестнадцати двоичных разрядов, причем старший разряд является знаковым.

Пожалуй, первое, что должен сделать программист, — это установить систему счисления для чисел, вводимых с клавиатуры и выводимых на дисплей. Командой HEX устанавливают шестнадцатиричную систему счисления (при "холодном" запуске эта система устанавливается автоматически), командой DECIMAL — привычную для всех десятичную систему. С этой команды и можно начать.

Надо сказать, что числа и команды можно вводить как по одному, нажимая после ввода каждого клавишу <ВК>, так и группами. Буфер ввода с клавиатуры вмещает до 128 символов. Введенные символы можно редактировать перемещением курсора и перепечаткой символов, введенных с ошибкой, между всеми вводимыми словами и числами должно быть не менее одного пробела. После нажатия клавиши <ВК> все числа помещаются в стек, а все слова немедленно исполняются, причем большинство слов работает с арифметическим стеком, являющимся универсальным средством передачи как числовых, так и логических операндов. Арифметическим стеком или стеком параметров в данной версии ФОРТа называется совокупность ячеек памяти и регистров процессора, обеспечивающая хранение двубайтных чисел и безадресный доступ к верхнему элементу стека.

Кроме арифметического, в ФОРТ-системах имеется еще и стек возвратов, но о нем разговор особый.

Логические операнды могут принимать значение либо FALSE, либо TRUE. Логическое значение FALSE (ложь) — шестнадцатиразрядное число, в котором все разряды равны нулю (т. е. 0), TRUE (истина) — любое шестнадцатиразрядное число, не равное 0.

Для распечатки (в текущей системе счисления) числа, лежащего на вершине стека, имеется слово . (точка).

Например,

1 2 3 . . .<ВК>

распечатает:

3 2 1

1 2 3 . . 4 . <ВК>

распечатает: 3 2 4

Первое число лежит на самом дне стека, и мы его еще не сняли. Взять число из стека может слово DROP или . (точка), т. е. при наборе

. <ВК>

распечатается:

1

Если же еще раз ввести:

. <ВК>

распечатается:

СТЕК ПУСТ

Продолжение. Начало см. в "Радио", 1995, № 7.

Таблица 4

Имя	Состояние стека	Комментарии
DUP	N1 > N1 N1	Дублирует верхний элемент стека
DROP	N1 > -	Снимает верхний элемент стека
SWAP	N1 N2 > N2 N1	Меняет местами два верхних элемента
ROT	N1 N2 N3 > N2 N3 N1	Поднимает третий элемент стека (T1)на его вершину
-ROT	N1 N2 N3 > N3 N1 N2	Кладет верхний элемент под два элемента, лежавших под ним
OVER	N1 N2 > N1 N2 N1	Кладет на вершину стека копию второго элемента (N1)

Таблица 5

Имя	Состояние стека	Комментарии
+	1 2 -> 3	Снимает со стека два слагаемых и кладет в стек их сумму
-	4 5 -> -1	Снимает со стека вычитаемое и уменьшаемое и кладет в стек их разность
NEGATE	8 -> -8	Меняет знак числа на вершине стека
2*	3 -> 6	Умножает число на 2
2/	8 -> 4	Делит число на 2 без остатка
*/MOD	7 3 5 -> 4 1	Умножает третий элемент (7) на второй (3) и делит с остатком на первый (5), оставляет на вершине остаток от деления, а под ним - частное от деления

Таблица 6

Имя	Состояние стека	Комментарии
AND	A B -> C	Производит поразрядное логическое умножение 16-разрядного числа A на 16-разрядное число B, т. е. в числе C остаются только единицы, совпадающие в числах A и B
OR	A B -> C	В числе C устанавливаются в единицу все разряды, установленные в единицу в числах A или B
XOR	A B -> C	В числе C устанавливаются в единицу только разряды, не совпадающие в числах A и B
>	A B -> C	C=-1, если A>B C=0, если A<=B
<	A B -> C	C=-1, если A<B C=0, если A>=B
=	A B -> C	C=-1, если A=B C=0, если A><B
0=	A -> C	C=-1, если A=0 C=0, если A><0
0>	A -> C	C=-1, если A>0 C=0, если A<=0
0<	A -> C	C=-1, если A<0 C=0, если A>=0

Таблица 7

Имя	Состояние стека	Комментарии
COLD	XXXX -> -	Холодный старт, очистка стеков, установка системных переменных и конфигурации системы, исполнение стартового файла
QUIT	XXXX -> XXXX	Очистка стека возвратов и переход в бесконечный цикл ввода (QUERY) и интерпретации (INTERPRET) команд с терминала
QUERY	ADR -> -	Снимает со стека адрес и вводит с терминала в ОЗУ строку символов, начиная с этого адреса
INTERPRET	ADR -> -	Снимает со стека адрес и интерпретирует текст, начиная с этого адреса
KEY	- -> COD	Ожидает нажатия клавиши и кладет в стек код символа в виде 16-разрядного числа
EMIT	COD -> -	Снимает со стека ASCII код и распечатывает его на терминале в виде символа
WORDS	- -> -	Распечатывает словарь ФОРТ-системы
SPACE	- -> -	Печатает на терминале один пробел
CR	- -> -	Выдает на терминал коды <ВК>, <ПС>
TYPE	ADR1 -> ADR2	Снимает со стека адрес и распечатывает все символы с кодами >0, но <80H; если встречается другой код, возвращает его адрес (ADR2)
HEX	- -> -	Устанавливает шестнадцатиричную систему счисления
DECIMAL	- -> -	Устанавливает десятичную систему счисления
NUMBER	- -> N	Преобразует строку символов, следующую после него, в число на вершине стека с учетом действующей системы счисления
.	N -> -	Преобразует число с вершины стека в строку символов с учетом действующей системы счисления и выдает на терминал
@	ADR -> N	Снимает со стека адрес и кладет в стек два байта, считанные с этого адреса
C@	ADR -> N	Снимает со стека адрес и кладет в стек один байт, считанный с этого адреса
!	N ADR-> -	Записывает двубайтное число N по адресу ADR
C!	N ADR-> -	Записывает младший байт числа N по адресу ADR
,	N -> -	Записывает двубайтное число N на вершину кодофайла
C,	N -> -	Записывает младший байт числа N на вершину кодофайла
HERE	- -> ADR	Кладет в стек адрес вершины кодофайла
'	- -> ADR	(Апостроф) по имени слова находит адрес исполняемой части (CFA) слова или -1, если слово не найдено
EXECUT	ADR -> -	Снимает со стека адрес и передает управление программе или подпрограмме по этому адресу
CREAT	- -> -	Создает в словаре новое слово с именем, стоящим после CREAT, т. е. CREAT <ИМЯ>
ALLOT	N -> -	Резервирует N байт на вершине кодофайла для переменных и массивов. Например, CREAT ARR1 20 ALLOT создает массив с именем ARR1 размером 20 байт. Исполнение слова ARR1 кладет в стек адрес массива
"	- -> -	Компилирует строку символов на вершину кодофайла. Например, CREAT ARR2 " ABRAKADABRA " создает массив, заполненный литерами
:<ИМЯ>	- -> -	Подобно CREAT создает новое слово и переключает интерпретатор в режим компиляции; используется для описания новых слов

Это же сообщение мы получим при работе некорректно написанной программы, пытающейся извлечь число из пустого стека; кроме того, в этом случае произойдет аварийное прекращение работы программы и выход в QUIT. Еще более худшие результаты получим, если будем помещать числа в стек и не будем их снимать: стек "наедет" на рабочие ячейки интерпретатора, и мы не получим никаких сообщений, а просто разрушим ФОРТ-систему.

Следовательно, при программировании на ФОРТе пользователь должен строго следить за всеми изменениями, происходящими на стеках (их два, но о втором — стеке возвратов — мы пока не говорим). Для надежной работы интерпретатора не следует помещать в стек более шестнадцати чисел; к тому же большие объемы чисел и символов лучше хранить в виде массивов, а стек оставить только для сверхоперативных переменных, число которых обычно не превышает трех.

Для трех верхних элементов стека в словаре имеется несколько слов, обеспечивающих быстрый доступ к любому из этих чисел. Для уяснения их работы проведем несколько экспериментов.

Наберем:

```
1 2 DUP . . . <BK>
```

получим:

```
2 2 1
```

Слово DUP положило в стек копию верхнего элемента.

Наберем:

```
1 2 3 DROP . . <BK>
```

получим:

```
2 1
```

Слово DROP уничтожило (безвозвратно) верхний элемент стека.

Наберем:

```
1 2 3 SWAP . . . <BK>
```

получим:

```
2 3 1
```

Слово SWAP поменяло местами два верхних элемента стека.

Коротко это можно описать в виде таблицы, в левой части которой находится само слово, а в правой — вначале состояние стека до введения этого слова, а затем (после разделителя в виде стрелки) — его состояние после того, как это слово введено. Пустой стек можно обозначить прочерком. В крайней правой колонке удобно поместить комментарии. Составим такую таблицу для слов, оперирующих верхними элементами стека (табл. 4). Попробуйте, используя эти слова и уже известное слово . (точка), убедиться в верности комментариев, приведенных в правой части таблицы.

Если простые операции с элементами стека уже наскучили, то можно перейти к более сложным экспериментам в области целочисленной арифметики (табл. 5). Последнее слово в этой таблице, кроме действий, указанных в комментарии, проверяет частное от деления на переполнение, и если оно имеет более шестнадцати двоичных разрядов, то происходит аварийное прерывание программы и выход в QUIT. Результат умножения может быть и 32-разрядным.

Убедиться в правильности комментариев можно так же, как и в предыдущем случае, вводом чисел и слов из табл. 5, проверяя полученные результаты словом . (точка). Для усложнения опытов можно вводить по несколько операторов сразу.

Например:

```
1 2 3 4 5 + + SWAP - . . <BK>
```

получим:

```
10 1
```

Когда почувствуете, что с арифметикой у вас все в порядке и обратная польская бесскобочная запись арифметических действий (когда сначала пишутся операнды, а затем операторы, выполняющие арифметические действия) уже не вызывает затруднений, можно перейти к освоению логических операций.

Так как логических операторов всего три, имеет смысл свести их в одну таблицу с операторами сравнения (табл. 6). Попробуйте (желательно в шестнадцатиричной системе счисления) произвести необходимые действия над числами, лежащими на стеке, с помощью слов из табл. 6, проверяя полученные результаты словом . (точка). Для примера приведем некоторые возможные результаты:

```
7 1 AND . <BK>
1
7 1 OR . <BK>
7
7 1 XOR . <BK>
6
7 1 > . <BK>
-1
7 1 < . <BK>
0
7 1 = . <BK>
0
```

Если ваши результаты совпадают с приведенными и вы поняли, почему это происходит, то можно считать, что освоение работы интерпретатора закончено. Прежде чем перейти к наиболее трудной части, приведем таблицу слов, которые часто употребляются при работе с интерпретатором (табл. 7).

Немного о самой процедуре интерпретации, которую производит слово INTERPRET. Вначале любое встретившееся слово с помощью слова ' (апостроф) отыскивается в словаре, и если оно найдено, его адрес передается слову EXECUT, которое тут же его исполняет. Если же слово не найдено, то с помощью слова NUMBER оно преобразуется в число, которое кладется в стек, а если слово не является и числом, то интерпретация прекращается, печатается знак вопроса, распечатывается весь текст входной программы, начиная с ошибочного слова, а затем — словарь системы, напоминая, что можно использовать только известные системе слова.

ПОДПРОГРАММНЫЙ ШИТЫЙ КОД

В предыдущем разделе мы рассмотрели работу системы ФОРТ в режиме интерпретации программы, вводимой с клавиатуры или из входного файла. У читателя, возможно, возник вопрос, как интерпретатор находит слова в словаре системы? Дело в том, что все слова, входящие в ФОРТ-систему, представляют собой единый список, называемый шитым кодом. Занимаемое им адресное пространство называется кодофайлом. В версии для "Радио-86РК" применен предложенный автором подпрограммный шитый код, который мы и будем разбирать.

Для примера приведем распечатку дампа двух рядовых слов, например, HEX и DECIMAL.

```
ADDR   LFA    NFA          CFA
0F99   0FA5   HEX 83       3E 10  32 4A 10 C9
0FA5   0FB5   DECIMAL 87   3E 0A  32 4A 10 C9
```

Как видим, слово HEX начинается не с имени, а с некоторого двубайтного адреса, указывающего на начало слова DECIMAL, которое, в свою очередь, также начинается с двубайтного адреса, указывающего на начало следующего за ним слова.

Будем называть адреса, с которых начинается каждая словарная статья, полем связи, так как именно они обеспечивают связь всех слов в словаре и позволяют, зная адрес первого и последнего слова, отыскать и все остальные. Заметим, что в описываемой версии список слов связан от начала к концу. Обычно поле связи обозначают аббревиатурой LFA. За LFA следуют ASCII коды имени и один стоп-байт. Поле имени кратко называют NFA.

Стоп байт отличается от ASCII кода имени тем, что имеет установленный в 1 старший (восьмой) знаковый бит. В младших шести разрядах стоп-байта указывается длина имени (ASCII кодов), а в седьмом бите — признак немедленного исполнения (IMMEDIAT), о котором мы поговорим позже. У слова HEX признака IMMEDIAT нет, и стоп-байт имеет значение 83H, а у слова DECIMAL — 87H. Так как под имя отведено только шесть бит, его длина не может превышать 63 символов. За NFA следует поле кодов CFA, состоящее из машинных команд процессора. Заметим, что именно адрес CFA возвращает слово ' (апостроф). CFA слова HEX состоит из трех машинных команд, которые на языке АССЕМБЛЕРа выглядят так:

```
MVI A,10H     ; записать в аккумулятор
              ; число 16
STA 104AH     ; переписать его в ячейку
              ; с адресом 104AH
RET           ; возврат через стек
              ; возвратов
```

Рабочая ячейка с адресом 104AH хранит текущее основание системы счисления, которое используется словами NUMBER и . (точка).

Если необходимо установить восьмиричную систему счисления, то следует подать команду

HEX 8 104A C! <BK>

или ввести в словарь новое слово OCTAL. Как это сделать, мы узнаем из следующего раздела.

За полем CFA может находиться поле параметров PFA. У переменных оно состоит из одного или двух байт, а у массивов быть любого размера. У констант и других слов ФОРТ-системы PFA отсутствует, т. е. имеет нулевой размер, хотя командой N ALLOT можно создать такое поле у любого слова, описанного последним. Следует отметить, что в других версиях языка ФОРТ поле LFA обычно следует за NFA, а последнее не заканчивается стоп-байтом, а начинается с байта-счетчика. Такую конструкцию называют строкой со счетчиком. Автор данной версии отказался от строки со счетчиком, так как использование строки со стоп-байтом имеет некоторые преимущества и более распространено в других системных и прикладных программах. Расположение LFA в начале словарной статьи также имеет преимущества при генерации и расширении языка подпрограммами, написанными на АССЕМБЛЕРе.

РАБОТА ЯЗЫКА В РЕЖИМЕ КОМПИЛЯЦИИ

Как мы заметили, работа с языком в режиме интерпретации позволяет вводить с клавиатуры или из входного файла числа и слова, имеющиеся в словаре, но в отличие от других языков программирования, ФОРТ не имеет фиксированного словарного запаса и открыт для пополнения списка слов самими пользователями. Попробуем проверить это на практике. Для начала опишем какое-нибудь простое слово, например OCTAL. Для этого введем с клавиатуры следующую строку:

```
: OCTAL 8 [ HEX ] 104A C! ; WORDS <BK>
```

После распечатки словаря мы увидим, что в нем появилось новое слово OCTAL, которым можно пользоваться так же, как словами HEX и DECIMAL .

Каким же образом новые слова включаются в словарь? Это делает слово : (двоеточие), которое генерирует в конце кодофайла LFA и NFA нового слова, причем в NFA оно заносит имя слова, стоящего после двоеточия (через пробел, разумеется). Затем интерпретатор переключается в режим компиляции, и число 8 не кладется в стек, а в кодофайл компилируются машинные команды, которые при исполнении слова OCTAL будут помещать в стек число 8, затем, если была установлена десятичная система счисления, необходимо установить шестнадцатиричную (чтобы ввести число 104AH), для чего необходимо временно снова выйти в режим интерпретации. В подобных случаях помогает слово [, которое не компилируется, а немедленно исполняется, потому что в стоп-байте имени этого слова имеется установленный в 1 бит признака IMMEDIAT.

Словом HEX устанавливаем нужную систему счисления и снова переводим интерпретатор в режим компиляции словом]. Все следующие за ним слова, если они не имеют признака IMMEDIAT, будут скомпилированы на вершину кодофайла в виде машинных команд. Все числа компилируются в виде двух трехбайтных команд типа:

CALL CFA (слова DUP)

LXI B, <число>

а все слова — в виде одной трехбайтной команды:

CALL CFA (слова)

И, наконец, слово ; (точка с запятой) скомпилирует команду RET и вновь переведет компилятор в режим интерпретации. Таким образом будет сформировано CFA слова (в данном случае — OCTAL). Надо сказать, что слово включается в словарь немедленно после описания LFA и NFA словом : (двоеточие). Это сделано специально для написания сложных рекурсивных программ, когда слово немедленно может обратиться к самому себе; правда, программист должен позаботиться о том, чтобы глубина вложенных обращений не превышала 32, так как под стек возвратов отведено только 64 байта.

Итак, на примере слова OCTAL мы разобрали, как пополнить список слов. Это и есть программирование на языке ФОРТ, когда вы пополняете его словарь словами, которые затем можно использовать в описаниях других слов, пока, наконец, не опишете слово, которое и будет выполнять необходимые действия. Тогда можете сразу исполнить это слово, введя его имя и нажав клавишу <BK>.

Как видите, программа составляется снизу вверх, когда сначала пишутся мелкие фрагменты, а затем окончательное слово, собирающее их воедино. Это, впрочем, не мешает конструировать программу или ее алгоритм и сверху вниз.

Для тренировки напишите слово, которое вводит, интерпретирует и распечатывает текст, расположенный по адресу 2100H. Вот одно из возможных решений:

```
: ПРИМЕР [ HEX ]

  2100 QUERY 2100 INTERPRET

  2100 TYPE CR ."КОНЕЦ ТЕКСТА" . ;
```

(Окончание следует)

ЯЗЫК ФОРТ ДЛЯ «РАДИО-86РК»

СОВЕТЫ ПО ПРОГРАММИРОВАНИЮ

Н. ШИХОВ, г. Козьмодемьянск, Республика Марий-Эл

СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ

Как и подобает любому солидному языку, а в этом он превзошел, пожалуй, и ПАСКАЛЬ, язык ФОРТ позволяет писать структурированные программы любой степени сложности без использования меток. Хотя словарь ФОРТа не фиксирован и даже начинающий программист может описать слово, вводящее понятие метки, он быстро убедится в полной бесполезности этого открытия, так как в базовом словаре имеются все необходимые слова для описания ветвлений, циклов с проверкой условия как в начале, так и в конце цикла, а также циклов со счетчиком. Основным словом, позволяющим реализовать упомянутые структуры, является слово IF. Как и прежде, рассмотрим его работу на простом примере:

```
: ПРИМЕР1 KEY DUP [ HEX ] 20 =
  IF ." ПРОБЕЛ " DROP
  ELSE EMIT
  THEN ." - ЭТА КЛАВИША ИСПРАВНА " ;
```

Как работает это слово? Слово KEY ожидает нажатия клавиши и помещает в стек код символа, введенного с терминала. Затем слово DUP кладет в стек число 20H, а слово = сравнивает эти числа и возвращает в стек либо TRUE (т. е. -1), если эти числа равны, либо FALSE (т. е. 0), если они не равны. Слово IF берет логическое значение, оставленное словом = , и если оно имеет значение TRUE, то выполняются операторы, расположенные за словом IF (см. пример 1), если был введен код пробела 20H, то будет напечатано слово "ПРОБЕЛ". Затем слово ELSE передаст управление на операторы, стоящие после слова THEN. Если же слово IF возьмет из стека значение FALSE, то оно передаст управление на операторы, стоящие после слова ELSE. Слово THEN не выполняет никаких действий, а лишь расставляет на этапе компиляции правильные адреса для ссылки вперед в слове ELSE или IF (если ELSE отсутствует). Конечно же, парность операторов IF THEN абсолютно необходима, и хотя в данной версии она и не проверяется самим компилятором, программист должен сам строго следить за соблюдением этого правила. В общем виде структуру слов с операторами ветвления можно описать примерно так:

полная альтернатива:

```
: имя <операторы, оставляющие в стеке условие>
  IF <операторы выполняются, если в стеке было TRUE>
  ELSE <операторы выполняются, если в стеке было FALSE>
  THEN <операторы выполняются всегда> ;
```

или неполная альтернатива:

```
: имя <операторы, оставляющие в стеке условие>
  IF <операторы выполняются, если в стеке было TRUE>
  THEN <операторы выполняются всегда> ;
```

или неполная альтернатива с инверсией:

```
: имя <операторы, оставляющие в стеке условие>
  IF ELSE <операторы выполняются, если в стеке было FALSE>
  THEN <операторы выполняются всегда> ;
```

Для инверсии логического значения на стеке можно применить и слово 0=, но это несколько медленнее, чем ELSE.

После столь подробного описания операторов ветвления желательно поупражняться в написании слов с их использованием. Заметим, что все структурные операторы имеют признак IMMEDIAT и могут использоваться только в описаниях других слов в режиме компиляции. В режиме интерпретации они не выполняют того, о чем было сказано выше. На самом деле эти слова только компилируют на вершину кодофайла несколько машинных команд, которые будут исполняться только при исполнении слова, внутри которого они скомпилированы. В табл. 8 приведен список слов, которые также используются только в режиме компиляции.

Для демонстрации работы операторов цикла приведем несколько примеров:

```
: ПРИМ2 BEGIN KEY DUP 20 = IF ELSE .
  REPEAT DROP ." ПРОБЕЛ" ;
```

Это слово циклически опрашивает клавиатуру и печатает ASCII коды нажатых клавиш, при нажатии клавиши с кодом 20H (пробел) цикл заканчивается. Следующее слово работает примерно так же, но цикл заканчивается при нажатии на клавишу управления с кодом < 20H.

```
: ПРИМ3 BEGIN KEY DUP 19 > IF EMIT
  REPEAT SPACE . ." Это код управления " ;
```

Следующее слово сразу после написания поздравит вас три раза:

```
DECIMAL : HELLO 22 10 6 +DO CR ."ПОЗДРАВЛЯЮ " REPEAT ; HELLO <ВК>
```

ПРОГРАММИРОВАНИЕ В МАШИННЫХ КОДАХ

До этого мы использовали только примитивы базового словаря, но подпрограммный шитый код позволяет вставлять в описания слов непосредственные машинные команды. Например, слово S. включает команды PUSH H и POP B и распечатывает содержимое арифметического стека без его изменения:

```
: S. DUP [ HEX E5 C, C1 C, ] 107A -2 +DO
  I @ . REPEAT ;
```

или то же самое:

Окончание. Начало см. в "Радио", 1995, № 7, 8.

Таблица 8

Имя	Состояние стека	Комментарии
[	--> -	Переключает систему в режим интерпретации
]	--> -	Переключает систему в режим компиляции
;	--> -	Компилирует команду RET и переключает систему в режим интерпретации
IF XXX	C-> -	Если C=TRUE, то исполняется XXX
ELSE YYY	--> -	Если C=FALSE, то исполняется YYY
THEN ZZZ	--> -	ZZZ исполняется всегда
BEGIN ZZZ	--> -	Начало цикла с условием. ZZZ исполняется всегда и оставляет в стеке логическое значение
IF XXX	C-> -	Если C=TRUE, то исполняется XXX, если C=FALSE, то выход из цикла
REREAT	--> -	Операторная скобка, парная к BEGIN IF, ограничивает операторы цикла ZZZ и XXX
+DO XXX N1 N2 N3	--> -	Снимает со стека конечное N1 и начальное N2 значения параметров цикла N3 и выполняет XXX(N2-N1)/N3+1 раза; при значении параметра =N1 выход из цикла
REPEAT	--> -	Операторная скобка, парная к +DO, ограничивающая операторы цикла XXX
I	--> N	Кладет в стек значение оператора цикла
J	--> N	Кладет в стек значение внешнего (объемлющего) цикла при вложенных циклах типа +DO +DO XXX REPEAT REPEAT
." "СООБЩЕНИЕ"	--> -	Распечатывает на терминале сообщение

Таблица 9

Адрес (HEX)	Назначение
0800H-0FFFH	Кодовайл базового словаря
1000H-103FH	Стек возвратов
1040H-1042H	JMP F803H вектор на подпрограмму ввода символов для KEY
1043H-1045H	JMP F809H вектор на подпрограмму вывода символа от EMIT
1046H-1047H	Хранит адрес LFA последнего слова в словаре
1048H-1049H	Хранит переменную HERE
104AH	Хранит основание текущей системы счисления
104BH	Хранит признак режима: 00 - интерпретация, FF - компиляция
104CH-104DH	Хранит адрес интерпретируемого символа (словом INTERPRET)
104EH-104FH	Хранит адрес, по которому QUERY запишет очередной введенный символ
1050H-1057H	Буфер для преобразования числа в строку словом . (точка)
1058H-107BH	Арифметический стек
107CH-107FH	Аварийные адреса
1080H-10FFH	Буфер для ввода команд с терминала
1100H	Начальный адрес области кодовайла слов пользователя

Таблица 10

Регистр	Назначение
PSW	Используется произвольно
DE	Используется произвольно
BC	Вершина арифметического стека
HL	Указатель арифметического стека
SP	Указатель стека возвратов
PC	Счетчик команд

```
: S. DUP [ HEX C1E5 , ] 107A -2
  +DO I @ . REPEAT ;
```

Программирование в машинных кодах является обоюдоострым свойством данной реализации, поэтому для того, чтобы не нанести непоправимого вреда ФОРТ-системе (особенно, если в программе происходят обращения по абсолютным адресам), приведем распределение адресного пространства, адреса рабочих ячеек интерпретатора и назначение регистров процессора (табл. 9 и 10).

Как видим, верхнее число, лежащее в стеке, на самом деле находится в регистровой паре BC, чем и объясняется высокая скорость работы ФОРТ-системы с вершиной стека (вспомните S.) Слово DUP делает копию содержимого регистровой пары BC примерно так:

```
MOV M,B
DCX H
MOV M,C
DCX H
```

а слово DROP снимает число с вершины стека так:

```
INX H
MOV C,M
INX H
MOV B,M
```

Регистровыми парами PSW и DE пользуются почти все слова базового словаря, поэтому сохранность информации в этих регистрах не гарантируется.

Для упражнения попробуйте написать в машинных кодах слова, повторяющие функции слов DUP и DROP. Следует избегать повторного использования имен слов, так как в данной версии их поиск производится от начала словаря, и всегда будет исполняться слово, написанное раньше. В версиях языка, где список связан от конца к началу, повторное описание слова делает недоступным более раннее описание.

В заключение приведем текст небольшой вирусной программы, которая, будучи один раз исполненной, перехватывает вектор вывода на дисплей и обрабатывает код 09H, как команду горизонтальной табуляции. Для ее написания потребовалось всего несколько минут. Читателю предлагается обратить внимание на действие слов, ограниченных словами [и], и расшифровать их смысл.

```
CREAT CUR 0 ,
: TAB DUP TAB [ 01 HERE 3 - C! ] 1044 !
  DUP 9 =
   IF 20
    BEGIN
     F809 EXECUT
     CUR @ 1 + DUP CUR ! 7 AND
    IF REPEAT DROP
   ELSE
    DUP 0D =
     IF 0 CUR !
     ELSE CUR @ 1 + CUR !
     THEN F809 EXECUT
  THEN ;
```

Как видим, написание вируса не такое уж и сложное дело. Но вряд ли стоит этим заниматься. Гораздо полезнее написать дельную программу. Вручая читателям свой труд, автор надеется, что именно этому они посвятят свое время и безграничную фантазию.

ПЕРЕМЕННЫЕ

В предыдущем примере мы использовали слово CUR, которое описано не через двоеточие, а словом CREAT. Что это за слово? Это — переменная. Переменными в языке ФОРТ называют слова, оставляющие на вершине стека адрес этой переменной (точнее, адрес PFA, в котором и хранится информация). Переменные могут быть одно- и двубайтными и образуются следующим образом:

однобайтные переменные:

CREAT <ИМЯ ПЕРЕМЕННОЙ> 1 ALLOT
или
CREAT <ИМЯ ПЕРЕМЕННОЙ> N C,

двубайтные переменные:

CREAT <ИМЯ ПЕРЕМЕННОЙ> 2 ALLOT
или
CREAT <ИМЯ ПЕРЕМЕННОЙ> N ,

В последних примерах переменной еще и присваивается начальное значение N. Доступ к переменным обеспечивают слова:

C! — запись байта из стека в переменную,
! — запись числа из стека в переменную,
C@ — чтение байта из переменной в стек,
@ — чтение числа из переменной в стек.

Попробуйте практически ознакомиться с переменными, так как личный опыт, к сожалению, ничем заменить нельзя.

МАССИВЫ

Массивы — более общий случай переменных и отличаются от последних только большим размером. Например, команда

```
CREAT ARR3 0 , 0 , 0 ,
```

или

```
CREAT ARR3 6 ALLOT
```

создает массив из трех слов (т. е. шесть байт). Особый интерес представляют массивы, заполненные литерами. Попробуем создать такой массив командой

```
CREAT ARR$ " МАССИВ, ЗАПОЛНЕННЫЙ ТЕКСТОМ"
```

Попробуйте распечатать этот массив командой

```
ARR$ TYPE CR .
```

Слово TYPE распечатает текст, записанный в массив, и вернет адрес стопбайта, записанного в конце массива словом " (кавычки). Так как в стоп-байте семь бит несут информацию о длине массива, то не рекомендуется записывать в массив более 127 символов (хотя этим правилом можно иногда и пренебречь). Эти же рекомендации относятся и к слову ." , которое пользуется услугами слова " .

КОНСТАНТЫ

Константы в данной версии языка проще всего описывать так же, как и другие ФОРТ-слова, через двоеточие. Например:

```
: TRUE -1 ;  : FALSE 0 ; и так далее.
```

Но иногда необходимо создать константу со значением, взятым из стека. В этом случае можно поступить двояко: либо создать константу и записать в нее новое значение, либо описать слово, которое будет само создавать константы. Одно из возможных решений — слово CONST:

```
: CONST CREAT HERE 3 - ! ;
```

Обращение к этому слову имеет вид

<ЗНАЧЕНИЕ> CONST <ИМЯ КОНСТАНТЫ>

Например:

```
-1 CONST TRUE    0 CONST FALSE
```

РАСШИРЕНИЕ СИСТЕМЫ ФОРТ

Конечно, расширение языка ФОРТ происходит и без вашего желания, хотя и не без вашего участия. Чтобы не "изобретать велосипед", старайтесь больше читать и использовать возможно большее число слов, уже достаточно устоявшихся или стандартизованных в других версиях. Это облегчит чтение ваших программ и позволит использовать программное обеспечение, написанное другими.

Начать легче всего с включения в ФОРТ-систему стандартных подпрограмм МОНИТОРа и программ или подпрограмм, написанных на АССЕМБЛЕРе. Для

примера опишем процедуры ввода кода нажатой клавиши и опроса состояния клавиатуры:

```
: ?KEY  0   F81B EXECUT [ 4F  C, ] ;
: ?TERM DUP F812 EXECUT [ 4F47 , ] ;
```

Подобным же образом, используя вставки в машинных кодах, можно описать обращения к любым программам и подпрограммам (например, так подключен редактор "МИКРОН"). Следует только позаботиться о стандартной передаче параметров через арифметический стек, т. е. регистровую пару BC и область памяти, адресуемую парой HL. Следует также позаботиться и о том, чтобы внешние программы не нарушали работы стеков. Исходные тексты программ, написанных на АССЕМБЛЕРе, можно включить в ФОРТ-систему и другим способом, используя тот факт, что область кодофайла пользователя и область трансляции АССЕМБЛЕРа "МИКРОН" расположены в одном и том же месте ОЗУ, начиная с адреса 1100H. Снабдите все ваши подпрограммы заголовком, аналогичным полям LFA и NFA других слов ФОРТ-системы.

На АССЕМБЛЕРе это выглядит так:

```
LFA: DW @PFA                ;поле LFA
NFA: DB 'ИМЯ'               ; ASCII- коды
                            ;имени
STB: DB STB-NFA+80H         ; стоп-байт
CFA: ...                    ; коды про-
                            ;граммы
...
...
RET
PFA: ...                    ; параметры
                            ;программы
...
@PFA:                       ; конец опи-
                            ;сания слова
```

Загрузите АССЕМБЛЕР "МИКРОН", откомпилируйте программу, а затем, загрузив ФОРТ, занесите в ячейки памяти по адресу 80CH адрес LFA последнего слова, а по адресу 80EH — слово, записанное по этому адресу, т. е. @PFA (см. выше). Запустив или перезапустив ФОРТ командой COLD, вы включите описанное слово в ФОРТ-систему. Подобным образом можно унифицировать и включить в систему любую программу, написанную на АССЕМБЛЕРе. Еще один способ позволяет написать программу, которая может использоваться и без ФОРТа. Введите команды:

```
HEX 7400 HERE — ALLOT
      2F3E , C23D , 7402 , 0021 ,
      3E21 , CD08 , F806 , 00FA ,
      7774 , C323 , 7409 ,
```

После этого по адресу 7400H будут записаны коды программы, которую можно запустить командой 7400 EXECUT или из МОНИТОРа командой G7400 <ВК>. Не вдаваясь в подробности, отметим, что эта программа позволяет вводить с кассеты тексты, записанные в формате редактора "МИКРОН", не только с начала, но и с любого места, даже с середины.

На этом описание работы с ФОРТом можно закончить. Остается только добавить, что после того, как написано и отлажено какое-либо слово в пультовом режиме, желательно командой 0 EXECUT выйти в редактор и "увековечить" находку в виде текста, который затем будет автоматически исполняться после входа в ФОРТ-систему или после "холодного" старта словом COLD.

Желаю успехов!

■