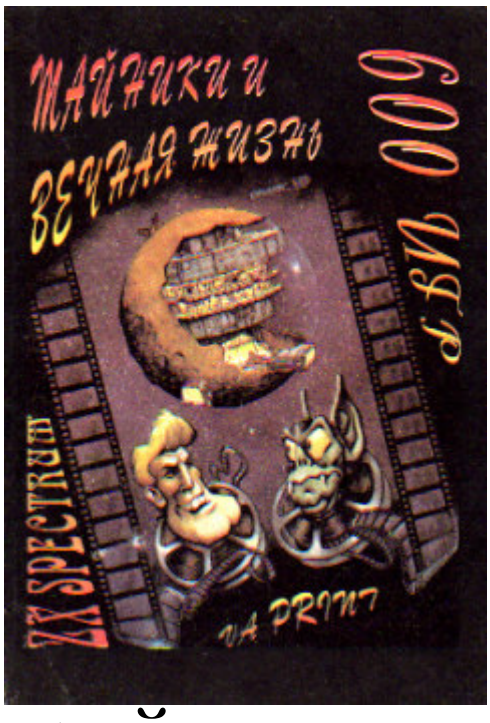

Т А Й Н И К И
ZX SPECTRUM
И
ВЕЧНАЯ ЖИЗНЬ

в 600 играх

МОСКВА
1993

Продукция фирмы VA PRINT

"VA PRINT" предлагает

Широкий выбор разнообразной литературы по Спектрум - совместимым компьютерам как для начинающих, так и для пользователей со стажем, бот краткий перечень того, что мы предлагаем сейчас:

1. Описание игровых программ:
   * 500 игр, часть 1
   * 500 игр, часть 2
   * 500 игр, часть 3
   * 500 игр, часть 4
2. Справочники пользователя прикладных и системных программ ZX Spectrum.
3. Ассемблер Z-80.
4. Основы программирования на языке BASIC, описание команд, примеры программ./
5. TR-DOS.
6. Описание редакторов, языков и программ для ZX Spectrum.
7. Лабиринты (описания секретов игровых экранов).
8. Графика, машинные коды, бессмертие игр и др.
9. Инструкция по пользованию компьютером 48К.
10. Инструкция по пользованию компьютером 128К.
11. Электронные устройства ZX Spectrum.'
12. Полный дизассемблер ПЗУ (1982г.).

Планируются к выходу 5 и 6 выпуски описаний игровых программ, а также другая литература. Мы поможем Вам войти в удивительный мир компьютера ZX Spectrum.

Ждем Ваших предложений о сотрудничестве.

По вопросам оптового приобретения продукции фирмы
"ВА ПРИНТ" обращайтесь по телефонам:
(095) * 354-94-39
495-67-49
491-24-06
или по адресу: Москва, 127322, а/я 50.

АННОТАЦИЯ

Мало кому нравятся программы, которые при первой же ошибке в ответе, либо при нажатии клавиши <BREAK> очищают всю память компьютера, не оставляя после себя никакого следа, либо "зависают", вынуждая нажимать <RESET>. Ситуация перестает быть забавной, когда мы имеем какую-нибудь программу, которую хотим приспособить для нетипового оборудования (например, JOYSTIC) или хотим заменить в программе все английские тексты на русские, а программу не удается остановить.

В этом описании будет последовательно дана информация о твоем компьютере, необходимая при такого рода работе, такая как карта памяти, способ записи в память отдельных строк BASIC, отдельные системныё переменные и т.п. Затем мы рассмотрим считывание программ и блоков данных "безопасным способом", т.е. так, чтобы они не стартовали автоматически, и можно было бы рассмотреть их содержимое, в конце мы займемся обезвреживанием предохранителей, написанных на внутреннем языке. Постараемся так же проиллюстрировать все это конкретными примерами на известных программах. Надеемся, что наши усилия не попадут зря, и Вы тоже научитесь добираться без препятствий к любой программе.

1. РАСПРЕДЕЛЕНИЕ ПАМЯТИ

В принципе память поделена на две особые части: ROM (ПЗУ) и RAM (ОЗУ).ROM занимает адреса 0…16383, а RAM адреса 16384…65536. Содержимым ROM мы сейчас заниматься не будем, но зато внимательно присмотримся к памяти RAM. Она поделена на блоки, выполняющие различные функции в системе BASIC.

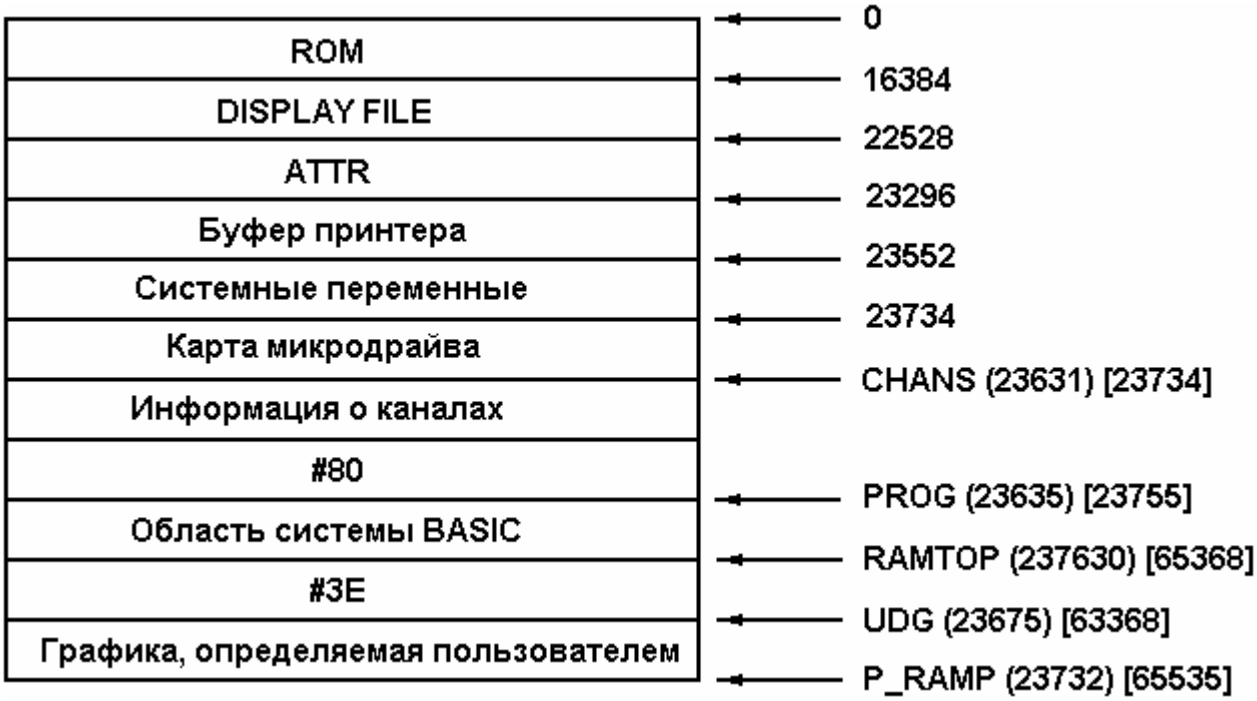

Рис. 1 Распределение памяти.

DISPLAY FILE

Область, в которой хранится информация о том, включены или погашены последовательные точки экрана. Это занимает 6144 байт.

ATTR

Эти 768 байт (с адреса 23558) определяют цвета последовательных полей экрана (8x8 точек).

БУФЕР ПРИНТЕРА

Эта область (от 23296 до 23551) используется лишь во время работы компьютера с принтером. Если не используются инструкции, касающиеся принтера (LLIST, LPRINT, COPY), то его содержимое не меняется и его можно использовать для других целей. Помни только, что использование какой-либо из указанных инструкций даже без включения принтера, произведет в этой области изменения.

СИСТЕМНЫЕ ПЕРЕМЕННЫЕ

Эти ячейки памяти используются системой для запоминания необходимых для ее безошибочной работы данных, таких как, например, адреса так называемых подвижных блоков памяти, информация о выполнении программы в БЕЙСИКе, т.е. какая строка выполняется, к которой должен осуществиться переход, появились ли какие-нибудь ошибки и т.п. В этой области находятся переменные, содержащие код по-

следней нажатой клавиши, продолжительность "BEEP" клавиатуры и еще много других. Подробнее мы займемся ими позднее.

Непосредственно за системными переменными, которые кончаются адресом 23733, начинаются так называемые подвижные области памяти. Это означает, что адреса их начал и концов (а также длина) могут изменяться в зависимости от того, подключены ли какие-нибудь внешние устройства, какова длина программы на БЕЙСИКе, сколько образуется переменных.

На рис. 1 и рис. 2 число за стрелкой означает адрес начала указываемого блока. Если адрес перемещается, то вместо числа записано имя системной переменной содержащей этот адрес, а в скобках - адрес этой переменной. В квадратных скобках значение этой переменной, которое она получает после включения компьютера (или выполнения RESET), но без подключения внешних устройств.

КАРТА МИКРОДРАЙВА

Если к твоему компьютеру подключен ZX интерфейс 1, то с адреса 23734 до адреса на 1 меньшего, чем содержимое переменной CHANS, находится карта микродрайва - область, используемая как буфер для трансляции данных, как набор добавочных системных переменных и т.п. Если интерфейс не подключен, то эта область попросту не существует - переменная CHANS содержит адрес 23734. Она определяет начало блока памяти, в котором содержатся данные о существующих каналах

ИНФОРМАЦИЯ О КАНАЛАХ

Необходима для безошибочного выполнения инструкций PRINT, LIST, INPUT и им подобных. В последней ячейке этой области находится число #80 (128), определяющее конец этого блока (это так называемый указатель конца).

ОБЛАСТЬ БЕЙСИКА

Эта область памяти содержит текст введенной программы на БЕЙСИКе. Адрес начала хранится в переменной PROG.Сразу за текстом программы (с адреса, указываемого переменной VARG) находится область, в которой интерпретатор размещает переменные, создаваемые программой. Она заканчивается указателем конца. Затем, начиная с адреса содержащегося в переменной E_LINE, находится область, используемая во время редактирования строки БЕЙСИКа, а так же ввода директив с клавиатуры (т.е. когда в нижней части мигает курсор и мы вводим инструкцию на БЕЙСИКе). В конце этой области находятся два байта с содержимым: 13 ("ENTER") и 128 (конец этой области). Сразу после, начиная с адреса, указываемого системной переменной WORKSP, находятся подобная область, но завершающаяся знаком "ENTER", за буфером INPUT (который автоматически удаляется после выполнения этой инструкции) находится "текущее рабочее пространство" - место памяти, используемое для самых различных целей. Туда, между прочим, загружаются заголовки считанных с лент программ. Туда считывается программа, размещаемая в памяти с помощью MERGE, прежде чем будет подсоединена к уже существующей программе. Эта область используется тогда, когда требуется на определенное время немного свободной памяти, но не только во временное пользование.

Однако, системе БЕЙСИКа принадлежит область памяти до ячейки, указываемой системной переменной RAMTOR. По этому адресу находится число #3E (62), которое устанавливает конец используемой БЕЙСИКОМ области. Двигаясь по памяти "вниз", мы сталкиваемся с одним не используемым байтом (этот байт как бы единое двухбайтовое число и является его младшим байтом), необходимым для верной инструкции RETURN. Если во время ее выполнения стек GOSUB будет уже пуст, то это число сыграет роль его продолжения. Но поскольку оно больше, чем 15872 (62*256), а строки БЕЙСИКа не имеют такой нумерации, то это будет восприниматься как ошибка и сигнализировать сообщением "RETURN без GOSUB". Сразу после этого байта (двигаясь "вниз" по памяти начинается) "стек GOSUB". В него заносятся номера программных строк из которых были выполнены инструкции перехода к подпрограмме, чтобы интерпретатор знал точно, куда должна вернутся инструкция "RETURN". Если интерпретатор не находился в подпрограмме, вызванной с помощью GOSUB, то этот стек просто не сущёствует - в нем не записано ни одного значения. Ниже находится "МАШИННЫЙ СТЕК", непосредственно используемый микропроцессором. Оба этих стека откладываются в сторону уменьшения адреса памяти.

Специальную роль играет системная переменная ERRSP. Процедура, обрабатывающая ошибку БЕЙСИКа (вызывается команда процессора RST 8), помещает значение этой переменной в регистр SP, после чего выполняется RET, считывая таким путем последний записанный в стек адрес (во время выполнения программы он равен обычно 4867). Под этим адресом находится процедура, выводящая сообщение об ошибке.

Рис. 2 Область памяти системы BASIC

ГРАФИКА, ОПРЕДЕЛЕННАЯ ПОЛЬЗОВАТЕЛЕМ

168 байт, зарезервированных для определения знаков UDG (их можно ликвидировать., например, с помощью CLEAR (65535) адрес последней ячейки памяти (равной 65535, если твой компьютер полностью исправен) запоминается в переменной P_RAMT. Если часть памяти повреждена, то эта переменная содержит адрес последней исправной ячейки.

2. СПОСОБ ЗАПИСИ ПРОГРАММ НА БЕЙСИКЕ

Помни, что практически каждая программа имеет хотя бы одно процедуру - загрузчик, написанную на БЕЙСИКе, а если не имеет, то вообще не защищена.

Начнем со способа записи на магнитную ленту программ - припомним, что видно и слышно во время считывания какой-либо программы в течение 1-2 секунд на экране видны широкие красно-синие полосы, а также слышен длительный звук. Это так называемый пилот, который позволяет компьютеру синхронизироваться с сигналом ленты, которую он будет считывать. Потом на момент появляются тонкие мерцающие желто-фиолетовые полоски, свидетельствующие о том, что компьютер считывает в память информацию. Ее 17 байт - это так называемый заголовок. Появляется надпись

"Bytes:", "Programm", "Charakter array:"

или

"Number array:",

потом после секундного перерыва, начинается второй пилот (более короткий), а после него считывается собственно программа.

Займёмся теперь заголовками, так как в них содержатся важнейшие данные о считываемых программах.

Заголовок содержит 17 байт. Пронумеруем их от 0 до 16 (смотри таблицу).

Нулевой байт означает тип блока. Он равен:

0 - если программа на БЕЙСИКе;

3 - если это блок машинного кода (записанный с помощью SAVE "…" CODE или SAVE "…" SCREEN$, которые означают тоже, что SAVE "…" CODE 16384, 6912). Если же этот заголовок предшествует набору, являющемуся массивом переменных БЕЙСИКа (записанный с помощью SAVE "…" DATA…), то равен:

1 - для числовых массивов;

2 – для символьных массивов.

Следующие 10 байт, это имя считываемого блока или текст, появляющийся после загрузки заголовка за надписью:
"Programm:", "Bytes:" и т.д.

Байты 11,12 содержат двухбайтовое число (первый байт младший) определяет длину блока, к которому относится заголовок. В зависимости от считываемого блока, байты с 13 по 16 интерпретирую разному. Начнем с заголовков программ, написанных на БЕЙСИКе.

Байты 13 и 14 содержат номер строки старта программы, если она была записана с помощью SAVE"…" LINE NR. Если программа была записана без операции LINE и после считывания не стартует автоматически, то значение этого числа больше 32767, Одним из способов нейтрализации защиты самостартующих программ на БЕЙСИКе является замена этих 2-х байтов на число больше 32767.

Байты 15 и 16 содержат число, определяющее длину самой программы на БЕЙСИКе т.к. SAVE "…" или SAVE "…" LINE записывает программу вместе со всеми переменными, т.е. содержимое памяти от байта указанного системной переменкой PROG, до байта, определяемого переменной E_LINE. Если от всей длины блока (байты 11 и 12) отнимаем это число, то узнаем, сколько байтов в этом блоке занимают переменные БЕЙСИКа. Это всё, если речь идёт о заголовках программ на БЕЙСИКе.

В заголовках машинного кода ("BYTES") байты 15 и 16 не используются, зато 13 и 14 составляют двухбайтовое число, определяющее по какому адресу надо считать следующий за заголовком блок.

В заголовках массивов из этих 4-х байтов используется только 14 байт, который представляет имя считываемого массива. Он записан также как имена всех переменных БЕЙСИКа (в области от VARS до E_LINE), т. е. три самых старших бита означают тип переменной (здесь это числовой или знаковый массив), а 5 младших битов - имя переменной

<table>
<tr><td>0</td><td>Тип блока:</td><td colspan="2">0 – программа</td></tr>
<tr><td>1
2
3</td><td rowspan="9"></td><td colspan="2">1 – NUMBER ARRAY (числовой массив)
2 – CHARACTER ARRAY (символьный массив)
3 – BYTES (данные)</td></tr>
<tr><td>4
5
6
7
8
9
10</td><td colspan="2">ИМЯ</td></tr>
<tr><td>11
12</td><td>Длина</td><td colspan="2">Сколько байт необходимо будет считать с ленты</td></tr>
<tr><td>13
14</td><td>Старт</td><td colspan="2">Bytes: Адрес загрузки блока
Programm: Номер строки старта программы
Array: Байт 14 – имя переменной
Байт 15 – не имеет значение</td></tr>
<tr><td>15
16</td><td colspan="2">Программа</td><td>PROGRAMM: длина самого Бейсика
(без переменных)
BYTES, ARRAY: не используются</td></tr>
</table>

Таблица значений байтов заголовка

На листинге 1 представлена программа, реализующая процедуру, делающую возможным чтение заголовков программы. Внимательно проверьте количество пробелов в строках с инструкциями DATA, запустите её RUN и подождите минуту. На экране появится информация об адресах начала и конца процедуры, а также о её длине, которая должна составлять 284 байт. Если не так, то проверьте, все ли строки программы введены без ошибок. Если вес правильно, то на твоей ленте окажется процедура "ЧТЕН.", благодаря которой ты сможешь прочесть каждый заголовок. После записи её на ленту, ее можно удалить из памяти. Процедура "ЧТЕН." будет работать правильно, независимо от того, по какому адресу она будет загружена. Можно считать её:

```
LOAD "CZVTACZ" CODE АДРЕС
```

а затем запустить (даже многократно) с помощью:

```
RUNDOMIZE USR АДРЕС
```

После запуска процедура считывает по адресу 23296 (буфер принтера) первый встреченный заголовок. Если из-за подключенных внешних устройств этот адрес не устраивает тебя, можно сменить его, заменяя в строке 200 листинга 1 число #005B, шестнадцатеричным адресом, по которому ты хотел бы считать заголовок (две первые цифры являются младшим байтом этого адреса). Чтобы после этой замены избежать

проверки контрольной суммы в этой строке, в конце текста, взятого в кавычки вместо пробела и 4-х цифр контрольной суммы нужно поместить литеру "S" с четырьмя ведущими пробелами

После считывания заголовка процедура считывает заключенную в нем информацию и возвращается в БЕЙСИК, но считанного заголовка не уничтожает, следовательно, если желаешь посмотреть его дополнительно, то можешь сделать это используя функцию PEEK.

Однако чтения заголовка мало, чтобы взломать блоки, записанные на ленте. Необходимо знать, что еще надо сделать с этими блоками, чтобы разместить их в памяти, не позволяя при этом начать работу.

В случае блоков типа "BYTES" достаточно, обычно, загрузить их под принудительный адрес повыше RAMTOR (т.е. повыше ячейки памяти, указываемой переменной RAMTOR), например:

```
CLEAR 29999: LOAD "" CODE 30000
```

Этот метод работает, если считываемый блок не очень длинный (может иметь максимально до 40К). Более длинные блоки могут просто не разместиться в памяти - тогда необходимо их разделить на несколько коротких частей. Скоро мы узнаем, как это сделать. Также и в случае массивов их загрузка не вызывает затруднений - достаточно применить обычную в таких ситуациях инструкцию

```
LOAD "" DATA…
```

Хуже выглядит считывание программ БЕЙСИКе. Они обычно записываются с помощью SAVE "…" LINE "…", а в самом начале строка с которой должны выполняться инструкции, закрывающие программу до останова. Простейшим решением является загрузка программы не с помощью LOAD"", а с помощью MERGE "", но этот способ не всегда дает результат. Из этой безнадёжной ситуации есть два выхода: подменить заголовок программы или использовать предоставленную ниже программу "LOAD/MERGE". Первый способ основан на замене записанного на ленте заголовка программы на такой же, но не вызывающей самозапуска программы. Можно с этой целью использовать программу "COPY-COPY", считать, заменить её параметр СТАРТ на число больше 32767 (т.е. выполнить, например, LET I=32678, если корректируемый таким образом заголовок был считан как первый набор). Модифицированный таким образом заголовок записываем где-нибудь на ленте. Убираем из памяти программу "COPY-COPY" и вводим LOAD. Считываем только что сделанный заголовок и сразу после его окончания останавливаем ленту. Теперь в магнитофон вставляем кассету с программой - так, чтобы считать только текст программы без ее заголовка.

Другой способ выгоднее. Вводим в память (с клавиатуры или ленты) программу "LOAD/MERGE", размещённую на листинге 2. После запуска она начинает ждать первую программу на БЕЙСИКе, которая находится на ленте, считывает совершенно также как инструкция LOAD, но после загрузки не позволяет программе запуститься, выводит сообщение "0 OK" с информацией с какой строки считанная программа должна была стартовать.

2.1. ЛИСТИНГИ ПРОГРАММ

Листинг 1.

```
10 CLEAR 59999: LET POCZ=60000
20 LET ADR=POCZ
30 RESTORE: READ A,B,C,D,E,F
40 DATA 10,11,12,13,14,15
50 LET NR=200; RESTORE NR
60 LET S=0: READ A$: IF A$="." THEN GOTO 130
70 FOR N=1 TO LEN A$-5 STEP 2
80 LET W=16*VAL A$(N)+A$(N+1)
90 POKE ADR,W: LET ADR=ADR+1: LET S=S+W
100 NEXT N
110 IF VAL A$(N)<>S THEN PRINT "OSN.W STROK";NR:STOP
120 LET NR=NR+1: GO TO 60
130 PRINT "VSE DANNYE HOROSHO"''"NASHALO:"; POCZ'"CONEC_:_";ADR-
1'"DLINA_:_";ADR-POCZ
140 SAVE "CZYTACZ" CODE POCZ,ADR-POCZ
150 REM
200 DATA "21920009E5DD21005BDDE51111  1246"
210 DATA "00AF37CD5605DDE130F23E02CD  1531"
220 DATA "011611C009DD7E00CD0A0CDDE5  1265"
230 DATA "D113010A00CD3C202A7B5CD1E5  1231"
240 DATA "D5ED537B5C010900CD3C20DD46  1346"
250 DATA "0CDD4EDBCD2B2DCDE32DE1DD7E  1872"
260 DATA "00DD460EDD4E0DC5A7202BEB01  1292"
```

```
270 DATA "1900CD3C20D5DD4610DD4E0FCD  1361"
280 DATA "2B2DCDE32DD1C178E6C0206EC5  1848"
290 DATA "011300CD3C20C1CD2B2DCDE32D  1280"
300 DATA "185EFE03203601700009EB0111  836"
310 DATA "0018E60D449C75676F9D9B2073  1281"
320 DATA "616D65676F2070726F6772616D  1313"
330 DATA "75200D4175747F737461727420  1161"
340 DATA "2D206C696E69120D2EE1BB0181  1239"
350 DATA "0009EB010900CD3C20C13E1FA0  997"
360 DATA "F660D7DD7E003D28033E24D73E  1383"
370 DATA "28D73E29D7E1227B5C3E0DD7C9  1538"
380 DATA "04081C2020201C000010181030  268"
390 DATA "100C000810384038047800OD41  430"
400 DATA "64726573209C61646F77616E69  1357"
410 DATA "61200D5461626C69636120      862"
420 DATA "."
```

Листинг 2.

```
1 REM LOAD/MERGE TS&RD 1987
2 FOR N=60000 TO 60025: READ A: POKE N,A: NEXT N
3 RANDOMIZE USR 60000
4 DATA 1,34,0,247,213,221,225,253,54,58,1,221,54,1,225,205,29,7,42,66,
92,34,69,92,207,255
```

Листинг 3 (язык ассемблера).

```
10;           LOAD/MERGE
20;
30            ORG         60000
40            LD          BC,34
50            RST         48
60            PUSH        DE
70            POP         IX
80            LD          (IY+58),1
90            LD          (IX+1),255
100           CALL        1821
110           LD          HL,(23618)
120           LD          (23621),HL
130           RST         8
140           DEFB        255
```

3. ЗАЩИТА ПРОГРАММ НА БЕЙСИКЕ

Прочитанная программа, как правило, не должна выглядеть нормально. Например, в программе есть строка с номером 0 или строки упорядочены по убыванию номеров. Нельзя вызвать EDIT ни для какой строки, видно подозрительную инструкцию RANDOMIZE USR 0 или просто ничего не видно, т.к. программа не позволяет листать себя. Если в программе, куда ты вламываешься, ты увидел что-то необычное, то лучше просматривать ее другим способом, несколько отличающимся от обычного - не с помощью LIST, а непосредственно, используя функцию PEEK, однако, сначала мы должны узнать каким образом размещен в памяти текст программы на БЕЙСИКе. Программа складывается из последовательных строк и так и хранится в памяти.

Отдельная строчка программы выглядит так:

MSB LSB	LSB MSB		
2 байта	2 байта		#0D
Номер строки	Длина текста+enter		ENTER

Рис 1.

Она занимает не менее 5 (а точнее 6, т.к. текст пустым быть не может) байт. Два первых означают её номер. Но он записан наоборот, от всех обычных 2-байтовах чисел, хранимых в памяти (MSB - старший байт, LST - младший байт).

Следующие 2 байта - это длина строки, т.е. число символов содержащихся в сроке вместе с завершающим ее знаком "ENTER" (#0D). За этими байтами находится текст строки, заканчиваемый "ENTER".

Если введем, к примеру, такую строку:

```
10 REM BASIC
```
и запишем её, нажимая клавишу "ENTER", то она будет записана в память как последовательность байт:

0	10	7	0	243	66	65	83	73	67	13
10		7		REM	B	A	S	I	C	ENTER
Номер		Длина		Текст						

Параметр "Длина строки" касается только её текста, следовательно, хотя строка занимает в памяти 11 байт, этот параметр указывает только на 7 байт: 6 байт текста и 1 байт "ENTER", заканчивающий строку.

Тебе, наверное, уже понятно, на чём основано часто применяемый трюк со строкой, имеющий нулевой номер. Достаточно впервые два байта занести число 0 (с помощью POKE), чтобы эта строка стала нулевой строкой. Если мы хотим изменить номер первой строки в программе (а к интерфейсу никакая быстрая память не подключена, т.к. в этом случае измениться адрес начала БЕЙСИКа), то достаточно написать:

```
POKE 23775,X: POKE 23765,Y
```
и строка получит номер 256*X+Y. Не зависимо от его значения строка останется в памяти там где была. Если введём:
```
10 REM НОМЕР СТРОКИ 10
20 REM НОМЕР СТРОКИ 20
POKE 23775,0: POKE 23765,30
```
то первой строке в программе будет присвоен номер 30, но она останется в памяти как первая, а на экране мы получим след:
```
30 REM НОМЕР СТРОКИ 10
20 REM НОМЕР СТРОКИ 20
```
следовательно, чтобы начать разблокировать программу, в которой имеются нулевые строки или строки, упорядоченные по убыванию номеров, следует найти адрес каждой строки и в их поле "НОМЕР СТРОКИ" последовательно размещать к примеру: 10,20… В памяти строки располагаются одна за другой, следовательно с обнаружением их начал ты не будешь иметь трудностей. Если X указывает адрес какой-нибудь программной строки, то следующий адрес равен:

X+PEEK(X+2)+256*PEEK(X+3)+4

т.е. к адресу сроки добавляется длина её текста, увеличенная на 4 байта, т.к. именно столько занимают параметры "НОМЕР СТРОКИ" и "ДЛИНА СТРОКИ".

Такой способ нахождения не действует, к сожалению, когда применяется другой способ защиты – фальшивая длина строки. Он основан на том, что в поле "ДЛИНА СТРОКИ" вместо настоящего значения даётся очень большое число – порядка 43-65 тысяч. Этот способ применяется очень часто, т.к. обычно делается невозможным считывание программы с помощью MERGE (т.е. так, чтобы не было самостарта). Делается это потому, что MERGE загружает программу в область WORKSPACE, а затем интерпретатор анализирует всю считанную программу строка за строкой: последовательно проверяет номер каждой из них, а затем размещает её в соответствующем месте области, предназначенной для текста программы на БЕЙСИКе. Для этой строки необходимо там подготовить соответствующее количество свободных байт, "РАЗДВИГАЯ" уже существующий текст программы. Если в поле "ДЛИНА СТРОКИ" стоит очень большое число, то интерпретатор старается выделить именно столько байт пространства в область текстов программы, что завершиться сообщением "OUT OF MEMORY" или просто зависание системы. Чтобы прочесть такую программу не вызывая её самозапуск, следует применить соответствующую отмычку, например, такую как представленная в разделе 2 программа "LOAD/MERGE". Дополнительным эффектом от применения фальшивой длины является невозможность в корректировке такой строки путём занесения в её поле. Это требует слишком большого количества памяти, следовательно, кончается только предупредительным звонком.

Если программа защищена таким способом, то адреса очередных строк приходится искать вручную или догадываться, где находятся, помня, что каждая строка заканчивается знаком ENTER (но не каждое число 13 означает ENTER). Чтобы просмотреть программу на БЕЙСИКе, введи такую строку:
```
FOR N=23755 TO PEEK 23627+256*PEEK 23628:
PRINT N; PEEK N; CHR$ PEEK N AND PEEK 31:
NEXT N
```
Она последовательно высветит: адрес, содержимое байта с этим адресом, а также символ, имеющий этот код, если это только не управляющий символ (т.е. с кодом 0…31).

После смены нумерации строк и изменения их длины, следующим способом защиты программ являются управляющие символы, не позволяющие последовательно просматривать программу, хотя и не только это.

После смены нумерации строк и изменения их длины, следующим способом защиты программ являются управляющие символы, не позволяющие последовательно просматривать программу, хотя и не только это.

Вернёмся к первому примеру (строка "`10 REM BASIC`"). Текст строки складывается из 7 символов – ключевого слова REM (все ключевые слова – инструкции и функции, а также знаки <=, >= и <> имеют однобайтовые коды из диапазона 165…255. Если ты не знаешь какие, то введи:

```
FOR N=165 TO 255: PRINT N, CHR$ N: NEXT N
```

и ознакомиться), а также пяти литер и символа "ENTER". Так бывает всегда, если в строке находится инструкция REM – все знаки, введённые с клавиатуры после этой инструкции, будут размещены в памяти без малейших изменений. Иначе выглядит ситуация, когда в строке находится другие инструкции, требующие числовых параметров (а обычно так и бывает). Введём, к примеру, строку:

```
10 PLOT 10,9
```

и посмотрим, каким образом она запишется в память (лучше вводя переменную выше строки H=23755 то …). Выглядит она так, как показана на рисунке 3.

0	10	18	0	246	49	48	14	0	0	10	0	0	44	57	14	0	0	9	0	0	13
10		18		PLOT	1	0	10						,	9	9						
Номер		Длина					Номер								Номер						ENTER

Рис. 3.

Как видно, текст был модифицирован – после последней цифры каждого числа, выступающего в тексте строки как параметр, интерпретатор сделал 6 байтов пространства и поместил там символ с кодом 14, а также 5 байтов, в которых записано значение этого числа, но способом, понятным интерпретатору. Это убыстряет, в определённой мере, выполнение программы на БЕЙСИКе, т.к. во время выполнения не должен каждый раз переводить числа из алфавитно-цифрового представления (последовательности цифр) на пяти байтовое представление, пригодное для вычислений. Готовое значение выбирается из памяти из-за управляющего символа CHR$ 14. Эта странная запись дает большие возможности в БЕЙСИКе в деле затруднения доступа к программам. Во многих программах загрузчиков (LOADER) присутствует такая строка:

```
0 RANDOMIZE USR 0: REM…
```

На первый взгляд эта программа после запуска должна затереть всю память, но так, однако, не происходит. После более внимательного просмотра (с помощью PEEK - строка FOR N=23755 …) оказывается, после USR 0 и символа CHR$ 14 совсем нет пяти нулей (именно так выглядит в пятибайтовой записи число 0). Если это целое число, то в пятибайтовой записи оно выглядит:

Байт 1 - 0
Байт2 - 0 (для +) и255 (для –)
Байт 3 и 4 - последовательно младший и старший (или дополнение до 2 для –)
Байт 5 -0

К примеру, 0, 0, 218, 92, 0 это равнозначно числу 23770. Функция USR не осуществляется, и тогда идет переход по адресу 0, а именно по 23770, а это как раз адрес бейта, находящегося сразу после инструкции REM в нашем примере. Там обычно находится программа - загрузчик, написанная на машинном языке.

Следующим управляющим символом, часто применяемым для защиты, является CHR$ 8 – "BACKSPASE" или пробел назад. Высвечивание этого символа вызывает сдвиг позиции ввода на одну позицию назад влево, следовательно, с его помощью можно закрывать некоторые позиции на листинге, печатая на их месте другой текст, если, например, в памяти находятся знаки:

```
LET A=USR 0: REM <<<<<<<<<<<<<<<<
LOAD "": …
```

(< обозначает CHR$ 8), то инструкция LOAD "" и последующий текст закроет инструкцию LET A=USR 0:. Хотя на листинге теперь видна только инструкция LOAD "":, но дальнейшая часть программы загружается не ею, а машинной программой, запускаемой инструкцией USR 0 (что очевидно, не должно обозначать переход к адресу 0). Такая защита, например, применяется в загрузчике программы BETA BASIC 1.0.

4. "ЗАЩИТНЫЕ" УПРАВЛЯЮЩИЕ СИМВОЛЫ

В этом разделе мы познакомимся с остальными управляющей символами. Три из них касаются изменения места вывода, шесть - смены атрибутов.

4.1. CHR# 6 – "COMMA CONTROL"

Этот символ (управляющая запятая) действует также как запятая, отделяющая тексты в инструкции PRINT, т.е. выводит столько пробелов (но всегда не менее одного), чтобы оказаться в колонке 0 или 16:

```
PRINT "1", "2"
```

а также

```
PRINT "1"+CHR$ 6+"2"
```

имеют идентичное значение.

4.2. CHR$ 22 – "AT CTRL"

Этот символ (AT управляющий) позволяет переносить позиции вывода в любое место экрана так же, как AT в инструкции PRINT. После этого знака должны появиться два байта, определяющие номер строки и номер колонки, в которой должен быть расположен следующий знак: PRINT AT 10,7,"!" равнозначно PRINT CHR$ 22; CHR$ 10; CHR$7; "!".

Чтобы убедиться, как с помощью этого символа делать программы невидимыми (листинги программ), введи, например:

```
10 RANDOMIZE USR 30000: REM НИЧЕГО НЕ ВИДНО!
```

После инструкции REM введи три пробела, а после восклицательного знака две управляющие запятые. Их можно получить непосредственно с клавиатуры, нажимая последовательно клавиши: EXTENT (или оба SHIFTa вместо) чтобы получить курсор "E", а затем клавишу "6" (курсор сменить цвет на желтый) и DELETE курсор перескочит к ближайшей половине экрана. После ввода этой строки заменим три этих пробела на знак AT 0,0.

С помощью: POKE 23774,22: POKE 23775,0: POKE 23776,0 попробуем теперь посмотреть программу На экране появится текст всей строки - начальная часть её закрыта надписью, находящейся после инструкции REM и знака AT CTRL. Такие же трудности возникают, если эту строчку перенести в зону редактирования (клавиша EDIT).

Координаты, заданные в символе AT CTRL должны находиться в поле экрана, т.е. номер строки не может быть больше 21, номер колонки не больше 31. Задание больших значений в случае PRINT или LIST вызывает сообщение "OUT OF SCREEN" и отмену дальнейшего вывода, если же листинг получен нажатием "ENTER" (автоматический листинг) - также наступит прекращение дальнейшего вывода, а кроме того, в нижней части экрана появится мигающий знак вопроса - сигнал ошибки. Следовательно, это практически способ защиты от просмотра текста для каждой программы.

4.3. CHR$ 23 – "TAB CTRL"

После этого символа (горизонтальная табуляция) следует два байта, определяющие номер колонки, в которую переносится позиция вывода. Они трактуются как одно двухбайтовое число (первый байт - младший). Поскольку колонок только 32, то число берется по модулю 32, т.е. старший байт и три старших бита младшего байта игнорируются. Второй существенной стороной является то, что TAB переносит позицию вывода с помощью вывода пробелов – также как и управляющая запятая, и, следовательно, может быть использован для закрывания уже находящихся на экране текстов.

5. СИМВОЛЫ СМЕНЫ АТРИБУТОВ

Эту группу управляющих символов составляют символы, меняющие атрибуты:

CHR$ 16 - INK CTRL
CHR$ 17 - PAPER CTRL
CHR$ 18 - FLASH CTRL
CHR$ 19 - BRIGHT CTRL
CHR$ 20 - INVERSE CTRL
CHR$ 21 - OVER CTRL

После каждого из этих символов обязателен один байт, уточняющий о каком атрибуте идет речь, после символов INK и PAPER это могут быть числа 0…9, после FLASH и BRIGHT 0, 1, 8 после INVERSE и OVER: 0 и 1. Задание других значений вызывает сообщение: "INVALID COLOUR" и, естественно, прерывание просмотра программы.

Высвечивая программу, защищенную управляющими символами цветов, принимаем для себя следующую последовательность действий. Например, если, просматривая текст программы мы встречаем код знака PAPER CTRL, то заносим в его второй байт значение 0, если INK CTRL - значение 7, в остальные управляющие цветами символы - значение 0. Кроме того, удаляем все знаки BACK SPACE < CHR$ 8) путем их замены пробелами (CHR$ 32) .Также ликвидируем знаки AT CTRL - заменяем с помощью POKE три байта символа на пробелы. После такой корректуры программу можно уже листать без всяких сложностей.

Бели ты хочешь взломать программу - загрузчик, то его не обязательно и в принципе не надо очищать - важно узнать, что эта программа делает, каким образом загружается в память и запускает следую-

щие блоки, а не стараться, чтобы она делала "ЛАДНО" и была написана чисто и прозрачно. Это тем более важно, пока не узнаешь точно программу, лучше не делать в ней никаких изменений - одна ловушка может проверяться другой, поэтому наилучший способ раскалывания программы это анализ ее работы шаг за шагом, считывая последовательные байты памяти:

```
    0 BORDER 0: PAPER 0: INK 0: CLS 0: PRINT #0, "LOADING";: FOR N=0 TO 20
STEP 4: BEEP 2,N : NEXT N: LOAD "" CODE: PRINT AT 19,0;: LOAD "" CODE: PRINT
AT 19,0;: LOAD "" CODE: PRINT AT 19,0;: LOAD "" CODE: PRINT AT 19,0; RANDOM-
IZE USR 24064
```

Помни о правильной интерпретации очередных байтов: сначала два байта номера строки, потом два байта содержания ее длины (которая может быть фальшивой), затем текст: инструкция БЕЙСИКа, потом ее параметры. За каждым числом записывается CHR$ 14 и пять байтов, содержащих значение этого числа за параметрами - двоеточие и следующая инструкция или ENTER и новая строка программы.

Это было бы все, если речь идет об управляющих символах, но есть еще одна вещь, которую требуется объяснить, что бы ты не имел неприятностей с чтением БЕЙСИКа. Речь идет об инструкции DEF FN. Введи, а затем внимательно просмотри такую строку:

```
10 DEF FN A(A,B$,C)=A+C
```

Кажется, что она должна занять в памяти 19 байтов (номер строки, ее длина, ENTER, а также 14 введенных символов), но это не так. Интерпретатор после каждого параметра функции поместил знак CHR$ 14 и дорезервировал зачем-то следом еще пять байт. Введи:

```
PRINT FN A(1,"125",2)
```

и снова посмотри содержимое памяти с адреса 23755. После первого параметра в определении функции далее находится CHR$ 14, но после него последовательно расположились: 0, 0, 1, 0, 0, что в памяти байтов записи обозначает 1. Также после третьего параметра функции находится CHR$ 14 и байты, содержащие число 2. После параметра B$ также находится значение использованного параметра: CHR$ 14 и пять байтов, которые последовательно содержат: первый для нас не имеет значения, второй и третий содержат адрес, по которому находится цепочка символов "125" (вызов функции был осуществлен в директивном режиме, следовательно этот адрес относится к области редактирования строки БЕЙСИКа), а байты 4-5 это длина цепочки - в нашем случае она составляет три знака.

Помни об этом, читая БЕЙСИК с помощью PEEK, а не LIST, иногда случается, что именно в этих байтах, зарезервированных для действительных аргументов функции скрыты проверки, определяющие работоспособность программы или даже машинная программа, загружающая последующие блоки (например, BETA BASIC 1.0).

В конце немного о программах-загрузчиках. Их задачей является считывание и запись всех блоков, составляющих программу. Обычно они это делают способом, максимально затрудняющим понимание их работы - так, чтобы запуск программы другим способом, а не через загрузчик (или на практике взлом программы) был не возможен. Посмотрим на загрузчики применяемые в большинстве продукции фирмы ULTIMATE (например, ATIC ATAC, KNIGHTLORE, PENTAGRAM, NIGHT SHADE и т.д.). Выглядят они так:

```
FOR N=23755 TO PEEK 23627+256*PEEK 23628: PRINT N;" "; PEEK N; CHR$
PEEK N AND PEEK>31: NEXT N
```

После такого загрузчика на ленте находятся пять следующих блоков: экран, закодированный блок программы, а за ним три коротеньких блока, защищающих программу: однобайтовый (код инструкции JP (HL), из нескольких байтов (это процедура, которая декодирующею программу) и последний двухбайтовый загружаемый по адресу 23627, или в переменную FRAMES. Значение этой переменной увеличивается на 1 через каждые 1/50 секунды. В машинной программе, запущенной с помощью RANDOMIZE USR 24064, её значение проверяется и если отличается от того каким должно быть (что означает, что где-то после загрузки программа была остановлена на какое-то время), наступает обнуление памяти компьютера. Взлом программ этого типа весьма прост. Достаточно загрузить все блоки за исключением последнего, а после просмотра программы или же приведения в ней определенных изменений (например, вписание POKE) достаточно лишь ввести: LOAD "" CODE: RANDOMIZE USR 24064 (но обязательно в одной строке, разделяя инструкции двоеточием), чтобы запустить игру.

Отдельной работой является декодирование программы, или запуск процедур, декодирующих так, чтобы она вернулась в БЕЙСИК. У читателя, знающего ассемблер, это не должно вызывать затруднений, однако с точки зрения на распространенность этого типа защиты, особенно в пользовательских программах (например ART STUDIO или THE LAST WORD), мы еще вернемся к этой теме.

6. ЗАЩИТА ЗАГРУЗЧИКОВ

Все игры имеют хорошо защищённую программу, написанную на БЕЙСИКе, т.к. это важнейший (с точки действенности защиты) элемент всей программы. Ведь с БЕЙСИКа начинается считывание всей программы. Если бейсиковский загрузчик защищен слабо, то взлом всей программы значительно облегчен. Примером этого являются загрузчики фирмы ULTIMATE, представленные в предыдущем разделе. Одним из способов снятия защиты загрузчиком является считывание их с помощью программы "LOAD/MERGE'" (смотри раздел 2). Однако иногда лучше поместить этот загрузчик не в память, предназначенную для БЕЙСИКа, а выше RAMTOR, чтобы можно было спокойно рассматривать его не опасаясь возможности случайных изменений в нем.

Для этого имеется очень действенный метод - считывание программы на БЕЙСИКе как блока машинного кода под удобный для нас адрес. Чтобы этого добиться нужно, знать длину программы, которую мы хотим считывать (можешь использовать процедуру "CZYTACZ" из раздела 2), хотя можно обойтись и без длины. Кроме того, требуется немного свободного места на магнитной ленте. Этот способ основывается на обмане инструкции LOAD путем подмены заголовков.

На свободной ленте записывается заголовок блока кода с помощью SAVE "BAS" CODE 30000,750. если мы знаем, что длина программы составляет 750 байт. Если мы ее не знаем - подаем соответственно большее значение даже порядка нескольких десятков килобайт, хотя программа может иметь всего лишь 100 байт длины. На ленте записываем только сам заголовок, прерывая запись, после этого нажимаем клавишу "BREAK". Теперь устанавливаем ленту точно перед записанным заголовком, а ленту с программой - сразу после заголовка программы, но перед требуемым блоком данных, вводим:

```
CLEAR 29999: LOAD "" CODE
```

или

```
CLEAR 29999: LOAD "" CODE 30000
```

и считываем заголовок.

Сразу после его считывания мы нажимаем СТОП в магнитофоне, заменяем кассету и вновь нажимаем ПУСК (все это время компьютер ждал блок данных). Теперь считывается программа на БЕЙСИКе, но под адрес 30000 - выше RAMTOR. Если мы задали в заголовке завышенную длину программы, то считывание кончается сообщением "Tape loading error", но это не мешает - теперь уже любым способом можно смотреть считанную программу на БЕЙСИКе. Кроме этого метода существует и второй, но для того чтобы им пользоваться, обязательно знание АССЕМБЛЕРа (а пользоваться стоит, т. к. он дает большие возможности в раскрытии программ, а знание ее позволяет расшифровывать работу загрузчика.

Очень часто (особенно в новейших программах) встречаются блоки программ, записанные и считываемые в память компьютера без заголовка. Это достаточно оригинальная мера защиты обычно отпугивает начинающих, но раскрытие такой программы не является вовсе трудным. Вся тайна основана на хранящихся в ROM процедур, используемых с помощью инструкций LOAD, SAVE, VERIFY, MERGE.

Под адресом #0556 (1366) находится процедура LOAD-BYTES, считывающая с магнитофона блок данных, или пилота и следующую за ним информацию. При этом не важно, будет ли это заголовок, или требуемый блок данных, которые следует поместить где-то в памяти.

Начнем же сначала. Каждая защищенная программа начинается с загрузчика написанного на БЕЙСИКе. Программа, применяющая загрузку без заголовков (с помощью процедуры 1366 или другой), должна быть написана на машинном коде, как каждая процедура, обслуживающая магнитофон. Чаще всего это программа помещается в одной из строк БЕЙСИКа. Например, после инструкции REM, или в области переменных БЕЙСИКа. После считывания, загрузчик на БЕЙСИКе запускается и выполняет инструкцию RANDOMIZE USR …, инициируя тем самым работу машинной программы. Процедура LOAD-BYTES требует соответствующих входных параметров. Они передаются в соответствующих регистрах микропроцессора. Так в регистре IX задаем адрес, под который хотим прочесть блок данных, а в паре DE - длину этого блока. В буфер помещаем 0, если хотим считать заголовок и 255 если это блок данных. Кроме того, указатель переноса устанавливаем (CARRY) в 1, т.к. иначе процедура 1366 вместо LOAD выполнила бы функцию VERIFY. Ниже дан пример процедуры, загружающейся с ленты без заголовка:

```
LD      IX,16384    ; Адрес считывания
LD      DE,6912     ; Длина блока
LD      A,255       ; Блок данных
SCF                 ; Установка CARRY
CALL    1366        ; Вызов LOAD-BYTES
RET                 ; Выход из подпрограммы
```

Процедура 1366 в случае ошибки считывания не выводит сообщение "Type loading error". Но существует еще одна процедура загрузки, которая это делает. Она находится под адресом 2050 и выглядит так:

```
2650  CALL      1366   ; Считывание блока данных
2053  RET       C      ; Возврат, если не было ошибки
2054  RST       8      ; иначе RST 8 с сообщением
2055  DEFB      26     ; "TYPE LOADING ERROR"
```

После возврата из процедуры 1366 указатель переноса содержит информацию о правильности считывания блока. Если он удалён, то это означает, что наступила ошибка. Некоторые загрузчики используют процедуру 2050, а не 1366.

Иногда загрузчики не пользуются ни той ни другой процедурами, а заменяют их собственными, но они однако, обычно очень похожи на процедуру 1366 или даже являются ее переделкой, благодаря которой, например, блоки данных загружаются в нижнюю часть памяти - с больших адресов к нижним или загрузка идет с другой скоростью. Такую программу следуёт анализировать с помощью дизассемблера (например, MONS), сравнивая некоторые её фрагменты с тем, что находится в ROM.

Сейчас мы объясним, как использовать процедуру из ROM для считывания БЕЙСИКа по любому адресу, а не в область, предназначенную для него: сначала с помощью "SZYTACZ" прочитаем заголовок программы, которую мы хотим вскрыть, и запоминаем ее длину (т.е. длину всего блока - программу вместе с переменными). Затем вводим соответствующую программу, которая прочтет БЕЙСИК под адрес, который мы установим (выше RAMTOR):

```
LD        IX,АДРЕС
LD        DE,ДЛИНА
LD        A,255
SCF
JP        2050
```

Также, как и при подмене заголовка, если мы не знаем длину программы, то можем задать завышенное значение, но тогда чтение завершится сообщёнием "Type loading error". Но считывание ассемблера каждый раз, чтобы ввести программу, приведенную выше, можно вызвать раздражение. Следовательно, лучше создавать эту программу с уровня БЕЙСИКа с помощью POKE:

```
10 INPUT "АДРЕС ЧТЕНИЯ BASIC?";A
20 RANDOMIZE A: CLEAR A-1
30 LET A=PEEK 23670: LET B=PEEK 23671
40 LET ADR=256*B+A
50 INPUT "ДЛИНА BASIC ?";C
60 RANDOMIZE C: LET CHPEEK 23670
70 LET D=PEEK 23671
80 FOR N-ADR TO ADR+11
90 READ X: POKE N,X
100 NEXT N
110 DATE 221,33,A,B,17,C,D,62,255,195,2,8
120 RANDOMIZE USR ADR
```

Устанавливаем ленту с обрабатываемой программой за ее заголовком. Затем запускаем программу, приведенную выше, вводим данные и включаем магнитофон. Результат аналогичен тому, который получаем при подмене заголовков, но первым же видимым достоинством этого способа является то, что мы не создаем беспорядок на кассетах.

В завершение стоит вспомнить еще об одной процедуре, размещенной в ROM под адресом 1218. Это процедура SAVE-BYTES обратная LOAD-BYTES, т. е. записывающая на ленту блок с заданными параметрами: перед ее выполнением в регистре IX размещаем адрес, с которого начинается здесь запись, DE содержит длину записываемого блока. В буфере помечаем должен ли это быть заголовок (0) или блок программы (255). Состояние указателя CARRY значения не имеет.

7. ИСПОЛЬЗОВАНИЕ СИСТЕМНЫХ ПРОЦЕДУР

В предыдущем разделе были представлены процедуры из ROM: SAVE-BYTES и LOAD-BYTES. Здесь рассказывается, как использовать эти процедуры для защиты программ. Займемся блоками машинного кода, которые запускаются удивительным образом. Первым таким способом является прикрытие загрузчика блоком, который им загружается. Такая защита, к примеру, применяется в игре "ТРИ НЕДЕЛИ В ПАРАДИС". Проследим способ чтения этой программы так, чтобы она не стартовала автоматически.

Начнем, как обычно, с БЕЙСИКа. Оказывается, практически, единственно важной инструкцией является RANDOMIZE USR… Лучше всего восстанавливать процедуру, начиная с адреса запуска RANDOMIZE USR (PEEK 23627+256*PEEK 23628). Т.е. в нашем случае с адреса 24130. Подобные процедуры используют известную нам процедуру 1366, считывающую блоки без заголовка. Так и в этом случае,

но перед ее вызовом с помощью команды LDIR, процедура загрузки переносит сама себя в конец памяти под адрес 63116 и переходит туда по команде JP:

```
24130   DI                  ; Запрет прерывания
24131   LD      SP,0        ; LD SP,65536
24134   LD      HL,(23627)  ; В HL адрес на 28
24137   LD      DE,28       ; Больше чем значение
24140   ADD     HL,DE       ; Переменной VARS
24141   LD      DE,63116    ; В DE Адрес, а в
24144   LD      BC,196      ; BC длина блока
24147   LDIR                ;
24149   JP      63116       ; продолжение выполнения
---------                   ; программы с другого
24152   LD      IX,16384    ; адреса
24156   LD      DE,6912
```

Теперь считывается картинка, а затем главный блок данных:

```
63116   LD      IX,16384    ; Подготовка загрузки
63120   LD      DE,6912     ; картинки на экран
63123   LD      A,255       ; с помощью процедуры
63125   SCF                 ; LOAD-BYTES ИЗ ROM
63126   CALL    1366        ;
63129   LD      IX,26490    ; Параметры главного
63133   LD      DE,38582    ; блока программы,
63136   LD      A,255       ; который считываясь
63138   SCF                 ; затирает эту процедуру
63139   CALL    1366        ; считывания блока
-----------
63142   JP      NZ,+79      ; После возврата из
63144   CP      A           ; процедуры 1366 здесь
63146   CALL    65191       ; же находится другая
63149   JP      NC,-2       ; программа
```

Но способ запуска считанной программы требует пояснения. Как вы знаете, каждая инструкция CALL заносит в машинный стек адрес, с которого начинает работать программа после выхода из подпрограммы, в этом загрузчике после выполнения второй инструкции CALL 1366 в стек заносится адрес команды, за CALL, т.е. 63142 и процедура загрузки самым обычным способом затирает сама себя, т.к. считывает байты с магнитофона в ту область памяти, где она была размещена. Существенное значение имеет способ запуска считанной программы: процедура 1366 кончается естественно инструкцией RET, которая означает переход по адресу, записанному в стек или в нашем случае по адресу 63142. В процессе считывания программы процедура, которая находилась там, была замещена считанной программой, но микропроцессор этого не замечает - он возвращается по адресу, с которого выполнил CALL 1366, не обращая внимания, что там находится уже совершенно другая программа. Это схематично представлено на рис 1.

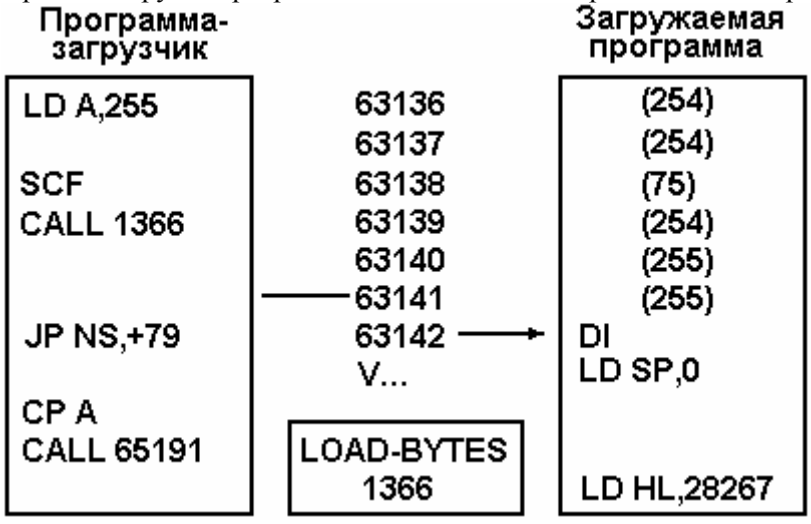

Рис. 1.

С левой стороны расписано содержимое памяти до, а с правой после считывания программы. Команды, отмеченные звездочками (*), ложатся на выполняемую программу.

Возникает вопрос, как распознать защиту такого типа и как ее ликвидировать. Начнем с БЕЙСИКа, считываем загрузчик (в ассемблере) и определяем адрес окончания считанных блоков (добавляя к адресу начала регистр IX длина/регистр DE). Если какой-либо из блоков накрывает процедуру загрузки, то это означает, что программа считывается и загружается именно таким способом.

Дальше все просто. Достаточно, опираясь на данные о блоках (адрес и длину), написать коротенькую процедуру, загружающую интересующий нас блок кода или подготовить соответствующий заголовок, затем с помощью CLEAR ADR установить соответствующим образом машинный стек (чтобы машинная программа не уничтожила стек) и, наконец, считать программу. После выполнения в ней необходимых изменений записываем ее на ленту, но таким же образом, каким был записан оригинал (длина блока должна совпадать прежде всего!). Если этот блок был без заголовка (а так и есть в нашем случае), то записываем его обычным SAVE"…" CODE …, но опуская заголовок, т.е. включаем магнитофон только в перерыве между заголовком и блоком кода. Также можно пробовать запускать считанный блок перехода на требуемый адрес командой RANDOMIZE USR…, но это не всегда может получиться. В игре ТРИ НЕДЕЛИ В ПАРАДИЗ" этим адресом будет 63142, и как ты можешь убедиться - этот метод срабатывает.

Другим интересным способом запуска блоков машинного кода является считывание программы в область машинного стека. Этим способом можно запускать блоки машинного кода, загружая их просто через LOAD "" CODE. Этот метод показан схематично на рисунке 2.

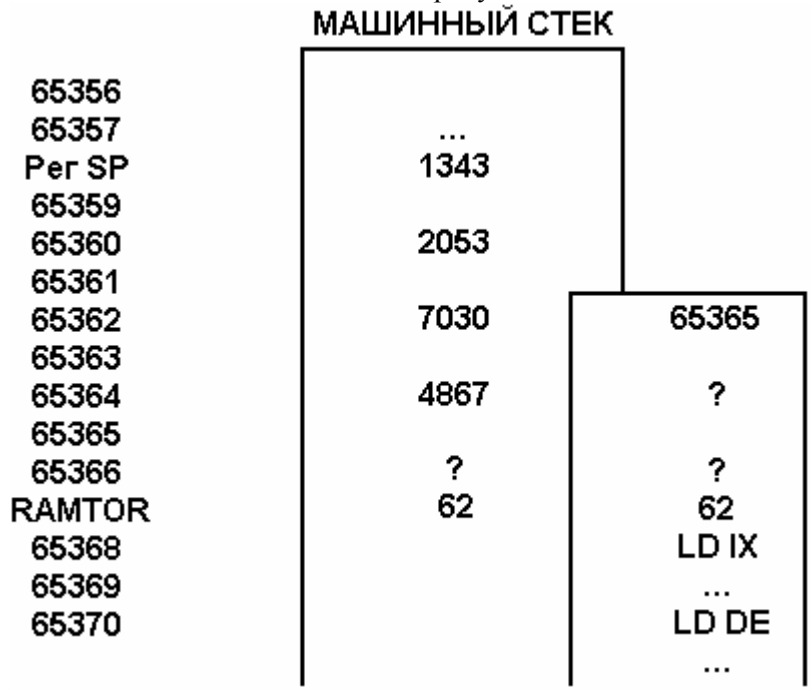

Рис. 2.

Указатель стека (регистр SP) принимает показанное на рис 2. состояние в процессе выполнения процедуры 1366 (вызванное из БЕЙСИКа через LOAD "" CODE). Способ запуска программы суммарно весьма прост. Адрес считывания блока рассчитан так, что блок считывается на машинный стек именно с того места, в котором находится (записанный интерпретатором БЕЙСИКа) адрес возврата из инструкции LOAD "" CODE (он тогда равен значению системной переменной ERRSP-2) или прямо из процедуры LOAD-BYTES (равный ERRSP-6). Тогда два первых байта программы обозначают адрес ее запуска. Этот способ очень похож на предыдущий, за исключением того, что там подменялась процедура загрузки, а здесь – адрес возврата из этой процедуры или просто адрес возврата из инструкции LOAD. После считывания блока кода, микропроцессор считывает содержимое стека и переходит по прочитанному адресу (который только что появился в памяти). По этому адресу в программе находится начало процедуры загрузки её последующих блоков. Это видно на рис 2.

Метод обхода такой защиты также весьма прост, достаточно заменить RAMTOR на соответственно низкое значение, а затем прочитать блок кода, который благодаря этому не запустится. Ситуация может осложниться, если блок очень длинный (что случается очень редко, но встречается) - тогда мы должны поступить также, как и с каждым длинным блоком, но помнить с какого адреса он запускается.

Займемся теперь расчленением блоков с длиной превышающей 42К. Взлом блоков этого типа основан на разделении их на такие фрагменты, чтобы в памяти еще осталось место для MONSa другого дизас-

семблера, исправления этих фрагментов, а затем их склеивания в одно целое либо написание новой процедуры загрузки. Обычно достаточно разделить длинный блок две части. Чтобы получить первую, используем процедуру 1366, но с другими параметрами (не с теми, которых требует разделенный на части блок). Ранее из процедуры загрузки или если такой нет, то из заголовка этого блока получаем ее длину и адрес загрузки. Просто задаем адрес, по которому хотим разместить блок (выше RAMTOR), а также длину, примерно 16К (несмотря на то, что блок этот значительно длиннее). Считываем теперь этот блок через CALL 1366 или CALL 2050, но во втором случае сообщение "Tape loading error", которое появится, не даст нам ни какой информации о верности считывания - загружаем часть блока и, следовательно, без контрольного байта, который находится в конце. Считанную таким образом первую часть блока записываем на ленту и сразу же приступаем ко второй части. Ее считывание труднее, но тоже возможно, несмотря на ограничения по памяти. Достаточно использовать тот факт, что SPECTRUM существуют 16K ROM, запись, в которую невозможна. Например, вызываем процедуру 1366 с адресом равным 0 и начальные 16K будут просто потеряны, а в память RAM считается только следующие 32К или меньше (в зависимости от длины блока). Считываемый блок займет в памяти RAM адрес с 16384 и дальше, заходя на системные переменные и оставляя без изменения лишь те байты, адреса которых больше длины блока. Поэтому необходимо позаботиться о том, чтобы машинный стек, а также написанную нами процедуру загрузки разместить в конце памяти. Надо также помнить о том, что система БЕЙСИКа будет уничтожена, и записать сразу считанный блок на ленту можно только процедурой, написанной на ассемблере. Кроме того, в промежутках времени между считыванием фрагмента блока и его записью нельзя разблокировать прерывания, т.к. они изменяют содержимое ячеек с адресами 23552-23560, а также 23672-23673, а там находится считанный блок. Чтобы выполнить это последнее условие войдем в середину процедуры 1366, благодаря чему после считывания блока не будет выполнена процедура 1343. Именно она еще и разблокирует прерывания.

С помощью CLEAR 64999 переносим машинный стек, а с адреса 65500 помещаем процедуру загрузки:

```
        ORG     65000
        LD      IX,0          ; Адрес считывания
        LD      DE,DL         ; Длина блока
        LD      A,255         ; Подготовка к считыванию
        SCF                   ; блока
        INC     D             ; Таким способом
        EX      AF,AF         ; заменяем начало
        DEC     D             ; процедуры 1366
        DI                    ; а затем входим
        LD      A,15          ; ее середину:
        OUT     (254),A
        CALL    1378
        LD      A,0           ; Черная рамка
        JR      C,OK          ; будет означать
        LD      A,7           ; верное считывание
OK      OUT     (254),A       ; белая - ошибочное
CZEKAJ  LD      A,191         ; Ожидаем нажатие
        IN      A,(254)       ;"ENTER"
        RRA                   ;
        JR      C,CZEKAJ
        LD      IX,0          ; Запись считанного
        LD      DE,DL-16384   ; блока на
        LD      A,255         ; ленту
        CALL    1218          ; а также
        LD      HL,64999      ; инициализация
        JR      4633          ; системы БЕЙСИКА
```

Вместо того, чтобы считывать ассемблер и вводить эту программу, можно запустить программу на БЕЙСИКе, представленную на листинге 1:

Листинг 1

```
10 CLEAR 64999
20 INPUT "DLINA BLOKA?"; DL.RANDOMIZE DL: LET X=PEEK 23670: LET Y=PEEK
23671: LET S=0
30 FOR N=65000 TO 65054: READ A: LET S=S+A: POKE N,A: NEXT N
40 IF S<>5254+2*(X+Y)-64 THEN PRINT "OSIBKA V DANNYH": STOP
```

```
      50 PRINT "WSE DANNYE OK=> WKLUCHI MAGNITOFON"
      60 RANDOMIZE USR 65550
      70 DATA 221, 33, 0, 0, 17, X, Y, 62, 2, 55, 55, 20, 8, 21, 243, 62, 15,
211, 254
      80 DATA 205, 98, 5, 62, 0, 56, 2, 62, 7, 211, 254, 62, 191, 219, 254,
31, 56
      90 DATA 249, 221, 33, 0, 0, 17, X, Y-64, 62, 255, 205, 194, 4, 33, 231,
253, 202, 25, 18
```

Как выглядит разделение блока? Устанавливаем ленту на блоке кода, который желаем разделить. Если он имел заголовок, то его опускаем. Запускаем процедуру и включаем магнитофон. Не пугайся, если увидишь в определенный момент, что программа считывается на экран - так и должно быть. После загрузки блока цвет рамки сигнализирует правильность считывания: если рамка черная, то все в порядке, если белая, то была ошибка. Тогда вложи в магнитофон другую кассету, включи запись и нажми "ENTER". Программа запишет на ленту, потом процедура вернется в БЕЙСИК, инициализируя систему, сообщением "© 1982…", но не очищая память (уничтожается только область от начала экрана до, примерно, 24000 адреса). Теперь, подготавливая заголовок или записывая коротенькую процедуру, можно прочесть полученный блок под любой адрес.

Когда ты найдешь то, что искал и захочешь запустить измененную программу, то придется немного помучиться и "склеить" раздёленную программу или написать для нее новую процедуру загрузки. Если программа заполняла полностью 48К памяти RAM возможен лишь второй метод.

Если надо соединить блоки - достаточно нависать процедуру, похожую на разделяющую, но такую, которая считывает первый блок под адрес 16384, второй - сразу за ним, а затем запишет их вместе как один блок.

В этом разделе был описан способ запуска блоков машинного кода путем считывания в область машинного стека. Чтобы такой блок запустить, достаточно ввести инструкцию LOAD "" CODE, которая запустит его. Чтобы убедиться в этом на практике введи и запусти программу с листинга 2:

Листинг 2

```
10 CLEAR 65361: LET S=0
20 FOR N=65362 TO 65395: READ A: POKE N,A: LET S=S+A: NEXT N
30 IF S<>3437 THEN PRINT "OSIBKA W STROKE DATA": STOP
40 SAVE "BEEP" CODE 65362,34
50 DATA 88,255,3,19,0,6,221,2,29,6,8,197,17,30,0,33,0,8,43,124,18
60 DATA 181,3,33,0,8,43,124,18,1,32,251,193,16,235,221,225,201
```

Она запишет на ленте короткий блок машинного кода, который будет запускаться самостоятельно. После записи этого блока освободи память с помощью RANDOMIZE USR 0 или RESET (по возможности переставь RAMTOR на нормальное значение с помощью CLEAR 65367) и прочти его, вводя LOAD "" CODE. Цель этого блока проинформировать о том, что он запустился - он делает это несколькими звуковыми сигналами. Ты услышишь их сразу после считывания программы, как только с рамки исчезнут гранатово-желтые полосы.

8. ДЕКОДИРОВАНИЕ ЗАКОДИРОВАННЫХ БЛОКОВ ТИПА "BYTES"

Что означает, что программа или блок закодированы? Кодирование - это род весьма простого шифра, делающего невозможным правильную работу программы. Для её запуска служит специальная декодирующая процедура, которая находится в той же программе, а также, что самое важное, не закодированная. Так для чего же служит кодирование? Это просто очередное затруднение доступа к тексту программы после того как все предшествующие предохранители взломаны и программа считана без самозапуска. В этом случае в памяти лежит "полуфабрикат", который только после переработки декодирующей процедурой становится программой. Кодирование может быть основано на инверсии всех байтов в программе. Хорошим примером является программа "ART STUDIO". Её бейсиковая часть практически не защищена никак, но главный блок программы ("STUDIO-MC" CODE 26000,30672) частично закодирован. Что делать, чтобы раскодировать "ART STUDIO" этот адрес 26000. Там находится инструкция JP 26024, которая осуществляет переход к декодирующей процедуре. Вот её текст:

```
      26024   21C165    LD      HL,26049       ; Запись адреса
      26027   E5        PUSH    HL             ; 26049 СТЕК
      26028   21C165    LD      HL,26049       ; Адрес начала
      26031   11476C    LD      DE,27719       ; Адрес конца
      26034   7E        LD      A,(HL)         ; Выбор байта из
      26035   D622      SUB     34             ; памяти, переко-
```

```
26037   07        RLCA                ; дарование его с
26038   EECC      XOR     #CC         ; помощью SUB, RLCA,
26040   77        LD      (HL),A      ; XOR и запись
26041   23        INC     HL          ; Следующий адрес
26042   B7        OR      A           ; Проверка признака
26043   ED52      SBC     HL,DE       ; конца, возврат в
26045   19        ADD     HL,DE       ; цикл или выход
26046   20F2      JP      NZ,26034    ; по адресу, записан-
26048   C9        RET                 ; ному в стеке
26049   B2EDF1                        ; или 26049
```

Сначала процедура помещает в стек адрес 26049. Теперь начинается декодирование: в регистр HL заново загружается адрес 26049 - как начало декодированного блока, в DE - 27719, как адрес последнего закодированного блока. Затем в цикле декодируются последовательно байты - инструкция SUB 34, RLCA и XOR #CC являются ключом, с помощью которого расшифровывается эта часть программы. Наконец, проверяется условие достижения адреса 27719, как последнего декодируемого (содержится в DE). Выполняется инструкция RET, но последним записанным в стек является не адрес возврата в БЕЙСИК, а записанный в начале с помощью PUSH HL адрес 26049 или адрес только что раскодированого блока, следовательно происходит его запуск.

Обратим внимание, какие инструкции осуществляют дешифрацию? никакая из них не теряет ни одного бита. Вычитание производится по модулю 256 и для двух разных входных данных. Результаты тоже различны. RLCA заменяет значения битов 7, 6, 3 и 2 на противоположные. Другими инструкциями, имеющими те же самые свойства, являются, например, ADD, INC, DEC, RRCA, NEG, CPL, но не OR или AND. Как раскодировать этот блок? Проще всего, вводя в БЕЙСИКе:

POKE 26027,0 (код инструкции NOP)

Тем самым ликвидировать инструкцию PUSH (RET в конце программы переведет в БЕЙСИК) и выполнить RANDOMIZE USR 26000, Стоит однако помнить о том, что декодирующая программа может проверяться другим фрагментом программы. Вот дальнейшая часть программы в "ART STUDIO":

```
26283   67        LD      H,A         ; В А уже находится
26284   6F        LD      L,A         ; значение 0
26285   E5        PUSH    HL          ; Запись его в стек
26292   3AA865    LD      A,(26024)   ; и проверка
26295   FE21      CP      #21         ; содержимого ячеек
26297   C0        RET     NZ          ; с адресами
26298   2AA965    LD      HL,(26025)  ; 26024…27027
26301   B7        OR      A           ; и если оно другое
26302   11C165    LD      DE,(26049)  ; то стирание
26305   ED52      SBC     HL,DE       ; памяти с помощью
26307   C0        RET     NZ          ; RET NZ
26308   3AAB65    LD      A,(26027)   ; (под адрес 0)
26311   FEE5      CP      #E5         ; Если все нормально
26313   C0        RET     NZ          ; то снятие адреса 0
26314   E1        POP     NZ          ; и нормальный
26315   C9        RET                 ; выход
```

Этот фрагмент проверяет: наверняка ли декодирующая процедура запустила программу, и если нет, то с помощью RET NZ стирает память (т.к. в стеке записан адрес 0).

Что делать в этом случае? Можно ликвидировать и эти меры защиты, но в некоторых программах это не поможет. Тогда остается другой выход: снова закодировать программу, т.е. сделать обратное, чем декодирующая программа. В нашем случае следовало бы выполнить.

```
        XOR     #CC
        RRCA
        ADD     34
```

В "ART STUDIO" неизвестно для чего была помещена кодирующая программа. Она находится под адресом 26003 и выглядит так:

```
26003   21C165    LD      HL,26049    ; Адрес начала
26006   11476C    LD      DE,27719    ; Адрес конца
26009   7E        LD      A,(HL)      ; Выборка байта из
26010   EECC      XOR     #CC         ; памяти, кодирование
26012   0F        RRCA                ; его с помощью XOR
```

```
26013   C622     ADD     34              ; RRCA, ADD и его
26015   77       LD      (HL),A          ; Запись
26016   23       INC     HL              ; Следующий адрес
26017   B7       OR      A               ; Проверка окончания
26018   ED52     SBC     HL,DE           ; Переход в цикл или
26020   19       ADD     HL,DE           ; возврат в БЕЙСИК
26021   20F2     JR      NZ,26009
```

Как видно, она построена аналогично декодирующей процедуре. Если таковой нет, то ее можно быстро и просто написать на основе декодирующей процедуры (например, ATIC ATAC, NIGHT SHADE, THE WORLD).

При защите программ применяются также малоизвестные и неопубликованные в фирменных каталогах команды микропроцессора Z80. Благодаря их применению программа становится мало читаемой, да и просмотр ее дизассемблером затруднен.

Наиболее часто встречаемыми не опубликованными инструкциями являются команды, оперирующие на половинках индексных регистров IX и IY в группе команд, которым не предшествует никакой иной префикс (т.е. CBH и EDH). Основываются они на префиксации кодом DDH или FDH команды, касающейся регистра H или L. В этом случае вместо этого регистра соответствующая половина индексного регистра. Через HX обозначается старшая часть регистра IX, через LX - младшая. Аналогично HY и LY. Вот пример:

КОД	КОМАНДА	КОД	КОМАНДА	КОД	КОМАНДА
24	INC H	DD 24	INC HX	FD 24	INC HY
2D	DEC L	DD 2D	DEC LX	FD 2D	INC LY
4C	LD C,H	DD 4C	LD C,HX	FD 4C	LD C,HY
64	LD H,H	DD 64	LD HX,HX	FD 64	LD HY,HY
26 01	LD H,1	DD 26 01	LD HX,1	FD 26 01	LD HY,1
B5	OR L	DD B5	OR LX	FD B5	OR LY

Это верно для всех команд однобайтовых пересылок между регистрами и восьмибитовых операций AND, OR, ADD, ADC, SUB, BC, CP - выполняемых в аккумуляторе.

Префикс FDH или DDH относится ко всем регистрам H, L, или HL, присутствующим в команде, следовательно, в одной инструкции не возможно использование ячейки адресованной как (HL), регистра HL, H или L одновременно с HX, HY, LX, LY (в дальнейшем ограничимся регистром IX, но все это относится также и к регистру IY), например:

```
66    LD     H,(HL)        DD66** LD      HX,(IX+*)
75    LD     (HL),L        DD75** LD      (IX+*),LX
65    LD     H,D           DD65   LD      HX,D
```

Несколько иначе представляется ротация ячейки, адресуемой индексным регистром, т.е. инструкцией начинающейся кодом DDCB. Инструкция типа RR (IX+**) и ей подобные подробно описаны во всех доступных материалах о микропроцессорах Z80, но мало кто знает об инструкциях типа RR (IX+***),% и им подобных, где % обозначает любой регистр микропроцессора. Они основаны на префиксировании кодом DDH или FDH инструкции типа RR%, также обстоит дело с инструкциями SET N,(IX+*),% а также RES N,(IX+*),% (но для BIT -уже нет). Выполнение такой инструкции основано на выполнении нормальной команды RR (IX+*) (или подобной), SET N,(IX+*) или RES N,(IX+*), а затем пересылки результата как в ячейку (IX+*) так и в соответствующий внутренний регистр микропроцессора, например:

```
CB13       RLE
DDCB0113   RL      (IX+1),E
DDCB0116   RL      (IX+1)
DD5E01     LD      E,(IX+1)
```

В конце рассмотрения инструкции этого типа следует вспомнить, что команд EX DE,IX или EX DE,IY нет. Префиксирование команды EX DE,HL не дает никаких результатов. Также префиксирование команд, коды которых начинаются с EDH, а также тех, в которых не присутствует ни один из регистров H, L или пары HL (например LD B,H, RRCA и т.д.).

Очередной любопытной командой является SLI (SHFIT LEFT AND INCREMENT), выполнение которой аналогично SLA с той разницей, что самый младший бит устанавливается в 1. Признаки устанавливаются идентично SLA и другим сдвигам:

```
CC37       SLI     A
CB36       SLI     (HL)
DDCB**36   SLI     (IX+*)
```

```
DDCB**57    SLI      (IX+*),A
```

Временами некоторые проблемы вызываю? построение флажкового регистра, особенно тогда, когда он используется достаточно нетипично, например:

```
PUSH    AF
POP     BC
RL      C
JP      NC,…
```

Его вид представлен на рис. 1.

7	6	5	4	3	2	1	0
S	Z	F5	H	F3	P/V	N	C

Дополнительной особенностью регистра F является то, что биты 3 и 5 (обозначенные как F3 и F5) точно отражают состояние восьмибитовой арифметической или логической операции INC %,(C) и IN F,(C), например:

```
XOR     A          ; Запись значения 0
ADD     A,15       ; Результат – 00001111
                   ; Указатели будут: F5=0; F3=1
```

Последняя (кто поручится, что их больше нет?) тайна Z80 это регистр обновления памяти P. А точнее то, что когда после очередного машинного цикла микропроцессора инкрементируется значение этого регистра, то его старший бит остается неизменным, следовательно, может быть использован для хранения любой, естественно однобитовой, информации. Важно также то, что инструкция LD A,P с помощью которой может быть получена эта информация, устанавливает также указатель S и, следовательно, не требуется дополнительная инструкция, проверяющая значение этого бита.

В конце несколько коротких, но часто приемлемых мер защиты. Их ликвидация основывается обычно на действиях обратным защитным. Например, переменная DFSZ (23659) определяет число строк в нижней части экрана, требуемое для вывода сообщения об ошибке или для ввода данных. Во время работы программы это значение равно двум, но достаточно занести туда 0, чтобы программа зависла при попытке прерывания (т.к. выводится сообщение для которого нет места). Если в программе находится инструкция INPUT или CLS, то она имеет подобный результат.

Распространенным способом защиты является также изменение значений переменной BORDCR (23624), которая определяет цвет рамки и атрибутов нижней части экрана. Значение отдельных битов следующие:

7 бит - FLASH
6 бит - BRIGHT
5,4,3 биты - BORDER
2,1,0 биты - INK

Способ защиты основывается на том, что цвет чернил тот же самый, что и у рамки, следовательно, при остановке программы, можно подумать, что она зависла (т.к. не видно сообщения). Разблокировать программу можно вписав BORDER 0 (или другой цвет).

Очередным способом защиты является изменение переменной ERRSP (23613-23614) или дна машинного стека (эта переменная указывает дно стека), где обычно находится адрес процедуры обрабатывающей ошибки языка BASIC (вызываемой с помощью RST 8). Уменьшение значения ERRSP на 2 вызывает защиту программы от "BREAK" и любой другой ошибки - программа заново запускается с того места, в котором произошла ошибка. Смена содержимого дна машинного стека или другая замена значений ERRSP может вызывать зависание или даже рестарт компьютера.

Известным способом защиты является также занесение значения больше 9999 в ячейки памяти, обозначающие номер строки БЕЙСИКа (первый байт - старший). Если он размещается в границах 10000 - 16383, а на листинге это выглядит, например 0000 (вместо 10000) и строки невозможно скорректировать (EDIT), если же превышает 16384 - дальнейшая часть программы считается не существующей. Можно также встретить защиту, основывающуюся на занесении минимальных значений (то есть 1) в переменные REPDEL (23561) и REPPER (23562), что затрудняет работу с компьютером, но для её ликвидации требуется лишь быстрая реакция (запиши с помощью POKE нормальные значения: 35 и 5).

Стоит вспомнить еще о переменной NMIADD, которая не используется из-за ошибки в ROM. Она должна была содержать адрес обслуживания прерывания NMI (всегда понимаемого). Ошибка в том, что микропроцессор переходит туда лишь тогда, когда он - 0. Если, однако, добавить к SPECTRUM соответствующую приставку, путь к задержке любой программы был бы открыт.

СЧАСТЛИВОЙ РАЗБОРКИ ПРОГРАММ!!!

ИСПОЛЬЗОВАНИЕ "АДРЕСОВ БЕССМЕРТИЯ"

Для пользования справочником "адресов бессмертия" желательно:

- знать хотя бы немного язык Бейсик;
- иметь представление о различных типах файлов;
- уметь пользоваться монитором - дизассемблером;
- владеть программированием в машинных кодах.

Если Вы все это знаете, то инструкции Вам не нужны. Если нет - читайте дальше.

Простейший случай. Например, игра "GHOTIK". Вы набираете команду "LOAD" и наблюдаете загрузку первого блока программы (с бейсик - блока начинается практически любая и каждая программа в наше время, хотя раньше было не так). Сначала появляется "тельняшка" на рамке, затем возникает название игры, снова "тельняшка" - и беспорядочные цветовые полосы, помелькав, исчезли. Теперь нажмите клавишу BREAK.

Все замерло. Нажимаете LIST и ENTER (в дальнейшем помните, что ENTER завершает ЛЮБУЮ команду). Если текст программы не возник на экране, дайте команду INK 9 и снова LIST. Теперь Вы видите текст бейсик-блока:

```
2 PAPER NOT PI: BORDER NOT PI: INK NOT PI: CLEAR 24999
4 LOAD "" CODE: RANDOMIZE USR 24500
7 RANDOMIZE USR 40960
```

В строке 3 первые три команды из четырех должны были помещать Вам прочитать текст - Вы преодолели их командой INK 9.

Теперь можно вносить изменения. Перед последним стартом программы (строка 7) надо внести желанные изменения:

```
6 POKE 41233,0: POKE 41318,0: POKE 43936,0
```

Набрав эту строку, нажмите ENTER - и она займет свое место а программе. Теперь можно продолжить загрузку (RUN) или записать измененный блок на магнитофон (SAVE"GOTHIK"LINE 3).

Не всегда так легко прочитать текст программы - он может быть защищен от просмотра. Есть несколько распространенных способов защиты:

- блокирование BREAK;
- блокирование MERGE;
- использование управляющих кодов.

Кроме того, преодолев защиту от просмотра, можно натолкнуться на защиту от редактирования (нулевой номер строки и т.п.). Подробно это рассмотрено в соответствующей литературе ("Тайники ZX SPECTRUM"), но один простой прием можно рекомендовать. Воспользуйтесь копировщиком COPY86/M. Считайте в него бейсик-блок игры и просмотрите (клавиша "B" - для просмотра, "Y" - для перелистывания -скроллинга). Если Вам полученный текст более - менее понятен, стоит попытаться. Нажмите "R" - с бейсик - блока будет снят автостарт. Теперь запишите этот файл на магнитофон, обнулите машину и введите записанный файл командой LOAD "". Программа не стартует и позволит себя просмотреть командой LIST (если не были применены управляющие коды). Вносите изменения как описано выше.

Бывают бейсик-блоки защищенные от вмешательства и редактирования. В этом случае нужно либо искать эту защиту и снимать ее, либо пойти другим путем. Допустим, изменение надо внести в ячейку 54321,0. Загрузим игру (целиком) в копировщик COPY86/M и найдем, какой блок (файл) перекрывает собой эту ячейку. Блоки грузятся либо командой LOAD "name" CODE, либо LOAD "name" CODE Address. В первом случае адреса загрузки кодов берутся из заголовков файлов - их отражает средняя колонка чисел на табло копировщика COPY86/M.

Во втором случае адрес загрузки блока - это Address (5-значное число).

Главное, внимательно просмотрев бейсик-блок, понять, какой блок какой командой загружается.

Допустим, блок "NONAME" длиной 15000 (крайняя колонка COPY86/M) загружается с адреса 50000.

Теперь пришла пора использовать копировщик COPY-COPY (или Pirate02). Загрузив его, дайте команду

```
LOAD AT 49983
```

и считывайте в него с магнитофона блок "NONAME". Почему 49983 ? Потому что заголовок блока имеет длину 17 байт и тоже считывается в копировщик - вдруг Вы пожелаете его изменить ? Итак, 50000 - 17 - 49983. Теперь последнее:

```
POKE 54321,0
```

Осталось сохранить измененный блок на магнитофоне.

В справочниках указывается не только POKE, но и адрес загрузки блоков в копировщик COPY-COPY, особенно когда эти блоки "безголовые" (без заголовков). Такие блоки загружаются только программой в машинных кодах, и адрес их загрузки иногда трудно определить. При загрузке "безголового" файла не надо отводить 17 байт под заголовок.

Возможно, самая неприятная, хотя и не самая главная, трудность - система распо-странения программ на рынке. Любой человек, торгующий на рынке программами, волен делать с ними все, что угодно. Авторского права фактически нет. И не составляет особого труда любому начинающему программисту перекорежить программу, "украсив" ее своим именем и телефоном (своим ли ?). Хорошо, если после этого программа еще и работает, но "адреса бессмертия" могут уже к ней не подходить. Поэтому, для некоторых игр дано несколько вариантов адресов - для нескольких известных вариантов игры.

Но, как показывает практика, в четырех случаях из пяти попытки вставить "адрес бессмертия" удаются. Так смелее! Хакер (взломщик) вышел на тропу войны!

БЕССМЕРТИЕ, НЕПОБЕДИМОСТЬ И ДРУГИЕ ПОЛЕЗНЫЕ ВЕЩИ.

Если указана команда POKE, значит надо изменить загрузчик. Если не указана, возможно, придется применить программу COPY-COPY в зависимости от Вашей версии игры.

A

ABU SIMBEL - PROFANATION

POKE 49290,N (число жизней): POKE 47684,0:POKE 42169,0:POKE 42170,0: POKE 42171,0

ACADEMY

31378,A;31386,B;31249,C;31305,D (увеличение лимитов на снаряжение)

ACE

32506,0;32507,0;32508,0

ACTION FORCE 2

50145,36(51450?) :51904,0 либо POKE 47874,182 либо 50262,N

ACTION REFLEX

Для фирменного варианта: POKE 23988,54: POKE 23349,201: RAND USR 23935: POKE 50770,0: POKE 50771,0: POKE 50772,0: POKE 50964,0: POKE 50965,0: POKE 50966,0: RAND USR 50000

AD ASTRA

28591,0;28592,0;28593,0 либо POKE 35853,0(182?)

AFTERBURNER

POKE 37934,0:POKE 37935,0:POKE 37936,0

AFTER WAR

48950,195;48951,217;48952,190

AGENT X

POKE 26099,0:POKE 25917,0

AGENT X - 2

ч.1-57821,0;ч.2-62499,0;ч.3-50561,0

AGENT 1000

Пароль: there's no escaping it

AH DIDDUMS

24920,255 либо POKE 24786,0

AIR FORCE 2

POKE 51904,2 либо 50262,N

AIRWOLF

45982,0

AIRWOLF - 2

53471,0

ALCHEMIST

47414,0;49745,195 либо 31055,0

ALIEN 8

43255,201 либо 43735,201 либо POKE 44461,97:POKE 44462,185: POKE 51736,0:RAND USR 25344 либо 44526,0;51736,0 либо POKE 43735,201 :POKE 51736,0:POKE 44460,201 :POKE 43255,201: POKE 44526,0: POKE 45121,1

ALIEN HIGHWAY

39443,0;39142,0;35125,0 либо 39410,201

ALIENS

31014,0(патроны):30738,0(жизни):34484,195

Пароли:

7324G
2727H
1506E
3761H
7100D
7123G

ALIENSINDROME

47503,182

AMAUROTE

46192,0 либо POKE 37311,0: POKE 37316,6: POKE 37321,0 либо POKE 40615,0:POKE 46312,0:POKE 46381,201 либо

```
10 CLEAR 24999: LOAD "" CODE: RAND USR 50000: POKE 23570,16
16 LOAD "" CODE: FOR I=1 TO 4: READ A: FOR J=A TO A+2: POKE J,0: NEXT J: NEXT I: RAND USR 36924: PRINT USR 26600: DATA 42912,42929,42455,38551
```

AMAZON WOMEN

57590,183

ANARHY

42405,255

ANDROID

52250,32;55249,24;53897,0 либо 33702,0 либо
25280,0;25312,0;46610,24

ANDROID 2

52262,0;53894,0 либо
POKE 52249,24:POKE 52250,62:POKE 53897,0

ANTIRIAD

57501,0;57502,0;57503,0 либо 54639,0;54640,0;54641,0

AQUAPLANE

POKE 25448,0:POKE 25449,195

AQUARIUS

POKE 31055,0

ARCADIA

POKE 25776,0:POKE 26197,0

ARKANOID

Загрузив в COPY-COPY последний сегмент с адреса 23552, ввести 33702,0 либо POKE 33842,0:POKE 35665,9:POKE 37120,0:POKE 39410,1 либо в таблице счета напечатать PBRAIN

ARKANOID 2

POKE 35417,0:POKE 37483,0 (для оригинальной версии) либо
POKE 33054,1 :POKE 37484,182:POKE 35418,182 либо
ввести пароль вместо имени: MAAAAH; либо
во время игры нажать одновременно BONK

ARMY MOVES

Ч.1-54597,0 ч.2-53772,0 либо 54598,0

ARTIC FOX

50242,0;49396,0 I

ASTEMEX

POKE 43518,34:POKE 43519,0:POKE 43520,0 либо
POKE 43516,0

ASTERIX & MAGIC CAULDRON

36276,0;37836,0;37837,5;37838,0

ASTRO BLASTER

POKE 27422,0:POKE 26396,255:POKE 23613,87:RAND USR 26368

ATHENA

50267,0;55268,61 ;51212,0

ATIC ATAC

POKE 36518,129 (36519,0?):POKE 36353,0:POKE 37260,175
Сначала введите POKE 23756,1 <ENTER>, затем LIST - и вводите POKE

ATTACK OF TOMATOES

25323,0;49433,81;42160,201;37002,0

AUF WIEDERSEHEN MONTY

POKE 42160,201 :POKE 37002,0:RAND USR 32799, либо
RAND USR 32768, либо загрузив программу командой LOAD"":REM MONTY,
поднять предмет слева

AUTOMATIA

64589,183

AVALON

POKE 23782,2:POKE 23786,201 :POKE 23878,204:POKE 23879,227 либо
POKE 58316,201 :RAND USR 62283

AVENGER

POKE 55519,0:POKE 51934,201:POKE 51527,0:POKE 51528,0:
POKE 51529,0:RAND USR 18434, либо 27453,0

B

BACK TO SCOOL

POKE 32748,1:POKE 29135,45:POKE 29131,80

BALLBREAKER 2

38874,0;35938,0;35729,99

BARBARIAN

51005,255;27580,0

BARBARIAN 2

27680,0

BARBARIAN, BARBARIAN2

```
 20 CLEAR 25000:LOAD""SCREEN$:POKE 23739,111:LOAD""CODE
 30 POKE 23739,244:POKE 35372,195:POKE 35378,8:POKE 35374,64
 40 FOR A=16392 TO 16406:READ S:POKE A,S:NAXT A
 45 RAND USR 16392:RAND USR 35240
BARB1 50 DATA 62,191,219,254,230,1,202,159,138,58,216
 60 DATA 182,195,47,138
BARB2 50 DATA 62,191,219,254,230,1,202,159,138,58,14
 60 DATA 183,195,47,138
```

BASL

POKE 41296,0:POKE 41968,201

BATMAN

36798,0 либо POKE 54067,0:POKE 54832,201:POKE 54708,0:POKE 54719,195 либо
POKE 37512,166(16?):POKE 36798,0:POKE 39908,201 :POKE 37430,0: POKE 37521,62:
POKE 37522,12:RAND USR 25984

BATMAN 2

46520,119;46526,24

BATTLE ZONE

44641,0

BATTY

Последний (9) сегмент загрузить в COPY-COPY с адреса A, затем A+1333,0; либо 48437,183

B.C.BILL

POKE 47589,201:RAND USR 25856

BEACH HEAD

POKE 45465,0

BEYOND THE ICE PALACE

38279,0(38278,0?)

BIGGLES 1

Ввести в таблицу счета слово DADD

BIONIC COMMANDO

34690,0 либо 34247,0;34274,0

BIRDS AND BEES

POKE 37229,0:POKE 37852,255:RAND USR 32700

BLACK CRYSTAL

Пароли: 512661220
1126690200
1126671220
3126641220
2126671220
1126290200

BLACK HAWK

POKE 34695,183

BLACK LAMP

33609,127;34487,127

BLADE ALLEY

POKE 58201,0

BLADE WARRIOR

POKE 37161,0:POKE 39898,182:POKE 39490,182(60?)

BLIND ALLEY

25284,0

BLOOD AXE

Ввести третий блок с адреса 23552, затем 26582,0;27957,0

BLUE MAX

43983,195;43984,163;43985,167

BLUE THUNDER

48552,0

BMX SIMULATOR

Нажать TAEHC

BOBBY BEARING

28094,36 либо загрузив в COPY-COPY с адреса 23296 ввести 29750,182 (время);
32173,0 (конец игры) либо POKE 29688,175

BOING

36610,0

BOMB JACK

49984,0 либо

```
5  CLEAR 29877:LOAD""CODE:POKE 65274,71 :POKE 65236,70
10 POKE 65237,85:FOR F=65517 TO 65535:READ A:POKE F,A:NEXT F
20 DATA 62,0,50,88,191,33,8,252,17,240,255,1,241,140,237,184
30 DATA 195,75,193:RAND USR 65465
```

BOMB JACK 2

Загрузив в COPY-COPY с адреса 23296, ввести 25379,0;31060,0

BOMBSCARE

POKE 56256,0:POKE 23606,46:POKE 23607,181

BOOTY

POKE 58294,0:RAND USR 52500

BOSCONIAN

POKE 33066,255

BOULDER DASH

36610,0 либо POKE 31007,0:POKE 31008,0:POKE 31009,0

BOULDER DASH 2

26028,0;26029,0;26030,0 (время)
31480,0;31481,0;31482,0 (бессмертие)

BOULDER DASH 3

26011,0 (время);26012,0;26013,0;26030,0;31480,0;31481,0;31482,0 (бессмертие) либо
45460,0;45461,0;45462,0;45464,0;40465,0;40466,0 (введя в COPY-COPY с адреса 24064)

BOULDER DASH 4

POKE 30960.N

BOUNDER

36610,0

BOUNTYBOB

50955,X (X-число жизней). Введя основной блок с адреса 24800 в COPY-COPY

BRUCE LEE

51795,0;51803,0

BUBBLE

57517,0

BUBBLE BUBBLE

43871,52

BUG EYES

43393,0 либо POKE 36003,201 :RAND USR 36000:POKE 43993,0:RAND USR 42200

BUGGY BOY

39086,0

C

CALL ME PSYCHO

48050,0

CAMELOT WARRIOR

POKE 53929,0:POKE 55918,201

CAPITAN KELLY

POKE 42982,0 (энергия) POKE 47145,0 (кислород) POKE 42517,0 (оружие)

CAR'S WAR

32337,0

CATCH 23

46813,0;61635,0

CAULDRON

POKE 40060,0;POKE 40061,0 либо 60060,0 либо

```
10 CLEAR 24599:FOR F=23296 TO 23309:READ A:POKE F,A:NEXT F
20 LET L=USER 23296:POKE 40060,0:LET L=USR 24600
30 DATA 221,33,24,96,17,232,159,62,255,55,205,86,5,201
```

CAULDRON2

POKE 52133,0 либо 52974,0 (для блока длиной 41986) либо
POKE 52718,0 (для блока длиной 41447)

CAVELON

POKE 47250,0 либо 47250,0

CAVERN FIGHTER

POKE 31683,0:POKE 31684,0

CHASE H.Q.

Переопределив клавиши управления на SHOCKED и нажав 1, получим разрешение загрузить 2 уровень, нажав 2 - 3-й уровень и т.д.

CHEEKAH'S EXPLOITS

28418,0

CHICAGO POKE

60264,0

CHILLER

34025,0

CHIMERA

POKE 23794,0:POKE 23799,0:POKE 23804,0:POKE 23805,0:POKE 23810,0: POKE 23815,0

CHRONOS

53407,201 (загрузив последний блок с адреса 23296 в COPY-COPY) либо в таблице счета наберите: JING IT BABY

CHUCKIE EGG

42873,0 либо 42508,3

CHUCKIE EGG 2

POKE 35453,0

CLIFF HANGER

POKE 25892,0

COBRA

POKE 36515,183(0?) (186?) либо 36518,0 либо 36491,24 либо 43647,255

COBRA FORCE

Переопределите управление на SIMON

COMBAT LINX

POKE 42525,0:POKE 42526,0:POKE 42527,0

COMMANDO

POKE 27652,175:POKE 27653,254:POKE 27654,0(10?) :POKE 27655,0: POKE 27656,0:
POKE 27657,0

CONQUEST

Загрузив в COPY-COPY с адреса 24000, ввести 38003,201 (основной сегмент)

COOKIE

POKE 35730,52 либо POKE 28698,0(28697?)

COSMIC CRUISER

25373,0 либо POKE 28567,0:POKE 28523,0:POKE 28560,0

COSMIC KANGA

29944,N (0 < N < 255) ;36212,0

CRITICAL MASS

56879,0(52?)

CRYSTAL CASTLES

63733,182

CURSE OF SHERWOOD

63033,0;64613,0

CYBERNOID

24917,255 либо POKE 39402,0 либо POKE 39403,0 либо 36678,0:31818,0; 39422,0;39423,0 либо нажать YXES

CYBERNOID 2

32202,0 либо POKE 36197,0 либо 36216,0;36217,0;36218,0 либо нажать ORGY

CYBERUN

POKE 36168,175 (для микродрайверной версии)

CYCLONE

38291,50;63892,160;63893,146 либо 37536,0; 16547,24;33429,0

D

DAN DARE

23974,168 либо для микродрайверной версии: POKE 36168,175: POKE 45954,104 либо
POKE 46888,201 :POKE 43256,0:POKE 46887,0: POKE 44413,201 :POKE 47710,201

DAN DARE 2

23453,237;23450,212 либо POKE 45891,0 либо 53822,255 либо 45891,0 либо POKE 60605,52: POKE 54390,0:POKE 56461,195:POKE 56462,101: POKE 56463,220:RAND USR 52390

DAN DARE 2/1

POKE 46459,0

DAN DARE 2/2

POKE 51797,0

DANDY

POKE 62004,201 ;POKE 49661,36:RAND USR 24100

DANGEROUS GARDENS

30361,201

DARK FUSION

50172,N;50407,0;50408,0

DEATHCHASE

26463,0

DEATH WISH 3

POKE 38768,183:POKE 39353,183:POKE 43301,183

DEFENDA FALL GUY

37530,52;37283,0;34164,0;27235,0 либо:

```
10 CLEAR 24100:LOAD""CODE:RAND USR 65100
20 LOAD""CODE:POKE 44204,0:RAND USR 41200
```

DEFENDER

30822,255;37815,255

DEFLEKTOR

34473,8:41784,0:42707,201

DIE ALIEN SLIME

POKE 32855,24:POKE 33227,195

DIGGER DAN

25559,0;26363,0

DIZZY

54216,0

DIZZY 3

POKE 30273,182

DIZZY 5

POKE 41815,0:POKE 51291,0

DONKEY KONG

35370,0

DOUBLE DRAGON

POKE 37693,0:POKE 37815,0:POKE 37813,0:POKE 37794,0

DOWN TO EARTH

POKE 40142,195

DRACONUS

62866,0;64215,0

DRAGON NINJA

41770,0 либо POKE 43455,8:POKE 38918,0:POKE 38684,1

DRAGON'S LAIR

51867,0

DRAGONTORC

POKE 58309,201:RAND USR 24700

DRILLER

48246,0

DRUID

24890,201

DRUID 2

30012,58

DUET

POKE 48270,57:POKE 48262,57:RAND USR 5E4

DUKES OF HAZARD

POKE 44246,0

DUN DARACH

34999,255 либо POKE 34378,24:POKE 34032,24:POKE 43333,201 :RAND USR 24100

DYNAMITE DAN

52678,0;57035,0;58770,201 ;59093,201

DYNAMITE DAN 2

POKE 33097,62:POKE 33098,192:POKE 33099,50:POKE 33100,217: POKE 33101,91:
POKE 33102,33:POKE 33103,16:POKE 33104,4: POKE 33105,0:POKE 34032,24:POKE 43333,201: POKE 33106,0: POKE 40077,201 либо
34032,24;43333,201 ;29003,24

DYNAMITE DUX

POKE 44277,0:POKE 44401,0

E

ELEVATOR ACTION

POKE 43820,0:RAND USR 39992

ELITE

POKE 46848,201 (45848?) (для микродрайверной версии)

EMPIRE STR.BACK

В режиме меню нажать одновременно ZXCV<Caps Lock>

ENDURO RACE

43542,0;43643,0

EQUINOX

POKE 41914,0 либо POKE 39858,52:POKE 39859,182 (жизни):
POKE 48691,0:POKE 48762,0(48752?) (горючее и лазер):
POKE 34022,0:POKE 34023,0:POKE 34024,0 (остановит время)

ERIC AND THE FLOAT

POKE 33245,0

ESKIMO EDDIE

24686,24;24687,76

EVERYONE'S A WALLY

28215,0 либо
POKE 58217,0:POKE 58218,0:POKE 58219,0:RAND USR 33156

EXOLON

Переопредели клавиши управления: ZORBA; либо загрузи в COPY-COPY с адреса 28000 основной файл (длиной 37536) и введи 40221,0 (бессмертие); 33646,0 (патроны) ;37456,0 (гранаты)

F

FAIRLIGHT

51893,0(61893?);58813,62;58814,6;62797,24;63478,24

FAIRLIGHT 2

30429,0;32027,24;32341,0(31341,0?)

FALL GUY

POKE 27235,0:POKE 44204,0

FANTASTIC VOYAGE

POKE 54227,0:POKE 54492,0:RAND USR 53248

FARENGEIT 3000

POKE 30818,0:RAND USR 29924

FAT WORM

30624,201

FINDERS KEEPERS

POKE 33969,0:POKE 34252,0:POKE 30394,N

FIREBIRDS

POKE 27235,0(27233?)

FIREFLY

44997,255;44998,255

FIRELORD

34509,0 либо POKE 39974,0:POKE 39975,195 либо:

```
10 CLEAR 65535
20 LET AA=USR "A"
30 READ N:IF N=999 THEN GO TO 70
50 POKE AA,N:LET AA=AA+1 :GO TO 30
70 RAND USR 65368
80 DATA 62,255,55,221,33,39,244,17,125,2,205,86,5,48,243,62
90 DATA 255,55,221,33,0,64,17,87,191,205,86,5,175,50,205,54
100 DATA 175,50,205,125,135,175,50,170,150,62,7,50,156,150
110 DATA 62,58,50,168,136,175,50,38,156,62,58,50,67,156,195
120 DATA 79,94,999
1987 SAVE CHR$ 22 + CHR$ 1 + CHR$ 0 + "LOAD" + CHR$ 6;LINE 0
```

FIST 2

POKE 27064,243:POKE 27065,243:POKE 27233,255:RAND USR 61710

```
1 CLEAR 65339:LET A=65400:READ N
2 IF N>256 THEN RAND USR 65400
3 POKE A,N:LET A=A+1 :GO TO 2
4 DATA 221,33,0,64,17,98,189,62,255,55,205,86,5,33,0,0,34
5 DATA 175,105,175,60,177,105,50,181,105,62,255,50,97,106
6 DATA 62,243,50,182,105,62,243,50,183,105,195,14,241,999
```

FLYING SHARK

54462,201;54379,3 либо загрузив 3 и 4 блоки, в COPY-COPY с адресов 25000 и 45000 (без заголовков): 43006,62;43007,2;43008,167;48980,0;48981,0;48982,0;48630,0;48631,0;48632,0;48605,0;
48606,0;48607,0;47035,0;42464,N

FRANK'N STEIN

POKE 28287,255 либо POKE 28277,0:POKE 28278,0:POKE 34124,0:POKE 33723,0

FRANK'N STEIN 2000

POKE 36047,0:POKE 36048,0:POKE 36049,0:RAND USR 36000

FRED

POKE 31171,0 либо POKE 31171,0:POKE 37729,0:RAND USR 30285

FREDDY HARDEST

64011,24

FREDDY HARDEST 2

61607,183

FREEZ BEES

POKE 34610,0

FROST BYTE

POKE 33805,182:POKE 36560,182:POKE 36687,0 либо
POKE 30992,183 (жизнь)
POKE 28237,183 (время)
POKE 33052,0 (жизнь)
POKE 33805,0 (время)

FULL THROTTLE

46608,0;46609,0;46610,0

FUTURE KNIGHT

31683,0

FUTURE PROJECT

POKE 27662,0 либо POKE 29332,0

G

GAME OVER 1

Код: 10218
31870,X (попытки);31880,Y (гранаты) либо POKE 39334,0 (попытки):
POKE 32417,0 (гранаты)

GAME OVER 2

Код: 18024 POKE 38704,0:POKE 32388,0

GARFIELD

33595,0

GAUNTLET

```
1 CLEAR 64999:LET T=0:LET W=0:FOR F=65000 TO 65032:READ A
10 POKE F,A:LET T=T+A*W:LET W=W+1 :NEXT F
20 IF T<>64702 THEN PRINT "ERROR IN DATA":STOP
30 DATA 221,33,218,254,17,81,1,62,255,55,205,86,5,48,241,33
40 DATA 1,254,34,57,255,243,195,0,255,62,201,50,82,184,195
50 DATA 0,132
60 PRINT AT 10,5;"START 'GAUNTLET' TAPE":RAND USR 65000
```

GEMINI WINGS

40076,0;41740,24;пароли уровней:

```
1. THESTART;
2. EYEPLANT;
3. WHATWALL;
4. GOODNITE;
5. SKULLDUG;
6.BIGMOUTH;
7. CREEPISH;
8. FINALFX
```

GHOSTBASTERS

Номера счетов

```
HERBIE:     05250624
PETER:      50338
HL:         70204700
ICH:        12345678
```

либо:

```
TANG BILLY: 15570011
```

GHOSTS'N'GOBLINS

33352,201 либо POKE 39857,135:POKE 39858,50:POKE 39859,180:
POKE 36057,36:POKE 33433,0:POKE 36083,0:POKE 39860,191 либо

```
10 CLEAR 24791 :LOAD""CODE:LOAD""CODE:RAND USR 24830
```

```
20 LOAD""CODE 16464:POKE 36058,0:POKE 36059,0
30 POKE 36060,0:POKE 18345.X-1 :RAND USR 24833
```

GIANT'S REVENG

POKE 24504,8(0?)

GILLIGAN'S GOLD

POKE 52881,0:POKE 52882,0:POKE 52883,0:POKE 52896,0: POKE 38935,18:POKE 38941,18:POKE 38939,0 либо POKE 52353,0:POKE 52354,0 либо
POKE 52436,0:POKE 52437,0:POKE 55219,0 либо
POKE 52786,0:POKE 38939,0:POKE 38935,18:POKE 38941,18: RAND USR 53379

GLIDER RIDER

34973,0 (неуязвимость)
37461,0;37462,0;37463,0 (ровный полет)
37439,0;37440,0 (выключает лазер)
34829,0 (замедляет время)
34818,0 (останавливает время)
34931,0 (бомбы не кончаются)

GLUG-GLUG

POKE 34139,0:POKE 34176,0:RAND USR 34349

GONZZALEZ

POKE 37085,0

GONZZALEZ 2

POKE 35749,0(жизнь):POKE 48058,0 (оружие)

GOONIES

33400,183;33409,0 либо 33247,N (1< N < 255)

GOTCHA

POKE 55926,0

GOTHIK

POKE 43934,58:POKE 42110,58

GO TO HELL

63275,5;63254,0 (загрузив в COPY-COPY с адреса 24296)

GREAT ESCAPE

41953,183;51243,201 ;47044,0;47045,0

GREEN BERET

42076,0 либо
46828,0;46829,0;46830,0 либо POKE 40074,0: POKE 40075,0:POKE 40076,0:POKE 40077,0 либо 40837,0;40844,0; 40845,0;40846,0;40850,251 ;40851,3;40852,208;40853,60;40854,50; 40855,227;40856,255 либо POKE 46827,183:POKE 46509,183: POKE 41651,0:POKE 41652,0:POKE 41653,0:POKE41654,0

GREET GURIANOS

34962,0

GROUND ATTACK

24872,0 либо POKE 29063,0

GRYZOR

33015,255;35477,255

GUNFIGHTER

65535,0

GUNFRIGHT

POKE 49233,54:POKE 49234,1:POKE 49235,0:POKE 49236,0 либо
POKE 47919,0:POKE 47920,0:RAND USR 23424 либо
POKE 23446,33: POKE 42355,0:POKE 64244,0:POKE 48484,0:PRINT USR 23424: POKE 43163,255:RAND USR 23446

GUNRUNNER

Для фирменной версии:

```
1 BORDER 0:POKE 23693,0:POKE 23624,0
2 CLEAR 25317:LOAD""CODE:POKE 64531,68:RAND USR 65412:POKE 65120,I
10 FOR I=23308 TO 23323:READ A:POKE I,A:NEXT I
15 RAND USR 65082
20 DATA 175,50,19,192,50,132,190,50,169,192,50,69,205,195,198,187
```

Для вскрытых версий:
POKE 49171,0:POKE 48772,0:POKE 49321,0:POKE 52549,0: RAND USR 48070

GYROSCOPE

53887,201 ;59149,0;54754,200

H

HADES NEBULA

POKE 49883,0

HALLS OF THINGS

POKE 32717,0:POKE 35923,255:RAND USR 24576

HAMMERFIST

Пауза - YCY; затем L - переход в следующий экран; бессмертие - NYC

HEAD OVER HEELS

38752,0;38473,0 либо 40756,0;40757,0;43132,0;42195,0; 35315,0 (записав в COPY-COPY с адреса 39999 сегмент N4) либо
POKE 41841,0:POKE 41842,0:POKE 41843,34(0?): POKE 41844,25(0?): POKE 41851,33:
POKE 41848,33:POKE 42185,0 либо35313,61;35314,39;35315,119;35316,167; 35317,200

HEARTLAND

41280,0;23563,201

HEAVY ON THE MAGIC

POKE 33102,0:POKE 33103,0:POKE 33201,48 либо POKE 33130,201 :POKE 33189,201:POKE 40207,24:POKE 33240,201: RAND USR 18434

HEIST 2012

36106,12(0?) ;36190,12(0?)

HELICHOPPER

Пароли: PASSWORDS
SHOW
FOREWER
RESTART
CLEAR

HERBERTS DUMMY RUN

POKE 51925,48 либо введите этот загрузчик. По его команде включите игру с начала

```
10 CLEAR 65535:FOR F=23296 TO 23321 :READ B:POKE F,B:NEXT F
30 DATA 221,33,0,0,17,17,0,175,55,205,86,5,221,33,224
40 DATA 252,17,63,2,62,255,55,205,86,5,201
50 PRINT AT 0,0;"INSERT TAPE NOW FROM START"
60 RAND USR 23296
70 POKE 65093,243:POKE65100,190
80 FOR F=65271 TO 65299:READ B:POKE F,B:NEXT F
90 DATA 17,128,91,33,254,83,26,174,203,164,174,203,228,18,43
100 DATA 19,123,254,154,32,241,62,48,50,213,202,195,148,91
110 RAND USR 65093
```

HIGHWAY ENCAUNTER

40905,0;40772,195;40773,123;40774,0

HOPPI'N'MAD

41968,0

HORACE AND SPIDERS

POKE 27680,0

HORACE GOES SCIING

POKE 29270,0:POKE 30027,0:POKE 30644,0

HUMAN KILLING MASH

35068,0;35069,0;35070,0

HUNCH BACK

POKE 24760,55 либо POKE 26888,0

HUNGRY HORACE

POKE 26426,0

HYDROFOOL

POKE 25859,201:RAND USR 64071 либо
POKE 64068,185:POKE 64069,248:POKE 64070,255

HYSTERIA

POKE 44588,201 либо 44527,201

ICE ATTACK

```
20 CLEAR 24999:POKE 23800,195:RAND USR 23760:POKE 53111,60:RAND USR 23803
```

I'BALL

POKE 49165,0:POKE 49483,0

I'BALL2

43384,0;49483,0 либо 49165,0;49483,0

IKARI WARRIORS

40272,0

IMPACT

POKE 54500,183

IMPLOSION

33538,182 (COPY-COPY 25000- основной сегмент)

IMPOSSABALL

41158,0;37706,0;37539,0 либо 45792,0;42142,201 ;42260,24

IMPOSSAMOLE

52919,0;52920,0

INDIANA JONES

POKE 33948,0:LOAD""CODE 16468:RAND USR 24833

INDIANA JONES AND LAST CRUSADE

POKE 33310,X

INTO THE EAGLE'S NEST

POKE 36640,0:POKE 36641,0:POKE 40512,0:POKE 40513,0: POKE 41136,0

IRON SOLDIER

Нажать GAD

J

JACK AND BEANSTALK

POKE 56115,0:POKE 56116,0:POKE 56388,62:POKE 56389,27: POKE 56390,0:POKE 42404,255

JACK THE NIPPER

POKE 44278,58:POKE 44285,58(0?):LOAD""CODE 16464: RAND USR 24833 либо 43519,201 либо для перехода в режим "CHEAT" набрать ZAPIT

JACK THE NIPPER 2

POKE 43251,0:POKE 23739,244:RAND USR 34240 либо
POKE 34232,56:RAND USR 54476 либо
POKE 44278,58:POKE 44285,58:RAND USR 24833

JAILBREACK

50651,0

JASONS GAM

Нахоясь в меню, нажать SWA

JAWS

33904,182

JETMAN

36966,224(244?) ;36945,3 либо POKE 36965,0

JETPAC

POKE 25020,255(0?):POKE 26075,0:POKE 36966,24:POKE 25018,0

JET SET WILLI

POKE 35899,0 (бесконечная жизнь)
POKE 36353,44 (возможность лезть по левой стене)
POKE 34795,X (старт из комнаты 0 < X < 60)
POKE 36477,1 (не разбиваешься, упав)
POKE 34275,10 (ключ к любой комнате - 9+другая цифра)
POKE 36358,0 (высокие прыжки)
POKE 35123,0 (убирает подвижные препятствия)

POKE 41983,255 (взяв пробку от ванной, вскочите на кровать и посмотрите, что получится)
POKE 59900,255 (преодоление карнизов)
RAND USR 33792
либо POKE 35899,0:POKE 34497,202:POKE 34498,135
либо введите в таблице счета: WRITETYPER

JET SET WILLI 2

POKE 30019,255:POKE 30027,32:POKE 34686,150 либо

```
10 CLEAR 64999:LET OBJ=150:LET ROOM=32
20 FOR N=65000 TO 65047:READ A:POKE N,A: NEXT A
30 PAPER 0:INK 0:BORDER 0:CLS:RAND USR 65000
40 DATA 221,33,0,64,17,56,185,62,255,55,205,86,5,243
50 DATA 48,240,33,6,254,17,197,100,1,59,0,237,176,195
60 DATA 0,95,62,255,50,67,117,62,OBJ,50,126,135,62
70 DATA ROOM.50,75,117,195,0,112
```

JOE BLADE

65032,50;65029,50

JUDGE DREDD

POKE 24936,24:RAND USR 24736

JUMPING JACK

30094,182 либо 30125,0;30093,0;30094,0;30095,0;30095,0

K

KAI TEMPLE

47783,0;47824,0

KARNOV

32972,0;32855,255 либо POKE 25620,0

KILLER RING

33636,0(33636?)

KING KONG 2

42523,0:43100,0:44111,0

KIREL

59322,154:35932,0

KNIGHT LORE

49629,175:53567,175:44947,50206,0
либо POKE 50205,0:POKE 50206,0:POKE 50207,0:POKE 53567,0

KNIGHT MARE

38686,16:38693,16

KNIGHT TIME

POKE 27813,0:POKE 32007,183 либо 24584,255;24585,255; 45322,255;45323,255;41456,0;41457,0

KOKOTONY WILF

42214,255 либо POKE 28929,9:POKE 28934,8:POKE 28939,8: RAND USR 41712
либо POKE 43742,0

KOSMIC KANGARO

POKE 36212,0

KOSMOS

В меню нажми <SIMBOL SHIFT>+K

KRAKOUT

POKE 45565,0:POKE 61534,0

KRION

POKE 45004,0(жизнь): POKE 44486,0 (горючее)

L

LAP OF THE GODS

POKE 47039,201 :RAND USR 47000:POKE 53790,201:RAND USR 57680

LASER WHEEL

32849,0(39249,0?)

LAST DUEL

POKE 37605,0:POKE 40063,0

LASTNINJA

POKE 36576,198:POKE 35993,198:POKE 36751,198

LASTNINJA 2

POKE 36578,175:POKE 36579,60(175?):POKE 41855,0 либо 36578,0;36579,0

LAZY JONES

POKE 56693,0(255?)

LEGEND OF AMAZON

57960,0 либо 57590,183

LEGEND OF KAGE

30609,255 либо POKE 37065,0:POKE 63427,240:POKE 63428,210:
POKE 54000,62:POKE 54001,0:POKE 54002,50:POCE 54003,57:POKE 54004,144
POKE 54005,195:POKE 54006,33:POKE 54007,249:
RAND USR 18434

LIGHT FORSE

POKE 40721,201 :POKE 40725,0:LOAD""CODE 16464:
RAND USR 24833 либо
POKE 39774,N(1<N<255) либо POKE 39764,0

LIVING DAIUGHTS

38913,201(206?)

LODE RUNNER

POKE 35427,24:POKE 35426,0

LUNAR JETMAN

POKE 36965,0:POKE 37999,201
либо POKE 23439,201 :POKE 36965,0(36936?)
либо 36966,224;36945,3

M

MAGLAXIANS

59354,7

MAG MAX

POKE 58472,12(60?) (0?) либо вставить строку:

```
20 CLEAR 24999:POKE 23800,195:RAND USR 23760:POKE 58472,0: RAND USR 23803
```

MANIC MINER

POKE 35136,0:POKE 36160,0:POKE 36106,0:POKE 36106,0 RAND USR 33792 либо
POKE 34798,0:POKE 34799,0:POKE 34800,0:POKE 35136,0 код телепортации 6031769 (набрать на первом экране) либо введите имя: TYPEWRITER

MARBLE MADNES

39579,0(38579?)

MARIO BROSS

44079,0

MARSPORT

POKE 34379,24:POKE 34057,201:POKE 43865,24:POKE 36607,24: POKE 36587,24:
POKE 44955,24

MASK 2

41560,0;41561,0;44944,0;45325,0;45326,0;43122,0 (загрузив с адреса 25088 в COPY-COPY)

MASK 3

46045,58:47778,58;49872,58;45589,201

MASTERS OF UNIVERS

42173,0;51406,0

MEGANOVA POKE

32382,0: пароли: 26719 ; 16640

METAL ARMY

POKE 48435,0:POKE 48559,0: (энергия)
POKE 42198,0:POKE 48700,107 (жизнь)
либо POKE 36897,0: (жизнь) POKE 42650,0 (энергия)

METROCROSS

43006,195;44490,0

MICKEY MOUSE
40873,0;40012,0;40091;40114,0

MISS PUCKMAN
52887,0

MONTY MOLE
38004,0 либо POKE 63467,X:FOR I=53981 TO 53904:POKEI,0: NEXT I (0<X<255)

MOON ALERT
POKE 42404,255 либо 39754,0;42654,195;37035,201

MOONLIGHT MADNESS
POKE 59945,X (1<X<255):POKE 57747,0:POKE 57145,l67:
POKE 57833,0:57834,0:POKE 57835,0
падая нажмите пару раз L

MOTOS
POKE 42241,0

MOVIE
64275,195 (загрузив третий блок с адреса 23532 в COPY-COPY)

MR. FREEZE
POKE 33823,0;POKE 33824,0

MR. WIMPY
POKE 33693,0:POKE 33501,0 либо 43012,0;42908,0;27504,62;27505,10;27506,50;27507,226;
27508,105;27509,0;27511,224;27512,105

MUTANT MONTY
POKE 54933,0 либо POKE 54959,0 либо POKE 55761,60:POKE 56483,183

N

NAVY MOVES
POKE 49962,0

NAVY MOVES 2
POKE 54047,0:POKE 55805,0:POKE 55837,0

NEBULUS
32921,0 либо POKE 32913,0:POKE 43650,0:POKE 43332,X: RAND USR 24831

NEMESIS
POKE 51949,0

NEMESIS THE WARL.
POKE 31851,0:POKE 31858,0:POKE 51949,0

NETHER EARTH
POKE 44399,18:POKE 44615,33 либо
POKE 51841,0:POKE51842,0:POKE51843,0:POKE 51844,0: RAND USR42496

```
10 FOR I=16384 TO 16411 :READ A:POKE I,A:NEXT I
20 RAND USR 16384:DATA 49,255,87,221,33,0,91,17,0,165,62,255
30 DATA 55,205,85,5,33,0,0,34,129,202,34,131,202
40 DATA 195,0,166
```

NEXOR
POKE 36212,201 :RAND USR 30720

NIGHT GUNNER
24763,182

NIGHTMARE RALLY
Нажать <SS>+Q или <SS>+W

NIGHTSHADE
POKE 58056,0:POKE 57499,0:POKE 53442,0: POKE 53443,12:POKE 51105,0

NODES OF YESOD
32661,0

N.O.M.A.D
POKE 40167,0:POKE 34569,201

NONTERRAQUEOUS
30596,24;36047,195;36222,0;36464,24;25699,24;25646,195

NORTHSTAR
POKE 44433,0(255) (устраняет врагов), либо POKE 48369,0:POKE 48370,0:RAND USR 47039

NOSFERATU
32499,0:39791,0(201?) либо, загрузив в COPY-COPY с адреса 25000 третий сегмент, ввести 36792,201

O

OCTAN
POKE 58309,0
ODISSEY
1 25300,0
OLLI & LISA
36060,201;36710,24; 35748,201
OPERATION GUNSHIP
56304,0 (бомбы)
55976,0 (ракеты)
57907,201;53912,201;57914,0;57915,0;57916,0;57901,0; 57902,0;57903,0 (бессмертие), либо
53926,0;57880,201;55988,0;55976,0;56304,0
OPERATION WOLF
40734,195 либо 40691,0; 40710,0
ORBIX
65529,191 ;32127,0;32188,0
ORION
POKE 37319,201
OTHER POKES
POKE 39754,0;POKE 37035,0:POKE 42654,195
OUTRUN
39204,0
OVERLANDER
29521,0 либо 27320,0;27296,0;61422,0

P

PACKLAND
35141,0
PANAMA JOE
38633,183
PANZADROME
POKE 25657,1:POKE 30410,0:POKE 32597,0:POKE 28658,0
POKE 28854,0(28864?)
PAPER BOY
POKE 48023,201 (для микродрайверного варианта)
PENETRATOR
49917,0;50751,0
PENTAGRAM
POKE 49917,0(9?) (201?):POKE 50751,0
PETER PACK-RAT
POKE 27243,100(0?)
PHANTIS 2
Код доступа 18757
PHANTOMAS 2
POKE 26606,0:POKE 28452,0
PHANTOM CLUB
POKE 56486,0
PHEMIX
29375,0
PHOENIX
POKE 32232,N(0<N<10)
PIBALLED
POKE 46441,0:POKE 44416,5:POKE 46457,0
PINBALL
31566,0

PINBALL WIZARD

POKE 48182,0:POKE 49054,0:POKE 45566,X (0<=X<=14)

PIPLINE 2

32511,18 (загрузив основной блок с адреса 23738 в COPY-COPY)

PIPEMANIA

Уровень	Пароль
5	DISK
9	NAIL
13	ONSE
17	ROPE
21	PENS
25	SLIP
29	EACH
33	RISE

PIRACURSE

56278,0

PI*R2**

38752,0;38173,201 (38473?)

POGO

POKE 44260,33:POKE 44259,0

POPEYE

26095,N (1<N<255) ;26242,0

PROFANATION

44015,201

PROHIBITION

POKE 25421,182

PROJECT FUTURE

POKE 27662,0 либо POKE 29332,0

PSSST

POKE 24984,0(24985?)

PSYHOSOLDIER

40123,0 либо 33372,62;33373,4;33374,0

PSYTRAXX

48930,N(1<N<255)

PSYTRON

28625,0;26143,255;26144,0;41100,1

PUD PUD

49287,0

PUNCHY

45632,0

PYJAMARAMA

48670,0;43883,2 либо 48683,0;33764,201

PYRACURSE

POKE 56278,0 либо

POKE 23325,201:RAND USR 23299:POKE 33466,201 (33446?):RAND USR 29600

PYRAMANIA

30254,0;30255,0;30256,0;30266,0;30267,0;30268,0;30278,0;30279,0;30280,0

PYRAMID

POKE 44685,0(44865?)

Q

QUAZATRON

Сегменты без заголовков грузятся по адресу 38403 в COPY-COPY

58243,0;58244,0;58245,0;58267,0;58268,0;58269,0

R

RAID OVER MOSCOW

40229,0;40300,195;43364,0;43365,0;49130,0;49136,0

RAMBO

27401,52;60263,0 либо 38841,24;27401,0(52?) ;27402,201

RAMBO 2

27145,0;27146,0

RANARAMA

```
20 CLEAR 24999:POKE 23800,195:RAND USR 23760:POKE 57421,0:POKE 57436,205:
```

POKE 57572,201 :POKE 59821,0:POKE 59836,0:RAND USR 23803

RASTAN

48909,255

RASTAN SAGA

55629,182

REBEL SQUAD

46840,255

REBEL STAR 2

POKE 28599,N

RED HEAT

Нажать <SYMBOL SHIFT>+все цифровые клавиши

RED STORM

POKE 37337,202

RENEGADE

41048,185(44048?) либо

```
100 MERGE "":POKE 23800,201 :CLEAR 24899:RAND USR 23760
110 POKE 23800,16(6?):POKE 41047,182:POKE 30301,195
120 RAND USR 23800
```

RENEGADE 3

В меню одновременно нажать Q и T

REVOLUTION

Загрузить с адреса 23296 в COPY-COPY 35653,183;47111,0;34831,X; (X-уровень 0...7)

REX1

POKE 39402,0:POKE 40057,0

RICK DANGEROUS

POKE 58356,0:POKE 64075,0:POKE 64166,0:RAND USR 17589
либо POKE 55480,0: (жизнь) POKE 61045,0: (гранаты)
POKE 60954,0 (патроны), либо 35376,0

RIVER RESQUE

33199,255;33426,0;33452,0;36193,N;36255,255
либо POKE 33420,0 (1 игрок), POKE 33542,0(33452?) (2 игрок)

ROAD RACER

POKE 27150,0

ROBIN OF THE WOOD

```
25 LOAD""CODE:POKE 48690,0:LET A=PEEK 23635
30 LET B=PEEK 23636:LET C=A+256*B+5:RAND USR C
```

либо 49911,0 (загрузив основной сегмент в COPY-COPY)

ROBOCOP

25917,0 бессмертие
25424,0 время
25795,0 без пауз
24039,0 TURBO

ROCKMAN

Пароли: ONIX; GURU; SAGE; CLAW

ROGUE TROPPER

30734,0 либо POKE 30924,0 либо 30874,0

ROLLER COASTER

38988,255 либо 36594,0;36595,0;36596,0

ROMMEL'S REVENGE

42976,0

R-TYPE
POKE 30873,0 либо 37374,0

RUFF & REDDY
POKE 33576,0

RYGAR
51216,0;61577,0

S

SABOTAGE
Пароли:
2) BUMBLE BEE 2
3) HONORARIUM 3
4) PHENOMENON 4
5) ONOMASTICS 5
6) SALMAGUNDI 6
7) PSEUDONIMOUS
8) ONOMATOPOEIA

SABOTEUR
POKE 29894,0:POKE 46998,0:POKE 47000,31 либо 47009,0;47010,0;47011,0

SABOTEUR 2
POKE 37121,0:POKE 37122,0:POKE 61338,172:POKE 61382,182: POKE 61362,0 либо
POKE 37122,0:POKE 61340,201 :RAND USR 25100

SABRE WULF
POKE 23756,1:CLEAR 65535:POKE 43575,255: (жизнь первого игрока)
POKE 45520,255: (жизнь второго игрока)
POKE 45559,X: (число попыток 0<X<255)
POKE 41725,255: (бесконечные призовые игры) RAND USR 23424
либо POKE 39702,30:POKE 43575,X либо POKE 44786,0

SAIGON COMBAT
POKE 63364,201:POKE 37421,1 (38421?)

SAIGON COMBAT 2
Пароль STARLIGHT

SAMANTA FOX STRIP POKER
POKE 23408,6 либо

```
10 LOAD""SCREEN$
20 FOR F-20728 TO 20735
30 READ A:POKE F,A:NEXT F
40 DATA 62,6,50,112,91,195,224,81
```

SAM SPADE
25215,0;26381,0

SAMURAI WARRIOR
Ввести "CHEAT" в таблице счета, либо:
33013,0;37866,0;37875,0

SANXION
POKE 36584,0 либо 36585,183 либо переопределить клавиши: CHEAT и вписать пароль: LYNN

SAVAGE
35454,X (попытки) либо 37557,0;39297,0;58584,0 либо 37443,0;37444,0;37445,0

SAVAGE 2
POKE 29177,60 либо 32668,195; пароль - SABATTA

SAVAGE3
POKE 44251,0 либо 57852,195; пароль – FERGUS

SCEPTRE OF BAGDAD
59858,0 либо POKE 58116,0

SCOOBY DOO
64027,86;64028,5;29614,0

SCRAMBLE
30192,201 ;28051,201

SCUBA DIVE

55711,X (попытки)

SEAS OF BLOOD

POKE 34242,195:POKE 34247,195:POKE 30241,201

SENTINELA

31684,0

SHAO LIN ROAD

44843,0;44844,0

SHINOBI

Переопределяя клавишу MAGIC нажать <CAPS SHIFT>+<ENTER>

SHIZOIDS

25102,0

SHORT FUZE

Пароли:
1) 000
2) 367
3) 157
4) 049
5) 281

SHORT CIRCUIT 2

33511,255;35485,0;34921,0

SHOWMAN

63197,0

SIDEARMS

29411,120

SIGMA 7

```
20 CLEAR 24999:POKE 23800,195:RAND USR 23760
30 POKE 34159,0:POKE 34164,0:POKE 60068,0:POKE 60073,0:POKE 60396,0:POKE
60401,0: RAND USR 23803
```

SIR LANSLOT

POKE 25726,0:POKE 25727,0 либо ввести блок без заголовка (длиной 9344) с адреса 33424 в COPY-COPY и затем:
33590,X (0<X<10)
33892,0;33893,0 либо 33892,24;33893,37 - бессмертие
33697,0;33698,0 - абсолютное бессмертие
33790,0;33791,0;33792,0;33690,195;33604,66;33605,66

SKATE CRAZY

46473,201 ;46646,126;46409,201

SKOOL DAZE

30464,201

SKYRANGER

Пароли:
1) ENTER
2) MAGIC
3) PILOT
4) STOMP
5) PARIS
6) EVENT
7) RECAP

SLAP FIGHT

```
10 CLEAR 24999:POKE 23743,VAL "80":LOAD""CODE VAL "5E4"
20 RAND USR 5E4:LET A=NOT PI:LET B=48872:LOAD""CODE
30 POKE B,A:POKE B+1 ,A:POKE B+2,A
40 POKE VAL "57115",VAL "201":POKE VAL "23743",VAL "88"
50 RAND USR VAL "64305"
```

либо 48872,0;48873,0;48874,0;57175,201

SMASHOUT
59500,0;59501,0;47004,0
SOLDIER OF FORTUN
44796,0
SOLDIER OF LIGHT
51119,0;50552,0;50035,255
SORCERY POKE
49823,0:POKE 40159,255
SOS
POKE 33951,0:POKE 33091,0:POKE 35238,0:POKE 34764,201: RAND USR 32768
SPACE HARRIER
POKE 41499,X:POKE 46543,195:POKE 46544,14: POKE 46545,182:POKE 46570,195:
POKE 46571,14: POKE 46572,182:RAND USR 18434
SPACE RIDERS
25962,0
SPACEWARS
26244,0;43364,182;46507,182;49130,182
SPACE ZOMBIES
29553,0
SPECTRAL INVADERS
POKE 25062,254
SPECTRAL PANIC
28522,0
SPECTRES
25680,183
SPELLBOUND
POKE 27038,N либо 32999,0;32996,0;32997,0;55121,0;55122,0;
55066,0;55070,0;55071,0;55072,0;27871,0;36133,0
SPINOIZZY
POKE 48272,201:POKE 48401,201 (для микродрайверного варианта), либо
FOR I=51398 TO 51404:POKE I,0:NEXT I
SPIRITS
POKE 51754,0
SPOOKED 89
POKE 60504,255
STAINLESS STEEL
POKE 46957,60 либо, нажав паузу, введите: ALIK или SLIK
STARCLASH
25381,183
STAR FARCE
Переопределите клавиши: TRONIC
STARION
POKE 46271,0:POKE 46272,0:POKE 46273,0
STARQUAKE
POKE 50274,0
STAR RUNNER
POKE 49560,183
STAR STRIKE
Нажав паузу, напечатать: I WANNA CHEAT. Чтобы выйти из этого режима, напечатайте: BORED
STAR STRIKE 2
Нажав паузу, набрать: HEAR AND OBEY
STAR WARS
POKE 45268,0
В коридоре, ведущем к реактору, нажмите одновременно 8, D, F, G, <SPACE>. Выйдите из коридора и нажмите Y, U, I, O, <SPACE>
STONKERS
24576,0

STOP THE EXPRESS

34464,183;34926,183;35257,183

STORMLOAD

58105,0;56877,0;34650,0 либо POKE 56877,201: POKE 56890,255 либо в таблице счета напечатать BRINGOTHEGIRLS, затем нажать цифру от 1 до 4 - попадешь на заданный уровень.

STRANGE LOOPE

POKE 63159,0:POKE 63161,0:POKE 63160,182

STREET GANG

POKE 39254,0:POKE 37127,39

STREET HASSLE

Нажать L для загрузки следующего уровня

STRIKE FORCE COBRA

48389,0;46499,0,46500,0;46501,0

STRIP DICE

27418,45;27656,45

STYX

31763,0;31764,0;31765,0;31766,0 либо POKE 29856,255

SUB NIGHTMARE

POKE 43435,0:RAND USR 3E4

SUPERHERO

POKE 49625,0:POKE 40444,0:FOR F-46334 TO 46337:POKE F,255: NEXT F:RAND USR 40341

SUPERTANK

37295,201;56813,201

SUPERTRUX

24498,134

SURVIVOR

Загрузить с адреса 23552 в COPY-COPY 37734,0;37735,0;36047,0; 36048,0

SWEEWO'S WORLD

POKE 33219,0:POKE 34912,0:POKE 35732,37:POKE 37008,255

T

TANK COMMAND

65535,0

TARGET RENEGATE

Набрать очков, погибнуть, в таблице счета вместо набора имени нажимать 2 любые клавиши до перекрытия счета на 2-3 знака. Затем игра продолжится с места Вашей гибели. Странный вид экрана нормализуется после перехода на другой экран.

TASK FORCE

Переопределите клавиши: CRASH

TECHNICIAN TED

44258,0

TERMINUS

Загрузить основной сегмент в COPY-COPY: 45583,0;47023,0;46970,201

TERRA CRESTA

POKE 35050,0:POKE 35051,0:POKE 35052,0 либо 37636,255;45282,0

TERRORDAKTIL

POKE 37629,0

THANATOS

```
10 CLEAR 24791:LET A=24:POKE 23570,16:LOAD""CODE
20 LOAD""CODE:RAND USR 24830:LOAD""CODE 16464
30 POKE 56058,201:POKE 56549,A:POKE 56816,A:POKE 58021,A
40 POKE 58220,A:POKE 58788,A:POKE 59174,A:POKE 59240,A
50 POKE 57603,A:POKE 57604,A:PRINT USR 24833
```

THEATRE EUROPE

На запрос "NUCLEAR CODE?" ответить: "MIDNIGHT SUN"

THOR

POKE 37550,195

THROUGH TRAP DOOR

```
1 CLEAR 24791:LOAD""CODE:LOAD""CODE:RAND USR 24830
2 POKE 44108,201 (часы для английской версии)
```

либо

```
2 POKE 41283,201 (часы для польской версии)
3 LOAD""CODE 16464:RAND USR 24833
```

THR.TRAP DOOR 2

47492,0

THUNDERBIRDS

POKE 58927,0:POKE 58928,0:POKE 58929,0:POKE 58930,0: POKE 62134,0:POKE 59119,201

THUNDERBIRDS 2

RECOVERY

THUNDERBIRDS 3

ALOISUS

THUNDERBIRDS 4

ANDERSON

THUNDERBIRDS 85

58927,0;58928,0;58929,0;58930,0

THANDERBLADE

POKE 33199,0:POKE 33145,7

TIGER ROAD

51489,195

TIME SCANNER

27448,0

TIR-NA-NOG

34202,255 либо POKE 31365,201:POKE 34751,201:POKE 33727,24: POKE 30801,195:
POKE 35421,24 либо, загрузив с адреса 24100 блок длиной 38674 в COPY-COPY, ввести: 33724,0;33725,0;27058,0;27069,0;31109,0;27057,0; 32403,0;26919,0

TLL

35006,0;33807,0

TOI ACID

Пароли: 517;124;500

TRANTOR

54236,0;56596,0; Коды телепортации:
KEMPSTON; JOYSTIK; SPECTRUM; SOFTWARE; KEYBOARD; COMPUTER; CASSETTE; SINCAIR; GRAPHICS; HARDWARE; TERMINAL; PRINTERS; CONTROLS; WARGAMES; WARRIORS; MEGA-GAME

TRANZ AM

POKE 25446,0

TRASHMAN

52037,0

TRAVEL W.T.

38656,183

TRIBBLE TRUBBLE

32110,0;32849,0;34214,0;35026,0;31981,0;31984,24

TRUENO 1

POKE 25100,N

TRUENO 2

POKE 24785,0:POKE 24984,0

TURMOIL

Загрузить с адреса 23296 в COPY-COPY; 57557,0

TUTANKHAMON

POKE 27783,0:POKE 27279,X (0 < X < 255)

U

U.C.M.

POKE 23296,255

UNDERWURLDE

POKE 59376,0:POKE 59380,0 либо
POKE 59376,0:POKE 38041,0:POKE 38042,0:POKE 59591,0:POKE 45019,201

UNTOUCHABLES

В таблице счета ввести HUMPHREY BOGART

URIDIUM

POKE 35403,34:POKE 38998,201:POKE 35427,195 либо

```
1 FOR A=23296 TO 23327:READ S:POKE A,S:NEXT A
2 RAND USR 23296
3 DATA 221,33,0,64,17,0,27,55,62,255,205,86,5,1,0,0
4 DATA 205,61,31,221,33,0,64,17,0,192,62,255,194,4,201
```

либо

```
1 FOR F=43773 TO 43814:READ A:POKE F,A:NEXT F
2 DATA 62,126,219,254,230,31,254,31,203,17,62,251,219,254,31
3 DATA 63,203,17,62,253,219,254,31,63,203,17,62,253,219,254
4 DATA 47,31,31,203,17,23,203,17,0,0,0,0
```

V

VAMPIRE

Нажать одновременно 1,2,3,4,5

VIEW TO A KILL

POKE 28032,255:POKE 28087,8:POKE 29243,8:POKE 30543,8

VIGILANT

POKE 48735,0:POKE 39921,255
либо в таблице счета набрать GREEN CRYSTALL

VINDICATORS

POKE 37913,0:POKE 38094,0

VIXEN

POKE 51794,175

VIXEN 3

POKE 41423,0:POKE 51794,174

W

WALLY

POKE 58215,182

WAR

POKE 38394,0:POKE 37033,0 (для микродрайверного варианта)

WAY OF EXPLOADING

POKE 27055,0:POKE 27056,0:POKE 27063,1 (255?);

WAY OF TIGER

POKE 45532,201:POKE 45178,201:POKE 45806,201:POKE 45554,183

WAY OF TIGER 3

45828,183

WEEK LE MANS

POKE 26110,34

WHEELE

Пароли:
1) WITTY
2) SHARK
3) BEBOP
4) XENON
5) ZX833
6) 2MQL3
7) HRME2

WHO DARE WINS 2

49743,255;49748,255 либо 51847,0;50833,0

WIZARD'S LAIR

25522,X (попытки)
Пароли:
1)HAWLO
2) CAIVE
3) VAULT
4)LIAYR
5) DUNGN
6) CRYPT
7) LYONS

WORSE THINGS...

33221,0:35303,0 либо 40133,0;42215,0

WRIGGLER

POKE 50173,0

X

XARQ

Введи XARQ

XECUTOR

47216,201:47320,201

XENON

Во время игры нажать <BREAK>, затем сразу 4 клавиши: TINY

XEVIOUS

POKE 53592,255 либо POKE 55333,62:POKE 55334,5: POKE 55335,50:POKE 55336,4:
POKE 55338,201:PRINT USR 24833

Z

ZAXXON

POKE 48825,X (попытки) либо POKE 50244,183
Для невидимости ввести "RED"

ZIP ZAP

POKE 54065,0:POKE 53382,N(1<=N<=99)

ZODIAC STRIP

Нажимайте J

ZUB

POKE 37473,201

ZUBEX

POKE 4522,0:POKE 45368,0

ZUNAPS

POKE 45592,24

ZUTHUM

54786,0;54790,0;54791,0;51270,0;51271,0 (загрузив третий блок без заголовка с адреса 24784 в COPY-COPY)

ZZOOM

POKE 24743,0:POKE 32692,0

***

1942

47001,255 или POKE 47007,255 (для микродрайверного варианта)
либо POKE 65368,198:POKE 65369.N (номер первого этапа);
POKE 65370,50:POKE 65371,9:POKE 65372,182:POKE 65373,201: POKE 52377,205:
POKE 52378,88:POKE 52379,255:RAND USR 24833

1985

49303,0

3D-SPACEWARS

26244,0;43364,182;46507,182;49130,182

3D-TUNNEL

29711,0