Krótka rozprawa z taśmą (cz. 1)

W listach od czytelników często pada pytanie: jak przerabiać gry, aby „chodziły" z dyskiem? Oczywiście nie ma na to prostej recepty, jednak szybko można sobie przyswoić zasady obowiązujące przy przerabianiu gier.

CZYM SIĘ BĘDZIEMY ZAJMOWAĆ?

Taśma magnetofonowa nie jest już uważana za „poważny" nośnik danych. Zalet szybkiej i niezawodnej pamięci dyskowej nie trzeba reklamować, zwłaszcza niecierpliwym graczom, którzy nie mogą się doczekać załadowania właśnie zdobytej, rewelacyjnej gry. Konieczność korzystania z magnetofonu do przechowywania programów może zniechęcić każdego. Definitywnym rozwiązaniem tego problemu jest zakup stacji dysków. Niestety, okazuje się, że autorzy programów (w tym także gier) nie przewidzieli, że akurat kupimy FDD 3000...

Celem niniejszego artykułu jest zaznajomienie czytelników z metodami „uzdatniania" gier dla systemu dyskowego FDD 3000 oraz z występującymi przy tym problemami. Rozpoczynając od sprawy tak prostej, jak „rozbrajanie" ładowaczy gier postaram się przedstawić najczęściej spotykane ich rodzaje, a następnie metody przenoszenia bloków kodu na dyskietkę. W kolejnych częściach artykułu omówione zostaną sposoby podmiany procedur (w kodzie gry) taśmowych na dyskowe, co daje pełną współpracę gry z dyskiem (ładowanie poziomów, zapamiętywanie bieżącego stanu gry itp.). Wszystkich czytelników uprzedzam, że do pełnego zrozumienia opisanych „zabiegów" wymagana jest znajomość języka maszynowego. Warto też przejrzeć cykl „TOS bez tajemnic", opublikowany w zeszłym roku.

NARZĘDZIA

Przed każdą poważną pracą zaczynamy od gromadzenia niezbędnych narzędzi. W naszym przypadku będą to oczywiście programy. Pierwszy z nich to **OPENER**. Program ten pomoże nam w oglądaniu taśmowych loaderów. Można również posłużyć się jakimś jego odpowiednikiem, który potrafi załadować z taśmy zabezpieczony wszelkimi znanymi sposobami „ładowacz". Do kopiowania plików z taśmy na dyskietkę warto używać programu **ZEBRA COPY**, lecz oczywiście można się bez niego obejść.

Rzeczą całkowicie niezbędną jest natomiast zestaw monitor-disasembler + asembler, na przykład **MONS** i **GENS**. Pierwszego z nich będziemy używać do analizowania kodu gry, drugiego — do wykonywania wszelkich zmian w tymże kodzie. Warto również mieć pod ręką ołówek i dużo, naprawdę dużo papieru...

DO DZIEŁA

Na samym początku należy przygotować dyskietkę. Najlepiej, jeżeli będzie ona zupełnie czysta — wszelkie zbędne pliki tylko się „plączą pod nogami" w trakcie pracy. Na tej dyskietce będziemy umieszczać kawałki przenoszonej gry (czy też innego programu). Bardzo wygodnie jest sobie przygotować drugą dyskietkę, na której znajdą się wszystkie wymienione programy narzędziowe. Następnie ładujemy do pamięci program OPENER (lub jego odpowiednik). Ustawiamy taśmę na początek gry i wgrywamy loader. Po pieczołowitym spisaniu parametrów nagłówka (najważniejszy jest numer linii startowej) listujemy loader na ekranie. I w tym momencie mamy dwie możliwości, gdyż występują dwa podstawowe rodzaje programów ładujących: składające się z instrukcji BASIC-a oraz napisane w kodzie maszynowym.

WARIANT PIERWSZY — PROSTSZY

Najprostszy jest ładowacz BASIC-owy. Może on wyglądać na przykład tak:

```
10 REM HACKER 007, 1990-02-10
20 PAPER NOT PI: INK NOT PI: BORDER NOT PI
30 CLEAR VAL "24499"
40 LOAD "AVENGER.1" CODE VAL "5E5": RANDOMIZE
USR VAL "5E5"
50 LOAD "AVENGER.2" CODE : LOAD "AVENGER.3" CODE
60 DRAW USR VAL "24500", USR VAL "61586"
```

Pierwsze dwie linie nie mają większego znaczenia. Rozkazy z linii 40 ładują i uruchamiają skompresowany obrazek. Linia 50 jest odpowiedzialna za załadowanie głównych bloków gry. Wreszcie linia 60 to dekompresja (USR 24500) i uruchomienie (USR 61586) gry. Zamiast swóch oddzielnych instrukcji RANDOMIZE wystąpiła komenda DRAW, gdyż ma ona dwa parametry, a funkcja USR nie jest zbyt wybredna, jeśli chodzi o poprzedzający ją rozkaz.

Zamiast ręcznie poprawiać loader (trzeba dopisać gwiazdki przed LOAD, czasami trzeba też dopisać odpowiednie nazwy plików w cudzysłowach bądź adresy ładowania), możemy w tym przypadku skorzystać z **ZEBRA COPY**. Loader zostanie zmodyfikowany, a kolejne bloki kodu przeniesione prawie automatycznie. Jednak warto gdzieś zapisać wszystkie dane, zawarte w nagłówkach plików. Ułatwi to „grzebanie" w grze, jeśli ma ona np. ładować z dysku dalsze poziomy gry.

WARIANT DRUGI — DUŻO, DUŻO TRUDNIEJSZY

Nasze zadanie jest trudniejsze, gdy cały (lub prawie cały) loader jest napisany w kodzie maszynowym. Ten przypadek można rozpoznać po tym, że wśród linii w BASIC-u nie występują instrukcje LOAD (lub jest ich mniej, niż bloków na taśmie). Ponadto występuje instrukcja RANDOMIZE USR adres. Adres ten musimy zapisać na boczku, a następnie przechodzimy do BASIC-a (komenda B — **OPENERA**) i nagrywamy loader na dysku (tak na zapas).

Tym razem nie obejdzie się bez programu monitora — disasemblera (np. **MONS**, **MAD**, **Z80**). Ładujemy go pod adres nie kolidujący z programem w BASIC-u ani ze stosem interpretera (UWAGA! Nie należy zmieniać położenia tego stosu instrukcją CLEAR, gdyż możemy w ten sposób skasować kawałek programu maszynowego, zawartego w obszarze zmiennych BASIC-a!).

Następnie oglądamy pamięć od adresu, który wcześniej zapisaliśmy na kartce. Bardzo często jest to liczba 23760. Pod tym adresem ujrzymy dłuższy lub krótszy program maszynowy. Bardzo często spotykany jest następujący programik, stosowany do ładowania gier „złamanych" za pomocą przerwania NMI:

```
LD   IX, 0000    ; parametry dla
LD   DE, 0011    ; wgrywania nagłówka
XOR  A
SCF
CALL 0556        ; ładuj nagłówek
LD   IX, 60D8    ; adres ładowania
LD   DE, 93EE    ; i długość pierwszego
LD   A,  FF      ; bloku - kodu gry
SCF
CALL 0556        ; ładuj kod gry
LD   IX, 0000
LD   DE, 0011
XOR  A
SCF
CALL 0556        ; ładuj drugi nagłówek
LD   IX, 4000    ; adres i długość
LD   DE, 045D    ; drugiego bloku
```

```
LD   A, FF      ; (skompresowanego obrazka)
CALL 0556       ; ładuj obrazek
CALL 60FE       ; rozpakuj grę i obrazek
LD   IX, 0000
LD   DE, 0011
XOR  A
SCF
CALL 0556       ; ładuj trzeci nagłówek
LD   IX, 4050   ; adres i długość trzeciego
LD   DE, 07B0   ; bloku - "uruchamiacza"
LD   A, FF
SCF
CALL 0556       ; ładuj trzeci blok
JP   6101       ; skocz do początku gry
```

Oczywiście, wszystkie liczby są zapisane szesnastkowo. Po zapisaniu adresów ładowania dla wszystkich (w tym wypadku trzech) bloków, tworzymy zupełnie nowy „ładowacz":

```
10 CLEAR 23791
20 LOAD*"NAZWA.1" CODE 23792
30 LOAD*"NAZWA.2" CODE 16384
40 RANDOMIZE USR 24830
50 LOAD*"NAZWA.3" CODE 16464
60 RANDOMIZE USR 24833
```

Następnie kopiujemy ZEBRĄ wszystkie bloki na dysk i uruchamiamy stworzony przez nas loader. Jeśli gra poprawnie się załaduje i uda nam się trochę pograć, to znaczy, że odnieśliśmy pełny sukces.

Może się zdarzyć, że po załadowaniu program się zawiesza lub resetuje się komputer. Prawdopodobnie oznacza to, że instrukcja CALL 60FE niszczy obszar BASIC-a (o ile nie został popełniony jakiś banalny błąd — radzę sprawdzić wszystko od początku). Dzieje się tak, gdy w oryginalnym ładowaczu występują rozkazy LDIR (tzn. procedura maszynowa sama się relokuje w „bezpieczny" obszar pamięci). W takim wypadku czeka nas napisanie loadera w kodzie maszynowym.

WARIANT OSTATNI — NAJTRUDNIEJSZY

Wpierw przypomijmy, jak wygląda procedura ładująca plik z dyskietki (zakładamy, że w buforze operacji dyskowych (#2000) znajduje się nazwa pliku zakończona znakiem CHR$ (0)):

PUSH IY; przechowanie IY
LD IY, 0
RST 8; włączenie ROM-u interfejsu
POP IY; odtworzenie IY
LD B, długość nazwy + 1
LD HL, 214Dh; *filetyp*
LD (HL), typ pliku; 3 dla typu Bytes
LD HL, długość pliku; (lub FFFh — wg nagłówka)
LD (214Eh), HL; *filelen*
LD HL, adres ładowania; (lub 0 — ładow. wg nagłówka)
LD (2150h), HL; *start*
CALL 0CC1h; *loadp*
JR NZ, obsługa błędów
CALL 0604h; powrót do ZX ROM-u
RET; koniec procedury

Ostatnie dwa rozkazy można zastąpić JP 0604h. Najprostsza procedura obsługi błędów może wyglądać np. tak:

LD HL, 210Dh; adres komunikatu o błędzie
CALL 03D8h; wydruk komunikatu na ekranie
; tu — oczekiwanie na klawisz lub tp.

Teraz możemy przystąpić do pisania loadera. Trzeba wybrać miejsce w pamięci, które nie zostanie zamazane w trakcie dekompresji gry (CALL 60FEh); tam umieścimy nasz program. Najprościej jest wzorując się na oryginalnym loaderze, wykonać „nakładkę" na procedurę wgrywania bloku z taśmy (listing 4 przedstawia takie rozwiązanie dla gry TETRIS). Jednak nie zawsze jest to możliwe — procedura do obsługi dysku jest przeważnie dłuższa od procedury dla taśmy. W takim wypadku należy napisać procedurę, która sama się przekopiuje w wolny obszar pamięci i tam się uruchomi.

Przypuśćmy, że takim obszarem jest bufor drukarki (tj. 23296 do 23551). Listing 5 przedstawia loader relokujący się do tego właśnie obszaru.

Zwróć uwagę, że konieczne jest przeliczenie niektórych adresów — np. zamiast:

LD HL, NAZWA

występuje

LD HL, NAZWA-START+23296

Wynika to właśnie z przemieszczenia programu pod inny adres.

Dla tej i poprzedniej wersji ładowacza trzeba jeszcze stworzyć część w BASIC-u, odpowiedzialną za uruchomienie procedury (i, niejako przy okazji, za ustawienie RAMTOP-u instrukcją CLEAR). Można również wyczyścić ekran, ustawić kolory lub coś napisać na ekranie. Po instrukcji REM w linii dziesiątej musi wystąpić pewna liczba spacji; jest ona równa długości naszej procedury maszynowej (może być też większa, lecz w żadnym wypadku mniejsza!). Linia dwudziesta ustawia stos interpretera poniżej wgrywanego bloku gry oraz uruchamia procedurę maszynową. Po wpisaniu linii w BASIC-u uruchamiamy **GENS** i wpisujemy część w asemblerze. Po kompilacji kod zostanie wpisany w linii dziesiątej za instrukcją REM.

Dlaczego procedura rozpoczyna się od adresu 23760? Otóż jest to adres początku programu w BASIC-u (23755) powiększony o pięć (dwa bajty dla numeru linii, następne dwa — dla długości linii i jeden dla instrukcji REM), czyli adres pierwszej spacji w linii 10.

Ci, którzy dokładnie śledzili cykl „TOS bez tajemnic" na pewno już zauważyli, że zamiast kopiować nazwy plików do bufora „na piechotę", prościej jest załadować do DE adres nazwy, do BC — jej długość i „skoczyć" pod adres 066Dh (oczywiście w ROM-ie interfejsu). Procerdura, która się tam znajduje, sama dopisze znacznik CHR$(0) na końcu oraz zwróci długość nazwy (ze znacznikiem) w rejestrach A i B.

Jeżeli dane w nagłówkach wszystkich plików są właściwe (tzn. pliki mają się ładować dokładnie pod adresy zawarte w swoich nagłówkach), to loader można znacznie skrócić. Wystarczy całą procedurę ładowania napisać w postaci pętli (powinna się ona wykonać tyle razy, ile jest plików do załadowania). W takim przypadku musimy jednak wpisać zero jako adres ładowania. W ten sposób procedura w interfejsie stacji pobierze potrzebne adresy z nagłówków plików, a nasz programik skróci się o ponad połowę.

HAPPY END

Z drugiej części artykułu dowiemy się, co robić, gdy nie znajdziemy miejsca w pamięci na loader. Będą także zaklęcia, uczące grę ładowania kolejnych poziomów (lub innych danych) z dyskietki, oraz zapisywania różnych danych na tejże dyskietce. A teraz możemy podłączyć joystick (o ile jesteśmy w jego posiadaniu) i czerpać podwójną przyjemność z własnoręcznie przerobionej gry.

JACEK TROJAŃSKI

LISTING 4

```
ORG  23760      ; zaraz po REM
PUSH IY
LD   IY, 0
RST  8
POP  IY
LD   HL, NAZWA  ; nazwa pliku
LD   DE, 2000h  ; adres bufora
LD   BC, KON-NAZWA
LD   A, C       ; przechowaj długość
INC  A          ; długość + 1
LDIR            ; skopiuj nazwę
EX   DE, HL
LD   (HL), 0    ; znacznik końca
LD   B, A
LD   HL, 214Dh  ; filetyp
LD   (HL), 3    ; Bytes
LD   HL, FFFFh
LD   (214Eh), HL; długość pliku
LD   HL, 60D8h  ; adres ładowania
LD   (2150h), HL
CALL 0CC1h      ; ładuj plik
LD   HL, EXT    ; adres rozszerzenia nazwy
INC  (HL)       ; następny plik
LD   HL, NAZWA  ; nazwa pliku
LD   DE, 2000h  ; adres bufora
LD   BC, KON-NAZWA
LD   A, C       ; przechowaj długość
INC  A          ; długość + 1
LDIR            ; skopiuj nazwę
EX   DE, HL
LD   (HL), 0    ; znacznik końca
LD   B, A
LD   HL, 214Dh  ; filetyp
LD   (HL), 3    ; Bytes
LD   HL, FFFFh
LD   (214Eh), HL; długość pliku
LD   HL, 4000h  ; adres ładowania
LD   (2150h), HL
CALL 0CC1h      ; ładuj plik
CALL 60FE       ; dekompresja
LD   HL, EXT    ; adres rozszerzenia nazwy
INC  (HL)       ; następny plik
LD   HL, NAZWA  ; nazwa pliku
LD   DE, 2000h  ; adres bufora
LD   BC, KON-NAZWA
LD   A, C       ; przechowaj długość
INC  A          ; długość + 1
LDIR            ; skopiuj nazwę
EX   DE, HL
LD   (HL), 0    ; znacznik konca
LD   B, A
LD   HL, 214Dh  ; filetyp
LD   (HL), 3    ; Bytes
LD   HL, FFFFh
LD   (214Eh), HL; długość pliku
LD   HL, 4050h  ; adres ładowania
LD   (2150h), HL
CALL 0CC1h      ; ładuj plik
CALL 0604h      ; włączenie ZX-ROM
JP   6101h      ; uruchomienie gry
DEFM "TETRIS."
DEFM "1"
KON
```

LISTING 5

```
        ORG  23760      ; zaraz po REM
        LD   HL, START
        LD   DE, 23296  ; dokąd kopiować
        LD   BC, KON-START
        LDIR
        JP   23296      ; skok do procedury
START   PUSH IY
        LD   IY, 0
        RST  8
        POP  IY
        LD   HL, NAZWA-START+23296
        LD   DE, 2000h  ; adres bufora
        LD   BC, KON-NAZWA
        LD   A, C
        INC  A
        LDIR
        ...
        ...
        (analogicznie, jak listing 4)
        ...
        ...
        CALL 0604h      ; włączenie ZX-ROM
        JP   6101h      ; uruchomienie gry
NAZWA   DEFM "ACADEMY."
EXT     DEFM "1"
KON
```

LISTING 6

```
10 REM
        tu odpowiednia liczba spacji

20 BORDER NOT PI : PAPER NOT PI : INK NOT PI
: CLEAR 23791 : RANDOMIZE USR 23760
```

Krótka rozprawa z taśmą (cz. II)

PAMIĘĆ POD PAMIĘCIĄ, CZYLI INTERFEJS STACJI

Po przeczytaniu pierwszej części artykułu, każdy średnio zaawansowany czytelnik (to znaczy taki, który na widok wydruku w asemblerze nie ucieka w najdalszy kąt osłaniając głowę jedną ręką, a drugą sięgając do wyłącznika prądu), powinien być w stanie zmusić niemal każdą grę do współpracy (z dyskiem, oczywiście). Napisałem: niemal każdą grę, ponieważ niektóre z nich są wyjątkowo „odporne" na wszelkie próby. Tak się dzieje na przykład, gdy któryś z załadowanych do pamięci bloków rozpakowuje się, niszcząc przy tym obszar programu w BASIC-u (a tam właśnie znajduje się procedura ładująca kolejne bloki). Trzeba wtedy pomyśleć o umieszczeniu ładowacza w innym miejscu pamięci — na przykład „pod sufitem", czyli od adresu 65000 i wyżej. Jednak nie zawsze to jest możliwe — niektóre gry korzystają z każdego wolnego skrawka pamięci, także z pamięci ekranu. Lecz na pewno nie używają one pamięci RAM interfejsu stacji — bo większość z programów przybywa do nas z krajów, w których stacja Timex FDD jest mało popularna.

Z instrukcji obsługi do stacji dysków możemy się dowiedzieć, że obszar od #2156 do #23FF jest wolny (nie wykorzystywany przez system). Miejsca jest więc aż nadto, ale pojawia się pewien problem: tej pamięci nie da się wykorzystać z poziomu BASIC-a w prosty sposób. Aby kod maszynowy loadera przenieść z linii REM do pamięci interfejsu, trzeba napisać krótką procedurkę, przełączającą pamięć i kopiującą zawartość: (listing 1). Także w tym przypadku obowiązuje przeliczanie etykiet (rozkaz LD HL, NAZWA-START+2156h).

Metodę tą można kombinować z loaderami omówionymi w 1 części artykułu; często zdarza się, że tylko jeden (ostatni) plik wymaga ładowania „spod ROM-u" (czyli z pamięci interfejsu). Loader składa się wtedy z trzech części: ładującej bloki „normalnie", relokującej ciąg dalszy oraz części relokowanej, ładującej blok w opisany właśnie sposób.

RATUJMY NORNIKI

Każdy ogrodnik ukamieniowałby mnie na sam widok tego tytułu (dlaczego? Spytaj ogrodnika). A tymczasem chodzi o specyficznego członka rodziny norników, niewielkiego gryzonia zwanego pod tajemniczą nazwą LEMING. Chyba wszyscy już wiedzą, na czym gra LEMMINGS polega, gdyż cieszyła się ona ogromną popularnością kilka miesięcy temu. Na każdym ze 120 (w przypadku Spectrum — z 60) poziomów mamy za zadanie uratować kilka z tych przemiłych stworzonek, zmuszając je do wytężonej pracy fizycznej. Jednak ta gra szybko może się znudzić, gdy kolejne poziomy ładujemy z taśmy...

Na początku „rozbrajamy" program ładujący:

```
10 CLEAR 24575
20 LOAD ""CODE
30 RANDOMIZE USR 25000
```

Może się on trochę różnić od zamieszczonego, gdyż każdy hacker lubi coś zmienić i dopisać się w loaderze. Linia dwudziesta ładuje blok kodu, a linia trzydziesta uruchamia ten kod. Oczywiście, zgodnie ze wskazówkami z pierwszej części artykułu przerabiamy programik:

```
10 CLEAR 24575
20 LOAD*"LEMMINGS.1"CODE
30 RANDOMIZE USR 25000
```

W ten sposób załaduje się pierwszy z trzech bloków programu — dwa pozostałe będą ładowane z poziomu kodu maszynowego — oryginalna procedura jest „zaszyta" w pliku LEMMINGS.1 (wszystkie pliki najlepiej skopiować przy użyciu ZEBRA COPY). Trzeba więc sięgnąć po MONS-a (lub inny monitor) i zlokalizować miejsca ładowania dalszych bloków. W tym celu kasujemy linię trzydziestą i uruchamiamy loader.

Najprościej jest szukać rozkazu skoku do procedury LOAD (CALL 0556h), lecz nie zawsze jest to metoda skuteczna. Znacznie prościej jest wyszukać rozkazy ładowania rejestru IX — jest on używany przede wszystkim przy operacjach taśmowych. W ten sposób odnajdujemy adres #6527. Warto jednak cofnąć się o kilkanaście rozkazów, by znaleźć właściwy początek procedury (listing 2). Z analizy listingu wynika, że pierwszy z dwóch bloków ładuje się pod adres C000h (49152) i ma długość 0DC0h (3520) bajtów, natomiast drugi — odpowiednio 8F0Dh (36621) i 6EF3h (28403) bajtów. Ponieważ mało kto używa ZX Spectrum 128 ze stacją Timex, najprostsza przeróbka procedury nie korzysta z banków pamięci (listing 3). Na uwagę zasługuje fakt, że nie jest sprawdzana poprawność ładowania — w przypadku dysku błąd jest mało prawdopodobny. Po kompilacji i zapamiętaniu (na przykład pod nazwą NAKLADKA.BIN) procedury musimy ją „nakładkować" na oryginalny blok kodu, chociażby w taki sposób:

```
LOAD*"LEMMINGS.1"CODE:
LOAD*"NAKLADKA.BIN" CODE 25867:
SAVE*"LEMMINGS.1" code 24576,10125
```

Wcześniej warto jest sporządzić kopię pliku LEMMINGS.1 na innej dyskietce, aby w razie pomyłki nie przenosić go ponownie z taśmy.

Następnie bierzemy się za procedurę wgrywania leveli. Może się ona znajdować w każdym z trzech bloków gry! Ładujemy więc wszystkie trzy pliki do pamięci (listing 4) nie uruchamiając ich, a następnie wgrywamy program monitora, najlepiej taki, który można umieścić w pamięci ekranu. Zlecamy programowi wyszukanie słów, które zauważyliśmy w trakcie ładowania leveli z taśmy — na przykład słówka LOAD. Znajdujemy je pod adresem 912Fh (czyli w pliku LEMMINGS.3). Cofając się o kilka rozkazów możemy obejrzeć taką oto procedurkę:

```
          ...
          ...
          RET
#9129     LD   BC, #1602
          CALL #C731
#912F     ; tutaj znajduje się napis
          ; load error, widoczny jako
          ; "dziwne" rozkazy
          ...
          ...
```

Procedura ta ma za zadanie wyświetlić odpowiedni tekst (leżący bezpośrednio za rozkazem CALL) w odpowiednim miejscu ekranu (określonym zawartością rejestrów BC). W tym wypadku jest to komunikat o błędzie w trakcie wgrywania — czyli procedura ta jest wywoływana zaraz po komendzie ładowania pliku. Szukamy więc sekwencji 29h, 91h; występuje ona kilkakrotnie. Fragment odpowiedzialny za ładowanie poziomów przedstawia listing piąty. Komórka F55Ah zawiera numer poziomu (od 1 do 60), a pliki są ładowane zawsze pod adres 5B68h i mają długość 20D5h.

Teraz bierzemy się za napisanie odpowiedniej procedury, którą zainstalujemy analogicznie, jak w przypadku programiku z listingu trzeciego. Ponieważ wszystkich poziomów jest 60, najlepiej jest numer zakodować dwiema literami. Załóżmy, że będziemy używać jedynie pierwszych szesnastu liter (od A do P). Niech 4 starsze bity będą kodowane na pierwszej literze, a 4 młodsze — na drugiej literze. Wobec tego level numer 1 powinien być oznakowany sekwencją AB, numer 2 — literami AC, a numer 60 — sekwencją DM. Widać, że wszystkie poziomy nie zmieszczą się na jednej dyskietce

LISTING 1

```
        ORG  23760      ; zaraz po REM
        PUSH IY
        LD   IY, 0
        RST  8          ; zamiana ROM-u
        POP  IY
        LD   HL, START
        LD   DE, 2156h
        PUSH DE         ; jako adres powrotu
        LD   BC, KON-START
        LDIR
        RET             ; ROM interfejsu aktywny
START   LD   HL, NAZWA-START+2156h
        LD   DE, 2000h  ; adres bufora
        LD   BC, KON-NAZWA
        LD   A, C
        INC  A
        LDIR
        ...
        ...
        (tak, jak loader z części 1 artykułu)
        ...
        ...
        CALL 0604h      ; włączenie ZX-ROM
        JP   8000h      ; uruchomienie gry
NAZWA   DEFM "FORCE.1"
KON
```

LISTING 2

```
        ...
        ...
        RET
#650B   LD   A, (#88B8)
        BIT  0, A
        JR   Z, #657E
        LD   A, (#88BA)
        JR   Z, 6520
        LD   BC, #7FFD   ; sprawdzanie, czy
        LD   A, #14      ; Spectrum 128
        OUT  (C), A
        JR   #6527
#6520   LD   BC, #7FFD
        LD   A, #16
        OUT  (C), A
#6527   LD   IX, #C000   ; początek bloku
        LD   DE, #0DC0   ; dł. bloku
        LD   A, #FF
        SCF
        INC  D
        EX   AF, AF'
        DEC  D
        DI
        CALL #05C2       ; ładowanie
        JP   NC, #6559   ; jeśli błąd - skocz
        LD   BC, #7FFD
        LD   A, #10      ; bank zerowy
        OUT  (C), A
#6542   LD   IX, #8F0D   ; początek bloku
        LD   DE, #6EF3   ; dł. bloku
        LD   A, #FF
        SCF
        INC  D
        EX   AF, AF'
        DEC  D
        DI
        CALL #0562       ; ładowanie
        JP   NC, #6559   ; skocz, gdy błąd
        JP   #8F0D       ; skocz do pocz. gry
#6559   ; tu - obsluga bledow
        ...
        ...
#657E   LD   IX, #C000   ; procedura ładowania
        LD   DE, #0DC0   ; bez użycia banków
        LD   A, #FF      ; pamięci ZX 128
        SCF
        INC  D
        EX   AF, AF'
        DEC  D
        DI
        CALL #05C2       ; ładuj
        JP   #6542       ; ciąg dalszy taki sam
```

LISTING 4

```
10 CLEAR 24575
20 LOAD*"LEMMINGS.1"CODE 24576
30 LOAD*"LEMMINGS.2"CODE 49152
40 LOAD*"LEMMINGS.3"CODE 36621
```

LISTING 3

```
        ORG  650BH
        PUSH IY
        LD   IY, 0
        RST  8          ; ROM interfejsu
        POP  IY
        LD   HL, NAZWA
        LD   DE, 2000H
        LD   BC, 9
        LDIR            ; kopiuj nazwę
        LD   A, 32H     ; kod cyfry "2"
        LD   (DE), A    ; rozszerzenie
        INC  DE
        XOR  A
        LD   (DE), A
        LD   B, OBH     ; dł. nazwy
        LD   HL, 214DH
        LD   (HL), 3    ; typ = CODE
        LD   HL, 0DC0H  ; długość
        LD   (214EH), HL
        LD   HL, C000H  ; start
        LD   (2150H), HL
        CALL 0CC1H      ; ładuj LEMMINGS.2
        LD   HL, NAZWA
        LD   DE, 2000H
        LD   BC, 9
        LDIR            ; kopiuj nazwę
        LD   A, 33H     ; kod cyfry "3"
        LD   (DE), A    ; rozszerzenie
        INC  DE
        XOR  A
        LD   (DE), A
        LD   B, OBH     ; dł. nazwy
        LD   HL, 214DH
        LD   (HL), 3    ; typ = CODE
        LD   HL, 6EF3H  ; długość
        LD   (214EH), HL
        LD   HL, 8F0DH  ; start
        LD   (2150H), HL
        CALL 0CC1H      ; ładuj LEMMINGS.3
        CALL 0604H      ; włącz ZX ROM
        JP   8F0DH      ; uruchom grę
NAZWA   DEFM "LEMMINGS. "
```

LISTING 5

```
#90B7   LD   A, (#F55A)  ; numer levelu
        CALL #9871       ; wydruk numeru
#90BD   LD   IX, #5B68   ; adres ładowania
        LD   DE, 1       ; długość nagłówka
        XOR  A
        SCF
        INC  D
        EX   AF, AF'
        DEC  D
        DI
        CALL #0562       ; ładowanie nagłowka
        JP   NC, #9129   ; skocz, gdy błąd
        ...
        ; wydruk napisu FOUND LEVEL :
        ...
        LD   A, (#5B68)  ; weź bajt z nagłówka
        PUSH AF
        CALL #9871       ; wydruk numeru
        POP  AF
        LD   HL, #F55A   ; porównaj z
        CP   (HL)        ; numerem poziomu
        JR   NZ, #90BD   ; skocz na początek,
        ...              ; gdy nie ten poziom
        LD   IX, #5B68   ; adres ładowania
        LD   DE, #20D5   ; długość
        LD   A, #FF      ; blok
        SCF
        INC  D
        EX   AF, AF'
        DEC  D
        DI
        CALL #0562       ; ładuj blok
        JP   NC, #9129   ; skocz, gdy błąd
        LD   A, 0
        OUT  (#FE), A    ; przywróć border
        LD   A, (#F55A)
        LD   HL, #5B68
        CP   (HL)
        CALL #F8C0
        RET              ; koniec
#9129   LD   BC, #1602   ; współrzędne
        CALL #C731       ; wypisz na ekranie
#912F   DEFM "LOAD ERROR."
```

LISTING 6

```
        ORG  90B7H
        LD   A, (F55AH)  ; weź numer levelu
        CALL 9871H       ; wydrukuj numer
        PUSH IY
        LD   IY, 0
        RST  8           ; ROM interfejsu
        POP  IY
        LD   HL, NAZWA
        LD   DE, 2000H
        LD   BC, 9
        LDIR             ; kopiuj nazwę
        LD   A, (F55AH)  ; nr levelu
        SRL  A
        SRL  A
        SRL  A
        SRL  A           ; tylko 4 starsze bity
        ADD  A, 41H      ; zamień na literę
        LD   (DE), A     ; 1 znak rozszerzenia
        INC  DE
        LD   A, (F55AH)  ; nr levelu
        AND  0FH         ; tylko 4 młodsze bity
        ADD  A, 41H      ; zamień na literę
        LD   (DE), A     ; 2 znak rozszerzenia
        INC  DE
        XOR  A           ; znacznik końca
        LD   (DE), A
        LD   B, OCH      ; dł. nazwy
        LD   HL, 214DH
        LD   (HL), 3     ; typ = CODE
        LD   HL, 20D5H   ; długość
        LD   (214EH), HL
        LD   HL, 5B68H   ; start
        LD   (2150H), HL
        CALL 0CC1H       ; ładuj LEMMINGS.nn
        CALL 0604H       ; włącz ZX ROM
        JR   NZ, 9129H   ; skocz, gdy błąd
        CALL 8FC0H
        RET
NAZWA   DEFM "LEMMINGS.  "
```

— trzeba więc obsługiwać błąd ładowania pliku, wyświetlając komunikat proszący o zmianę dysku (w tym wypadku można wykorzystać analogiczną procedurę znajdującą się pod adresem 9129h). Po kompilacji programu z listingu szóstego i zapisaniu go jako LEVELS.BIN, warto jest przeszukać plik LEMMINGS.3 i pozamieniać komunikaty dotyczące taśmy na odpowiednie, dotyczące dysku. Następnie wpisujemy:

```
LOAD*"LEMMINGS.3"CODE 36621:
LOAD*"LEVELS.BIN" CODE 37047:
SAVE*"LEMMINGS.3" CODE 36621,28403
```

a następnie stajemy na rzęsach, by na 47 poziomie lemingi nie spadły w przepaść, bo kończy się to obfitym chlapnięciem krwi we wszystkie strony.

Należy pamiętać, żeby nie wciskać BREAK w trakcie ładowania z dysku, gdyż kończy się to najczęściej pozdrowieniami od producenta komputera. Można tego uniknąć przez odpowiednią obsługę BREAK, lecz prościej jest po prostu o tym pamiętać i w ten sposób zaoszczędzić nieco pamięci.

W trzeciej, ostatniej części, mowa będzie o tym, co zrobić z grą, która chce coś nagrać na taśmie.

JACEK TROJAŃSKI

Krótka rozprawa z taśmą *część 3 (i ostatnia)*

Zgodnie z obietnicą, zajmiemy się teraz przystosowaniem gry do zapisywania plików na dyskietce. Przeważnie zapisywany jest bieżący stan gry lub tabela najlepszych wyników.

Jako przykład niech nam posłuży TAU CETI II - gra stara ale dobrze napisana i często korzystająca z taśmy. Umożliwia ona zapamiętanie stanu gry oraz własnoręcznie zaprojektowanych pojazdów. Ładowac z tasmy możemy właśnie stan gry projekty statkow, a także kolejne poziomy (jest ich pięc)

Procedury obsługi znajdujemy dosyc szybko stosując opisane wczesniej metody Najprosciej jest poszukac (za pomocą monitora) nazwy pliku nagrywanego, np."level..." W ten sposób odnajdujemy tablicę·

#89F0· chr(3) "Game File."
#8A01: chr(3) 'Ship Data.'
#8A12. chr(3), "level III "
#8A23· chr(3) "Extra lev "

Pod adresem #8224 znajdujemy tablicę, służącą do dopisywania końcówek z numerem levelu:

#8224: 'EXTR" #8228: "I "
#822C: "II " #8230· "III "
#8234: "IV " #8238: "V "

Odpowiednia końcówka jest kopiowana pod adres #8A19, czyli po spacji za słowem "level". Widać, że prócz wymiany procedur, będziemy musieli zmodyfikować tablicę z nazwami tak, by nie zawierały one wewnątrz spacji. Wpierw jednak zajmiemy się samymi procedurami. Po żmudnych poszukiwaniach odnajdujemy adresy· #89BE (LOAD) i #89B3 (SAVE) Są to procedury niskiego poziomu, wspolne dla wszystkich operacji, jednak bardzo trudno je przystosowac do współpracy z dyskiem. Szukamy więc wszystkich wywołan (rozkazów· CALL 89BEh i CALL 89B3h, czyli ciągów #CD, #BE, #89 i #CD, #B3, #89). Następnie cofamy się, ustalając właściwy początek badanej procedury. ostatecznie otrzymujemy adresy. #87C0 (LOAD) i #88FA (SAVE) Warto przeanalizować umieszczony tam kod w całości, jednak zostawiam to jako ćwiczenie dociekliwym czytelnikom.

Każda z tych procedur, poza ładowaniem lub nagrywaniem odpowiednich bloków, ma za zadanie rozróżnić, czy jest to level, projekt statku, czy stan gry, a następnie dobrać nazwę z omówionej wcześniej tabeli. Ponadto wyświetlane są różne komunikaty. Porównajmy treść procedur z zamieszczonymi listingami. Dodatkowych wyjaśnień wymagają parametry wejściowe Dla procedury LOAD:

level C = 4, B = 4
Ship Data C = 2, B = 2
Game File C = 9, B = 0

Z tych wartości obliczany jest adres nazwy. W przypadku, gdy ładowany ma byc poziom (level), akumulator zawiera jego numer.

Dla procedury SAVE występują tylko dwie wartości parametru. B = 2 dla Ship Data i B = 0 dla Game File (poziomów przecież nie można zapisywać).

Najlepszym wyjściem będzie zdeasemblowanie bloku pamięci od #87C0 do #89B3 z zapisaniem na taśmę (lub dysk). Następnie zbędne linie zastępujemy tymi z listingów (obszary "wykropkowane" na wydrukach nie ulegają zmianie). Po kompilacji programu z listingu 1, nagrywamy go np. pod nazwą OVL1.BIN. Musimy jeszcze sprawdzić, jaki adres otrzymała etykieta ROMON (przełączanie ROM-ów, wspólne dla obu procedur). Adres ten przypisujemy etykiecie ROMON w listingu drugim (powinien on wynosić #88C1, o ile listing 1 nie był modyfikowany). Drugi utworzony blok kodu zapamiętujemy w pliku OVL2.BIN. Teraz trzeba załadować główny blok gry i "dograć" nasze pliki pod odpowiednie adresy (#87C0 i #88FA). Jednak to nie wszystko.

Trzeba jeszcze zmodyfikować tablicę z nazwami. Powinna ona wyglądać następująco:

#89F0 DEFM chr(3), "GAMEFILE"
#8A01 DEFM chr(3), "SHIPDATA"
#8A12 DEFM chr(3), "LEVEL "
#8A23 DEFM chr(3), "EXTRALEV"

Drugą tablicę (z rzymskimi numerami leveli) pozostawiamy bez zmian; zauważmy, że numery te będą przepisywane pod adres #8A18, czyli bezpośrednio po słowie "level". Ostatnią czynnością będzie wyszukanie w pamięci słów "TAPE, "tape" i "Tape" i ich zamiana na "Disk". W ten sposób w menu gry zamiast "Tape menu" będzie napis "Disk menu" itp.

Po zapamiętaniu głównego bloku gry, przystępujemy do kopiowania leveli. Nada-

LISTING 1

```
10 : LOAD
20 *D+
30         ORG  #87C0       ; początek
40 LOAD    PUSH AF
50         LD   A,B         ; B = kod op.
60         SRL  A           ; B/2
70         LD   (#8A62),A   ; zapamiętaj
80         POP  AF          ; nr levelu
90         BIT  2,C         ; czy ład level
100         JR   Z,L87D8    ; nie
110         INC  A          ; nast. level
120         ADD  A,A
130         ADD  A,A        ; level * 4
140         LD   E,A
150         LD   D,0        ; A -> DE
160          D   HL,#8224   ; tabela numerów
170         ADD  HL,DE      ; adres numeru
180         LD   DE,#8A18   ; dokąd kopiować
190         PUSH BC
200         LD   BC,4
210         LDIR            ; kop'uj numer
220         POP  BC         ; odtwórz BC
230 L87D8   LD   A,C
240         LD    #8A61),A  ;przech. kod op.
250         LD   DE #7 2F
260         CALL #993A
270         PUSH DE
280         CALL #878D
290         POP  DE
300         CALL #96D5
310         CALL ROMON      ; przełącz ROM
320         LD   A,(#8A62)  ; kod op./2
330         LD   L,A
340         LD   H,0        ; HL = kod op./2
350          D   E,A
360         LD   D,H        ; DE = HL
370         ADD  HL,HL
380         ADD  HL,HL
390         ADD  HL,HL
400         ADD  HL,HL      ; HL * 16
410         ADD  HL,DE      ; HL * 17
420         INC  HL         ; HL * 17 + 1
430         LD   DE,#89F0   ; tabela nazw
440         ADD  HL,DE      · HL-adres nazwy
450         LD   BC,8
460         LD   DE,#2000
470         LDIR            ;kopiuj do SRAM
480         XOR  A
490         LD   (DE),A     ; znacznik końca
500         LD   HL,#214D
510         LD   (HL),3     ; kod LOAD (TOS)
520         LD   A,(#8A62)
530         ADD  A,A
540         LD   E,A
550         LD   D,0
560         LD   HL,#8A59
570         ADD  HL,DE
580         LD   E,(HL)
590         INC  HL
600         LD   D,(HL)     ; DE = długość
610         LD   (#8A63),DE
620         LD   (#214E),DE
630         LD   HL,#ED00   ; HL =adres ład.
640         LD   (#2150),HL
650         PUSH DE
660         LD   A,9        ; długosć nazwy
670         LD   B,A
680         CALL #0CC1      ; LOAD
690          D   A,(8450)
700         CALL #0603      ; ZX ROM
710         POP  DE
720         OR   A          ; czy błąd ?
730         JP   NZ,#89A6   ; tak
740         LD   BC,(#8A63)
750         CALL #89D9      ; spr. checksum
760         LD   IX,#ED00
...              ; tak, jak
...              ; w oryginale
1290         LD   (#6772),A
1300 L88F5  CALL #87A2
1310        SCF             ; CY = 0 (OK)
1320        RET
1330 ROMON  PUSH IY         ; przełączanie
1340        LD   IY,0       ; ZX ROM -> TOS
1350        RST  8
1360        POP  IY
1370        RET
```

LISTING 2

```
1399 ; SAVE
1400 ;ROMON WPISAC PO KOMPILACJI OVL1.ASM !
1410 ROMON  EQU  #88C1
1420        ORG  #88FA       ; początek
1430 SAVE   LD   A,B         ; kod operacji
1440        PUSH AF
1450        SRL  A           ; kod/2
1460        LD   (#8A62),A   ; przechowaj
1470        CALL #878D       ; 'Disk
option:'
1480        POP  AF
1490        LD   B,A
1500        LD   DE,#712F
1510        CALL #993A
1520        CALL #96D5       ; 'Press fire'
1530        CALL #85A8       ; czekaj na
fire
1540        LD   A,(#8A62)   ; kod operacji
1550        ADD  A,A
1560        LD   E,A
1570        LD   D,0
1580        LD   HL,#8A59
1590        ADD  HL,DE
1600        LD   E,(HL)
1610        INC  HL
1620        LD   D,(HL)      ; długość w DE
1630        LD   (#8A63),DE
1640        CALL ROMON       ; włącz TOS ROM
1650        LD   A,(#8A62)
1660        LD   L,A
1670        LD   H,0
1680        LD   E,A
1690        LD   D,H
1700        ADD  HL,HL
1710        ADD  HL,HL
1720        ADD  HL,HL
1730        ADD  HL,HL       ; HL * 16
1740        ADD  HL,DE       ; HL * 17
1750        LD   DE,#89F0    ; tabela nazw
1760        ADD  HL,DE
1770        INC  HL          ; omiń chr(3)
1780        LD   DE,#2000
1790        LD   BC,8
1800        LDIR             ; kopiuj nazwę
1810        XOR  A
1820        LD   (DE),A      ; znacznik
końca
1830        LD   A,(#8A62)
1840        ADD  A,A
1850        ADD  A,A
1860        ADD  A,A
...              ; dalej tak, jak
...              ; w oryginale
2180 L897D  LD   BC,(#8A63) ; długość bloku
2190        CALL #89D9       ; oblicz
checksum
2200        LD   (#ED00),HL ; wpisz
checksum
2210        LD   HL,#2134
2220        LD   (HL),1
2230        LD   A,9         ; długość nazwy
2240        EX   AF,AF'
2250        LD   A,3         ; kod TOS
(SAVE)
2260        LD   BC,(#8A63)
2270        LD   DE,#ED00    ; adres start.
2280        CALL #0A61
2290        LD   A,(8450)
2300        CALL #603        ; SAVE
2310        OR   A           ; błąd?
2320        JP   NZ,#89A6    ; tak
2330        CALL #96CF       ; 'File Saved'
2340        DEFW #7198
2350        CALL #85B0       ; czekaj na
fire
2360        CALL #87A2
2370        SCF
2380        RET              ; koniec proc.
```

jemy im oczywiście nazwy "LEVELI", "LEVELII", aż do "LEVELV".

I to właściwie już wszystko. Tych, którym opisane czynności nie wydają się proste, odsyłam do poprzednich części ninejszego artykułu. Innym, w ramach rozgrzewki, polecam przeróbkę SIM CITY lub TETRIS (jest ona prostsza od opisanej). Tego typu gier jest całe mnóstwo; niektóre zajmują całą dyskietkę, lub jeszcze więcej. Przystosowanie gier do współpracy z dyskiem daje satysfakcję, a przy okazji sama gra zyskuje na wartości i wzrasta przyjemność zabawy.

JACEK TROJAŃSKI