ZX Magazín

časopis pro uživatele počítačů ZX Spectrum a kompatibilních
číslo: 2/99 cena: 34 Kč

Vítejte na DoxyConu ’99

Cože? Vy se neúčastníte? To děláte VELIKOU chybu!

DoxyCon ’98: spousta nohou, písmena ZX složená z různých odrůd ZX Spectra a pohár pro vítěze DemoCompo (uprostřed písmene X)

Úvodník

Hello to all Speccy-fans!

Je tomu již více než rok, co vyšlo první číslo obnoveného ZXM. A vyšlo pouze pět čísel z původně plánovaných sedmi, což je alarmující, ano, bohužel je tu zcela neplánovaný skluz. Naštěstí je teď ZXM vydáván za podpory firmy CONQUEST, což je velmi pozitivní, neboť se rapidně snížily náklady, tudíž bychom mohli, tedy pokud bude dostatek článků, vydávat ZXM častěji. A abyste nemuseli číst články stále od těch samých autorů, musíte posílat své vlastní, bez příspěvků to prostě nejde.

Co se týče obce předplatitelské, tak ta se nepatrně s každým vydaným číslem zvětšuje, většina z Vás má však předplaceno následující číslo nebo dvě, proto Vás prosím, abyste po vyjití příštího čísla poslali předplatné na dalších šest popřípadě na další tři výtisky. Samozřejmě nic není zadarmo, a tak pokud bude ZXM dál vycházet, a to bude, tak Vám nepředplacená čísla nepřijdou.

Doufám, že se Vám tento ZXM bude líbit stejně, ne-li více, jak ty předešlé, že se Vám ze článků o packování nezapackuje mozek, a že nezapackujete Speccy pod skříň a nezdrhnete k PeCím.

Přeji Vám hezký zbytek prázdnin a uvidíme se na DOXYCONu (viz vedlejší stránka).

Matěj Kryndler – šéfredaktor

Tak tohle číslo na mě Matěj přímo nehodil část úvodníku, ale ten svůj dodal moc krátký, tak aby tu nebylo prázdno, zas pár slov ode mě.

V tomto čísle je na rozdíl od toho minulého méně recenzí na hry a naopak je více technické. Když jsem poprvé viděl Gamův článek o pakování a pakerech, zhrozil jsem se. Délka přes 30 kB čistého textu... Ale rozdělovat se mi ho nechtělo, neboť si ještě živě pamatuji svou nechuť k „rozděleným“ článkům. Prostě to není ono. Něco jiného jsou originální seriály přímo od autorů (CP/M, seriál o D40). A protože Gamův článek „sežral“ 5 a půl strany, už na hry moc nezbylo :-). Pokud někoho z vás metody komprese vůbec nezajímají, kupte si příští ZX Magazín. Gama dodal podobné (i když ne tak dlouhé) články o ruských klonech spektra a assemblerech (editorech a překladačích) pro ZXS. Novinky samozrejmě chybět nebudou a i na recenze her snad vyjde místo. Co jsem tak prohlížel, co všechno mám na disku připravené k otištění, ke komprimování se ještě vrátíme, o hardware se tu taky něco najde. Chystám s PVL podrobnější článeček o HDD a CD-ROM (nejen připojení, ale snad kompletní „programming tutorial“ :-).

A málem bych zapoměl. **VELIKÉ DÍKY Tomášovi Hauerlandovi (TDM)** za fotku na obálku, kterou sem bez jeho svolení použil. Doufám, že mi za to na DoxyConu neutrhne hlavu... :-).

Lubomír Bláha – Tritolsoft

Obsah

ZX Magazín – časopis pro uživatele počítačů ZX Spectrum a komp.

Vydavatel a šéfredaktor: Matěj Kryndler
Sazba: Lubomír Bláha
Grafická úprava: Jan Hanousek, Lubomír Bláha
Příprava obálky: Tomáš Hauerland, Lubomír Bláha
Tisk: QMS 3260 EX PrintSystem
Adresa redakce: Matěj Kryndler, Lotyšská 8/645, 160 00 Praha 6

Za obsah příspěvku a jeho původnost ručí autor. Inzerci přijímá redakce. Za její obsah ručí inzerent. Cena inzerce dle dohody. Distribuce formou předplatného a soukromými prodejci.

Vychází nepravidelně. Doporučená cena: **34 Kč**

Toto číslo mohlo být vydáno díky podpoře firmy CONQUEST a.s.,
Areál VÚ, 190 11 Praha 9 – Běchovice, http://www.conquest.cz/

DoxyCon ’99

Jelikož jsem z Inter(fer)netu stáhnul tuhle hezkou (no, byl to jpeg) mapu, která se akorát pěkně vešla na stránku, přikládám i oficiální pozvánku a informace od pořadatelů.

Tritolsoft

Team E.S.A. si Tě dovoluje pozvat na akci „**Doxycon ’99**“, která se koná jak byla ohlášena, tedy **23. až 25. 7. 1999**, v kulturním středisku „**Pension Starý Mlýn**“, vedle restaurace „Sklípek“ (u potoka) ve **STARÝCH SPLAVECH** (cca 2 km od Doks – na druhém břehu Máchova Jezera) .

Jak se tam dostat:

Na mapě jsou vyznačeny možné trasy:

1) ČERVENĚ *(tady je to ta tmavší – pozn. tnt)* pokud přijedete vlakem – Jeďte vlakem do stanice „Staré Splavy“ a odtud jděte po hlavní ulici (ul. Dalibora z Myšlína) až k místu konání (cca 300 m od nádraží).

POZOR! Spěšné vlaky ve stanici Staré Splavy zastavují pouze ve směru od Mladé Boleslavi, proto pokud tímto chcete jet, vystupte již ve stanici DOKSY a zde počkejte na místní spoj do Starých Splavů nebo jeďte autobusem, který má zastávku nad nádražím, viz. mapa.

Nebo můžete jít po svých podél Máchova jezera (asi 1,5 km), když přejdete most nad nádražím a poté se dáte vlevo. Ve St. Splavech vyjdete v ulici Jarmilina stezka).

2) ZELENĚ *(zde logicky ta světlejší – pozn. tnt)*, když přijedete autobusem – Vystupte na zastávce „Staré Splavy“ a jděte dále do města. Asi 100 m odtud je vlaková zastávka a železniční přejezd a odtud již stejně jako v bodě 1.

3) AUTEM (nevyznačeno) – Na křižovatce u Doks neodbočujte (nevjíždějte) do Doks, ale jeďte směrem Česká Lípa. Asi po 2 km je zde viditelná odbočka vpravo do Starých Splavů.

Je zde problém s přespaním, proto si účastníci budou muset zajistit nocleh sami. Kdo nechce, může samozřejmě přenocovat (probdět noci) na místě, ovšem bez zbytečného hluku.

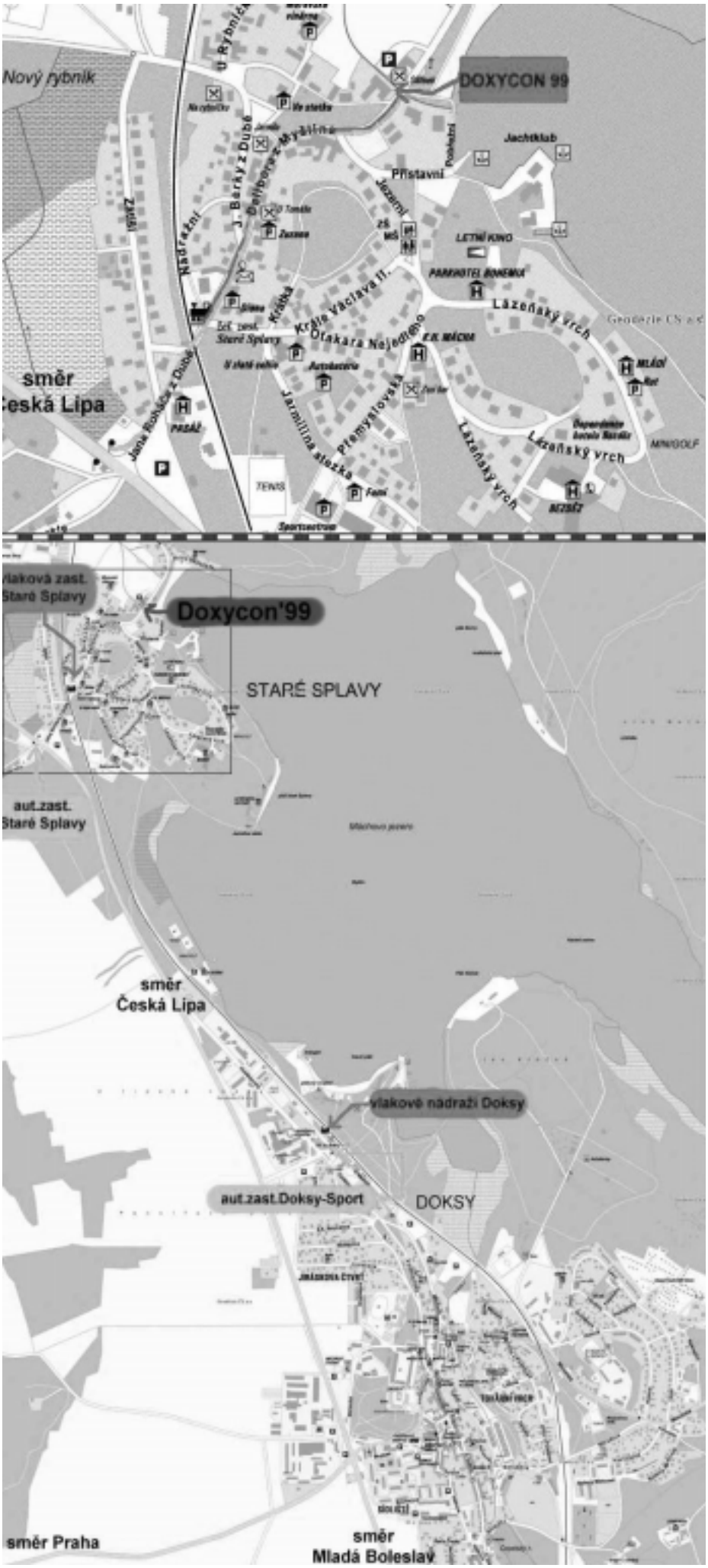
Účastnický poplatek se nemění, tedy 100 Kč/osoba. Peníze slouží k zaplacení nájmu z poskytnutých prostor.

Těšíme se na Vás!

Team E.S.A.

Jé, zbyl mi tady ještě takovej cancourek... DoxyCon ’98 byl super a letos to určitě bude ještě lepší. Přijede mnoho známých osobností jak z ČR tak zahraničí, takže pokud chcete být „IN“ a ne „OUT“, určitě přijeďte.

Tritolsoft



NoWinky

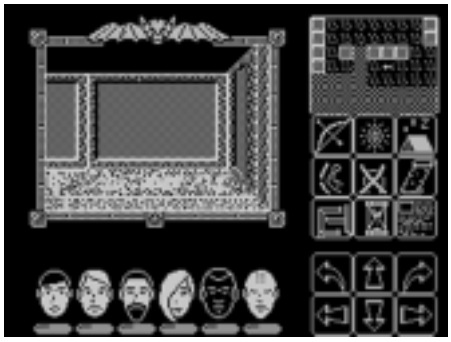
Když jsem v předminulém čísle nadával, že nemám ani demoverzi SMAGLYho 3, netušil jsem, že se mi asi měsíc po vydání sejdou dema hned dvě, první pro 48 kB s ošizenou grafikou a jednou krátkou úrovní, a druhá only 128 kB se špičkovým intrem – viz obrázek, skvělou grafikou i ozvučením.

Jednou z mála novinek bez přiloženého obrázku bude plná verze **ICE CLIMBER**a, ze kterého jste mohli screenshot vidět v předminulém čísle (alespoň si to myslím).

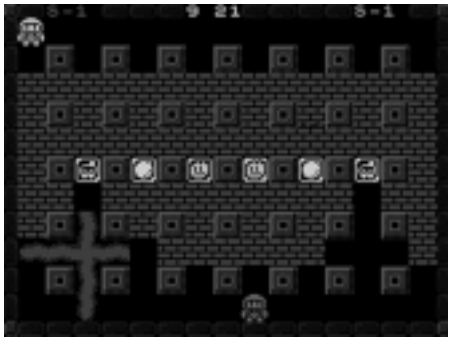
Mezi nejaktivnější skupiny v bývalém SSSR patří bezesporu grupa FATALITY (http://ftl.da.ru), která se, po dokončení logickoakční hry PUSSY – The Titanic Story, rozhodla předělat z Amigy pařbu FANTASTIC DIZZY.

Zcela neznámá MULTYPLEX GROUP předělává WORMS. V plné verzi bude možno hrát proti počítači, budete moci použít všechny zbraně, a přibude ozvučení. *(Menu je na Spectru v interlace módu, ovšem v časopise to není vidět :-)*

Pro příznivce dungeonů tu máme lahůdku COURSE OF XEEN, od které bohužel neznáme ani autora.



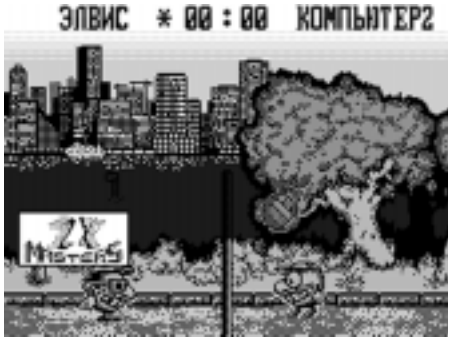
ALLIANCE TEAM připravuje hru MEGA-B💣💣M – předělávku Dynablastera, která bude zabírat celou BETAdisketu, samozřejmostí je ozvučení pro AY a General SOUND, multikolorové animace, bonus levely a další vychytávky.



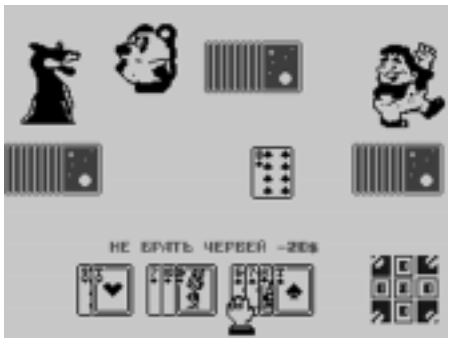
MANIC MINER TECHNOLOGIES (pravděpodobně z ANGLIE) přinesli na obrazovky ZX Specter další pokračování dobrodružství horníka Willyho, tentokráte pod názvem EUGENE – LORD OF THE BATHROOM. Ve hře naleznete vylepšenou grafiku, vydařené levely na motivy známých her, změněnou hudbu a další. Hra pojede i na 48kB Spectru.

MASTER HOME COMPUTERS GROUP připravuje hru WORM, jedná se o klasickou „sežíračku“ čísel, plusů, mínusů a v plné verzi možná i dalších pochoutek, výjimečný je na této hře pohyb červa, neboť nezatáčí o 90 stupňů, ale plynule po kruhové trajektorii.

A pro zaryté počítačové sportovce máme eso v rukávu - hru HEADBALL od firmy ZX-MASTERS. Jedná se v podstatě o předělávku starčičkého ARCADE VOLLEYBALLu z PeCe, provedení je ovšem na vysoké úrovni.



BSOFT z Běloruského města Grodno přetvořil a vylepšil PéCéčkářskou hru M$ HEARTS. Jeho výtvor se jmenuje KING a pro hráče karet bude asi přínosem.



DEECKOBRUZ a RUFF ze skupiny AVALON přinášejí hru KICK DA GAGA, prazvláštní akční hra pro dva playery nebo jednoho a počítač. Cílem je sebrat všechnu energii svému soupeři pomocí nárazů. ■

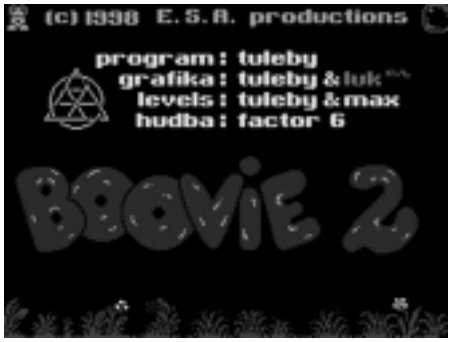
Three „Weeks“ in paradise 128

+GAMA

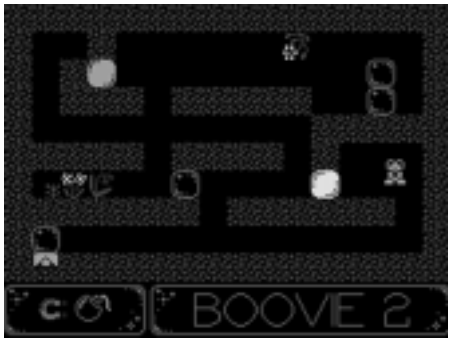
Ano, je to tak. Slavná hra Dave Perryho ještě z dob, kdy psal pro firmu Microgen (slavný rok 1986, kdy vznikala ta nejlepší Spectrácká klasika), má už svou 128 verzi (a to oficiální, žádný hloupý remix nebo tak něco). Nevím, jestli se dá někde normálně sehnat, mně to trvalo dost dlouho a ještě jsem ji dostal jenom jako .Z80 snap, dva dny jsem se nad ním trápil, hlavně nad zásobníkem a přesilou pushů a popů, než jsem rozldirovanou hru dal zase do kupy a zapakoval, ale povedlo se. Aspoň že tak. Znovu říkám, jinak než ve snapu se mi hru nepodařilo nikde najít. Proč 128 verze? Čím se liší od té 48ičkové, kterou všichni tak dobře známe a na kterou vycházejí návody v každém Fifu (a dokonce i v jednom starším ZX Magazínu)? No tak nepočítám-li takové drobnosti, že úvodní obrázek se ukládá do druhé VRAMky, že ta slavná úvodní hudba, kterou všichni tak vesele vykrádají, hraje na AY (a zní to moc dobře), je tu ještě spousta věcí. Bohužel, já nemohu přesně posoudit, co ve 128 verzi proti 48ičkové je navíc, protože 48 verzi jsem nikdy neměl (když totiž víte, že existuje 128, tak už se se 48 nikdy nespokojíte, zvlášť když víte, že 128 verze je lepší). Zkusil jsem ji ale hrát podle návodů v ZXM a zažil jsem několik překvapení. Podle něj by v místnosti u vos měla být tupá sekera. Ne, je tam plazmový deštník. Projdeme-li do moře a z podmořské jeskyně (z té se scrolltextem „Where is Pete?“) odejdeme DOLEVA (plotem se dá projít se štípačkami), stále doleva podzemním zbrojním komplexem či co to je, za plazmovou stěnou teprve leží sekera (tou stěnou se dá projít s deštníkem). To by byl jeden rozdíl – ten vojenský komplex ani nebyl na žádné mapě, pokud se pamatuji... Rozdíl druhý byl v tom, jak šaman nebo bůh deště či co je to za postavu, reagoval na horký popel. Nebo ne on, ale ten mrak. Začal hrlit blesky, ale z místa se nehnul. Což by se podle návodu hnout měl. Takže pokud se 128 verze podle starých 48čkových návodů nedá hrát, znamená to, že tu skutečně jsou věci navíc (mohu potvrdit, v každé paměťové stránce je něco, co asi bude mít nějaký význam). No dobře. Ti, co tvrdili, že Three Weeks in paradise není špatná hra pro 48ku by si honem měli 128 verzi opatřit. Až bude čas, budu se o ní snažit zjistit víc, ale rozhodně by nikomu neměla chybět! Už jenom proto, že hráč se na osvědčený návod nemůže zcela spolehnout a musí hledat a zkoušet sám. Doufám, že rozdílů bude ještě víc (nepohrdl bych nějakým animovaným a ozvučeným závěrečným outrem). ■

BOOVIE 2 48 © 1998 E.S.A. Team

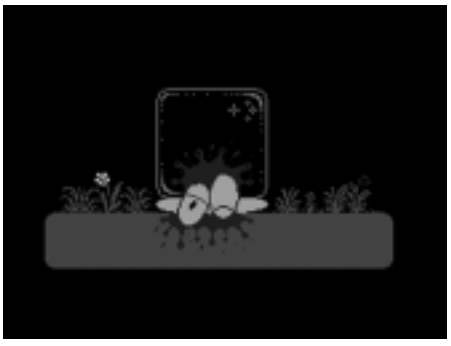
Johny X



Vítejte u pokračování skvělé logické hry BOOVIE. Tentokrát (jak jste jistě postřehli) ji nemá na svědomí KVL, což jí ale vůbec neubírá na kráse. V tomto pokračování ubylo pár věcí, ale taky pár (tuším, že dvě) věcí přibylo. Boovie má na pomoc dva předměty – magnet a plo... teda bombu. Magnet umí přitahovat kvádry (jsou totiž z magneticky bohaté horniny); bomba zas likviduje zelený porost, který občas překáží v cestě. Pokud nevíte o co ve hře jde, pak vězte, že „jenom“ o dostrkání bílého kvádru na plošinku. Zní to jednoduše, ale člověk míní, realita mění...



Teď k technickému zpracování. Grafika je jednoduše nádherná, nevím, kolik jí je z původní hry, ale animace samotné postavičky je super, pokud necháte chvíli Boovieho bez pohybu, začne se krásně nudit. Po pravdě řečeno, nejvíc z grafiky se mi líbí ob-



rázek, který se objeví po projevení gravitace na kvádr, který vlivem volného pádu změní vlastní energii potencionální na energii kinetickou, z dopadem měnící se na energii deformační, což se patřičně projeví i na hlavní postavě (ovšem pokud se Boovie snaží padající kvádr zastavit vlastní hlavou nebo jinou částí těla). Vzhledem k brutalitě, kterou obrázek obsahuje a nemožnosti zákazu čtení ZXM před 22-tou hodinou, musíte si hru opatřit. Beeper se projeví pouze při vkládání hesla, AY-čko hraje jednu ze tří hudeb, které má na svědomí FACTOR 6, ze kterých si každý vybere tu svou. Mě osobně se líbí hudba č.3, ale to je otázka názoru. Ovládání je popsáno ve hře samotné, pokud vím, nepodporuje žádnou veselou páku (joystick). Před hrou je ještě intro, ve kterém se dozvíme, že hra je věnována BONNY B. k jejím narozeninám a další méně či více důležité informace. No, nakonec abych jenom nechválil, jediné, co mi ve hře vadí, je neustálé vkládání hesla kola, což je ale malinká skvrnka na obrovském slunci. Jinak řečeno, hra je výborná, má něco přes 40 levelů (tuším) a vydrží vám dlouho. Pokud ještě nemáte důvod hru si opatřit, pořiďte si ji už jenom kvůli vinikající hudbě a výše zmíněnému brutálnímu obrázku.

Pokud bych měl ohodnotit, hra by dostala 8/10 bodů. ■

(Jelikož ten „gravitační“ obrázek je fakt super, neudržel jsem se a dal ho sem. To ovšem neznamená, že byste si hru neměli pořídit – pozn. tnt.)

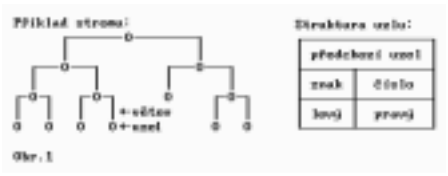
KomprimAce dat je výhodný způsob, jak ušetřit paměť nebo záznamovou kapacitu média. Nyní vám nabízíme možnost poodhalit její podstatu.

Komprimuji, kmrmj, krj, kj, k, ...

Petr Žabenský

O komprimAci jste již určitě někde četli. Něco již vyšlo FIFU, něco i v jiných časopisech, které se už věnují jenom PeCím. Pokaždé, když někdo píše o komprimAci, tak mluví o jednoduché komprimaAci. Ta je založena na tom, že se zaměříme na posloupnosti stejných znaků, nebo si zaznamenáváme polohu nenulových bytů. Všechny tyto algoritmy jsou jednoduché a hlavně rychlé. Jejich nevýhodou však je, že se klidně může stát, že soubor po komprimAci může být až dvakrát větší, než před ní. A pravděpodobnost této situace je u některých algoritmů značně vysoká. Komprimační programy využívající takovéto metody se dají využít spíše ke kompresi obrázků a některých speciálních dat, pro které dosahují nejlepší účinnosti.

My si dneska řekneme o metodě, která je velice účinná obecně. Používá se ke komprimAci souborů větších než asi 2 kB (proč, to si vysvětlíme později). Ta metoda se nazývá **Huffmanův kód**. Myšlenka kódování (nezaměňujte s jiným kódováním, my budeme komprimovat!) je založena na podobném principu jako Morseova abeceda. Když si ji totiž položíte před sebe, zjistíte, že písmena, která se používají nejčastěji, mají i nejkratší kód. A tuto myšlenku se pokusíme aplikovat. V „Morseovce“ se používají pouze dva znaky: čárka a tečka. Když nahradíme tečky jedničkou a čárky nulou, už bychom se dopracovali k nějakému výsledku. Princip je tedy takový, že posloupnost bytů zakódujeme do posloupnosti bitů, kde vynecháme nepotřebné bity. Ale to ještě není vše. Musíme si ještě určit nějakou zarážku, která nám při dekompresi určí, kdy končí jedno písmeno a začíná druhé. To však není tak jednoduché, protože mimo 1 a 0 už nám v binární aritmetice nic nezbývá. Musíme si totiž uvědomit, že když zkomprimujeme data a potom je zase odkomprimujeme, musíme dostat úplně shodné soubory. Jak si tedy vytvořit takovou zarážku?

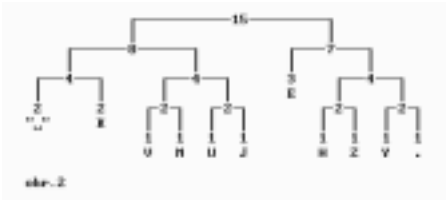


Obr.1

Nebudu vás dlouho napínat a rovnou vám prozradím, že tento problém vyřešíme pomocí tzv. binárního stromu. Nejdříve si však tento pojem objasněme. Pokud jste již programovali v PASCALu a používali jste DDS (Dynamické Datové Struktury), tak jste se s ním již určitě setkali. Jeho název vychází z podobnosti se skutečným stromem. Jeho větve jsou spojením mezi jednotlivými uzly, ve kterých je zaznamenána informace. Každý uzel obsahuje údaj (u nás to bude znak a číslo), odkaz na předchozí (otcovský) uzel a na následující (dceřinné) uzly. Binární se mu říká proto, protože následující uzly jsou pouze dva: levý a pravý. Proč jenom dva? Protože máme jen dva symboly: 0 a 1. A takovýto strom je právě vhodný pro naši potřebu (viz. obr.1)

Jak tedy pracuje Huffmanovo kódování? Předpokládá se dvojí průchod souborem. Nejdříve je nutné sestavit tabulku četnosti jednotlivých znaků v souboru a z této tabulky vytvořit binární strom. V druhém průchodu už probíhá samotné kódování. Předveďme si to na příkladu. Máme následující text: VENKU JE HEZKY.

Tabulka četnosti bude: V (1×), E (3×), N (1×), K (2×), U (1×), mezera (2×), J (1×), H (1×), Z (1×), tečka (1×). A nyní si sestavíme binární strom. Napíšeme si do řádku všechny znaky, které jsou obsaženy v textu právě jednou. Jsou to: V, N, U, J, H, Z, Y. Nad každý znak si napíšeme do kroužku jedničku (počet výskytů). Nyní znaky pospojujeme do dvojic, tak vytvoříme o řádek výš pro každé dva znaky otcovský uzel (ten už ale nebude obsahovat znak, znaky jsou totiž jen v koncových uzlech), k němuž napíšeme součet četností (tedy 2). Jelikož bylo v prvním řádku 8 uzlů, vzniknou nám 4 otcovské uzly. Do téhož řádku už můžeme přidat i uzly se znaky, jejichž četnost je dvě. I tyto uzly spárujeme. Ve třetím řádku získáme 3 uzly, ke kterým přidáme E se třemi výskyty. V následujícím řádku už budou uzly se součty výskytů 7 a 8. Když je spojíme, získáme kořen stromu, odkud budeme vycházet při dekódovaní. Jeho součet četností je stejný jako počet znaků v souboru. Vytvořený binární strom je na obrázku 2.

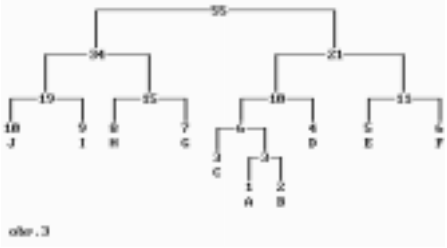


obr.2

Nyní si popíšeme vlastní postup při kódování a dekódování. Pokud budeme kódovat, budeme procházet strom od umístění znaku ke kořeni. Pokud vstupujeme do otcovského uzlu zleva, zaznamenáme si 0, pokud zprava, píšeme si 1. Po vstuu do kořene musíme zapsat získanou posloupnost do souboru obráceně. Najděme si tedy kód písmen E a Y. Písmeno E bude znamenat cestu levá-pravá, tedy 01, obráceně 10. Je tedy kódováno do dvou bitů. Písmeno Y je levá-pravá-pravá-pravá, tedy 0111, pozpátku 1110. Je tedy kódováno 4-mi bity. Celá věta bude tedy zakódována takto:
0100-10-0101-001-0110-000-0111-10-000-1100-10-1101-001-1110-1111
(V-E-N-K-U-_-J-E-_-H-E-Z-K-Y-.)

Celá věta je zakódována do necelých 7 bytů, přitom normálně by zabrala 15 bytů! Dekódování je jednoduché. Začínáme u kořene. Podle pravidla 0 vlevo, 1 vpravo se noříme hlouběji, až se dostaneme do koncového uzlu. Následující bit už je začátek kódu jiného znaku.

Pro názornost si ještě uvedeme další příklad. Máme tyto četnosti: A (1×), B (2×), C (3×), D (4×), E (5×), F (6×), G (7×), H (8×), I (9×), J (10×). Výsledný strom najdete na obrázku 3.



obr.3

Ještě jednu velice důležitou věc! Pokud budete kódovat nějaký soubor tímto způsobem, nesmíte zapomenout někde si zaznamenat strukturu stromu, protože bez něj soubor nelze rozkódovat. Nejlepší je, když si ho uložíte na začátek souboru a od dat si ho oddělíte nějakou značkou. Před dekomprimací si ho pouze „natáhnete“ a můžete pracovat.

A jaká je efektivita tohoto algoritmu? Je vhodné ho používat ke komprimaci souborů větších než 2 kB, protože tak veliký je asi binární strom (záleží na struktuře, kterou zvolíte). Pokud byste dosáhli nulové komprese (tj. žádné úspory), tak vám velikost souboru naroste maximálně o ty 2 kB, nikdy ne o více. Taková situace může nastat tehdy, když se všechny znaky vyskytují ve stejném počtu a je jich v souboru obsaženo všech 256. Schválně si zkuste napsat všech 256 znaků vedle sebe a začněte spojovat vždy dva sousední, potom spojte vždy dva takto vzniklé uzly atd. až se dostanete k jednomu uzlu, který už není s čím spojit, a to je kořen. Když si teď spočítáte „vrstvy“, zjistíte, že jich je právě 8. Proč? Zkuste si to spojit s binární aritmetikou a s binární reprezentací každého znaku. Ještě vás nic nenapadá? Tak ještě chvíli přemýšlejte.

Když tedy zhodnotíte všechny uvedené informace, zjistíte, že algoritmus Huffmanova kódování je vysoce efektivní. Můžeme s ním dosáhnout vysoké komprese dat. S tímto algoritmem by se dala provádět i vícenásobná komprese a komprimovat tak dlouho, až se dosteneme do stavu, že výsledný soubor je větší než původní (taková malá rekurze).

Tento algoritmus nemusíme nutně použít pouze pro kompresi. Velice výhodně ho lze využít například pro zakódování programu, aby nám ho nikdo nenaboural. To už však záleží jen na vaší fantazii.

Samozřejmě, že popsaný algoritmus není ještě ten zcela nejlepší. Existuje několik dalších, které jsou ještě efektivnější než tento, jako např. aritmetická komprese, vylepšený Huffmanův kód atd. ■

Packeři a packování

+GAMA

Úvodní blábol

Pokud se vám název tohoto sloupku nezamlouvá, máte smůlu. Vypůjčil jsem si ho ze starších ZXM, kde vycházel na pokračování seriál Crackeři a crackování od JSH. A o čemže to bude řeč? To neuhodnete. Já vám to ovšem prozradím. Bude řeč o packování a samozřejmě se speciálně budeme bavit o packování na Spectru. Basta. KomprimAce se dělí na ztrátovou a neztrátovou. Ztrátová se s oblibou používá na PeCi (haha, dobře jim tak), neztrátová je potom jediná z těchto dvou, kterou je možno použít i na Spectru. Ztrátovou totiž nelze použít snad ani na obrázky. Tedy šla by, ale pak samozřejmě o obrázek přijdete. Na PeCi jsou na pomrvené obrázky zvyklí a nějaké ztrátové JPeGy je už z míry nevyvedou. Neztrátová komprimAce se potom dělí na symetrickou a asymetrickou, a to podle časové náročnosti packu a depacku. Pokud pack trvá stejně dlouho jako depack, je komprese symetrická. Pokud je depack ďábelsky rychlý a pack trvá déle, je komprese asymetrická. Pakliže pack proběhne bleskově a depack trvá dlouho, nakopejte autora do prdele (příklady z praxe: MrPack, PackMaker).

Komprese se ještě obecně dělí na pozitivní a negativní, negativní je taková, při které se data nesmrsknou, ale nafouknou. Nemusí to ovšem záviset na metodě, někdy to visí i na datech. Však o tom ještě bude pokec.

Nezapomeňte ale, že ta úplně nejlepší komprese je data smazat. Cokoliv naprosto nutně nepotřebujem, smažme. Je-li nějaká hra prostě blbost, nemá cenu ji na disku nechávat. To je asi největší komprimační moudro.

Kde jsem vzal tu drzost

Ptáte se asi, proč o pakování píše nějaký blbý +GAMA. Já sám nejsem žádný teoretik oboru, spíše empirik a zkoušeč. Jak to začalo, nevím, ale prostě jsem se začal o pakátory zajímat, na počátku byla prostá touha zjistit, který je ten TOP ONE, ten THE BEST, abych ho mohl používat a ostatní abych mohl s klidem zahodit. I počal jsem pakovat a pakovat různá data různými kombinacemi packerů. Protože mi ale vycházela moc podivná čísla, začal jsem se v packerech šťourat víc. Ještě předtím ale přišel Matsoft s nehorázným požadavkem, abych napsal packer, který by mohl pakovat stránky 128ky, tedy aby depakoval jinam, než se blok nahrál. To jsme ovšem ani jeden netušili možnosti CHARPRESSu (tedy TOMpackeru). Tak tedy vznikl perfektní, ale nikoliv účinný, packer Horatius. Přiznám se, ta rutina není ode mne, přinesl mi ji Matsoft a dnes už vím, že to byl nějaký shrink (RLE metoda) od Busysoftu, já jsem spáchal jenom prostředí a ovládání, rutinku jsem samozřejmě upravil tak, aby byla relokovatelná a podobně. Upozorňuji, že relokovat neumí ani CHAR. Ale měl jsem radši vyvinout nějakou svou metodu, ta Busyho rutina od Matsofta byla fakt debilní. Další peckou byl textový packer Tolkien, který se mi dostal do rukou a nemohl jsem přijít na to, jak ho ovládat. První, co mne napadlo, bylo napsat packer vlastní, bohužel jsem asi dvě nebo tři věci moc nedomyslel a tak nepakuje tak účinně, jak by mohl. Horší šok byl, když jsem Tolkiena konečně prokoukl a zjistil jsem, jak ty tři drobnosti řeší George K., škoda, byl jsem dost blízko... Horatius i Caesar jsou na ftp://sorry.vse.cz/pub/speccy, spíš jen pro zajímavost, normálně použitelné asi nejsou.

Když už jsem se ale o packery zajímal, proškolil mne důkladně Zilog, který je skutečně odborník na věc (škoda, že tuhle slátaninu nepíše on, autor packeru, který trumfne i ruský multimetodový DSQ 4...). Hodil bych sem jeho emaily s popisem jeho packeru, ale smazal jsem je, musíte se spokojit s vysvětlením mým.

Packery se kdysi hodlal zabývat i PVL, nepříliš známý, ale neuvěřitelně schopný spectrista. Když jsem se ho ptal, proč snahy zanechal, odpověděl, že si napsal nějaký packer na obrázky, samozřejmě ne jen obyčejný shrink, ale důkladnou implozi. Packer zkracoval obrázek přímo na obrazovce, aby bylo vidět, jak se to hezky pakuje. No pak to PVL spustil. A ono to zkracovalo, zkracovalo, trvalo to už asi dvě a půl hodiny a pořád to šlo ještě zkrátit... a PVL se naštval, prohlásil, že to nefunguje, a nechal toho.

Co je lepší packovat

Pokud se divíte, že je taky možné být tak blbý a nevědět, že packovat se dá všechno, uklidněte se. Tenhle poněkud zavádějící nadpis měl nadhodit asi následující problém: Pokud si data zaarchivuji (ARJ, RAR, ZIP na PeCi), jsou sice smrsklá, ale nepoužitelná. Při použití se musí na disku rozpakovat a žerou místo, které jsem chtěl původně packem ušetřit. Pak je taky možnost spáchat samorozbalovací kód, který se nerozpakuje rozpakovat až po natažení do paměti (PKLITE na PeCi, většina Spectráckých packerů). To je samozřejmě řešení daleko výhodnější, i když se pak nemůžete v bloku svobodně šťourat, hledat texty a tak, musíte ho rozbalit (a dobře zapakovaná hra je tak celkem slušně chráněna proti crackerům, pokud tedy nepoužívají SNAP). Zabalený soubor se taky zaarchivováním moc nesmrskne, mnohdy naopak, viz třeba problém s JPeGy na PeCi.

Pokud už říkáte, že nás se to netýká, pak je to tím, že na Spectru existuje takové množství diskových systémů, až to použití archiverů brání. Tudíž jediné skutečné archivery, které jsem pro Spectrum viděl, pocházely z Ruska a byly BETASHiT only. A přitom nejsou nezajímavé. Za pozornost stojí hlavně LZ Compress, který má opravdu nádherné uživatelské prostředí ovládané šipkou (kdyby alespoň nepodporoval tu debilní ruskou myš a použil místo ní A-Mouse), a ZX Zip, bohužel jsem neměl možnost ověřit jeho kompatibilitu se ZIPem PeCoidním (ale asi to bude jen shoda názvu).

Ono abych byl přesnější, platformově závislé jsou i „obyčejné“ Ruské packery, většinou vyžadující betu. Jen abych vyjmenoval ty nejrozšířenější: LZSS (nelíbil se mi), MS Pack (Rusové si ho velmi libují, vypadá hezky a nemá špatné výsledky), Hrust (často trumfne MS Packa), DSQ 4 (DSQ znamená Data SQueezer, pochází od Code Busters) ...a to by mohlo stačit. Jak jsem už hlásal, žádný z těchto se na Zilogův packer nehrabe (sám Zilog má kontakt na autora Hrustu a získal od něj Hrust II s basicovým interfacem, aby ho mohl ozkoušet i bez Bety). Packery rozšířené po Čechách snad vyjmenovávat nemusím.

RLE, značkový bajt a bit (SHRINK!)

Předem se přiznám, nemám rád nic, co smrdí bitovou kompresí. Držím se hezky na pevné půdě celých bajtů. I když ty bajty necelé přijdou na přetřes rovněž. Ukážu vám ale, že v RLE ty bity svůj význam mají. Co je RLE? Run Length Encoding! To jste z toho chytří... Já tomu ovšem říkám shrink. Metoda je to neztrátová, jak jinak,

symetrická a obvykle ukrutně rychlá, a to za cenu nepříliš spackaných dat. Ovšem výsledek je na datech hodně závislý. Ukažme si teď ten značkový bajt. Struktura dat pak vypadá nějak takhle: Normální nespakovaná data... pak značkový bajt. To je bajt, který se jinak v bloku dat nevyskytuje, nebo se na tento účel používá bajt nejméně se v bloku vyskytující (to používá Horatius), či je daný nějak fixně a pak je důležitá další struktura dat (pakující kopírák Gargantua). Po značkovém bajtu se ukáže, co a kolikrát se opakuje. Celkem tři bajty. Pak následují další nespakovaná data. Otázka je, jak to vyřešit, když se v bloku vyskytuje značkový bajt a je osamocený, tedy to není skupina bajtů, která by se dala spakovat. Emulátor Z80 na pakování snapů používá značkový dvojbajt #ED ED xx yy, kde xx je kolikrát a yy je co, když je v bloku potom jenom jedno #ED, bere se jako nespakované. Dvě ED za sebou se spakují jako #ED ED 02 ED. Takže na spakovaná data se spotřebují ne dva, ale hned čtyři bajty. Tam už zbývá jen ošetřit situaci třeba #ED 00 00 00 00 00 00. Nabídlo by se spakovat to jako #ED (nespakovaná data) #ED ED (značka) 06 00, ale počítač by to pak bral jako #ED ED (značka) #EDkrát 06, 00. Proto se při pakování nesmí bajt za singl #ED zahrnout do bloku: #ED #00 #ED ED 05 00. Universumův kopírák Gargantua pro rychlé pakování během kopírování používá kvůli rychlosti rovněž shrink. Jako značku používá bajt 127, kde se vyskytuje 127ka jako singl, spakuje ji tak, že 127ka se opakuje 1x. Tedy prodloužení z jednoho bytu na 3. Zkuste kopírovat blok ze samých 127ček. Bude spakován na trojnásobek původní délky! V případě jiného bajtu by se vám ale každých až 126 stejných bajtů smrsklo na pouhé tři. Tož tu vidíte, jak záleží na datech. Proto je dobré dělat specializované shrinky na obrázky nebo texty (ne, to je blbost, ale k textům se ještě dostaneme), obecně to takhle moc nejde, ledaže byste měli chytrou rutinu, která by si data osahala, zjistila, co jsou zač, případně by zjistila, jaký značkový bajt je nejlépe použít. Takhle to dělá Horatius. Ale příklad si z něj neberte.

Busy při psaní svého komprimujícího kopíráku použil jiný způsob, a protože podobný používá i moudrý Zilog a je zabudován i do PK Lite a Turbo imploderu, samozřejmě s drobnými odlišnostmi, zase neuškodí, když ho trochu víc rozvedu. Ale v podstatě není ani co rozvádět. Značkový bajt jako takový není žádný bajt, ale 1+7 bitů, celkem osm. První (většinou sedmý) udává, zda následují data pakovaná nebo „blbá“, nepakovatelná. To je tedy značkový bit. Zbylých 7 udává kvantitativní množství, tedy až 128. A pak následuje buď určený počet blbců, jak je trefně terminuje Zilog, nebo jeden bajt udávající, co se má opakovat, ten se pak jmenuje „proudová data“.

Absolutně nepakovatelný blok se vám pak vlastně prodlouží o 1 bajt na každých 128 bajtů, tedy blok ze samých neopakujících se blbců se vám na kilo prodlouží o 8 bajtů. A bajty pakované jsou smrsknuty do pouhých bajtů dvou, tedy lépe, než u značkového bajtu. Jestli jste si všimli, tady je značkový bit ten, co ukazuje, zda jde o blbce, nebo packdata. No a pro pochopení ukázka: [blbci,4] blb blb blb blb [pack,4x] 4 [blbci,2] blb blb.

Pro úplnost, PK LITE a Turbo imploder používají značkový bit stejně jako Zilog. Turbo Imploder má ale rutinu o 14 bajtů kratší než PK LITE (neplatí pro další metody, tam je to naopak). U MrPacka je to s RLE těžší. Starší PackMaker používá rovněž značkový bit, ale ne sedmý, nýbrž nultý (je to o něco výhodnější, časem si ukážeme, proč). RLE nebo shrink s značkovým bitem se nemusí používat jen na celé bajty, původní RLE bylo na opakující se bity, ale je s tím pomalá a piplavá práce, rutiny jsou pak složité a stejně, podle odborné literatury pak takové spakované bity sežerou celý jeden bajt, v případě pakování RGB pouhé 4 bity, ale ten efekt asi nebude moc good. V celobajtové podobě, jak jsem ho popsal, se dá použít i na screeny, a to nejen v pakování po zpřeházených mikrořádcích, jak to známe z loadování screenu, ale i po řádcích, sloupcích, ba i cikcak nebo po dlaždicích. Koneckonců slavný Pressor VI používá 4 metody shrinku pro pixely a další 4 pro atributy... Když jsem mluvil o různě velkých paketech, jistou modifikaci používá, narozdíl od staršího PackMakeru, MrPack. Vyzkouší normální shrink bajtový (stejný jako u MRPACKu, jen rutina je o 6 bajtů delší, snad kvůli BASICMOVE) a pak se snaží pakovat dvojbajty (16 bitů, tzv. word), použije ten, který dopadne lépe, což je pokaždé nějaký jiný. Takže až budete porovnávat účinnost packerů, zjistěte si, kterou shrinkovací metodu MrPack použil. Shrink se v opravdových velkých pakátorech používá pro alokaci volného místa pro rozsáhlá data jiných metod, například pro Huffmannův strom, CCITT slovník, nebo druhý screen při pakování MrPackem. Co nesmím zapomenout zmínit je to, že použijete-li tuto metodu jako první a značkový bit u blbců vám blok při pakování prodlouží, ten vám přeteče přes pakovaná data a jste v hajzlu. Zilog to řeší tak, že nechává blok přetéct dolů, pakujete-li tedy blok od 23296 do konce paměti, stoprocentně vám přeteče do atributů, pokud pakujete od 26000 výš, musíte počítat s tím, že narozdíl od ostatních se projde i po prostotu pod 26000, pozor tedy na CLEAR 25999! PK LITE, CHAR, Turbo imploder a další se při přetečení jednoduše složí, znáte to – PACK ERROR, DATA LOST, a to i tehdy, kdy by data šla dále spakovat (škoda jich).

CCITT aneb co z PeCe nevleze do SPECCe

CCITT je řada algoritmů Consultative Commitee for International Telegraphy and Telephony, spousta lidí si je plete s Huffmannem. Jedna z jejích verzí sice Huffmannův strom používá, ale jinak je to něco jiného. Jsou to metody symetrické a kupodivu rychlé. Tím, že jsou symetrické, se liší od implodu a LemplZiva, případně LempelZivWelche. Princip je asi ten, že se vyhledají časté motivy, určuje se četnost a délka. Čím delší je motiv a větší jeho frekvence výskytu, tím kratším kódem je nahrazen. To by bylo hezké, ale, jak jsem zjistil, tabulky četnosti motivů byly jednou provždy spočítány a zveřejněny, metoda tedy není adaptivní a vyžaduje při práci slovník motivů. Výsledný kód bývá až 16krát kratší, než zdroj, nevýhodná data se ale můžou 5krát prodloužit. Na PeCi se takhle kóduje TIFF, CGM a telefonní aplikace, třeba faxy, a to zejména díky vysoké rychlosti a symetrii komprese. Jak jsem ale kdesi slyšel, do spakovaných paketů se přidávají redundantní, tedy zbytečná data, která slouží pro vzpamatování se z poruch při přenose, například každý bajt se vysílá dvakrát po sobě a podobně, takže PCčkáři si komprese stejně moc neužijou, když ji pak zas nafukujou... a to by k CCITT bylo vše, na Spectru jsem na ni doufám ještě nenarazil...

Lempel Ziv Welch (a IMPLOD!)

Zilogova geniální úprava tohoto algorytmu z roku 1977 od pánů Lempela a Ziva, zvané LZ77, roku 1984 vylepšené Welchem pro použití v diskových řadičích, se nazývá LemplZiv. A je skutečně geniální a od toho, co tito pánové zplodili původně, se i dost liší. Tak začnu obecně tím, co to imploze je. Je to metoda neztrátová. Světe, div se... a totálně asymetrická. Pack není z nejrychlejších, ale aspoň nemusíte pak při depacku čekat. Pro LemplZiv to platí jen částečně, vypadá to, že ďábelskej Zilog spáchal dost ďábelský program, takže i pack je na implozi dost ďábelsky rychlý. Spočívá ve vyhledávání stejných motivů, které se odkazují buď samy na sebe, do okna nebo do slovníku, a to i rekurzivně (i když nevím, kdo tu rekurzi používá), ovšem s tím, že slovník je plně dynamický, některé varianty metody dokonce mohou poznat pokles efektivity a příslušným způsobem samomodifikovat už existující slovník! Nezlobte se, když budu používat dost slaboduché ukázky, ale sám jsem ducha mdlého, tak abych tento článek vůbec pochopil... VODOVOD VODOPŘEVODY PŘEVODY jsou nespakovaná data. [1]o[1] [1]o[2] [2] jsou data spakovaná. Slovník je

pak takovýto: 1: VOD 2:PŘE[1]Y No a teď jsem vám doufám vysvětlil smysl rekurze, ještě jednou si prohlédněte slovník a zkuste si „rozpakovat“ spakovaná data. A jak vidíte, pro texty je to metoda vhodnější než RLE (nepočítáme-li pohádky pana Wericha a „táááááááááááááákovýhle fousy“), pack textu v Desktopu sice opravdu zkracuje (je to specializovaná RLE, jakýsi hybrid), ale ani ne moc ten text, jako spíš nadbytečné mezery, podtržítka a podobně.

Odkazování do slovníku jste snad pochopili taky, prostě máme extra text a extra slovník, dá se to použít na textové pressory, které čtou text zahuštěný, pro inspiraci otevřte knihu ASM a ZXS 1. díl nebo se podívejte na Caesara.

Odkaz samo na sebe nebo do okna vypadá přibližně následovně: V závorce je [od kolikáté pozice, kolik znaků], při počítání uvažuji v závorce dva znaky: VODO[1,3] [1,3]opře[1,3]y [11,5]. Pozice se ukazuje adresou buď absolutní, tedy skutečnou, nebo relativní, kdy se počítá od začátku bloku. Protože adresa spotřebuje moc bitů, vlastně dva bajty, trochu se to omezuje. Navíc nějaké chytré hlavy stejně zjistily, že optimum je asi čtyřkilové „okno“, je dost malé, aby nezdržovalo a dost velké, aby komprese moc neutrpěla (není to sice na první pohled tak účinné, ale zas se ušetří adresací). Zilog používá okno 4kilové, PK LITE, CHAR a Turbo Imploder mají okno jen dvoukilové a odkazují se ne od začátku okna, ale zpátky, od pakovacího kurzoru. A pro zajímavost – Caesar okno nemá vůbec, tedy lépeřečeno ho má nekonečně velké. Abyste to pochopili, ukážu vám takové desetiznakové okno: VODOVOD VODblaqwoertyu VODOVOD jsou nespakovaná data. vodo[-4,3] [-7,3]blaqwoertyu vodo[-4,3] jsou data spakovaná. Jak vidíte, druhému vodovodu už první vodovod vytekl z okna a už se na něj nemohl odkazovat, vypadá to, že úplná adresace by to ještě o znak zkrátila, ale protože adresace do okna je kratší než adresová a my tu máme adresace tři, tedy trojí zkrácení na jedno prodloužení, je výsledek vlastně supr. V praxi to tak dopadnout nemusí, ale podle hlav matematiků to za optimálních podmínek dopadne takhle. Pokud na optimální podmínky nevěříte, při psaní svého packeru vyzkoušejte obě možnosti, s oknem i s celkovou adresací.

Jak teď rozeznat pack? Nabízí se značkový bajt a skutečně to páni Lempel, Ziv a Welch dělali přes něj. Ne tak náš hodný a geniální Zilog, ten si zavedl značkový bit. Jeho metoda pak pakuje takto: Jeden bajt: 7. bit značka pro blbce, zbytek délka (až 128 blbců). Následuje patřičný počet nespakovatelných dat. Jeden bajt: 7. bit značka packu další tři bity je délka motivu zmenšená o 3 další čtyři+další bajt (12 bajtů) pozice ve 4kilovém okně (všimněte si, že 12ti bity naadresujete právě 4 kila). Sbalená data se nám pak vejdou do dvou bajtů. Když tak sbalíme trojbajtový motiv, o bajt se zkrátí. Balit kratší nemá cenu, šetříme tedy bity a těmi třemi v osmi kombinacích můžeme označit motiv dlouhý až 11 bajtů. Musím říct, že tohle se mi celkem líbí. PK LITE, Turbo Imploder a Char používají, zdá se, značkový bajt, pokud je to blbec, následuje za ním nula (tím se pack zbytečně prodlouží, bohužel), další dva bajty pak určují adresu a délku motivu, vyberte si sami, která metoda je lepší, značkový bit obecně má tu nevýhodu, že prodlužuje nespakovatelná data, ale má kratší data spackovaná, značkový bajt, pokud je to bajt, který se v bloku nevyskytuje (což za obecných podmínek zaručit nelze a je to nepravděpodobné), k prodlužování příliš nepřispívá (za obecných podmínek přispívá, viz prodlužování u shrinku), hlavně mu ale nevadí libovolný počet blbců. U LemplZiva od Ziloga se prodlužují jen blbci (8 bajtů na kilo), sbalená data jsou o bajt na každé sbalení kratší. Jde jen o to, čeho je víc. Blbců nebo sbalených dat. Abyste měli jasno, sdělím vám tajemství. Zilog data stejně skoro vždycky spakuje líp než nějaký Turbo Imploder... IMPLOD v PK LITE a CHARu je jedno a to samé, programátoři moc fantazie neukázali. Totéž platí pro Turbo Imploder, který má ale rutinu o jeden bajt delší... Další zajímavostí u těchto pakátorů je fakt, že někdy můžete implozi pustit až třikrát přes sebe, výsledek bývá ohromující, většinou ale při třetím, někdy až čtvrtém průchodu, zahlásí přetečení. Nepříjemná skutečnost, že programátoři opisovali jeden od druhého staré, byť osvědčené, rutiny, je příčina toho, proč máme na Spectru sice dost pakátorů, ale málo metod, mezi kterými bychom si mohli vybírat.

Huffmanne, Huffmanne, Huffmanne, vždycky na tě dojde

Už jsem snad říkal, že nemám rád bitové komprese a držím se pevné půdy celých bajtů. Chcete-li se stát milovníky Huffmannova kódování, což bych vám ještě nedávno nedoporučoval, musíte celé bajty opustit ať chcete nebo ne. Princip jsem nastínil už u CCITT, u Huffmanna je to v podstatě děláno tak, že si spočítáte výskyty jednotlivých bajtů (možná by šly i dvojbajty, ale zkuste si pořizovat asi tak 65535 kódů), ten nejčastější nahradíte nejkratším bitovým kódem a ten nejčastější nejdelším. Pokud jsou všechny v datech obsažené bajty přibližně stejně četné, Huffmann zklame. Ty kódy ale musejí nějak vypadat a my musíme poznat, který je který a jak je dlouhý. Ideální se jeví ukončovací sekvence (ovšem jenom jeví, už dávno se nepoužívá, tedy od té doby, co vznikl Huffmannův strom). Vybrali-li byste si jako konec řetězce bitů dvě jedničky, tedy 11, nemůžete je použít nikde uvnitř řetězce, musíte se pak spoléhat na sérii 10101, občas proloženou nulou. Délka řetězců tak brzy naroste. Pokud použijete jako ukončovadlo jedničky tři, jsou kódy zpočátku trochu delší, ale neprodlužují se s časem tak rapidně. Bohužel k vyrovnání dojde až při délkách kódu větších než 1 bajt. No prosím. To znamená, že u prvního způsobu se nám zkrátí výskyty dvaceti bajtů, u druhého pouhých patnácti, úspora je bohužel vidět až u málo četných výskytů, ale tam jsou už kódy dlouhé hodně přes bajt. Tady vidíte, že až zas tak moc supr geniální metoda to není. Vždyť jsem to říkal. Ukončovací sekvence se nepoužívají, protože máme dávno něco lepšího. Huffmann se jmenuje Huffmann podle Huffmanna, který si lámal svou matematickou hlavu tak dlouho, až vymyslel strom. Princip je asi tento: Jednička nebo nula znamená větvení doleva nebo doprava (obrazně řečeno). No a nějakou matematickou metodou se dá odvodit, jestli se ta větev na kterou se takhle dostanu, větví dál, nebo je slepá. Pokud je slepá, tak odpovídá nějakému znaku (tedy bajtu), kterému, to se přiděluje podle četnosti jeho výskytu v surových datech a délky kódu větve. Protože je Huffmannova metoda příbuzná s CCITT, vyžaduje slovník kódů, který umisťuje buď do místa vyšetřeného předchozími metodami (MrPack), nevejde-li se, tak do obrazovky (MrPack), u TREE PRESSu a PK HUFFu si adresu volíte sami, v podstatě si ale vybíráte z výše uvedených možností. Protože slovník musí být součástí rutiny a není zrovna nejkratší, má rutina MrPacka a PackMakera skoro dvě kila. PK Huff a Tree Press sice taky na slovník spotřebují 1024 bajtů plus asi 100 bajtů pro rutinu, součástí rutiny je ale krátký podprogram, který strom kódů sám vygeneruje, proto se blok skládá jen z dat a stobajtové rutiny. Zilog to dělá podobně, generuje strom, původně asi čtyřicetipatrový, ale při optimalizacích ho stále zmenšoval, zdá se, že v hotovém packeru to bude jen něco kolem 30ti, tím se značně strom zkrátí a nebude žrát tolik paměti. Ještě jednu výhodu Zilogova rutina má. Je drsně rychlá. Pomalu ani nevěříte, že je to Huffmann. Nevěříte-li, nahrajte si jako ukázku MGSuxx, upozorňuji ale, že tady je depakování zpomaleno, aby se dalo lépe sledovat, co to vlastně dělá. Nepletu-li se, Matsoft sliboval, že má krásný popis nějakého konkrétního Huffmanna, modlete se, aby ho do ZX Magazínu dal, možná z toho pak budete moudřejší. Tree Press a PK Huff jsou úplně jedno a to samé, liší se jen trošku vzhledem, třeba Tree Press nemá ukazatel úspěšnosti, ale program je vlastně tentýž. Jsou plně interaptovatelné, jako imploze, což se ale nedá říct o MrvPacku. PK Huff A Tree Press mají jednu velkou nectnost.

Strašně rády hlásí PACK ERROR, DATA LOST. Výkonem je sice huffmann z Packmakera předběhne, ale ten nejde použít samostatně bez shrinku. Jak se liší MrPack a Packmaker? MrPack má rutiny o 6 bajtů delší než Packmaker, použitelné z něj je ale Basicmove, to jiné packery nemají, a pokud chcete použít loader vytvořený Pressorem VI, musíte použít i SAVE z MrPacku (nemusíte v něm pakovat, stačí skutečně jen SAVE), které přidá do hlavičky souboru data pro loader. Co je na packerech od Mixoftu zajímavé, jsou strašně prasecké rutiny (i když je v nich pár zajímavých nápadů), které se nesnášejí s programky pod přerušením. Rutina se ukládá defaultně na zásobník, prostě počítá s tím, že je tam pro ni místa dost). Nejzajímavější ale je, že pack netrvá až tak dlouho (mluvím o PackMakerovi), zato depack (a teď mluvím o Huffmannovi z PackMakeru) dlouho trvá. Ona holt práce s bity a neustálé rotace zdržují a je to pomalé.
K chvále Ziloga musím podotknout, že on to takhle pomalé nemá. Poslední poznámka k Huffmannu, a mám tím na mysli MrPack a PackMaker. Jsou sice účinnější než jiné Huffmanny (mysli Tree Press a PK HUFF, ne Zilogův packer), ale na implozi prostě nemají ani co do výkonu, ani co do rychlosti depacku. Takže imploze Huffmann 1:0. Připsal bych ještě něco o fraktální kompresi, která je žhavou novinkou ve světě PeCí, ale je ztrátová, tak se jí my zabývat nebudem...

Strašidelná otázka

Říkal jsem, že mne k zájmu o packery hnala otázka, který používat. Abyste nemuseli hledat sami, dám vám odpověď. Píšete demo, data se vám do paměti naráz nevejdou, potřebujete pakovat, ale nechcete vypínat IM 2 a nějaké ty efekty. Přitom chcete, aby depack nebyl bleskový, abyste během depakování mohli nechat běžet nějaký efekt (viz Dies Irae, například). Tady se dá použít PK HUFF. Ve všech ostatních případech použijte PK LITE. Imploze je rychlá a rovněž plně interaptovatelná. Tady máte dvě možnosti: nechat proběhnout shrink a pak až do omrzení prohánět blok implozí, nebo se na shrink vykašlat a prohnat blok více implozemi. Doporučuji pouštět imploze po jedné a blok sejvovat (co kdyby se vám packer při příští implozi sesypal), při dalším průchodu nezapomeňte, že blok se zkrátil (v odpovědi na dotaz Lenght). Nebojte se, že je to nevýhodné, kdybyste dali více průchodů automaticky, byl by výsledek stejný, navíc Turbo Imploder víc průchodů neumí a musíte je datlovat ručně. Pozor na shrink z Turbo Imploderu, občas špatně depakuje, asi díky o 14 bajtů zkrácené rutině. Vyvarujte se tedy samotného shrinku. U PK LITE to neplatí, tam je shrink naprosto bezpečný (málokdy ale potřebujete samotný shrink). Potřebujete-li blokem nějak hýbat, použijte CHARpress a volbu Modify depacker. CHARpress umí nahrávat soubory, hodí se tedy pro pakování her, které jsou rozprostřeny od 23296, defaultní umístění depakovací rutiny pak musíte přesměrovat buď do volného místa (vždycky nějaké je, v paměti přece bývají loadery, zbytky basicu a tak), do obrazovky jen v nouzi (umisťovat do VRAM jakékoliv depakovací nebo LDIRovací rutiny je prasárna). Potřebujete zobrazit panel (druhá obrazovka MrPacka a PackMakeru). Pressor VI a Packmaker s volbou 2nd screen je jen pro mentáčky. Šikovnější zapakuje obrázek CHARem (depack je pak rychlejší než z PRESSORu a i výsledný blok je kratší, jen pozor na ztracenou relokovatelnost) a přilepí ho do místa uvolněného prvním průchodem packu hlavního bloku (autostart je dobré nastavit na depack hlavního bloku). Výsledek se většinou dá ještě zdrcnout implozemi (autostart na depak obrázku).

Potřebujete balit textovku nebo program, který obsahuje basic. Nelze-li kompilovat (Softek FP umí kompilovat i kazetový LOAD a SAVE), použijte Basicmove v MrPacku, ale nepakujte. Blok pak nějak dostaňte do PK LITE (většinou ho nemusíte ani sejvovat, stačí nahrát PK LITE a nadatlovat správná data).

Na normální obrázky používejte radši PRESSOR kvůli relokovatelnosti (úvodní obrázky ale stejně umisťujte na 4E4).

Chcete něco zapakovat velmi rychle, ale chcete, aby se uživatelé zbláznili při čekání na depack. Pak použijte PackMaker...

Pozor ale! Tohle bude platit jen do té doby, než Zilog dokončí svou KomprimAci! Vzhledem k tomu, že má mít zabudovánu i speciální podporu packu obrázků, je možné, že většinu starších packerů budete moct zahodit daleko, předaleko... Před čím vás tedy budu varovat? Já to sem už napsal, ale stejně to zopakuji. Vsssžžžž! Slyším MrvPacka, vidím MrvPacka! Tady se podepsal zas někdo neinteligentní. Říkám, imploze je lepší jak časově, tak výkonově... Na obrazovce problikl úvodní obrázek, teď už je vidět jen tma a v dálce je tušit depack. Pokud si VRAMku dobře prohlédnete, najdete tam depakovací a LDIRovací rutiny. Nechci nikoho pomlouvat, ale je to častá vizitka JSH. Klackson Hollis se už asi nepolepší, ale vy ho v tomhle radši nenásledujte.

Zpackaný text

Ano! Spáchali jste text a chcete ho zabalit. RLE na pakování textů není vhodná, to už víme. Huffmann se nehodí. CCITT by bylo na místě, ale pokud vím, Spectristé ho neprovozují (škoda). Tolkien je specializovaný implod. Tak a teď víte skoro vše. V textech se především nevyskytují, jako v obecných blocích, všechny znaky. I s češtinou je písmenek méně než 128. Budete-li se snažit narvat text do 7mi bitů, ušetříte na 8mi bajtech jeden bajt, na kile pak 128 bajtů (87,5%). Není to špatné, ale není to zas tak výkonné. Dal by se použít značkový sedmibit, ale dělejte se s implodem v neustálých rotacích... Vraťme se radši k celým bajtům. Když jsem páchal Caesara, srazil jsem češtinu (desktopáckou, jak jinak) do kódů 1 až 16, ENTER skonvertoval na 31 (mám aspoň ten dojem) a krom několika nevyužitých bajtů (škoda jich, ale to jsem zjistil až dodatečně) se mi vše vešlo od 1 do 127. Nula znamenala oddělovník v tabulce slovníku. Kódy 128–255 nesly (krom značkového bitu) i číslo, které znamenalo pořadí slova ve slovníku. Vlastní pack pak pracoval asi takto: [1]o[1] [1]o[2][2]0vod0pře[1]y0.

Pokud nevíte, co jsem to pakoval, byl to náš cvičný text z kapitoly o implodu. Nejprve je vlastní text, pak nula, první slovo slovníku, nula, další slovo slovníku, etc. V podstatě tedy implod se slovníkem. Jediná chyba byla ta, že v zájmu nečitelnosti textu jsem pakoval i takové kombinace, které se packem nezkrátily, ale tabulku okupovaly. Vidíte, že jsem měl místo na 128 odkazů. Takhle se rychle zaplnily a účinek nebyl tak velký, jaký by mohl být. George K. to udělal fikaněji. Všechny znaky si srazil do intervalu 0 až [počet znaků]. Potřeboval sice tabulku navíc, ale nešť. Písmena v textu pak nahradil jejich pořadovým kódem v tabulce a právě tím se stal text nečitelný, i kdyby ho nikdo nepakoval. Vzhledem k tomu, že málokdy máte v textu 127 různých písmen, měl víc odkazů na slovník než já a tím i lepší kompresní poměr (navíc pakoval skutečně jen to, co text zkrátilo, a neplýtval tak zbytečně odkazy). Proč se u textů používá slovník a u normálního implodu ne? LemplZiv (píšu teď opět o Zilogově modifikaci) nemá volnou paměť, kterou by slovníkem mohl obsadit, protože slovník musí být k dispozici až do konce depaku. Zato texty zůstávají v podstatě stále zabalené a slovník nikde nepřekáží, naopak odkazy samy na sebe (ne-li dokonce do okna) narážejí právě na to, že se nerozbalují a odkážete-li se na něco, co se za chvilku zabalí a už se to nerozbalí, dost tvrdě ztroskotáte. Pokud jste četli implod pozorně, všimli jste si, že se obvykle odkazuje do části, která je při depaku už depakovaná.

Crack!

Crackujete-li pakované hry, máte to jednoduché hlavně u MrPacka. Ten se totiž vždy vrací přes return. Volá-li se pack ze strojáku, je to jen obyčejný CALL. Většinou ale stačí hru po depacku BREAKnout a jste uvnitř (skoro, ještě se totiž může

všelijak LDIRovat...). U PK LITE a odvozenin bývá nastavený autostart. Pokud rutina začíná LDIRem, je to implod a vám stačí jet po kódu tak dlouho, dokud nenajdete 3x za sebou RRCA. Za chvilku přijde jeden JR NZ a hned za ním je příslušný JP, který spouští hru nebo další průchod. Začíná-li rutina JUMPem na konec spakovaných dat, je to shrink. Pokud se hned v úvodu pushuje hl, bude se pravděpodobně skákat na adresu v něm uloženou. Nepushuje-li se hl, bude se asi za chvilku pushovat de a to znamená, že rutina skáče normálně na začátek depakovaného bloku. Rutiny si ještě ukážeme. Není nic jednoduššího, než Devastací změnit JP na RET (201) nebo PUSH na NOP (0). Pak nechte proběhnout depack a podívejte se, zda se chystá další průchod, nebo zda je už hra rozpakována. Hotovo. Jak potom cracklou hru packovat, to už snad vysvětlovat nemusím...

Jak má depack vypadat

Mluvil jsem o PUSHích čehosi, na co se pak skáče RETem. I MGSuxx začíná instrukcemi LD HL,xxx:PUSH HL:LD HL,yyy:PUSH HL:LD HL,zzz:PUSH HL. Tím se nastaví volání dalších průběhů a startu programu, příslušné adresy se ocitnou na zásobníku a rutině pak stačí mít v sobě už jen RET a o víc se nemusí starat. Pokud na závěr nic nepushneme, program hru nespustí, ale vrátí se. Ne vždy je to ale výhodné. Lze to dělat, plní-li se nám nějaký registr číslem adresy, na kterou si přejeme skákat. Tím ušetříme 1 bajt. Pokud ale musíme registr plnit uměle, prodlužujeme rutinu o 2 bajty (v případě MGSuxx, kde jsou volání tři, je rutina proti třem JUMPům prodloužena o 6 bajtů).

```
        LD DE,16384
        LD HL,40000
        LD BC,6912
        LDIR
        JP 16384
```

To je asi 13 bajtů. Střední hodnota.

```
        LD DE,16384
        PUSH DE
        LD HL,40000
        LD BC,6912
        LDIR
        RET
```

A tohle je jen 12 bajtů. To 16384 bychom do dvojregistru DE cpali tak jako tak.

```
        LD HL,50000
        PUSH HL
        LD DE,16384
        LD HL,40000
        LD BC,6912
        LDIR
        RET
```

A tohle je 15 bajtů. Ne že by na tom záleželo. Je to jen detail, ale tady záleží na každé maličkosti. Chcete, abych vám ukázal, jak takovéhle drobnosti úplně znemožňují jeden známý packer? Fajn, ještě chvilku vydržte.

Takhle vypadá imploze z PK LITE nebo CHARu:

```
START   ld hl,RUTINA
        ld de,DEPRES
;obvykle se rutina přesune na 234800 nebo 23500, tedy pod systémové proměnné. U Turbo Imploderu umístění rutiny nemůžete ovlivnit, u ostatních ano.
        push de
;tady se využije toho, že adresu tak jako tak v de máme.
        ld bc,RUTLEN
        ldir
        ret
;proti tomuto skoku není námitek.
RUTINA  ld de,START
        ld bc,DATALEN
        ldir
;v hl se ukazovalo na data.
        ld hl,DATAEND
        ex de,hl
;depakuje se odzadu, v de je teď konec depakovaného bloku, v hl konec pakovaných dat.
NEDEP   dec hl
        dec de
        ld a,(hl)
        ld (de),a
        sub 137
        jr nz,NEDEP
;ta 137 není povinná, je to prostě značkový bajt.
        dec hl
        or (hl)
        jr z,DEPAK
;pokud následuje 0, nebude se pakovat, protože to byla 137ka skutečná.
```

Následuje část, kterou ani nebudu popisovat. Z následujících dvou bajtů se vyfiltruje začátek a délka odkazu a lddrne se. Součástí této části jsou i ony nápadné tři rrca. Okno je na rozdíl od Zilogova LemplZiva pouze dvoukilové (adresované 11ti bity). Délka odkazu je zmenšena o 3 (je tu proto add a,3). Je doufám jasné, že odkazy směřují do už rozpakované části. O tom, že lepší než značkový bajt by byl značkový bit, jsem už psal. Trochu nepochopitelné je, že shrink z toho samého packeru značkový bit používá. Stejně tak je zajímavé, že rutiny shrinku a implodu jsou jinak organizovány, ačkoliv je snad psal jeden a ten samý autor...

```
DEPAK   sbc hl,de
        add hl,de
        jr nz,NEDEP
        jp START
;do tohohle jumpu se píše startovací adresa. V případě, že je stejná jako START, bylo by lepší mít za návěštím RUTINA, kde LD DE,START máme, PUSH DE a tady RET.
DATABEG
;následují pakovaná data.
```

To nebyla věru špatná rutina. V CHARpressu, když si dáte Modify depacker, se vám číslo, které dáte jako Unpacked at, uloží za návěští RUTINA. Protože se LDIRuje a ne LDDRuje, můžou se bloky překrývat, pokud jimi hýbete dolů, ale nesmí se překrývat, pokud jimi hýbete nahoru. Nevyplácí se na to zapomínat. Asi jsem na to měl upozornit už dříve.

Jak nemá vypadat depack

Slíbil jsem znemožnit jeden packer. Podívejte se nejdřív, jak deshrinkuje PKLITE.

```
START   jp RUTINA
;zbytečné tři bajty navíc. Lepší by bylo umístit na začátek rutinu rovnou a až pak data
DATA    defb 0
        defb 228
;blok se skládá ze sta nul (228-128 je 100), data jsou ukládána odzadu.
RUTINA  ld hl,RUTIN2
        ld de,DEPRES
        ld bc,RUTLEN
        push de
;teď už víte proč...
        ldir
        ld hl,DATA
        ld de,START
        ld bc,DATALEN
;zbytečné plýtvání místem, všimněte si, že narozdíl od implodu tu rutina nemůže použít starý obsah registrů. Evidentně je tedy výhodnější umístit nejprve rutinu a až pak data.
        ldir
        ld hl,PCKEND
        ld de,DPCKEND
        ld bc,DPCKLEN
;depakuje se odzadu, ale to už jsem už jednou nastínil.
        ret
RUTIN2  ld a,(hl)
        rlca
        srl a
;tímhle se do CARRY hodí značkový bit (kdyby byla data „předrotovaná“ RLCAčkem, stačilo by SRL, zas by se bajt ušetřil).
        dec hl
NEDEP   ldd
        ret po
;v tomto případě se rutina normálně vrací a nic se nevolá.
        dec a
        res 7,a
        jr z,RUTIN2
;po rozbalení určitého počtu proudových dat nebo určitého počtu blbců se zas skáče na čtení značkového bitu.
        jr nc,NEDEP
;blbci mají značkový bit 0. Pokud se depakuje, platí c.
        inc hl
        jr NEDEP
```

A to je všechno. Sama rutina je celkem krátká. Dala by se ještě vylepšit, to ano, ale asi by to musel dělat šikovně chytrý program, který by ji modifikoval tak, aby byla pokud možno co nejkratší.

A teď tu trapnost. Porovnejte prosím se shrinkem PackMakeru, metoda je sice prakticky stejná, ale metody hrůzně různé.

```
START   jp RUTINA
```

```
;jako předtím, veskrze zbytečné 3 bajty.
DATA    defb 0
        defb 201
;zkuste si to zrotovat doprava, vyjde vám
228 jako u PK LITE. Je hezké, že má Pack-
Maker značkový bit předrotován, ušetří tak
jednu instrukci. Bohužel tento náskok ztratí
na jiných, odporných a zbytečných kravi-
nách.
RUTINA call 82
;ne, rutina není relokovatelná. Já si myslím, že
se sem občas poukuje zavolání depacku dru-
hého screenu. Protože se ale panel využívá
jen výjimečně, jsou to další zbytečné tři bajty.
        ld hl,81
        push hl
        ld a,i
        jp pe,RUTIN2
        ld a,i
        jp pe,RUTIN2
        pop hl
;pokud platí PE, zůstane nám na zásobníku
hnusné číslo, ukazující do ROMky na in-
strukci EI (kterou následuje RET). Tahle
snůška instrukcí testuje stav přerušení
(IFF2). Já ale nepředpokládám, že bych byl
uvnitř NMI rutiny (vy snad ano?), podívejte
se, co ten vůl vyplácal prostoru. Necpat nic
na zásobník, odpadlo by mu celé složité tes-
tování DI a EI. Proč se čte stav přerušení
dvakrát, taky nevím, asi to není moc spole-
hlivý způsob a tak to radši zkouší dvakrát,
ale podle toho, co o tom vím, by jednou sta-
čilo, ušetřily by se asi čtyři bajtíky. Při povo-
leném přerušení je ho třeba obnovit, při za-
kázaném se neobnovuje nic, PackMaker se
totiž o zakázání postará sám.
RUTIN2 di
;normálně by to nebylo třeba, ale tady je to
venkoncem nutné. Proč? Protože zásobník.
        ld de,START
        ld hl,DATA
        ld bc,DATALEN
        ldir
        dec de
;tohle známe. V de teď máme ukazatel na
konec spakovaných dat (a zas depakujem
od konce).
        push de
        scf
        sbc hl,hl
        add hl,sp
        ex de,hl
        ld hl,RUTEND
        ld bc,RUTLEN
        lddr
        ex de,hl
        inc hl
        ld de,DPCKEND
        ld bc,DPCKLEN
        jp (hl)
;hrůza všech hrůz, odsune rutinu na zásob-
ník. Ta má sice 16 bajtů, ale nezdá se mi to
být moc geniální. Vidíte, kolik to jenom té
nebohé paměti užírá? Proto je PackMaker a
MrPack naprosto neinteraptovatelný. Pře-
psala by se mu depakovací rutina.
        pop hl
;tady se obnoví to odložené de. V podstatě
od toho dec de až sem to je snůška kravin.
Stačilo pár instrukcí a nemuseli jsme se in-
teraptu bát.
NEDEP   ld a,(hl)
        dec hl
        srl a
;tady jsme oproti PK LITE jednu instrukci
ušetřili (rlca nám udělal už packer).
DEPAK   ldd
        ret po
;tady se buď program vrátí, nebo skočí na
EI a pak teprve se vrátí. Skákat nikam neu-
mí. Kdybyste ale přepoukovali tu 81, skákal
by podle toho, co byste mu tam napískali.
        dec a
        jr z,NEDEP
        jr nc,DEPAK
;blbci mají značkový bit 0, jako u PK LITu. A
proč ne, když je to vlastně rutina z PK LITE
upravená nějakým amatérem.
        inc hl
        jr DEPAK
```

V podstatě se dá říct, že autor zase tradiči nezklamal a obšlehl, co se dalo. Metoda je ta samá, vlastní depakovací rutina až na dvě instrukce (RLCA a RES 7,A) jakbysmet, navíc je tam akorát ta hrůza se zásobníkem a hrátky s přerušením.

Zdravím vás a doufám, že se příště budeme otravovat zase s něčím jiným. ■

Loader s počítadlem pro MDOS

+GAMA

Tak co, uživatelé D40, pamatujete se ještě na ty překrásné kazetové loadery s počítadly??? Pokud jste si hned po této první větě řekli, že něco takového musíte okamžitě spáchat, vězte, že něco už jsme udělali za vás. Před vámi se placatí hrůzovina nejzacyklenější, mající za účel natáhnout z disku do paměti blok bytes, např. hlavní část pakované hry. Od LOADu z ROM D40 se liší tím, že se snaží během práce zobrazovat vpravdě křovácké počítadélko. Výpis by měl být komentovaný, uvedu tedy to, co se tam už nevešlo. Vlastní loader by měl být umístěn tam, kde ho nic nepřehraje, podle mého v nejvyšší části paměti. Pakovaná hra by ho tam během nahrávání neměla přepsat a když už loader nebude potřeba, vaše oblíbená pařba se přes něj jednoduše rozpakne. Další místo je třeba na FATku, která se do paměti nahraje dříve než blok. Tu však můžeme blokem přepsat, a to také doporučuji. Umístěte ji někam, kde se později bude blok vyskytovat, a je po problémech. Nejhorší je to asi s 200 bajty mapy souboru, která se normálně ukládá do DRAM, ale myslím si, že polohou a velikostí nám skvěle poslouží printbafr. Hlavně si hlídejte, aby byla v paměti přítomna během nahrávání (bez ní to prostě nemůže fungovat...). Tyto tři smrtelně důležité hodnoty se definují za příkazem ORG a u návěští FATKA a ZASOBA. U návěští JMENO je jméno nahrávaného souboru, ZACATEK je adresa, kam se soubor bude nahrávat. Nejdůležitější je samotný tisk počítadla za návěštím PRD, s tím si hrajte – svěřuji ho vaší programátorské intuici, snad vám vznikne něco hezkého, jako obzvlášť dobrý nápad doporučuji ručičkové (analogové) hodiny... Poslední poznámka – volané strojové rutiny fungují i při použití MDOSu 2.0 (tj. už i na Kompaktech to musí fungovat).

```
        org 65000       ;Tady to začíná.

YEDEME  ld a,2          ;Otevřít kanál pro tisk
        call #1601      ;počítadla do obrazovky
        ld hl,ZACATEK   ;a začátek souboru uložit
        ld (LDADR+1),hl ;do hloubi programu.
PAGE    ld a,#4F        ;Přestránkujeme na
        ld de,TAB-26;   ;DROM.
        call #25AB
        ld hl,0
        ld (TAB),hl
        ld hl,#3EF7
        ld (TAB+2),hl
        rst 0
CMUCHEJ call 7311       ;Atakujeme disketu.
        ld hl,FATKA     ;Na známou adresu si
        ld de,5*256+1   ;uložíme FATku.
        ld bc,1
        call 8866
FINDFILE ld hl,JMENO    ;Fajl s tímto jménem...
        ld de,16010
        ld bc,11
        ldir
        call 8491       ;...zkusíme najít.
        jp nz,NENITU    ;Odskok, není-li tady.
        ld (15986),hl;  ;Délku souboru...
        push hl
        pop ix
        ld d,(ix+12)    ;... kterou sebereme
        ld l,(ix+17)    ;tuhle, s prvním sekto-
        ld h,(ix+18)    ;rem souboru...
        ld ix,ZACATEK
        srl d           ;...vydělíme 512...
```

```
        inc d           ;To je počet sektorů,
        ld a,d          ;který by bylo hezké si
        ld (POCIT+1),a  ;uložit...
        call PRD        ;...a vytisknout.
        ld bc,ZASOBA
DF1     push hl
        call FDBLOCK    ;Hledáme další sektor.
        ld hl,3072      ;Test na prázdný soubor.
        sbc hl,de
        ret z           ;Soubor má délku 0, návrat
        ld a,d          ;Test na poslední sektor
        cp 14           ;souboru.
        jr nc,POSLEDNI
        pop hl          ;Číslo sektoru si uložíme.
        ld (ULOZ+1),bc
ULOZ    ld (0),hl
        inc c
        inc c
        call FDBLOCK    ;Hledáme další sektor
        ex de,hl        ;souboru,
        jr DF1

POSLEDNI xor a          ;uděláme si koncovou
        ld (bc),a       ;značku (dvoubajt 00)
        inc c
        ld (bc),a
        inc c
        pop hl          ;a za ni uložíme číslo
        ld (ULOZ1+1),bc ;posl. sektoru v souboru.
ULOZ1   ld (0),hl
        ld h,b
        ld l,c
        inc l
        inc l
        ld (hl),e       ;Číslo sektoru, který
        inc l           ;se teď bude ládovat
        ld (hl),d       ;z disku,
        ld bc,ZASOBA    ;vyzvedneme z mapky.
LOAD    ld a,(bc)
        ld l,a
        inc c
        ld a,(bc)
        ld h,a
        inc c
        or l            ;Je-li to dvounula, násle-
        jp z,POSLEDN1   ;duje už jen posl. sektor.
        push bc         ;Zbavíme se ukazovátka
        push hl         ;a čísla sektoru,
        ld a,(POCIT+1)  ;číselník snížíme
        dec a           ;o jedničku
        ld (POCIT+1),a
        call PRD        ;a zobrazíme si ho.
        call DIVIDE     ;Číslo logického sektoru
LDADR   ld hl,0         ;se přepočte na stopy
        ld de,257       ;a fyzické sektory,
        call 8866       ;aby se mohl načíst.
        ld hl,(LDADR+1)
        inc h           ;Zvětšíme si adresu o 512
        inc h
        ld (LDADR+1),hl ;a napoukujeme!
        pop hl          ;Obnovíme
        pop bc          ;ukazovátko
        jp LOAD         ;a jdeme na další sektor.

POSLEDN1 ld h,b         ;Po koncové značce si
        ld l,c          ;načteme číslo posled-
        ld c,(hl)       ;ního sektoru.
        inc l
        ld b,(hl)
        inc l
        ld e,(hl)
        inc l
        ld d,(hl)
        push bc
        xor a
        ld (POCIT+1),a  ;Opět snížíme počítadlo
        call PRD        ;a vytiskneme číselník.
        ld a,d
        and 1           ;Ořízneme nepotřebné
        ld d,a          ;bajty ze sektoru
        or e
        jp nz,CHYC
        ld d,2
CHYC    pop hl
        push de
        call DIVIDE     ;ten si přepočítáme na
        ld hl,14848     ;fyzický a do diskové RAM
        push hl
        ld de,257
        call 8866       ;ho načteme.
        pop hl          ;V potřebné délce
        pop bc          ;ho přeneseme
        ld de,(LDADR+1) ;tam, kam patří.
        ldir            ;Nahráno, vypnout motory
        jp 737          ;a odstránkovat. Návrat!

DIVIDE  ld a,(15979)
        call 8620
        ld e,(ix+3)
        ld d,0
        ld b,-1
        or a
DV0     inc b
        sbc hl,de
        jr nc,DV0
        add hl,de
        ld c,l
        ret

FDBLOCK push bc
        ld bc,FATKA-512
        ld de,341
        or a
        LESS inc b
        inc b
        sbc hl,de
        jr nc,LESS
        add hl,de
MORE    ld e,l
        ld d,h
        srl d
        rr e
        ex af,af'
        add hl,de
        add hl,bc
        ld e,(hl)
        inc hl
        ld d,(hl)
        dec hl
        ex af,af'
        jr nc,NODD
ODD     ld a,e
        ld e,d
        jr BOTH
NODD    ld a,d
        rrca
        rrca
        rrca
        rrca
BOTH    and 15
        ld d,a
        pop bc
        ret

NENITU  ld a,7          ;Pokud se námi hledaný
SRS     out (254),a     ;fajl odmítl nalézat na
        dec a           ;tomto disku, sršíme
        or a            ;a čekáme, až chudák
        jp nz,SRS       ;uživatel vymění disk
        ld a,127        ;a zmáčkne mezeru.
        in a,(254)
        rra
        jr c,NENITU
        jp CMUCHEJ

PRD     push hl         ;Tisk číselníku je tento:
        push bc         ;Aby se neměnily registry,
        push de         ;tak je uložíme.
        ld a,22         ;Print AT 20,14
        rst 16
        ld a,20         ;(souřadnice si
        rst 16          ;klidně změňte).
        ld a,14
        rst 16
POCIT   ld a,0
        ld d,100        ;Oddělíme stovky.
        ld e,0
STO     sub d           ;Odečti sto.
        jp c,NESTO      ;Přehnal jsi to?
        inc e           ;Ne - jedna 100 k dobru.
        jp STO          ;Zkus další.

NESTO   add a,d
        ld hl,CIF1+1    ;Stovky naládujem do
        ld (hl),e       ;programu.
        ld e,0          ;jdeme na
        ld d,10         ;desítky.
DESET   sub d           ;Odečti deset.
        jp c,NEDESET    ;Přehnal jsi to?
        inc e           ;Ne - taky dobrá.
        jp DESET        ;Každá desítka se hodí.

NEDESET add a,d
        ld hl,CIF2+1    ;Teď naládujeme desítky
        ld (hl),e
        ld hl,CIF3+1    ;a jednotky, co nám po té
        ld (hl),a       ;machinaci zbyly.
CIF1    ld a,0
        add a,48        ;A vyplivnem stovky
        rst 16
CIF2    ld a,0
        add a,48        ;a desítky
        rst 16
CIF3    ld a,0
        add a,48        ;a jednotky!
        rst 16
        pop de          ;Tak číselník je,
        pop bc          ;teď už jen obnovit si
        pop hl          ;registry
        ret             ;a vrátit se v program.

FATKA   equ 25000       ;Místo v paměti pro FAT
                        ;(2,5 kB).
ZASOBA  equ 23296       ;Místo pro mapu souboru
                        ;(200 B).
JMENO   defm „FileName“ ;Jméno fajlu doplněné
        defb 0,0,“B“    ;nulami a příponou.
ZACATEK equ 25000       ;Počáteční adresa fajlu.
TAB     defw 0          ;Tabulka pro přestránko-
        defw #3EF7      ;vání do diskové ROM.
END
LENGHT  equ END-YEDEME
```

■

Seriál

Konečně pořádně o D40 podruhé

Petr Žabenský

Dneska si řekneme, jak to vlastně v ROM D40 funguje, hlavně o přechodu ze ZX ROM do ROM D40. Jak už všichni víte, jsou vstupními body do ROM D40 adresy 0 a 8. Pokud obsahuje registr PC hodnotu 0 nebo 8, je místo ZX ROM připojena ROM D40. Tyto dva vstupní body ale musíme rozlišovat. Adresa 0 slouží pro speciální operace: RESET, SNAP, návrat z rutiny ZX ROM, návrat pro tisk chybového hlášení, čtení a zápis znaku z/do sekvenčního souboru. Adresa 8 je obsazena interpretem BASICu, kdy po zavolání RST 8 ze ZX ROM je přestránkováno do ROM D40, kde se zpracovávají příkazy pro disketovou jednotku.

Nejdříve něco o RST 0. Možná vás zarazily ty možnosti, ale je to skutečně tak. Co se tedy děje po přestránkování? Nejdříve se skutečně uloží stav přerušení v době volání skoku na adresu 0, uloží se obsahy registrů a kontroluje se tabulka na adrese #3EEF. Pokud bude narušena, provádí se reset počítače. V dalším kroku se kontroluje obsah adresy #3EF7, jestli se neprovádí návrat z rutiny ZX ROM (#4F) nebo chybového hlášení (#45). Pokud je zde hodnota #4F, jsou obnoveny registry a provede se návrat zpět do volajícího programu s tím, že zůstane přistránkována ROM D40. Při hodnotě #45 jsou obnoveny registry, potom se provede část programu na adrese #145, který je opsán s malými změnami ze ZX ROM a provede se buď tisk hlášení MDOSu (#185) a návrat do ZX ROM, nebo se přímo vrátíme do ZX ROM, kde se tiskne hlášení ZX ROM. V obou případech je ale původní hodnota přepsána hodnotou #20, tedy žádná činnost. Pokud ale není žádná z těchto hodnot, začne se kontrolovat tabulka povolených návratových adres (předtím je ještě takový malý pozůstatek z ladící rutiny, ale tím se nebudeme zabývat).

O co vůbec tímto jde? Když voláme CALL 0 nebo RST 0, je na zásobník uložena návratová adresa. My si tuto adresu vyzvedneme a porovnáme ji s povolenými návratovými adresami. Pokud nebude v tabulce, provede se reset. Začátek tabulky je na adrese #19A a obsahuje celkem 5 položek. První slovo je návratová adresa, druhé slovo je adresa programu, kam se bude skákat, pokud návratová adresa souhlasí. Tabulka je zakončena slovem 0. Jsou zde adresy programů pro čtení a zápis znaku z/do sekvenčního kanálu a SNAPu. Nyní si asi řeknete, co je to za nesmysl?!

Tak nejdříve SNAP. Jistě jste si již všimli toho tlačítka na konektoru, který je připojen na sběrnici. Ano, je to tlačítko SNAP. Nebudu znovu popisovat, co to je SNAP. Důležité je, že po jeho stisku se vyvolá NMI. Autor článku v ZX Magazínu 3&4/1994 má pravdu, že se na adresu #66 vnutí instrukce RST 0, která vyvolá skok na adresu 0. Takže to není nic neobvyklého. Zastavíme se u čtení/zápisu znaku z/do sekvenčního kanálu. Je to velice jednoduché. Pokud se podíváte na struktury kanálových dat, zjistíte, že pro každý kanál je vyhrazeno 5 bytů. První 2 byty je adresa rutiny pro výstup, druhé dva pro vstup a poslední je kód jednopísmenného názvu kanálu. Nyní ještě zůstává rozlišit vstup a výstup do kanálu.

Je jasné, že se jako adresa nemůže do obou uložit 0 (zkuste přemýšlet proč). Ale stačí najít v ZX ROM instrukce RST 0 (v komentovaném výpisu ZX ROM je nenajdete, ale jsou tam), jeden bude pro vstup, jiný pro výstup. Adresy těchto instrukcí si uložíme jako vstup a výstup a podle návratových adres potom rozeznáváme, jakou operaci budeme požadovat. Ale abych se ještě trochu opravil. Kdo už trochu zkoumal práci se sekvenčními soubory, tak ví, že se pro kanál se sekvenčním souborem nevyhrazuje pouze 5 bytů, ale více (nechci přijímat dopisy, ve kterých mě upozorňujete na tuto dezinformaci, proto to zde uvádím, na principu se ale nic nemění). Přesný popis práce se sekvenčními soubory si ale popíšeme jindy (ono je to trochu složitější). Více se toho k adrese 0 snad ani říci nedá.

Za zmínku ještě možná stojí popis stránkovací rutiny od pana Svozila, publikované v ZX Magazínu 6/1993. J. Flaška to popisuje jako simulaci BASIC příkazu POKE. On ten pojem „simulace“ je trochu zavádějící, protože to není přímo interpretace příkazu POKE, ale výsledek těchto operací je stejný. Tento prográmek využívá principu práce s proudy (kanálovými, ne mořskými) a principu zápisu znaku do sekvenčního souboru.

Ale popořádku. Když se provádí výstup znaku na kanál, je voláno RST #10. Vyzvedne se adresa začátku kanálových dat pro otevřený kanál, do HL se vyzvedne adresa programu výstupu znaku na kanál, je dána do HL a do DE je adresa uložení vyššího byte adresy pro výstup znaku. V A je znak, který se má zapisovat. Nyní se vyvolá program pro výstup znaku. Důležitý je právě obsah registru DE a A. Když se podíváte do stránkovacího prográmku, všimnete si těchto dvou instrukcí:

```
ld a,#4F
ld de,TAB–#1A
```

Hodnota #4F způsobuje návrat s přistránkovanou ZX ROM (zapisovaný znak). V DE je ukazatel na začátek hlavičky otevřeného souboru.

Teď něco nakousnu z kanálových informací v ROM D40. Kanálová data pro otevřený soubor mají velikost 544 bytů, 32 bytů je hlavička a 512 bytů je bufer. Hodnoty za návěštím TAB jsou velice důležité z hlediska zápisu hodnoty do bufferu. První slovo je počet zapsaných znaků v bufferu (v našem případě 0) a druhé slovo je adresa v bufferu, kam se bude daný znak zapisovat (v našem případě #3EF7). Protože tyto informace jsou uloženy až na konci hlavičky, musíme do DE uložit právě TAB–#1A, kde #1A je relativní posun na tyto parametry v hlavičce. Potom zavoláme #25AB, kde je instrukce RST 0. Po přestránkování je zjištěno podle návratové adresy, že se jedná o zápis znaku do sekvenčního souboru a je proveden skok na adresu #E23. A nyní už výpis programu:

```
0E23    ld hl,#001A       ;posun na informace o počtu zapsaných znaků a pozici v buferu
0E26    ei                ;povol preřušení
        add hl,de         ;nastav ukazatel do hlavičky na patřičné informace – nyní ukazuje na začátek TAB
        ld e,(hl)         ;vyzvedni počet zapsaných znaků (nula)
        inc hl
        ld d,(hl)
        inc hl
        ld c,(hl)         ;a do BC aktuální pozici v buferu v našem případě #3EF7
        inc hl
        ld b,(hl)
        ld (bc),a         ;uložíme hodnotu #4F na #3EF7
```

```
        inc de          ;zvýšíme počítadlo
zapsaných znaků
        bit 1,d         ;a testujeme, jestli
bylo zapsáno 512 znaků
        jr z,#0E12      ;pokud ne, skoč
na uložení DE a BC
```

Protože je v DE hodnota 1, je skok proveden a řízení se vrací zpět do volajícího programu. Obsah TAB se tedy změní, proto je ho zapotřebí obnovit, což dělají další instrukce v stránkovací rutině. Nyní máme na adrese #3EF7 uloženu hodnotu #4F, takže můžeme klidně volat RST 0 a máme k dispozici ROM D40. Nepotřebujeme žádné systémové proměnné. Velice mazané, ne?

Nyní se budeme věnovat těm návratům. Nejdříve návrat z rutiny ZX ROM. Protože ROM D40 používá některé rutiny, které jsou již v ZX ROM, bylo zbytečné je do ROM D40 znovu vkládat. Autoři vymysleli lepší způsob, a to volání rutin ZX ROM z ROM D40 s návratem zpět do ROM D40. Volání rutin se provádí přes RST #28 a to takto:

```
        rst #28
        defw adresa_rutiny
```

A funguje to takto:

```
003B    push af         ;ulož si a příznaky
        ld a,(#3EEE)    ;vyzvedni informa-
ci o snapu
        and a           ;a otestuj, jestli se
provádí snap (pokud se provádí SNAP, nelze
volat rutiny ze ZX ROM)
        jp nz,#034A     ;pokud ano, skoč
na návrat přes snap
        pop af          ;obnov A a příznaky

 Nyní vyzvedneme adresu volané rutiny.

        ex (sp),hl      ;nastav HL na ulo-
žení adresy rutiny která se bude volat
        ld (#3E66),de   ;schovej si DE
        ld e,(hl)       ;vezmi adresu rutiny
        inc hl
        ld d,(hl)
        inc hl          ;v HL je nyní ná-
vratová adresa do volajícího programu

Nastavíme si adresy na zásobník.

        ex (sp),hl      ;ulož si novou ná-
vratovou adresu zpět na zásobník
        push hl         ;ulož si HL
        ld hl,#3EF7     ;do HL adresa ulo-
žení kódu činnosti
        ld (hl),#4F     ;nastav činnost
„volání rutiny ZX ROM“
        ld hl,#0000     ;do HL dej návra-
tovou adresu z rutiny ZX ROM do ROM D40
        ex (sp),hl      ;ulož ji na zásob-
ník a obnov HL
        push de         ;ulož adresu vola-
né rutiny na zásobník
        ld de,(#3E66)   ;obnov si DE

A provedeme rutinu s návratem do ROM D40.
        jp #1700        ;skoč na přestrán-
kování do ZX ROM
```

návratová adresa do volajícího programu
0000 – návratová adresa do ROM D40
adresa volaného programu v ZX ROM (sem ukazuje SP)

Jak to tedy funguje? Po JP #1700 dojde k přestránkování do ZX ROM. Je vyzvednuta adresa volané rutiny a proveden skok na tuto rutinu. Po ukončení rutiny je vyzvednuta ze zásobníku hodnota 0, dojde k přestránkování do ROM D40, podle obsahu #3EF7 dojde k návratu s přistránkovanou ROM D40. Je vyzvednuta návratová adresa do volajícího programu, kam se předá řízení za DEFW. A to je vše.

Nejde tedy volat rutiny, které využívají ROM D40 (i když šlo by to, ale musely by vrátit zpět hodnotu #4F na #3EF7, aby se nám nezhroutil systém). A nyní něco o návratu pro tisk chybového hlášení. MDOS má v sobě zabudovanou kontrolu obsahu adresy adresované ERR_SP. Má zde být uložena hodnota #1303 pro návrat po vykonání příkazu do BASICu. Pokud ale máte zabudované ošetřování chybových hlášení (ON ERROR GOTO), musí být zabezepečena i hlášení ROM D40. Pokud dojde k chybě, skáče se na adresu #2A3.

```
02A3    push bc         ;ulož si adresu ulo-
žení chyby (při při chybě syntaxe to je návra-
tová adresa do ZX ROM, při chybě v MDOSu
to je ERR_NR)
        ld hl,#000B     ;vracet se budeme
na adresu #000B, aby jsme se nezacyklili
        push hl         ;uložíme ji na zá-
sobník

Nyní otestujeme, jestli se při chybě vrací
řízení přímo do BASICu na adresu #1303
nebo byla tato návratová adresa přepsána
(obsah adresy adesované proměnnou
ERR_SP). Pokud je adresa #1303 a je chy-
ba MDOSu, tiskne se hlášení na obrazovku
a řízení se vrací zpět do BASICu. Pokud byla
tato adresa přepsána a došlo k chybě MDO-
Su nebo ZX ROM, vrací se řízení na tuto ad-
resu bez toho, že by se tisklo jakékoliv hlá-
šení. Je třeba dát pozor, že nedošlo k nulo-
vání některých důležitých systémových pro-
měnných MDOSu, které můžou způsobit
dost závažné problémy (dokonce i chybný
zápis na disketu).

        ld hl,(#5C3D)   ;adresa položky na
zásobníku při chybě do HL
        ld e,(hl)       ;do DE adresa,
kam se skáče při chybě
        inc hl
        ld d,(hl)
        ex de,hl        ;dej ji do HL

Nejdříve otestujeme, jestli někdo nepře-
psal návratovou adresu do BASICu, protože
používá ošetření chybových hlášení.

        ld bc,#1303     ;do BC adresu
MAIN-4 v ZX ROM
        and a           ;nuluj CY
        sbc hl,bc       ;jsou adr. shodné?
        jr nz,#02C2     ;ne, skoč na návrat
do ZX ROM bez tisku chybového hlášení
        ex de,hl        ;do HL dej adresu
položky na zásobníku při chybě

Není přepsána návratová adresa do BA-
SICu, bude se tedy tisknout chybové hlášе-
ní na obrazovku.

        ld (hl),#00     ;nuluj obsah adresy
        dec hl
        ld (hl),#00     ;po obsloužení
chyby se řízení vrací na adresu #0000
        ld hl,#3EF7     ;do HL adresa
kódu činnosti
        ld (hl),#45     ;ulož kód činnosti
„výpis chybového hlášení“

Návrat do ZX ROM.

02C2    call #2536      ;zastav mechaniky
        ld hl,(#5C5D)   ;do HL adresa zna-
ku pro dekódování nutné pro návrat do ZX
ROM
        jp #1700        ;skoč na přestrán-
kování zpět do ZX ROM
```

A teď něco o adrese 8. Jak už jsem napsal, je tato adresa využita interpretem BASICu. Někteří lidé ji využívali pro stránkovaní pomocí IM 2, ale to uz je minulost. Jak vůbec rozpozná ROM D40, že se jedná o její příkaz. Hlavní věc je, že některé příkazy obsahují znak „*“, některé „#“ a některé mají trochu jinou syntaxi, než je v ZX ROM (kanály). Všechno se to zase děje přes určení návratové adresy, odkud bylo voláno RST 8. Tabulka všech návratových adresa je na adrese #5FF. Její struktura je následující:

```
defw adresa_příkazu_v_tabulce_syntaxe
defw odkud_bylo_voláno_RST_8
defw počet_návrat._adres_při_volání_RST_8
defw adresa_progr._vykonávajícího_příkaz
```

Celá tabulka je opět zakončena 0. Jsou tedy postupně vyzvedávány adresy příkazů v tabulce syntaxe a porovnávány s obsahem adresy #5C74 (T_ADDR). Pokud jsou stejné, je vyzvednuta návratová adresa při RST 8. Pokud jsou i tyto stejné, je vypočtena výška zásobníku při zavolání RST 8 (vrchol zásobníku pop RST 8 mínus obsah adresy #5C3D (ERR_SP)). Když jsou i tyto stejné, je proveden skok na podprogram.

Poslední test nám vylučuje například použití rozšíření BASICu například programem PRO-DOS, který zvyšuje výšku zásobníku o 1 návratovou adresu, čímž je tento test negativní u všech příkazů. Prohlížení tabulky a porovnávání hodnot zajišťuje program na adrese #266. Při zpracovávání příkazů se již potom postupuje jako v ZX ROM.

To by bylo pro dnešek asi všechno. ■

(pokračování příště)

Seriál

Pár rutinek MDOSu, které se hodí

Petr Žabenský

A je tady další dodatek k mému článku o zajímavostech disketové jednotky. Ono těch rutinek v ROM D40 je strašně hodně, některé nakousl ve své miniknize (to slovo se mi strašně líbí) George K. Možná, že jako autor komentovaného výpisu, který je už snad mezi lidmi, jsem sám proti sobě, ale co se dá dělat. Ti, kdo ho ještě namají, ať si ho pořídí. Ale tady je několik adres rutin.

ANALWDNM (#10E2) – analyzuje jméno disku v DNZONE1
Vstup: jméno disku v DNZONE1
Výstup: Z, NC není jméno v DNZONE1
NZ, C v DNZONE1 je normální jméno
NZ, NC v DNZONE1 je jako jméno určení mechaniky (A, B, C, D), v A je číslo mechaniky (0, 1, 2, 3) pokud je v A 255, bylo chybné jméno disku

ARRANGNM (#107C) – upraví jméno souboru v FNZONE1 na masku, převede „*“
Vstup: ve FNZONE1 je jméno souboru
Výstup: ve FNZONE1 je maska
C bylo použito wildchars
NZ byla vložena přípona
Z nebyla vložena přípona

BWRITE (#2296) – slouží pro zápis sektoru na disketu
Vstup: HL adresa paměti, odkud se bude zapisovat
B číslo stopy, kam se bude zapisovat
C číslo sektoru, kam se bude zapisovat
D počet sektorů, které se budou zapisovat
E počet opakování při chybě CRC (1–žádné 2–jedno, 3–dvě)

Tady se na chvíli zastavím. V miniknize byla totiž chyba. V E není počet opakování při CRC, SEEK etc., jak se tam píše, ale pouze při chybě CRC, a není 1=dva pokusy, ale žádný pokus. Tady je kousek výpisu:

```
23B2   bit 3,d          ;byla chyba CRC ?
23B4   jr z,#23B9       ;ne,skoč na konec operace
23B6   dec e            ;sniž počet opakování
23B7   jr nz,#2387      ;další pokus? ano, skoč na opakování operace
```

Dále A nemusí obsahovat číslo disku, protože se bere obsah adresy #3E6B. Ale pokračujme dále.

BREAD (#22A5) – slouží pro načtení sektoru z diskety, parametry stejné jako u BWRITE

BFORMA (#229C) – slouží k naformátování stopy, parametry stejné jako u BWRITE

CMPDSK (#1CD5) – načte BOOT z disku, jehož číslo je na #3E6B, porovná se jménem v SRAM, pokud je jiné, načte nové parametry a jméno diskety v drivu
Výstup: Z jméno diskety v drivu a jméno v SRAM je stejné
NZ jména jsou jiná

DFILER (#1F88) – vymaže soubor z diskety
Vstup: HL adresa položky adresáře v DIRBUF

DRWCMP (#21AC) – vypočte adresu parametrů mechaniky
Vstup: A číslo mechaniky
Výstup: IX adresa parametrů mechaniky
Z mechanika není připojena
NZ mechanika je připojena

DSKSTP (#2536) – zastaví mechaniky

FINDEMPTYFAT (#20F6) – najde první prázdnou položku FAT od položky v HL
Vstup: HL číslo položky FAT, od které se začne hledat
Výstup: HL číslo položky, která je volná
Z taková položka existuje
NZ taková položka neexistuje

FIRSTEMPTY (#215C) – najde první volnou položku adresáře
Vstup: IX adresa parametrů drivu
Výstup: A číslo nalezené položky
HL adresa položky v DIRBUF
Z taková položka existuje
NZ taková položka neexistuje

FIRSTMASK (#212B) – najde první položku adresáře, která vyhovuje masce v FNZONE1
Vstup: maska v FNZONE1
IX adresa parametrů drivu
Výstup: HL adresa položky v DIRBUF
Z byla nalezena
NZ nebyla nalezena
A číslo položky v adresáři

FREECOUNT (#1DC2) – spočítá volné sektory na disketě
Výstup: BC počet volných sektorů

FYZLOG (#1DE9) – převede fyzickou stopu a sektor na logický sektor
Vstup: IX adresa parametrů drivu
B číslo stopy
C číslo sektoru
Výstup: HL číslo logického sektoru

GETFAT (#1D04) – vyzvedne obsah položky v HL z tabulky FAT
Vstup: HL číslo položky
Výstup: DE obsah položky

GETPAR (#1EA1) – přečte BOOT disku, jehož číslo je na #3E6B, nastaví parametry a uloží je do systémových proměnných, uloží jméno diskety do SRAM
Vstup: #3E6B číslo drivu

INITALLDR (#1F49) – nastaví parametry všech připojených disků z BOOTů disket

LOADBLOCK (#19AE) – nahraje do paměti soubor
Vstup: IX začátek uložení dat
DE délka dat
#3E72 uložení adresy, kde je v DIRBUF uložena hlavička souboru
DNZONE1 jméno disku, odkud se bude nahrávat

LOAWITHF (#1FAB) – vyhledá soubor se jménem v FNZONE1 a nahraje data do paměti
Vstup: HL počáteční adresa uložení dat
DE délka dat
IX adresa parametrů drivu
FNZONE1 jméno souboru upravené na masku

LOGFYZ (#1DF9) – převede logický sektor na fyzickou stopu a sektor
Vstup: HL číslo logického sektoru
IX daresa parametrů drivu
Výstup: B stopa
C sektor

NEXTEMPTY (#215E) – najde další volnou položku v adresáři
Vstup: A číslo položky–1, od které se bude prohledávat
IX adresa parametrů drivu
Výstup: stejné jako u FIRSTEMPTY

NEXTMASK (#212D) – najde další položku adresáře vyhovující masce v FNZONE1
Vstup: A číslo položky–1, od které se bude prohledávat
IX adresa parametrů drivu
Výstup: stejné jako u FIRSTMASK

RDNOEMPTY (#216C) – najde první neprázdnou položku od A
Vstup: A číslo položky–1, od které se bude prohledávat
IX adresa parametrů drivu
Výstup: stejný jako u FIRSTEMPTY

SAVEFILE (#2046) – uloží soubor na disk
Vstup: HL adresa začátku dat
DE délka dat
#3E78 počáteční adresa uložení dat
#3E7A délka BASICu bez proměnných
FNZONE1 jméno souboru

SECPERDISK (#1DDC) – vypočte počet sektorů na disketě
Vstup: IX adresa parametrů drivu
Výstup: HL počet sektorů na disketě

SETACT (#1C8F) – nastaví drive podle jména v DNZONE1 jako drive, se kterým se bude pracovat, pokud toto jméno nenajde ve jménech drivů, načte parametry všech připojených drivů a zkusí to znovu
Vstup: DNZONE1 jméno disku
Výstup: Z takový drive byl nalezen
NZ takový drive nebyl nalezen

SETDRV (#1F16) – nastaví drive, se kterým se bude pracovat, podle jména v DNZONE1
Vstup: DNZONE1 jméno drivu
Výstup: Z disk s takovým jménem byl nalezen
NZ disk s takovým jménem nebyl nalezen
#3E6B číslo drivu

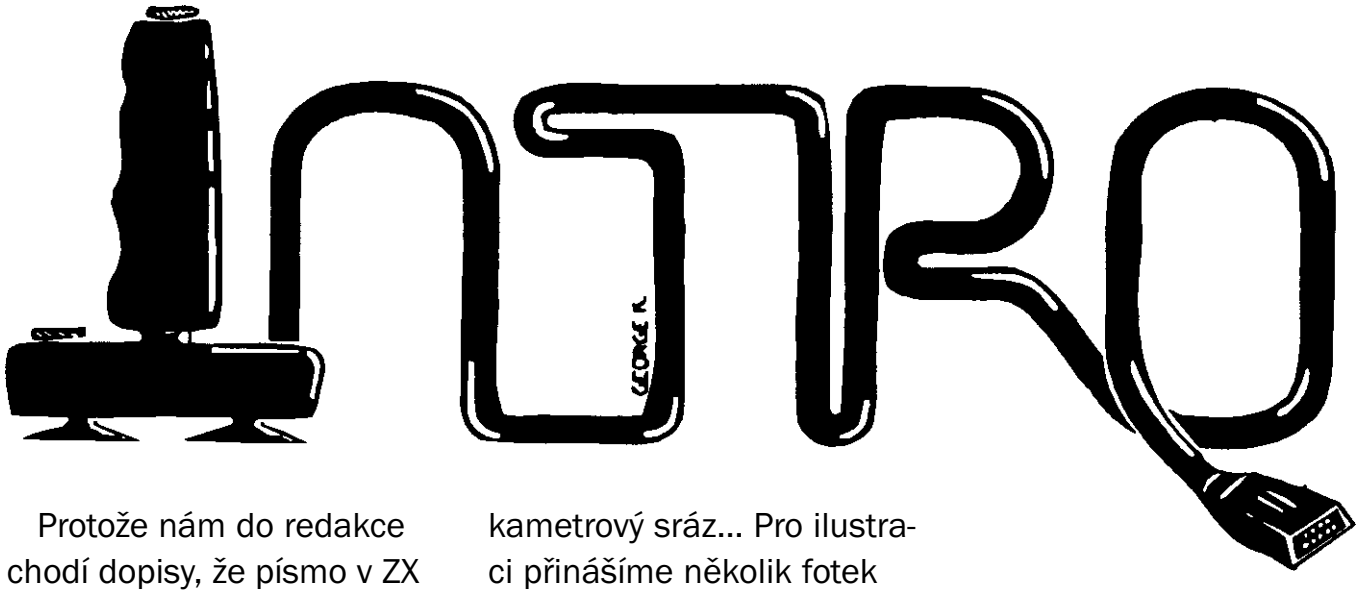
TESTMSK (#2137) – zjistí, jestli jméno souboru odpovídá masce v FNZONE1
Vstup: FNZONE1 maska souboru
HL adresa uložení jména souboru
Výstup: Z vyhovuje masce
NZ nevyhovuje masce

WFATIFCH (#1D9D) – zapíše sektor FAT, který je v FATBUF, pokud byl jeho obsah změněn

WRTOFAT (#1D1E) – zapíše do FAT obsah DE do položky v HL
Vstup: HL číslo položky
DE co se má zapsat

WSCADR (#1E65) – zapíše sektor adresáře, který je načten v DIRBUF
Vstup: #3E6F stopa a sektor, kam se bude zapisovat

■



Protože nám do redakce chodí dopisy, že písmo v ZX Magazínu je příliš malé, rozhodli jsme se pokusně jednu rubriku udělat písmem o něco větším. Ta rubrika není žádná jiná, než všemi oblíbené Intro. Teď jej již budou moci číst i téměř slepí lidé...

ZX Magazín inmobilní!

Při jedné ze zahořovacích jízd Matsoft nezvládl řízení a opřel se svým autem v serpentýnách předním blatníkem o jednu stranu vozovky. Naštěstí si vybral tu, kde byla celkem pevná skála, na rozdíl od té druhé, kde byl poměrně řídký vzduch a několikametrový sráz... Pro ilustraci přinášíme několik fotek postižené Škodivky – pro lepší vizuální zážitek hned z několika různých úhlů.

Driver Required!

Tohle není zvolání nebohého majitele Windows XX (i když by klidně mohlo být, obzvláště, interpretujeme-li XX jako NT – No Thanx), ale Tritola a Dizzyho, kteří se na DoxyCon '99 odmítli plahočit vlakem či autobusem a Dizzy pro účel dopravy na DC pořídil nové auto. Jediný problém zatím je, že ani Tritol ani Dizzy nemají řidičský průkaz.

New Cassanova?

Velmi schopný programátor a velice úspěšný hardwarový experimentátor PVL, jehož přezdívku někdo zcela nesmyslně interpretoval jako „Productivity Very Low", učinil zajímavý objev svých dalších schopností. Jak se exkluzivně podařilo ZX Magazínu zjistit, při PVLových RARovacích aktivitách dívky z ne zcela jasných příčin omdlévají blahem...

Šijí s Matsoftem všichni čerti?

To je otázka, která zajímá nejednoho spektristu. Jelikož nebylo možno tento ZXM sešít strojově, bude Matsoft muset všechny výtisky sešít ručně sešívačkou. Možná by se mu pár ďábelsky rychlých pomocníků hodilo...

Dementi

Za dementa byl prohlášen DIHALT, neboť se nechal odvést do Green Hell (pro neznalé – na vojnu).