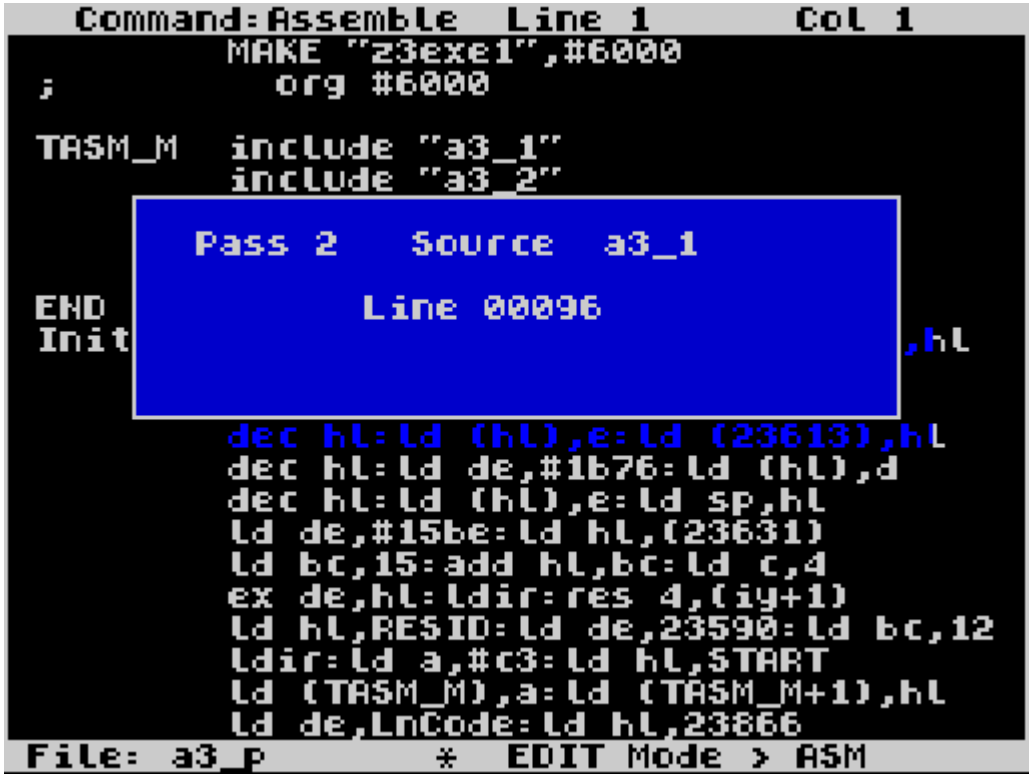
ZX- TURBO ASSEMBLER 3.0 PRO MB02+



Doba prací: 14.12.2008- 14.01.2009- cca 30 dnů

Datum vydání: 12.01.2008

Autor úpravy: Hood, email: hood@znojman.cz

http://hood.speccy.cz

tel: +420-777-192-191

OBSAH

PŘEDMLUVA- ANEB JAK TO CELÉ VZNIKLO

Vážení příznivci a uživatelé řadiče MB02+. Bůh a osud tomu znovu chtěly a dostává se vám do rukou můj další výtvor – MB02+ verze ruského programu ZX- TURBO ASSEMBLER 3.0 (dále jen „zasm“). Jak to celé vzniklo?

Vše vlastně „zavinil“ +GAMA, kterému tímto děkuji. Na JHCONu 2008 totiž v emulátoru psal v zasmu. A jakmile jsem ho uviděl, tak po krátké Gamově prezentaci jsem se do něj zamiloval, a okamžitě jsem jej chtěl mít na Mbčku. A jak to u mě bývá, mnoho dobrých věcí pochází z emocí a nadšení, zasm nebyl výjimkou. Okamžitě jsem se pustil do zkoumání kódu, zburcoval jsem ruské fórum, vyhledal autora, který se mi naštěstí ozval a po zjištění, že zasm používá korektní volání TRDOSu, dal se do předělávání. Nejhorší a nejdelší bylo se v zasmu zorientovat, ovšem práce významně urychlil autor zasmu3.0, Vladimir Rubcov, který na moji žádost uvolnil zdrojáky. Tedy po základním zorientování se co a jak se kde provádí, bylo třeba najít volné místo. S tím byl trošku problém a po té, co jsem je musel asi třikrát přestěhovat, začaly pomalu vznikat první mbčkové diskové operace. Občas přišly perné chvíle, viz dále, ale podařilo se je všechny překonat. Dost časově náročná byla i finalizace, zasm pro mbčko byl funkční již na Silvestra, ale prostě vše doladit, dohledat bugy, učesat zdroják, napsat tento komentář zabralo hodně času.

Možná jsou uživatelé jiných systémů než MB02+ naštvaní, že jsem to neudělal univerzálně z BASICu. Toto však nebylo možné, zasm pracuje přímo se sektory a to jak při ukládání, tak při nahrávání.

Tuto moji dokumentaci jsem pojmul jen jako základní orientaci pro úplné nováčky. Není to v žádném případě manuál pro práci v zasmu. Obecně však lze říci, že na práci se zasmem si lze rychle a dobře zvyknout. První rada pro vás: EXT+h vyvolá pomoc. Originál dokumentace je bohužel v ruštině, myslím, že dobrý popis najdete v ZX Magazínu 3-4/99 od +Gamy- ten všem, kdo se chtějí se zasmem blíže obeznámit vřele doporučuji. Další popis je už v ruštině v TAPce, nebo na MBD imagi, v ReadMe souboru. Online verze je pak na stránkách zasmu http://zasm.by.ru/index.html. Na této stránce uvidíte i zasm3.10, což už je velká lahůdka.

ZASM 3.0- ASSEMBLER, EDITOR

Rozložení 128k stránek, jak s nimi ZASM pracuje

Tedy nutno předeslat, že na 48k si se zasmem neškrtnete. I když v ZX magu i manuálu popis je, přidám pár dalších informací. Vezmu to napřeskáčku.

- page 3- zde je tzv. pool, neboli buffer. Jeho velikost máte možnost měnit setupu. Tento buffer je využíván funkcemi INCLUDE a INSERT. Je zvláštní, že když jsem dal velikost poolu maximální, tak mi u velkých zdrojáků zasm zahlásil NOT ENOUGH MEMORY. Toto je i v originále, nevšímejte si toho a pool size prostě snižte. Za poolem se od f412h nachází veškeré mbčkové rutiny.
- page 7- zde je defaultně přítomen a nahrán STS monitor a debugger, přesněji od d000h (platí pro verzi sts 6.22+ distribuovanou a speciálně upravenou pro zasm30), ostatní verze stsek začínají od db00h. No, a od c000h až po začátek stska je druhý buffer, jeho velikost si nastavit nemůžete, a ten slouží pro funkci MAKE.
- page 0- do této stránky se překládá hotový strojový kód
- page 6- zde je uložen zdrojový text
- page 1- sem se při výskoku do BASICu ukládá kód z 8000h- bfffh
- page 4- zde od c000h do dfffh je kód zasmu. A při výskoku do BASICu se sem od e000h nahoru schovává kód z 6000h- 7fffh.

Jak to přesně je s tím přesouváním obsahu stránek při výskoku do BASICu? Jak je popsáno výše, prostě vše od 6000h do bfffh se uschová do stránek 1 a 4. V BASICu, pak máte připnutou stránku 0, klidně i s již přeloženým strojákem.

Co ZASM umí- bližší popis

- zdrojový text začíná na adrese 894fh a pokračuje dále po ffffh, ve stránce 6. Jeho maximální délka je tedy 76b1h/ 30385bajtů.
- tuším, že libovolně dlouhé návěští, odlišuje malá i velká písmena
- délka řádku je 128 znaků, zobrazuje na obrazovce 40 znaků
- komentáře před i za instrukcemi uvedené středníkem
- na jednom řádku několik instrukcí oddělených dvojtečkami (skvělá věc)
- import/ export txt souborů (další skvělá věc)
- má grafické znaky
- možnost změny fontů
- horké klávesy
- knihovny- podmíněný překlad- IFDEF, IFNDEF, IFUSED, IFNUSED, ELSE ENDIF
- funkce INSERT, INCLUDE- píše se přímo do textu a obě se vykonají při příkazu ASSEMBLY. Umožní vložit soubor patřičného jména. Opět skvělá věc. INCLUDE vkládá jiný zasm zdrojový text, INSERT vkládá binární soubor. Pakliže se vám zdá

oněch 30kB na zdroják málo, vězte, že těmito užitečnými funkcemi si můžete váš zdroják prodloužit prakticky libovolně. Funkci INSERT lze myslím dobře na mbčku použít pro spojování souborů. Když zadáte:

MAKE“výsledek“,adresa

INSERT“soubor1“

INSERT“soubor2“

Soubor „výsledek“ vznikne spojením souboru 1+2. Ovšem lze to takto dělat pouze do velikosti výsledného souboru ffffh. Prostě je to takto nastaveno, pokus o spojení do souboru o větší velikosti selže. I když udělat by to šlo, ovšem primárně je tato funkce určena pro účely zasmu, nikoliv pro spojování souborů na mbčku. Soubory >ffffh by na disku zabraly mnohem více místa. Bližší info v kapitole o předělávání na MB02+ verzi.

Co ZASM neumí- aneb srovnání s promíkem

Ke srovnání jsem si vybral vynikající assembler Prometheus, protože ten znám a používám. Rozdíly vyplývají již z předchozí kapitoly. Obecně lze říci, že v ovládání se zasm promíkovi podobá. PageUp/Down jsem předělal stejně jako je má promík. Zasm je ovšem v ovládání a práci komfortnější, má tabulátor, instrukce lze mít na řádku kdekoliv.

V čem je promík lepší je kontrola syntaxe již při zadávání řádku. Syntax kontroluje zasm až při překladu, který má jako promík dva průchody.

Co zasm nemá a co může rovněž chybět je seznam všech návěstí, na krásně abecedně setříděná návěští prostě zapomeňte, ty jsou až ve verzi 3.10.

Co mi okamžitě od začátku chybělo je promíkovská funkce PUT, tedy překlad kódu jinam, než fyzicky leží a ve 128k verzi i umístění do stránek. Toto řeší zasm jinak a to pomocí funkce MAKE, která se opět uvádí do textu. Prostě místo ORG zapíšete MAKE“jméno“,adresa a po zahájení ASSEMBLY se provede jakýsi promíkovský PUT, ovšem opět na disketu. Když chcete překládat do stránek, musíte si na to napsat zvláštní kód.

Co naopak zasm má je funkce SAVE OBJ- uložení na disketu přeloženého strojáku od adresy ORG. Tady pozor! Bezpečná zóna pro uživatele je od 6000h nahoru- tady je vše zálohované a na této adrese i začíná kód zasmu. Oblast níže než 6000h ale chráněná zasmem není, takže si ji pomocí assembly a ORGu <6000h můžete přepsat/ poškodit (sys. proměnné BASICu, např.) Na to pamatujte! Rovněž nelze přímo překládat do VRAM. Úspěšní budete až od 5b00h/23296- sem a odtud lze překládat/sejvovat váš kód. Jinak překlad do VRAM, nebo <4000h řeší opět funkce MAKE.

Co dále zasm neumí (a je to spíše věcí monitoru) je disassembling, tedy překlad strojového kódu zpět do zdrojového textu. To si myslím je velká škoda.

Specifika MB02+ verze

Mbčková verze je verzí plnohodnotnou a z hlediska funkčnosti je shodná s betadiskovou verzí. Jediné s čím jsem se nenamáhal (doslova) je CATALOG DISKETY a VOLBA DISKU. Tedy catalog disku tam je, ovšem není pěkně srovnán do okna, jak originál, ale je normální jak jej znáte z BSDOSu. Volba disku tam také není. Tyto dvě funkce jsem

nepředělal záměrně a to z důvodu času. Kdybych je předělával, možná bych na tom seděl do teď. Totiž kdysi, když jsem předělával pro mbčko ProTrackera, tak volbu disku i catalogu jsem udělal ve stejné podobě jako originál, ale stálo mě to mnoho úsilí. Prostě nejen za tímto účelem jsem v nmi menu vytvořil volbu disku a adresáře, takže ji použijte i v zasmu.

Zasm musel být samozřejmě přepracován pro práci s 1024b sektory na místo betadiskových 256b.

Opravil jsem drobnou chybku. Již zmíněný buffer ve stránce 7 má v originále 6912 bajtů, což znamená, že při aktivaci funkce MAKE buffer přepíše začátek STS verze 6.22+. Snížil jsem tedy buffer na velikost 4096 bajtů, čímž lze krásně provozovat i sts6.22+.

Další oprava spočívá v tom, že zasm umožňuje přihrát na pevnou adresu libovolný monitor (nelépe tedy sts, ale zkoušel jsem i devastaci a funguje také). No, a sts6.22+ je poněkud větší a loaduje se níže než všechny ostatní stska, a originál i moje předělávka má sice sts6.22+ defaultně nastavený, ale nelze jej kvůli jeho délce již podruhé přihrát, tak jsem to opravil a mb verze umí přihrát i tento.

STS- MONITOR, DEBUGGER

Ano, všemi nebetadiskaři chtěný, téměř mýty ověnčený a pověstný sts monitor k nám spadl z nebe. Ovšem mbčkáři, neradujte se, mbčkového tam nenajdete nic. Vezměte si, že stsko má tyto diskové funkce: catalog, help, load/save souboru, load/save sektoru, setup. Co z toho je skutečně potřeba a co chybí? Catalog, load/save souboru umí moje nmi menu. Help je přiložený v TAPce, můžete si jej přečíst v zasmu, bez setupu se dá snad žít, jediné, co by se mohlo hodit je load/save sektoru, ovšem originál tam vůbec nemá políčko pro výběr strany, jak toto řešit na mbčku? Když jsem zvážil důležitost (lépe řečeno nedůležitost) těchto diskových funkcí, tak jsem si řekl, že v této etapě stsko pro mbčko předělat by bylo jedině za cenu potu a krve a do toho teda zatím nejdu. Měl jsem připravenu pro předělávání pro mbčko alespoň funkci help, ale jedna rutina prostě pořád padala, takže nyní po nahrání zasmu máte v paměti a i na disketě verzi sts6.22+mb, která má všechny diskové vstupy zaslepeny. Je to pro to, aby při používaní stska vám nespadl program, jen proto, že jste omylem stiskli klávesu pro trdos diskové operace. Jediné, co mě mrzí je to, že jsem takto musel zaslepit i funkci G- počítání taktů, ale podle mě sahá dost nízko až na betadisk. Snad se mi podaří ji někdy v budoucnu odblokovat.

POZOR!!! na disketě jsou přiloženy i jiné verze sts, tyto nejsou pro mbčko upraveny a pravděpodobně vám budou při stisku klávesy pro diskové služby padat!!!

ÚPRAVA PRO MBČKO- ANEB ZEVRUBNÝ PLÁN VESNICE A PŘESNÝ POPIS DOROTY MÁCHALOVÉ:))

Následující část je určena pro ty, kteří by chtěli předělat zasm na jiný diskový systém. Najdete v ní co a jak předělat a podstatu mojí předělávky. Probereme si funkce po funkci, adresu po adrese, bajt po bajtu:). Kromě nových rutin bylo třeba i kód zasmu na některých místech pozměnit. K detailnímu zkoumání doporučuji můj zdroják, kde jsou jednotlivé funkce přehledně odděleny.

Celkově jsem předělávku postavil tak, že veškeré odskoky do mojí rutiny se sbírají v jednom bodě, a tam se pak rozhodne podle čísla služby, které je v registru c, o jakou funkci se bude jednat a dle toho kam má vlastně program běžet a co udělat. Tedy, jak už jsem zmínil, můj kód se nachází za poolem ve stránce 3 od adresy f412h. CALL nebo JP do trdosu jsem nahradil callem nebo jumpem na adresu 8438h, kde se provede přistránkování strany 3 a skok na f412. Tam se zazálohuje oblast cca 1000 bajtů od adresy 7000h a na její místo se ldirne můj kód. Ten je schválně dole, protože mnoho diskových operací pracuje právě se stránkami. Až se potřebné provede, vše se zase vrátí na původní místo. Bohužel od 6000h po začátek zdrojového textu není moc místa, našel jsem asi jen 15 bajtů, zbytek jsem musel hodit na zásobník, a to nás přivádí k dalšímu tématu aneb:

KDE V ZASMU NAJÍT VOLNÉ MÍSTO PRO SVOJE RUTINY ?

Ve stránkách je místa poměrně hodně, dosáhnete toho například snížením maximální velikosti některého z bufferů. Jenomže někde dole potřebujete mít stránkovací rutiny pro 128k. Jediná oblast padající do úvahy je 6000h- 894eh. Výše začíná zdroják, níže není ochrana proti assemblování. Místo je 11 bajtů od 8438h, po rutině, která vyskakuje do trdosu. Ale dobrý trik vymyslel Velesoft, kde totiž před startem zasmu se na čtyřech místech přepíše nový, nižší start zásobníku, no a vy zbylé místo můžete využít, já tam mám cca 15 bajtů. Nebýt tohoto triku, kód dole je tak natřískaný, že bych neměl šanci svých pár bajtů někam umístit a musel bych použít mbčkovou SRAMku.

JAK ZMĚNIT SEKTORY, KDYŽ TRDOS JE MÁ 256 BAJTŮ DLOUHÉ?

To je podstatná věc, kterou musíte udělat jako první a bez které se nepohnete. Je třeba donutit zasm, aby pracoval s vaší délkou sektorů. V podstatě musíte omezit oba buffery na celé násobky délky vašich sektorů. K poolu pro INCLUDE/INSERT- na c947h musíte dát call na svoji vlastní rutinu, která vám přepočítá násobky trdos sektorů na násobky sektorů vašeho systému. Omezení druhého bufferu hledejte na adresách 65d9 a 6619 v registru HL.

LOAD/MERGE

obě funkce používají stejnou cestu, samotný call je na adrese 85e5h. Jedná se o službu eh- nahrání souboru z diskety, kde v hl a de je start a délka. Stupid.

SAVE/SAVE SETUP/SAVE BLOCK

opět řešeno stejnými rutinami, na adrese 84cbh je erase file služba, tu můžete přeskočit, pokud nechcete soubor mazat a na 84d6h je samotný save, služba bh. Opět de a hl a důležitá adresa, kde trdos pracuje 5cddh- tam se ukládají a odtud berou názvy souborů, ale i identifikace diskety. Jinak stupid.

SAVE OBJ

Tak tady začal oříšek. Princip této funkce je ten, že vám sejvne na disk přeložený kód od 5b00h nahoru. (Jak zasm řeší v tuto dobu zásobník se mě neptejte, to kdybych zkoumal, tak to předělávám ještě teď). Co vy musíte udělat je dát sejvovací rutinu tak, aby končila na adrese 57ffh. Tedy respektive, takto to má orig. trdos, který ldirne sejvovací rutinu od 57f1h a má nějakých 15 bajtů. Tedy, asi můžete přesáhnout i výš, ono stejně do oblasti atributů se překládat nedá, si myslím, a já i naopak posunul začátek této save rutiny níže (na 57eah). Ale princip je tento, čeká vás spousty ldirů, no celkem humus funkce na úpravu.

LOAD FONT

Ultraeasy .Font se loaduje napevno do stránky 4 od d6e7h a má pevnou délku 800h bajtů. Je to na adrese ca19h.

LOAD STS

Easy. Stska, nebo i jiné debuggery se napevno loadují do stránky 7 od adresy db00h. V mých rutinách je úprava, která zvládne loadnout i sts6.22+, které začíná už od d000h. Tento load najdete na drese 891bh.

INCLUDE/INSERT

Tak, a dostáváme se na sektory. Jde o funkce, které z disku čtou. A úprava je už mírně náročnější. Samotné čtení z disku do paměti tedy probíhá ve stránce 3 v bufferu, jehož velikost si můžete určit v setupu. Na 66a3h se zjišťuje, zda je soubor na disketě, na 66aeh se děje služba č. 8h, kterou ale přeskakuji a samotné load po sektorech je na adrese 6718h a na vstupu je v b počet trdos sektorů a v hl start (tedy vždy začátek poolu). Stačí jen tuto poslední rutinu odchytávat a zjistit, kolik sektorů přesně z daného souboru nahrát, to je vše.

MAKE

Vedle SAVE OBJ je toto největší maso celé předělávky. Jde o ukládání sektorů s přeloženým strojovým kódem z bufferu ve stránce 7 na disk. Trdos se v originále volá celkem 5x, na mbčku to však stačilo jen třikrát. Největší problém mbčka je ten, že neumí sejvovat do souboru neznámé délky. Nejprve jsem chtěl zasm donutit proběhnout assembly 2x, a při prvním průchodu si zjistit délku. Zasm však odolal (nehledě na časovou náročnost tohoto principu pro uživatele) a tak Velesoft a nezávisle na něm Busy:) vymysleli metodu, kterou jsem nakonec použil. MB verze zasmu je tedy patrně první program pro mbčko, který

používá zálohovací soubor. Princip je naprosto jednoduchý. Vytvořit si na mb disku soubor „zasm3.0bak“ o délce 65535 bajtů, do něj se uloží výsledek funkce MAKE a po posledním sejvu pak víme, jak velký tedy má soubor být, a už stačí jen vytvořit soubor o již správné délce a nakopírovat do něj obsah souboru bak. Soubor bak můžete z disku klidně smazat, nebo ho tam nechat, to je úplně jedno, program si jej příště vytvoří nebo uloží stroják do již existujícího baku. Hračka, ne? Jenže než to vymyslíte a uděláte....65b2h (vynechaná, trdos otevírá soubor), 65b9h zjistí velikost volného místa na disku a pokud je volno, založí se soubor bak, 6618h samotné sejvování do souboru po sektorech, 6664h přesunutí obsahu bak do nové souboru se správnou délkou, 6676h (vynechána, trdos uzavírá soubor).

CATALOG DISKU

Neptejte se mě jak upravit zasm, aby kočkoval disketu pěkně do svého okna. Nevím to. Obešel jsem tuto funkci pomocí služby bsdosu. Tím pádem vzniklo po trdos rutině CAT ve stránce 4 hodně místa, které jsem použil pro vlastní potřebu. Catalog začíná na adrese c6b7h, kde jsem první call nechal origoš a následuje ld c,64h a skok na vyhodnocení do mojí centrální rutiny. Do c jsme dal tuto hodnotu schválně, podle ní identifikuji snadno, že se jedná o katalog disku.

ERASE FILE

mega easy. Bsdos má na to službu. jméno se bere opět z 5cddh, adresa c8c5h ve stránce 4.

ZMĚNY KÓDU ZASMU

Následuje seznam adres a co vše bylo změněno, aby to spolupracovalo s mbčkem, (seznam teda slouží hlavně pro mě:)

LOAD, MERGE- 85e5h call 8438h

LOAD FONT- str. 4- ca19h call 8438h

LOAD STS- 891bh call 8438h

SAVE/SETUP/BLOCK- 84cbh 3xnop, 84d6h call 8438h

SAVE OBJ- 8564h 3xnop, 859ah jp 8438h, 853fh ld c,101: call 8438h- ošetřuje volné místo na disku u této funkce

- 8578h, 8583h ld de,57f1h změnit na 57eah

- 8598 ld a,13h: ld bc,7ffdh: xxxxx, na místo xxxx ldirne můj kód toto: out (c),a: ld c,bh: jp f412h

MAKE- 65b2h 3xnop, 6664h call 8438h, 6676h ret, 6618h jp 8438h, 65b9h call 8438h,

65d9h, 661dh- ld hl,násobek sektoru- mění velikost bufferu pro MAKE

INCLUDE/INSERT- 66aeh 3xnop, 6718h call 8438h, 66a3h call 8438h- zjištění přítomnosti souboru

CALL TRDOS- 8436h jr 841eh- přesměrování výskoku trdosu do BASICu

ERASE FILE- c8c5h jp 8438h

CATALOG- c6b7h call cb00h: ld c,100: call 8438h: xor a: ret str. 4, c6d6h- c7ceh je patrně volno, pokud zrušíte trdos catalog

Pokud smažete screen, originál špatně zobrazuje levou a pravou svislou čáru, toto problém odstraní:

- c3ceh jp c6c1 tedy hned za ret, str. 4, dát sem ld (hl),b: inc h: ld (hl),b: inc h: ld (hl),d: jp c3d1h

- c3d8h jp c6c9h, str. 4, dát sem ld (hl),d: inc h: ld (hl),b: exx: ret

Aby se správně vykonal výpis 10 znaků při assembly u INCLUDE/INSERT funkcí, je třeba udělat tyto čtyři věci:

- cf79h 5 ->3, str. 4
- 7788h 1bh ->19h
- 77b7h místo ldiru dát 2xnop
- 7795h přepsat na „Memory“,6,4,0- kde 6 je znak pro mezerník a za ním následuje počet mezer, místo je na max. 9 znaků!!

prohození kláves PgUp/PgDwn- 6aedh 68h, 6aef 7dh

rozšíří INPUT jména na 10 znaků- c8a4h ld c,10, str. 4, 67b8h ld b,10

POOL- 894d 18h, rutinka počítá násobky 1024, nutné pro změnu velikosti poolu- c947h call c6d6h- str. 4, c950h cp 35h- str. 4, omezí max. velikost poolu

dává délku do hlavičky, vyhodit, přepisuje 10. znak jména- 6635h 3xnop

úprava výpisu do informačního řádku, str. 4:

- cef3h „ File:“

- 6926h 3xnop- vypisuje číslu trdos drivu, to nechceme

- c8cbh ld de,cefah

- c8cfh 8 -> 10, c893h ret, c894-9h volno, str. 4- zobrazení 10 znaků v informačním řádku dole v obrazovce

oprava a úprava nápisu SetUp3.0 na 10 znaků, str. 4:

- ca41h nechat ld hl,5ce6: d de.... a dát jp c6ceh, tam dát ld (hl),32: ld hl,892fh: jp 84c7h

- ccc0h 20h

zrušení výběru trdos disků, str. 4:

cb88h ret- za tím je hodně místa až po PUTPoint

vytvoření místa na zásobníku pro vlastní rutiny, před startem zasmu:

- 69d6h, 6942h, 77ddh, 697dh- zde všude jsou instrukce, kde se stanovuje startovní zásobník, originál má zde hodnotu 61c4h, já to přepsal na 61a8h, odkud jsem umístil stránkovací rutiny pro 128k. Mají tvar push af: ld a,c. strany: jp stránkuj, viz, můj zdroják.

Hlavní vstupní bod do mých rutin je na adrese 8438h:

call 61abh, přistránkuje str. 3

jp f412h, skočí do ní

push af, zde se vyskakuje z mých rutin

jp 6511h a stránkuje se strana, kterou nastavuji v reg. a

ZMĚNY V STS6.22+MB

V stsku byly provedeny změny následující:

e7f8h catalog (ret), e7b0h help (nahrává se na adresu d000h), setup, klávesa G (ret), e66eh load souboru (ret), e60ah save souboru (ret), e301h load sektoru (ret), e2fch save sektoru (ret), e465h volba mechaniky (ret), eb99h 28h, eb9ch 1bh- přehodí PgUp/PgDwn, f6a1h- na defdh ld a,0, na df02h ld de,0- quit to DOS- záslepka

PODĚKOVÁNÍ

- Vladimir Rubcov (RubtsOFF): autor zasmu- za rady i zveřejněni zdrojáků
- kluci z konference zx.pk.ru
- Velesoft za testování, rady při finalizaci a poskytnutí dvou rutin s posuvy, za ideu zálohovacího souboru
- +GAMA za rady i za to, že mi zasma ukázal
- Busy za rady+ duplicitně za ideu zálohovacího souboru:)