DALŠÍ MOŽNOSTI

PROGRAMOVÁNÍ NA TI 57

Již několik let k nám dováží PZO TUZEX programovatelné kalkulátory firmy Texas Instruments, zejména TI 57, 58 a 59. Téměř současně se zahájením tohoto dovozu vznikly v technicky zaměřených časopisech rubriky, v nichž si majitelé těchto kalkulátorů vyměňují zkušenosti. Zpočátku to byly články obecné, později obsahovaly programy řešící monotématické problémy a nakonec došlo i na články popisující zajímavé a nečekané možnosti těchto kalkulátorů (např. prohlížení mikroinstrukcí). Právě v tomto posledním typu článků byla TI 57 poněkud opomíjena, proto jsem se rozhodl pokusit se vyplnit tuto mezeru, neboť možnosti tohoto kalkulátoru nejsou zdaleka tak omezené, jak se mnohému zdá.

V zahraniční literatuře se objevily články, jejichž autoři popisovali zobrazení písmen A až F na displeji TI57. Při systematickém zkoumání jsem narazil na některé další funkce, které rovněž popíši.

Postup tvoření písmen:

RST LRN 2 Fix SST Lbl 2 R/S LRN RST R/S LRN

Na displeji se objeví 04 00 0, dále zadáme číslice 2 až 7, čímž vznikne kód *10* až *15*, tedy písmena A až F. To si můžeme ověřit stiskem **BST**, případně ještě s následným **LRN SST**. Použijeme-li celý tento postup, objeví se písmeno na displeji. Pokud chceme tvořit další písmena, necháme kalkulátor v módu LRN nastaven na krok, na němž je kód již vytvořeného písmena, zadáme **Ins LRN RST** a postup tvoření písmena opakujeme. Samozřejmě, že se takto tvoří dané slovo pozpátku. Nechceme-li již tvořit další písmena, lze celý postup kromě kódů písmen přepsat programem nebo vypustit pomocí **Del.**

Ještě k samotnému postupu: Lze v něm zaměnit **Fix SST** za **Exc SST** nebo **Prd SST** (důležité je SST, neboť zajišťuje, že v programu zůstane nedoplněná instrukce). Místo číslice *2* je možno použít jakoukoli číslici kromě nuly, místo Lbl 2 libovolný label.

Pokud budeme v programu z různých důvodů používat písmena, je nutné před každé písmeno nebo souvislou řadu písmen umístit **CLR** a za nimi opět, jinak se zapíše jeho číselná hodnota, zmenšená o 10. Za písmenem lze použít i jinou funkci, nikdy však +, –, x, ÷, nebo sdruženou instrukci. Číselné hodnoty písmen lze však využít, a to zejména v exponentu, kde zůstane jejich plná hodnota. Např. místo **1 EE 15** lze zadat **1 EE F**, čímž se ušetří jeden krok. Pokud potřebujeme písmeno např. na adrese *47*, není časově efektivní, vytvořit je na adrese *04* a potom pomocí Ins posouvat o 43 adres. Potom lze postup zahájit

GTO 2nd 42 LRN Lbl 4 2 Fix SST Lbl 2 R/S LRN

a spouštět stiskem **SBR 4.**

Na závěr popisu písmen ještě k zajímavé instrukci **Fix F**, tedy **Fix 15**. Vytvoříme ji tak, že před kód *15* (tedy F) zapíšeme **Fix** bez indexu. Tato instrukce zaokrouhluje na řád desítek a vpravo je umístěna pomlčka. Pokud zavedeme exponenciální notaci, mantisa úplně zmizí, mezi číslicemi exponentu svítí desetinná tečka, pokud znovu stiskneme tlačítko **EE**, tečka zmizí, ale zadané číslo se při příští operaci ztratí. Zmizení mantisy lze využít všude tam, kde je třeba před obsluhou utajit číslo, které za normálního stavu problikne na displeji (např. při hrách Hi-Lo, Master Mind apod.). Výhodou je, že instrukci vepíšeme do programového registru, přečteme a můžeme ji přepsat programem hry s jediným omezením: v programu se nesmí vyskytovat Fix 0 až 9 nebo INV Fix. Zobrazení výsledků hry upravíme tak, že je posuneme do řádu desítek a těsně před zobrazením zrušíme exponenciální notaci. Pokud v režimu Fix F stiskneme **CLR** = +/– +/– (pokud bylo na displeji číslo menší než 5 lze CLR vynechat) displej bude úplně tmavý. Toho je možno využít jako náhradu konstantní paměti, protože v tomto stavu odebírá kalkulátor pouze 7,5 mA na rozdíl od rozsvícené jedničky, kdy odebírá 9 mA (svítí-li 88888888-88 je odběr asi 50 mA). Pokud chceme tento postup zařadit na konec déle trvajícího programu, můžeme ztížit nepovolanému přístup k výsledkům tak, že naprogramujeme

CLR = +/– +/– **RST**

a na krok *00* umístíme **RST.** Displej je teď tmavý, program běží a jediná možnost rozsvícení displeje je **R/S** a např. **RCL 1.** Displej se však rozsvítí až po stisku **RCL 1.** Jedinou nevýhodou tohoto „zámku" je nepatrné zvětšení odběru způsobené během programu.

Modifikací postupu, tvořícího písmena, lze získat skryté kódy, se kterými se mnohý majitel TI57 jistě setkal v okamžiku, kdy akumulátory kalkulátoru byly na hranici vybití a zasunutý adaptér zachoval vzniklé kódy v programové paměti. Jsou to např. kódy 2nd, LRN apod. Lze je získat následujícím způsobem:

RST LRN Fix SST Lbl 2. (desetinná tečka) R/S LRN RST R/S

objeví se opět 04 00 0 a stiskem *0* až *7* vytvoříme jeden z kódů **2nd, LRN, SST, BST, GTO, SBR, RST, R/S,** tedy *11, 21, 31, 41, 51 . . . 81.* Užitečnost těchto kódů je poněkud sporná.

Kód **2nd** (tedy *11*) se chová podobně jako funkce Last X u kalkulátorů firmy Hewlett Packard. Ukažme si to na malém příkladu: Nastavte programový čítač na adresu s instrukcí *11* a zadejte **LRN,** dále libovolné číslo např. 555=a vymažte kalkulátor postupem **INV C. t.** Stisknete-li teď **SST**, objeví se 00 11 a po stisku = se rozsvítí vymazané číslo 555. Po důkladnější analýze zjistíte, že funkce *11* vrátí na displej číslo v absolutní hodnotě mantisy a absolutní hodnotě exponentu, tedy z –0,5 vytvoří 50 apod. Pokud počítáte 5×6 a přečtete pomocí **SST** funkci *11*, vrátí se na displej první činitel, tedy 5. V tomto případě je nutno zaměnit = následující za 11 za funkci |x|, neboť jinak se vyvolané číslo 5 dosadí za druhého činitele a dostaneme součin 5 × 5, tedy 25.

Kód *21* představuje funkci **LRN**, neboli přepnutí do programového módu. Přečteme-li ručně *21* pomocí **SST**, přepne se kalkulátor do programového módu stejně jako po stisku **LRN**, ale přečte-li kalkulačka kód *21* při běhu programu, běží dále v módu **LRN**, na displeji blikají čísla adres a instrukcí, konkrétně byl-li kód *21* na sudém kroku, běží program pouze po lichých a naopak. Zapsané instrukce přirozeně nevykonává a reaguje pouze na stisk **SST** pauzou, po stisknutí **R/S** se nezastaví, ale dále běží po všech adresách a funkce **R/S** se na všechny proběhnuté adresy zapisuje. Zpět do normálního módu se přepne pouze druhou instrukcí **LRN** umístěnou těsně za první, jinak se přepne po doběhnutí na konec programového registru.

Funkce **SST**, reprezentovaná kódem *31*, funguje při běhu programu jako **R/S**, při krokování **SST** program přečte ještě následující instrukci.

Ještě méně se projevuje funkce **BST** (kód *41*) – při běhu programu se neprojevuje vůbec (kromě dosazení čísla do operace), pouze při běhu v modu **LRN** se projeví tak, že program doběhne např. na krok *25*, přečte **BST**, skočí o krok zpět, opět čte **BST**, což na displeji uvidíme jako poblikávání 25 41, které narušíme pouze stiskem **R/S.**

Pro úplnost uvádím, že podobně lze tvořit i kódy dalších sloupců tlačítek.

Postupem

LRN Fix SST Lbl 2 A (kód 10) R/S LRN RST R/S LRN

svítí znovu 04 00 0 a po stisknutí 0,1, . . . vytvoříme kódy *12, 22* atd. Místo kódů číslic se tvoří chaoticky běžné sdružené kódy. Jediným zajímavým z takto vzniklých kódů je *12*, tedy **INV**, což je nová funkce zasahující při několikanásobném aplikování těžko definovatelným způsobem do exponentu čísla. Např. máme číslo 1 EE 55, po aplikaci funkce *12* vznikají postupně čísla 1 46, 1 37, 1 28, 1 19, 1 10, 1 01, 1–08, 1–01,1–08 atd.

Tytéž kódy v inverzní podobě můžeme vytvořit tak, že do postupu vsuneme před **Fix** číslici *1* až *9*; na funkci to však nic nemění.

Pro všechny skryté kódy platí, že po vytvoření kódu lze celý ostatní postup vymazat.

Dále lze postupem

LRN Fix SST Lbl 2 EE R/S LRN RST R/S LRN

obdobně vytvářet kódy *13, 23* atd., ale ty jsou naprosto běžné.

Úplně na závěr bych se chtěl zmínit o nekonvečním adresování skoků na TI57. Místo **GTO N** lze psát **GTO SST N**, jak bylo popsáno v AR A8/82. Dále lze počet labelů rozšířit použitím písmen – za N dosadíme např. *15* (F). Tyto skoky lze uskutečňovat pouze uvnitř programu a to jen nazpět. Podrobněji se tímto adresováním zabýval Ivan Přenosil ve svém článku v ST 4/1980. Stejně tak se zabýval i nepřímým adresováním takovéhoto číselného labelu. Spočívá v nedoplnění instrukce **SBR** postupem **SBR SST**. Program potom skočí na číslici podle absolutní hodnoty exponentu. Zde však nelze použít písmena, lze ale použít množství dalších kódů instrukcí; pak je možno nepřímo adresovat postupem:

RCL N (0 až 7) INV log SBR SST

pokud máme nepřímou adresu na displeji, je možno vynechat **RCL N.**

Zjištění adresy: Máme-li např. kód *81* (**R/S**), vznikne exponent nutný ke skoku na tuto instrukci tak, že čteme kód odzadu a odečteme 1: pro **R/S** to bude 18 – 1 = *17*. Program tedy skočí na první **R/S** v programovém registru, bude-li při **SBR** na displeji 1 EE 17. Podobně lze odvodit změny kódů pro mnohé další instrukce, ovšem v mezích 10 až 99. Lze uplatnit i popsané skryté kódy, např. pro 1 EE10 program skočí na *11* neboli **2nd**. Pro inverzní kódy platí podobný předpis: opět číst pozpátku a přičíst 7, např. **INV CLR** –15 . . 51+7=58.

Pro některé exponenty nenajdete těmito postupy adresu, potom buď není mezi běžnými kódy, nebo je její kód sdružený; např. pro 1 EE 95 je adresa **SBR 6**. Tuto problematiku nechávám čtenářům k samostatnému zkoumání, pouze upozorňuji, že popsané výpočty neplatí při přechodu přes desítku!

Zbyšek Bahenský