Kompilátory jazyka Basic pro ZX Spectrum

Ing. M. Veverka, A. Mareš

ZX Spectrum obsahuje jeden z nesčetných dialektů jazyka Basic implementovaný jako interpret. Co to je interpret a jak pracuje? Zdrojový program (ten, který si uživatel napíše) uložený v paměti RAM je čten příkaz za příkazem a každý příkaz je ihned přeložen a vykonán. Překlad probíhá tedy zároveň s výpočtem. To má své výhody i nevýhody. V podstatě jedinou nevýhodou (dost velikou) je to, že kolikrát chceme program spustit, tolikrát bude interpret dešifrovat zdrojovou verzi. Pokud má být nějaká část programu vykonána třeba stokrát, stokrát se také přeloží. Proto programy prováděné interpretry běží velmi pomalu. Výhodou je interaktivní režim práce a velmi snadné ladění programů. Interpret ve Spectru upozorní na chybu v syntaxi blikajícím otazníkem za chybou a chybný řádek či příkaz prostě 'nevezme'. Tato syntaktická kontrola zabrání ukládání chybných řádek do paměti a ušetří spoustu času při ladění.

Druhým způsobem implementace programovacího jazyka v počítači je kompilátor neboli překladač. Kompilátor zdrojový program v několika průchodech přeloží přímo do strojového kódu daného mikroprocesoru. Podstatné je, že překlad proběhne pouze jednou a po překladu se spouští přímo cílový kód (tj. to, co kompilátor vyrobil z našeho programu). Nevýhodou této implementace je těžkopádnější ladění programu. Nejprve se opraví chyba ve zdrojovém programu, pak se program přeloží a překladač označí chyby. Znovu opravíme zdrojovou verzi, po (teď již) bezchybném překladu spustíme cílový kód a objevíme další chybičku ... Ladění programu je tedy mnohem (nejen časově) náročnější, ale jsme poctěni rychlým během přeloženého programu. Je vidět, že kompilátor je vhodný pro často se opakující výpočty, kde nám záleží na krátké době výpočtu.

Pro ty, kteří chtějí zvýšit výkon svých programů (po časové stránce) a nechtějí se (zatím) zabývat jinými jazyky (Pascal, FORTH, C jazyk) či přímo strojovým kódem, jsou tu kompilátory jazyka Basic. Pro Spectrum jich u nás kolují asi dvě desítky (včetně jednotlivých verzí), čili je z čeho vybírat. Při práci s kompilátorem na Spectru využijeme vlastně těch lepších vlastností interpretu a kompilátoru. Pomocí interpretru program velice snadno odladíme a pak bezchybný program kompilátorem přeložíme a spouštíme cílový kód. Ovšem zároveň se projeví i nevýhoda při použití většiny kompilátorů - v paměti počítače musí být zdrojový program, cílový kód a vlastní kompilátor, protože cílový kód využívá podprogramy kompilátoru.

Kompilátory lze rozdělit do dvou skupin - na celočíselné a s plovoucí řádovou čárkou. Celočíselné (Integer) kompilátory pracují s celými čísly v rozsahu od -32768 do 32767. To jim umožní vytvářet velice rychlé cílové kódy, ale nemají (ve většině případů tolik potřebnou) aritmetiku s desetinnými čísly. Tu zase umožňují kompilátory s plovoucí čárkou (Floating Point - FP), ale jimi vytvořený cílový kód běží pomaleji než u

celočíselných kompilátorů. Každý kompilátor má své specifické vlastnosti, pro které je vhodný či nevhodný pro to konkrétní použití. Proto je dobré znát omezení jednotlivých kompilátorů, abychom svůj program nemuseli rozsáhle 'překopávat' a mohli se možnostem kompilátoru podřídit. Od toho je jich tu celá řada, abychom si vybrali ten nejvhodnější.

Vyzkoušeli jsme všechny nám dostupné kompilátory, celočíselné i s plovoucí řádovou čárkou. K prověření jsme použili mírně upravené testovací programy (benchmarks) určené spíše pro posouzení rychlosti interpretrů. Rychlosti jednotlivých kompilátorů jsou shrnuty v tabulce na konci textu. Rychlosti uvedené v tabulce je nutno brát s rezervou, protože jsou testovány jednoduché programové konstrukce. Výsledné zrychlení na 'normálních' programech je u celočíselných kompilátorů asi 10 až 50 krát a u FP kompilátorů 2 až 10 krát, velmi totiž záleží na struktuře programu.

Předložený text je v podstatě 'výcuc' z nám dostupných návodů anebo naše dosavadní zkušenosti z používání kompilátorů, u kterých jsme návod neměli k dispozici (SUPER C, PUMMA)

Testovací programy

```
BM 1
100 POKE 23674,0:POKE 23673,0
    :POKE 23672,0
400 FOR N=1 TO 10000
500 NEXT N
900 PRINT PEEK 23672+256*PEEK
    23673
910 BEEP 1,30
930 STOP
```

```
BM 2
100 POKE 23674,0:POKE 23672,0:
     POKE 23672,0
400 LET N=0
500 LET N=N+1
600 IF N<10000 THEN GOTO 500
900 PRINT PEEK 23672+256*PEEK
    23673
910 BEEP 1,30
930 STOP
```

```
BM 3
100 POKE 23674,0:POKE 23673,0
    :POKE 23672,0
400 LET N=0
500 LET N=N+1
510 LET A=N/N*N+N-N
600 IF N<10000 THEN GOTO 500
900 PRINT PEEK 23672+256*PEEK
    23673
910 BEEP 1,30
930 STOP
```

```
BM 4
100 POKE 23674,0:POKE 23673,0
    :POKE 23672,0
400 LET N=0
500 LET N=N+1
510 LET A=N/2*3+4-5
600 IF N<10000 THEN GOTO 500
900 PRINT PEEK 23672+256*PEEK
    23673
910 BEEP 1,30
930 STOP
```

```
BM 5
100 POKE 23674,0:POKE 23673,0
    :POKE 23672,0
400 LET N=0
500 LET N=N+1
```

```
BM 6
100 POKE 23674,0:POKE 23673,0
    :POKE 23672,0
400 LET N=0
420 DIM D(5)
```

```
510 LET A=N/2*3+4-5
520 GOSUB 950
600 IF N<10000 THEN GOTO 500
900 PRINT PEEK 23672+256*PEEK
    23673
910 BEEP 1,30
930 STOP
950 RETURN
```

```
500 LET N=N+1
510 LET A=N/2*3+4-5
520 GOSUB 950
530 FOR L=1 TO 5
540 LET D(L)=A
550 NEXT L
600 IF N<10000 THEN GOTO 500
900 PRINT PEEK 23672+256*PEEK
    23673
910 BEEP 1,30
930 STOP
950 RETURN
```

BM 7

```
100 POKE 23674,0:POKE 23673,0
    :POKE 23672,0
400 FOR N=1 TO 200
500 LET A=SQR N
510 LET A=A^2
520 LET A=SIN N
530 LET A=ASN A
540 LET A=LN N
550 LET A=EXP A
560 LET A=SGN N
570 LET A=ABS N
580 LET A=INT N
590 LET A=RND
600 LET A=COS N
610 LET A=ACS A
620 LET A=TAN N
630 LET A=ATN A
700 NEXT N
900 PRINT PEEK 23672+256*PEEK
    23673
910 BEEP 1,30
930 STOP
```

BM 9

```
100 POKE 23674,0:POKE 23673,0
    :POKE 23672,0
300 FOR A=0 TO 175
400 FOR B=0 TO 255
500 PLOT B,A
600 NEXT B
700 NEXT A
900 PRINT PEEK 23672+256*PEEK
    23673
910 BEEP 1,30
930 STOP
```

BM 9

```
100 POKE 23674,0:POKE 23673,0
    :POKE 23672,0
300 FOR A=0 TO 87
500 CIRCLE 127,87,A
700 NEXT A
900 PRINT PEEK 23672+256*PEEK
    23673
910 BEEP 1,30
930 STOP
```

BM10

```
 10 POKE 23674,0: POKE 23673,0: POKE 23672,0
 20 LET X=11: LET R=87
 30 FOR A=0 TO X-1
 40 FOR B=0 TO A
 50 LET P=360*A/X: GO SUB 200: GO SUB 260
 60 LET J=128+R*C/100: LET L=87+R*S/100
 70 LET P=360*B/X: GO SUB 200: GO SUB 260
 80 LET F=128+R*C/100: LET G=87+R*S/100
 90 IF ABS (L-G)>ABS (J-F) THEN GO TO 140
100 LET U=J: LET H=F: LET O=G-L: LET V=L
110 IF F<J THEN LET U=F: LET H=J: LET O=L-G: LET V=G
120 GO SUB 320
130 GO TO 170
140 LET U=L: LET H=G: LET O=F-J: LET V=J
150 IF G<L THEN LET U=G: LET H=L: LET O=J-F: LET V=F
160 GO SUB 360
```

```
170 NEXT B
180 NEXT A
190 GO TO 400
200 LET Q=P: LET M=1
210 IF P>270 THEN LET Q=360-P: LET M=-1: GO TO 240
220 IF P>180 THEN LET Q=P-180: LET M=-1: GO TO 240
230 IF P>90 THEN LET Q=180-P:LET M=1
240 IF Q<35 THEN LET S=M*10*Q/6: RETURN
250 LET W=90-Q: LET S=M*(100-W*W/70): RETURN
260 LET Q=P: LET M=1
270 IF P>270 THEN LET Q=360-P: LET M=1: GO TO 300
280 IF P>180 THEN LET Q=P-180: LET M=-1: GO TO 300
290 IF P>90 THEN LET Q=180-P: LET M=-1
300 IF Q<=55 THEN LET C=M*(100-Q*Q/70): RETURN
310 LET W=90-Q: LET C=B*10*W/6: RETURN
320 FOR N=U TO H
330 PLOT N,INT ((10*V+5+10*(O*(N-U)/(H-U+1)))/10)
340 NEXT N
350 RETURN
360 FOR N=U TO H
370 PLOT INT ((10*V+5+10*(O*(N-U)/(H-U+1)))/10),N
380 NEXT N
390 RETURN
400 PRINT PEEK 23672+256*PEEK 23673
410 STOP
```

Popis jednotlivých kompilátorů

1. SUPER C — SOFTEK COMPILER 1982 ANDREW GLAISTER

```
SUPER C     BASIC    10     707
            CODE  40500    1500
            CODE  49000    4800
```

Celočíselný kompilátor.
Nahrání: LOAD ""
Spuštění překladu: RANDOMIZE USR 49152
Při běhu cílového kódu musí být kompilátor v paměti.
Nepřeložitelné příkazy vyvolají standartní chybová hlášení s číslem řádku i číslem příkazu.
Při psaní programu je nutno přejít do módu velkých písmen.
Po úspěšné kompilaci je zdrojový program vymazán a nahražen přeloženým a jsou vygenerovány řádky

```
   1    PROPERTY OF SOFTEK
ALL COMMERCIAL RIGHTS RESERVED
   4 RANDOMIZE USR 25394
```

Cílový program se spustí příkazem RUN, nejde brejknout.
Zdrojový program je vhodné ukončit příkazem STOP.
Po NEW zůstává přeložený program v paměti a spouští se příkazem RANDOMIZE USR 25394.
Kompilátor zpracovává celá čísla od -32768 do 32767.
Příkazy, které neumí přeložit: REM, DEF FN, FN, DIM, CLEAR N, AND, OR, BIN, nezná řetězce a z toho plynoucí příkazy pro práci s nimi, barevné kódy (PAPER, INK,...), příkazy pro pásku a mnohé další.
Poněvadž k tomuto kompilátoru nemáme návod, neumíme ho plně využívat a ani některé základní funkce jsme neobjevili - např. záznam přeloženého programu na pásku.

2. ZIP Issue 1.3 02.08.84

ZIP	BASIC	10	287
	CODE	23000	224
ZIP-LIB	CODE	53247	1500
zip basic	BASIC		16095

Nahrání: LOAD ""
Po nahrání se vypíše chybové hlášení C Nonsence in BASIC, 70:3 a je nutné vložit INK 0.

Celočíselný kompilátor napsaný celý v Basicu (16 KB + 1500 bajtů knihovna). V paměti zbývá asi 13 KB pro zdrojový program a asi 10 KB pro cílový kód.
Zdrojový text je možno vložit ručně anebo jej nahrát z pásky příkazem MERGE "" (příkaz LOAD by vymazal vlastní překladač). Je třeba zajistit, aby čísla řádek zdrojového programu nepřesáhla 4999.

Překlad se spustí příkazem GOTO 5000 a uživatel je o překladu průběžně informován. Vlastní překlad trvá velmi dlouho, zato výsledný cílový kód je nejrychlejší ze všech zkoušených kompilátorů.
Po skončení překladu ZIP oznámí potřebné údaje ke spuštění a k zaznamenání přeloženého programu na pásek.

Příkazy, které ZIP umí zpracovat: ABS, AND, AT, ATTR, BIN, BORDER, BRIGHT, CHR$, CLS, FLASH, FOR, IF, IN, INK, INVERSE, LET, NEXT, NOT, OR, OUT, OVER, PAPER, PAUSE, PEEK, PLOT, PRINT, POKE, REM, RETURN, RANDOMIZE, SGN, STEP, STOP, TAB, THEN, TO.

Příkazy, pro které platí jistá omezení: CLEAR, DIM - velikost pole smí být konstanty, DRAW - nelze užít třetí parametr, GOTO a GOSUB pouze číselný parametr (ne proměnná), INPUT - jen pro jednu proměnnou a nelze užít zprávu, INT je ignorováno, RANDOMIZE USR - jen číselný parametr.
Další vlastnosti:
- 26 jednoznakových proměnných, 26 polí
- operace + - * /
- celá čísla v rozsahu od -32768 do 32767
- lze použít č. do 65535 bez následných operací
- nepřípustné goniom. funkce, stringy, streamy (PRINT #...), RND, INKEY$.

Cílový kód se spouští od adresy uvedené po ukončení překladu (vhodne si poznamenat u každého přeloženého programu). Na pásek se s cílovým kódem nahraje i knihovna podprogramů nutná pro běh cílového kódu, takže ten je možno nahrát do 'čistého' počítače příkazem LOAD "" CODE (po předchozím CLEAR 53246 nebo nižším) a k jeho činnosti není překladač nutné mít v paměti.

Rozsah návodu: 1 blok v Taswordu (139 řádek).

Kompilátory - 2.část

3. MCODER 2 SPECTRUM INTEGER COMPILER v 2.0

1983 Hodgson and Threlfall

```
MCODER2      BASIC      1     870
MCODER2      CODE   59990    5373
```

Celočíselný kompilátor.
Nahrání: LOAD ""
Spuštění překladu: RANDOMIZE USR 60000

Při překladu se jednotlivé řádky zobrazují na obrazovce (při obou průběžích) spolu s délkou výsledného kódu (v bílém obdélníčku vpravo nahoře). Při nepřeložitelném příkazu se za chybou zobrazí blikající otazník a tento řádek je připraven k editaci. Po ukončení překladu je vhodné si poznamenat startovací adresu cílového kódu a jeho délku. Počáteční adresa je 40000, je možno ji příkazem CLEAR N posunout jinam (vhodné pro postupnou kompilaci podprogramů).

Spuštění cílového kódu: RANDOMIZE USR 40000 nebo dle nastaveného RAMTOPu - kompilátor musí být v paměti počítače.
Záznam přeloženého programu na kazetu: SAVE "JMENO" X,Y
kde čísla X a Y udávají startovní adresu a délku cílového kódu (uvedené na konci překladu).

Zpracování čísel: od -32768 do 32767, u příkazů POKE a PEEK čísla do 65535.

M-CODER 2 zpracuje tyto příkazy:

AND, OR - jen uvnitř příkazu IF
BORDER, BRIGHT, FLASH, INK, INVERSE, OVER, PAPER
CIRCLE, DRAW, PLOT
ABS, ATTR, BEEP x,y - pro kratší tón než 1 sec. BEEP 1/3,20 .
CHR$, CLS, CLEAR, CODE, COPY, DATA, FOR..TO..NEXT (STEP je vždy 1), GO SUB, GOTO, IF..THEN, IN, INKEY$, INPUT i s komentářem jako u PRINT, INT, LET, LEN a$ - a$ nemůže být uvnitř funkce zpracován, LPRINT, NEW, OUT m,n , PAUSE n - zastaví program na n/50 sekund, PEEK, POINT, POKE m,n ,
PRINT "řetězec"
      číslo
      CHR$ n
      AT n,m
      TAB n
      barevný příkaz
      řetězcová proměnná

RANDOMIZE, READ, REM, RESTORE n , RETURN, RND - celé číslo v rozsahu 0 až 32767, SGN, STOP - poslední basicovský příkaz, který má být přeložen - v tomto místě kompilátor končí práci, DIM - pouze jednorozmérné číselné nebo stringové pole

řetězce: normálně lze překládat délku 32 znaků, (při překročení jsou bajty nad 32 přepsány). Při požadované jiné délce je nutno vložit POKE 60200,n kde n je požadovaná délka řetězce v rozsahu 1 až 255. Toto nastavení je nutno provést dříve než řetězec definujeme.

Při použití příkazu TO pro členění řetezců se musí užít obecná forma A$ (n) nebo A$(n TO m), nikoliv vynechávání jako např. A$(TO m).

Použití příkazu REM :
REM #0 - funkce BREAK použita jen při dotazu 'scroll?' a v příkazech INPUT. Nejrychlejší běh programu.
REM #1 - BREAK funguje normálně, inicializován na začátku, rychlost menší.
REM #2 - BREAK funguje normálně, vpravo nahoře se zobrazuje právě prováděný řádek.

Rozsah návodu : 1 blok v Taswordu.

4. MCODER 1

SPECTRUM INTEGER COMPILER v 1.0
1982 Hodgson and Threlfal

M-CODER	BASIC	10	875
COMPILER	CODE	28990	3600

Celočíselný kompilátor pro 16 kilové Spectrum provozuschopný i na 48 kilovém Spectru. Oproti verzi MCODER 2 má některá omezení, ale zůstává volná oblast paměti od adresy 32768 až do konce (48KB verze), kterou lze využít např. pro data, poněvadž příkazy POKE a PEEK mohou mít argument v rozsahu 0 až 65535.

Nahrání: LOAD ""
Spuštění překladu: RANDOMIZE USR 29000

Spuštění cílového kódu: RANDOMIZE USR 26500 nebo dle nastaveného RAMTOPu (v rozmezí 24000 až 28000), kompilátor musí být v paměti počítače.

V podstatě pro něj platí co pro MCODER 2 až na tyto rozdíly:
- po spuštění překladu proběhne první průběh i s listingem překládaného programu, program pák čeká na stisk libovolného tlačítka - odstartování druhého průběhu opět v výpisem programových řádek, vpravo nahoře se zobrazuje poslední obsazená adresa cílovým kódem

- adresu spuštění cílového kodu je si třeba pamatovat, ponévadž kompilátor ji nezobrazí (standartně nastavena na 26500)
- zdrojový program není nutno ukončit příkazem STOP, je-li však v programu obsažen, bude program přeložen jen do tohoto místa

Omezení oproti verzi MCODER 2:
- nelze užít AND, OR, CHR$, LEN a$, NEW
- CODE e omezeno na e="a" nebo INKEY$
- DIM z(x) - jenom jedno pole čísel s jménem 'z'
- u příkazu PRINT nelze užít řetězcovou proměnnou
- není možné zavádět řetězcové proměnné a operace s nimi

Uložení cílového kódu na pásek: SAVE "jméno" CODE a,b
kde a je počáteční adresa (26500 nebo podle nastavení CLEAR a)
b je délka cílového kódu (zobrazená hodnota v bílém obdélníčku po ukončení překladu minus hodnota a)

Platí též možnosti užití příkazů REM # n

5. COLT COMPILER

HiSoft COLT COMPILER v 1,0
1985 Hodgson and Threlfall

```
COLT COMP.  BASIC     1   2920
MC          CODE  53750  11619
```

COLT COMPILER je v podstatě přepracovaný a doplněný komplilátor MCODER 2. Vznikl tak docela slušný celočíselný kompilátor plně slučitelný s Interface 1 a Microdrive. Použitím určitých fint lze na něm pracovat i desetinnými čísly. Navíc obsahuje blok EXECUTIVE, který nabízí řadu dalších příkazů Basicu.

Zavedení do počítače: LOAD "". Potom následuje dotaz, zda chceme pořídit pracovní kopii kompilátoru (Y/N), a chceme-li změnit počáteční adresu 40000 (Y/N).

Vlastnosti COLTu:
- číselný rozsah od -32768 do 32767 (celá čísla)
- plné zpracování řetězců
- plné pojmenování proměnných (např. LET colt=100)
- přístup k proměnným Basicu z přeloženého programu
- přístup k proměnným COLTu z Basicu
- kompilace až 32 KBajtů Basicu
- řetězcové výrazy nesmí obsahovat závorky jako např. a$=(b$+c$)
- v příkazu PRINT musí být porovnání řetězců v závorkách např. PRINT ("x"<"y")

Příkazy:
- BORDER, BRIGHT, FLASH, INK, INVERSE, OVER, PAPER
- CIRCLE, DRAW, PLOT, POINT
- LOAD, SAVE, VERIFY, MERGE - všechny tvary mimo ...DATA,

- GO TO, GO SUB, RETURN
- READ, DATA, RESTORE - DATA nesmí obsahovat výrazy
- DIM a(v), DIM a$(v) - jenom jednorozměrné, řetězec max.255 zn.
- AND, OR, IN, OUT, POKE, PEEK, ABS, BIN, CHR$, CODE, INT, SGN, STR$, LEN a$ - a$ nelze dále členit (TO), RND - celé od -32768 do 32767
- PRINT, AT, ATTR, SCREEN$, TAB
- FOR a=u TO v STEP w - w volitelné, rozdíl mezi u a v nesmí přesáhnout 32767
- IF e THEN
- CLS, CLS # pro Interface 1
- CLEAR - maže jen kompilované proměrné, CLEAR # pro Interface 1 CLEAR n není povoleno
- CONTINUE - nelze užít v programu, používá se k označení konce kódu ke kompilaci
- INKEY$, INKEY$ #, INPUT, INPUT # e pro RS-232 či Microdrive
- TO - pro členění řetězců, všechny tvary povoleny
- USR e, USR "řetězec"
- VAL a$, VAL$ a$ - jestliže a$ obsahuje odkaz na proměnnou, užije se Basicovské proměnné
- BEEP, COPY, LET, LINE, LPRINT (#),NEW, PAUSE, STOP
- CAT, CLOSE, COPY, ERASE, FORMAT, MOVE, OPEN - všechny tvary
- DEF FN, FN - NEJSOU povoleny

Použití příkazu REM # :
REM # 0 - BREAK vypojen s výjimkou scroll? , INPUT a během příkazů pro Microdrive, nejrychlejší běh programu
REM # 1 - BREAK funguje normálně, pomalejší
REM # 2 - jako # 1, navíc se zobrazí číslo právě prováděného řádku (je-li v EXECUTIVE aktivováno)

Spuštění překladu: RANDOMIZE USR 60000

Chybný řádek je zároveň editačním. Po ukončení kompilace se objeví na obrazovce několik informací, z nichž nejdůležitější jsou (vhodné si poznamenat):
Begin - adresa počátku cílového kódu
End - adresa konce cílového kódu
Ctop - poslední volná adresa v paměti použitelná pro cílový kód

Uložení cílového kódu na pásek: SAVE "..." CODE Begin,End-Begin
nebo SAVE "..." CODE Begin,65536-Begin i s COLTem.

Spuštění přeloženého programu (COLT musí být v paměti):
RANDOMIZE USR Begin (Begin po inicializaci je 40000)

Nahrání cílového kódu do počítače: LOAD "" CODE Begin

Blok EXECUTIVE:
Součástí COLTu je blok EXECUTIVE, který obsahuje řadu dalších příkazů použitelných jak v Basicu, tak i v přeloženém programu. Popis příkazů přesahuje rámec tohoto příspěvku, zde jen pro informaci vyjmenujeme některé vlastnosti (vlastní popis je součástí návodu ke COLTu):
použití SPRITE, oken, definice uživatelských kláves, vnitřní hodiny a jejich řízení, dvoubajtové POKE a PEEK, konverze mezi

čísel, soustavami (hexa, bin, dekad.), dotaz na volnou paměť, příkaz ON ERROR GOTO a další.

Po nahrátí COLTu je nastaven mód zobrazení hodin vlevo nahoře. Komu se to nelíbí, nechť vloží: *fx0

Blok EXECUTIVE zabírá v paměti místo od 58750 do 59200, Inicializuje se (např. po použitém NEW) RANDOMIZE USR 55020, odpojí se (ale nesmaže) RANDOMIZE USR 55010. Je-li třeba zvětšit prostor v paměti pro přeložený program, lze blok nenávratně smazat příkazem RANDOMIZE USR 59190.

Rozsah návodu: 5 bloků v Taswordu.

6. IS COMPILER

SOFTEK 'IS' INTEGER COMPILER
v 1.2 1983 Martin Lewis

```
COMPILER    BASIC      10    705
MACHINECODE CODE    59300   6000
```

Celočíselný kompilátor.
Nahrání kompilátoru: LOAD ""
Spuštění překladu: RANDOMIZE USR 59300
Během překladu kompilátor vypisuje sdělení:
START ADDRESS - počátek cílového kódu, možno změnit CLEAR n
END ADDRESS - konec cílového kódu
VARIABLES END - konec proměnných

Konec překladu je indikován hlášením NO ERRORS.
V případě chyby se chybný řádek zobrazí a za chybou bliká otazník. Tento řádek je zároveň editačním.

Rozsah čísel od -32769 do 32767. U některých funkcí je rozsah od 0 do 65535 (PEEK, POKE, USR).

Příkazy:
BORDER, BRIGHT, FLASH, INK, INVERSE, OVER, PAPER
CIRCLE, DRAW, PLOT
BEEP, CLS, CLEAR, COPY, LET, NEW, PAUSE, PRINT, RANDOMIZE, STOP
READ, RESTORE, DATA - jen čísla a řetězce, nikoliv proměnné
FOR...TO..STEP
GOTO, GOSUB, RETURN
IF...THEN...
INPUT - mimo LINE
LPRINT -jen s #, OUT, POKE
LOAD "..." CODE - jen jeden parametr
SAVE, VERIFY "..." CODE e,f

REM příkazy:
REM B - testuje stisk BREAK
REM M,k,l,m,... - vložení strojového kódu a jeho volání
REM S,a,x,y - tisk znaku s ASCII kódem 'a' na PLOTovou pozici x,y

řetězce: max. délka 256 znaků. Povoleny tvary i s TO. Dále je možno užít LEN STR$, CODE CHR$, CODE INKEY$, SCREEN$, + pro spojování.

Funkce: ABS, AND, ATTR, CHR$, CODE, IN, INKEY$, LEN, NOT, OR, PEEK, POINT, RND (od 0 do 32767), SCREEN$, SGN, STR$, USR, VAL, operátory + - * / a všechny porovnávací.

Spuštění přeloženého kódu: RANDOMIZE USR start address (kompilátor musí být v paměti).

Záznam přeloženého programu na pásku:
SAVE "..." CODE start address,65536-start address -včetně kompilátoru, anebo jen samotný cílový kód :
SAVE "..." CODE start address, end address - start address

7. FP COMPILER SOFTEK 'FP' FULL COMPILER
v 1.7 1983 Martin Lewis

FP48K v1.7	BASIC	20	36	verze 1.7
	CODE	23500	1000	
m/CODE	CODE	59300	6100	
COMPILER	BASIC	10	707	verze 1.1
MACHINECODE	CODE	59300	6050	

Kompilátor s plovoucí čárkou.
Je to 'vlastní bratr' IS kompileru, a tudíž pro něj platí všechno, co bylo u IS kompileru uvedeno s tím rozdílem, že zpracovává čísla s plovoucí čárkou ve stejném rozsahu jako Basic. Mezi oběma verzemi jsme neshledali žádný rozdíl.

Nové příkazy: n - celé kladné číslo, $ - řetězec
CLEAR n
CLOSE # n,$
DIM a(n) nebo a$(n) - pouze jednorozměrné pole
OPEN # n,$

REM příkazy:
REM B - testuje stisk BREAK
REM M,k,l,m,... - vložení strojového kódu a jeho volání
REM E,n - při chybě je proveden skok na řádek číslo n, tím se ale poškodí zásobník návratových adres, proto návrat do Basicu může být proveden jen skokem z programu nebo příkazem STOP
REM N - nastaví v případě chyby běžný mód (provést před návratem do Basicu)
REM O,a,k,l,... - simulace Basicovské funkce ON a GOTO k,l,... a musí být jednoduchá proměnná, je-li větší než nejvyšší číslo řádku, pokračuje se za příkazem

Nové funkce: SIN, COS, ATN, LN, ... a další

Rozsah návodu: 1 blok v Taswordu společně s IS kompilerem.

8. BLAST 1985 OCSS I3.7

BLAST	BASIC	5	619
logo1	SCRN	16384	6912
CBLAST	CODE	30200	35350
CHCHK	CODE	23301	50
RBLAST	CODE	23734	4958
TOOLKIT	BASIC	10	492
tk code	CODE	61917	3550

verze I3.7

toolkit k verzi I3.7

BLAST	BASIC	0	801
logo1	CODE	16384	16384
CBLAST	CODE	30800	24800
CHCHK	CODE	23301	50
RBLAST	CODE	23734	5277

verze I2.0

TKBLAST	BASIC	10	227
tkcode	CODE	60497	5039

toolkit k verzi I2.0

Kompilátor s plovoucí čárkou.

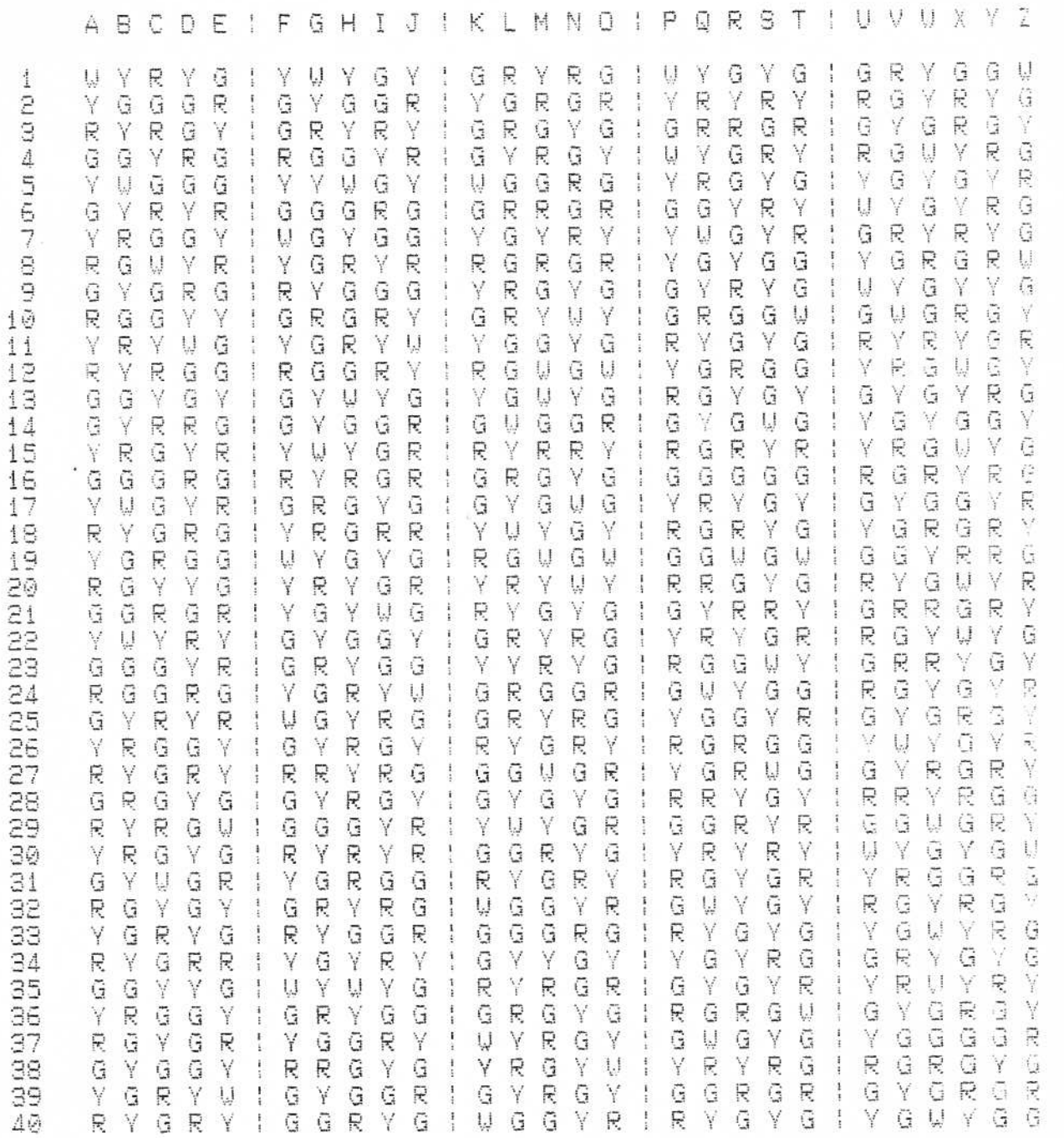
Schopen překládat libovolně dlouhé programy.

Nahrání BLASTu: LOAD ""

Po nahrání se dotáže, zda chceme vytvořit pracovní kopii na Microdrive, poté provádí kontrolu ochrany prpgramu. K tomu slouží dekódovací tabulka, která je součástí návodu.

Verze I3.7 se bez této tabulky neobejde, verze I2.0 je již upravena a stačí na dotazy odmačkavat ENTER.

Dekódovací tabulka pro BLAST I3.7:



Takto je BLAST připraven k činnosti. Je schopen překládat programy do délky asi 2,5 KB. Pro delší překládané programy je nutno užít předzpracování programem TOOLKIT. Ten obsahuje řadu příkazů pro editaci, přečíslování a pod. Zde si všimneme jen nutných příkazů pro převedení našeho Basicovského programu do strojového kódu.
BLAST umožňuje překlad přímo do strojového kódu nebo do tzv. p-kódu. Strojový kód běží nejrychleji, nejde pak ovšem brejknout, p-kód je kratší než str. kód, lze ho brejknout, ale běží pomaleji. Typ cílového kódu se specifikuje příkazem (musí být jako první řádek programu):
REM !PCODE nebo REM !MACHINECODE

Zpracování programu TOOLKITem:
- nahrát TOOLKIT
- vložit Basicovský program (z pásky nebo ručně)
- vložit příkaz *B<jméno souboru>
- řídit se hlášeními TOOLKITu, který si žádá zapínat a vypínat magnetofon (zpracování delších programů trvá velmi dlouho)

Překlad krátkých programů (asi do 2,5KB) z paměti do paměti:
- nahrát BLAST
- nahrát nebo vložit zdrojový program
- příkazem *C spustit překlad
- spuštění přeloženého programu - *R
- vymazání zdrojového programu *N (NEW vymaže i BLAST)

Překlad dlouhých programů (delších než 2,5 KB) s použitím magnetofonu jako periferie:
- zpracovat program TOOLKITem (viz výše)
- resetnout počítač a nahrát BLAST
- příkazem *I nastavit vstupní zařízení (T = magnetofon)
- příkazem *O nastavit výstupní zařízení (T = magnetofon)
- příkazem *C spustit překlad - BLAST postupně vyzívá obsluhu ke vkládání zdrojové (zpracované TOOLKITem) a cílové (pro přeložený program) kazety
- po překladu BLAST sám provede reset počítače
- přeložený program se nahraje do počítace příkazem LOAD ""

Je vhodné přeložený program (skládá se z více bloků) nahrát do kopíráku FREE COPY nebo ISO COPY 2 a pak znovu uložit na pásek, protože BLAST nevytváří mezi hlavičkou a tělem bloku mezery a 'normální' kopíráky si s tím nevědí rady.

Výsledný 'zblastovaný' program obsahuje dvě řádky:
```
     5 RANDOMIZE USR 0: STOP
    10 LOAD "...." CODE 0: RANDOMIZE USR 0: GOTO 5
```

Lze ho spustit GOTO 5 nebo RUN, či je možné obě řádky vymazat a spustit RANDOMIZE USR 23792.

Uživatel může připisovat své programy v Basicu a spuštět je, aniž by to přeloženému programu nějak vadilo. Proměnné jsou normálně přístupné přeloženému programu i Basicu. BLAST posouvá počátek Basic programu za svůj cílový kód. O tom se lze přesvědčit ze systémové proměnné PROG na adresách 23635/6.

BLAST je schopen přeložit všechny příkazy Basicu, pouze příkaz CONTINUE nebude pracovat.

Rozsah návodu: 3 bloky v Taswordu.

9. TOBOS FP

FLOATING POINT COMPILER
Jerzy Borkowski Wojciech Skapa 1986

```
Tobos FP    BASIC     1    442
tobosc      CODE  53100  12269
```

Kompilátor s plovoucí čárkou.
Používá 4bajtového formátu čísel, z toho vyplývá přesnost čísel na 7 cifer, v tomto formátu je napsaná celá aritmetika včetně výpočtu funkcí, to vše se zřetelem na krátký čas výpočtu (v tomto bodě je informace uvedená ve Zpravodaji 7/8 Karolinky poněkud nepřesná). To znamená, že TOBOS je vhodný pro aritmetické výpočty, kde ostatní kompilátory nepřinesou výrazné zrychlení (viz program BM 7).

Nahrání TOBOSu: LOAD ""
Pak vymazat pomocný Basic příkazem NEW. Teď je možno vložit zdrojový program v Basicu z pásky nebo ručně.

Spuštění překladu: RANDOMIZE USR 53100
Po ukončení překladu kompilátor ohlásí počátek cílového kódu (nastavitelný příkazem CLEAR číslo) a jeho délku.

Omezení TOBOSu - nepřekládá příkazy obsluhy magnetofonu a Microdrive, CONTINUE a CLEAR s argumentem.

Záznam cílového kódu na pásek - dle instrukcí po překladu.

Spuštění cílového kódu - dle pokynu po překladu, kompilátor musí být v paměti. Zdrojový program zůstane zachován.

Dále lze užít možnosti překladu, která zruší program v Basicu. Tímto způsobem je možné překládat programy dlouhé asi 27 KB.
Postup: - CLEAR 53099 vymezí max. prostor pro Basic
- POKE 53240,0
- vložit zdrojový program (z kazety nebo ručně)
- spustit překlad RANDOMIZE USR 53100
- zdrojový Basic bude vymazán, RAMTOP se nastaví na 23999 a přeložený program se spouští RANDOMIZE USR 24000

U TOBOSu byla ověřena možnost relokace (přemístění) cílového kódu. Přeložený program je možné nahrát od jiné adresy, než pro kterou byl přeložen a po spuštění pracuje bezvadně. Možná je to pravidlem, možná jsme měli štěstí, že námi zkoušené programy to umožňovaly.

Rozsah návodu: 1 blok v Taswordu.

10. PUMMA KOMPILATOR 1996

compERE	CODE	23536	80
			4308
			3584

Pozn.: V hlavičce prvního bloku je oznámena délka 80 bajtů, ale ve skutečnosti se nahraje 81 bajt.

Kompilátor s plovoucí čárkou.

Postup při překladu:
- vložit Basicovský program z kazety nebo ručně
- smazat případné proměnné příkazem CLEAR
- vložit LOAD "" CODE a nahrát první dvě části kompilátoru a zastavit pásek - první průběh překladu
- vyčkat hlášení PLAY, spustit pásek a nahrát poslední část kompilátoru - druhý průběh překladu, v dolním řádku je uživatel informován o čísle právě překládaného řádku (tento průběh je pomalejší)
- konec překladu oznámí hlášení 0 OK s číslem posledního řádku

Při pokusu o vylistování přeloženého programu se na obrazovce objeví dvě řádky:
?000 RANDOMIZE USR 0
KOMPILATOR PUMMA software 1986

Obsahuje-li Basicovský program řádky s příkazy DATA nebo DEF FN, jsou tyto řádky zobrazeny před výše uvedenými.

Přeložený program lze spustit GOTO 1 nebo RANDOMIZE USR 26256. Je možné ho libovolně brejknout, vypíše se hlášení o čísle řádku a příkazu v místě zastavení, ale CONTINUE pak nefunguje. Nespornou výhodou PUMMA kompilátoru je možnost spustit přeložený program od libovolné řádky příkazem GOTO čř. Nedoporučuje se spouštět přeložený program příkazem RUN (někdy tak spustit program lze, ale většinou ne).

Po doběhnutí přeloženého programu se objeví zajímavé chybové hlášení: 0 OK, ?000:1

Záznam přeloženého programu na pásek: SAVE "jméno" (jako by šlo o Basic), není možné užít tvaru SAVE "..." LINE číslo

Omezení kompilátoru PUMMA: V příkazu DIM je nutné užít čísla, nikoliv proměnné. Při brejku nelze užít CONTINUE.

Nové řádky je sice možné připisovat, ale potom nejde spustit přeložený program, poněvadž nové řádky odsunuly přeložený program na vyšší adresy.

Námět pro experimentátory - vytvořit z přeloženého programu řádek např. 0 REM ..., pak by nově připsané řádky ležely za tímto programem. Podaří-li se to někomu, dejte vědět, prosím, na konci uvedenou adresu.

Kompilátory - 4.část

11. LASER COMPILER

COMPCODE CODE 59800 kompilátor

RTCODE CODE 59800 run-time modul (=knihovna podprogramů)

Kompilátor s plovoucí čárkou.
Určen ke kompilaci programů napsaných v LASER BASICu. Takto přeložené programy lze pak šířit jakýmkoliv způsobem bez licencí a poplatků (tolik původní návod). Tento kompilátor dává nejhorší výsledky ze všech námi zkoušených kompilátorů.

Postup při překladu:

- CLEAR 59799
- nahrát kompilátor příkazem LOAD "COMPCODE" CODE
- vložit Basicovský program z pásky nebo ručně
- spustit překlad RANDOMIZE USR 59800, konec je oznámen zprávou OK

Uložení přeloženého programu na pásek: SAVE "jméno" nebo SAVE "jméno" LINE 1.

Po provedení LIST se ukáže:
0 PRINT 0
1 RANDOMIZE USR 59800

Při použití uživatelských funkcí (DEF FN) se vytvoří další řádek s číslem 0 se všemi definicemi, stejně tak při použití DATA, opět s číslem 0 uložených v jediném řádku (RESTORE ale funguje normálně).

Spuštění přeloženého programu:
- vložit CLEAR 59799 (nebyl-li již vložen)
- nahrát run-time modul příkazem LOAD "RTCODE" CODE
- nahrát přeložený program
- spustit GOTO 1 nebo RANDOMIZE USR 59800

Přeložený program nejde brejknout.

Omezení LASER kompilátoru:

- u příkazů GOTO, GOSUB, RUN, RESTORE nepoužívat vypočítávané parametry, pouze jen čísla
- u DIM užívat čísla, nikoliv proměnné
- nezná CLEAR, MERGE, CONTINUE, LIST, LLIST
- RUN nemaže proměnné
- OPEN # a CLOSE # funguje ve vztahu k streamům "s", "k" a "p"
- SAVE, LOAD, VERIFY funguje normálně, ale program nahraný LOAD je považován za program přeložený kompilátorem a modul run-time se jej snaží rozběhnout od začátku

Chybové hlášení 'Program not compiled' znamená, že začátek programu neodpovídá začátku, jaký by vytvořil kompilátor.

Rozsah návodu: 2 bloky v Taswordu.

Porovnávací tabulka kompilátorů

U jednotlivých kompilátorů je v prvním řádku uvedena doba chodu programu v sekundách, v druhém řádku je relativní rychlost vůči interpretru Basicu (hodnoty jsou zaokrouhlené).

	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8	BM9	BM10
BASIC	42	81	200	192	229	770	199	366	66	307
SUPER C	.98 43	.54 150	11 19	10 19	10 22	-	-	19 20	76 .85	-
ZIP	.44 95	.64 126	5.1 39	1.1 171	1.2 191	5.3 143	-	17 21	-	11 29
M CODER 2	.26 162	.5 162	5.6 36	10 19	10 23	16 48	-	21 17	65 1	13 24
COLT	.38 111	.64 127	5.7 35	10 19	10 22	19 41	200 1	22 16	66 1	13 24
IS 48K	.48 88	.64 127	5.5 36	11 17	11 20	-	-	10 37	65 1	13 24
FP 48K v1.7	17 2.4	21 3.8	87 2.3	95 2	95 2.4	223 3.5	191 1	101 3.6	65 1	124 2.5
BLAST I3.7	3.5 12	20 4.1	79 2.5	93 2.1	97 2.4	345 2.2	200 1	82 4.4	68 .96	148 2.1
TOBOS FP	5.5 7.7	8.3 9.8	20 10	14 14	22 11	70 11	4.6 43	58 6.3	65 1	23 13
PUMMA	1.1 38	8.4 9.6	66 3	69 2.8	70 3.3	100 7.7	191 1	66 6.6	5.8 11	105 2.9
LASER	31 1.4	55 1.5	169 1.2	162 1.2	165 1.4	612 1.3	197 1	292 1.3	66 1	273 1.1

Některé poznámky k tabulce

- některé programy při zpracování některými kompilátory bylo nutno maličko upravit
- u kompilátorů MCODER 2 a COLT byl použit režim REM #1 (nastaven implicitně po nahrání)
- u COLTu bylo zrušeno zobrazení času příkazem *fx 0
- u BLASTu byl realizován překlad do p-kódu, program BM10 musel být zpracován Toolkitem
- machine code u BLASTu je asi o 15% rychlejší než p-kód
- MCODER 1 je trocu pomalejší než MCODER 2
- BLAST I2.0 je (většinou) o něco pomalejší než BLAST I3.7
- FP48K v1.7 a FP48K v1.1 jsou stejně rychlé
- na programu BM9 je patrné, že v případech, kdy se volá časově náročná ROM rutina (CIRCLE) sama v cyklu, nelze tento program urychlit žádným kompilátorem (PUMMA má novou rychlou rutinu kreslení kružnice); podobně je to s programem BM7 (výpočet funkcí), pouze TOBOS díky novým (méně přesným - jako v Pascalu) aritmetickým rutinám vykazuje výborné časy
- k programu BM10: vykresluje se pravidelný jedenáctiúhelník bez použití funkcí SIN a COS, ty jsou s ohledem na celočíselné kompilátory počítány jiným způsobem (SIN na ř. 200 až 250, COS na ř. 260 až 310), k propojení bodů úsečkou nebyl užit příkaz DRAW, nýbrž byl napsán algoritmus lineární interpolace přímo v Basicu (řádky 320 až 390), záměrně byl zaveden velký počet proměnných

Dále jsme jako zkušební program pro FP kompilátory užili 3D GRAFY od Ing. Martina štěpánka otištěné v nabídkovém čísle časopisu Elektronika. Z nabídky programu jsme ponechali funkci SIN (x*x+y*y)/(x*x+y*y), nastavené intervaly x a y, 31 řezů v obou směrech a nastavili režim potlačení neviditelných hran.

Do měření času bylo zařazeno:

- automatický odhad mezí osy z
- výpočet matice dané funkce bez dodatečné úpravy v ose z
- vykreslení grafu funkce s potlačenými neviditelnými hranami

Doby výpočtů včetně doby překladu jsou v následující tabulce.

	doba výpočtu	doba překladu	relativní rychlost
FP 48K v1.7	4min 51s	13s	9,5
BLAST I3.7	6min 35s	13min	6,3
TOBOS FP	2min 32s	10s	16,9
PUMMA	4min 59s	4min 29s	8,3
LASER	20min	8s	2,1
Interpret BASIC	41min 12s	-	1

Závěr

V tomto příspěvku jsme se bránili možnosti vyzdvihnout jeden kompilátor nad druhý, poněvadž každý z nich má své klady i zápory. Z popisu u každého kompilátoru vyplývají základní vlastnosti a způsob jeho použití. Je zřejmé, že novější kompilátory budou (a také jsou) dokonalejší než starší, proto se těšme na nové (budou-li nějaké).

Pisatelé tohoto příspěvku mají zájem o tyto objekty:
- návod k PUMMA kompileru
- kompilátor fy HiSoft (včetně návodu), o kterém byla zmínka v Amatérském radiu číslo 5/87 na str. 184 (nejde o COLT!)
- kompilátor M CODER III i s návodem
- BLAST verze I4.0 i s IBLASTem a návodem
- návod k ZIPu, dle našeho jsme nedokázali umístit cílový kód na námi zvolenou adresu
- případné jiné typy v příspěvku neuvedených kompilátorů i s návody

Pomůže nám někdo? Věříme, že ano. Děkujeme.

==========

V Praze a Kutné Hoře r 1987.

Ing. Miroslav Veverka
Jemenská 582
160 00 PRAHA 6

Antonín Mareš
Kaňk 152
284 04 Kutná Hora