1988/1-2

KLUB MIKROELEKTRONIKY

ATARI®

ZPRAVODAJ

OLOMOUC

Vážení přátelé ATARI.

Váš zájem o našeho olomouckého zpravodaje v uplynulém roce, podněcuje hrstku nadšenců pokračovat ve své záslužné činnosti a připravit pro Vás i v letošním roce další zpravodaje, které budeme vydávat ve vyšším nákladu a dvojnásobném rozsahu, tzn. celkem 10 čísel o 32 stranách, abychom uspokojili vzrůstající hlad po informacích týkajících se výpočetní techniky zaměřené na osmibitové počítače ATARI.

Odběr celého ročního zpravodaje je vázán příspěvkem na vydání, které letos vzhledem k většímu rozsahu činí 30,- Kčs. objednávky a distribuce bude opět organizována ZO svazarmu, na základě v současné době podepisovaných smluv o spolupráci. Redakce tímto zároveň oznamuje, že není v našich silách a možnostech reagovat na individuální objednávky jednotlivců.

Redakce s politováním konstatuje, že přestože je zpravodaj určen pro širokou obec ataristů, je z převážné části zajišťován příspěvky členů olomouckého klubu. Vyzýváme pro ještě větší zpestření zpravodaje i ostatní dopisovatele o příspěvky.

-rk.-

ATARI A BASIC

K programování většiny dnešních osobních mikropočítačů se používá programovací jazyk BASIC, nejčastěji ve formě interpretačního překladače nebo-li interpretu, zpravidla s tzv. programovou interpretací. BASIC, jehož označení představuje zkratku z anglického názvu Besinners All-purpose Symbolic Instruction Code, (symbolický instrukční kód pro všeobecné použití začátečníky) vznikl ve Spojených státech amerických v roce 1964. Dnes má v oblasti osobních počítačů zcela dominantní a počet různých verzí je odhadována na několik desítek. Přepis řešeného problému do programovacího jazyka BASIC je díky jeho jednoduché a logické konstrukci snadný a také přenositelnost programu v jazyce BASIC z jednoho osobního mikropočítače je poměrně dobrá. Z hlediska rychlosti průběhu příkazů je však interpret jazyka BASIC velmi pomalý. Uvádí se, že proti jazyku symbolických adres (assembleru) je BASIC asi 1000x pomalejší. Podle funkčních vlastností interpretu, které jsou zhruba úměrné jejich délce (velikosti zabrané paměti) se ustálilo rozdělení zhruba do čtyř kategorií:

- tiny (celočíselný), 2-4 kB
- standartní, 8kB
- rozšířený, 16 kB
- diskově orientovaný, 17 a více kB

Obecně platí, že čím dokonalejší interpret, měřeno výkonem, přesností a komfortem obsluhy, tím větší kapacitu paměti zabírá a tím pomalejší je při výpočtu.

Pro 8-bitové počítače ATARI dnes existuje několik verzí jazyka BASIC. Je těžké rozhodnout, který interpret je nejlepší - musíme vždy vycházet z potřeb uživatele. Zajímavé však může být srovnání.

Existují tři důležité vlastnosti, na které se zaměříme - kompatibilita, provozní sady (runtime packases) a kompilátory.

Kompatibilita je vnitřní podobnost mezi jazyky. Potom "vzestupně slučitelný s Atari BASICem" znamená, že interpret může vykonávat program napsaný v Atari BASICu bez jakékoliv změny.

Provozní sady zajišťují přenos programu mezi interprety. Jestliže máme program napsaný v Atari BASICu, nemůžete ho spus-

tit bez zabudovaného jazyka BASIC. Provozní sady vám zabezpečí běh programu, i když nevlastníte daný interpret.

Kompilátory zrychlují činnost programu. Před spuštěním je standartní program převeden do kódů, kterým počítač rozumí. U většiny verzí jazyka BASIC se tak děje po zadání příkazu RUN. To znamená, že první řádek programu je přeložen a potom vykonán, následující řádek je přeložen a vykonán atd. Kompilátory dokáží najednou přeložit celý program a nahrát ho v upravené verzi. Rychlost se zvýší 10-20x, protože konečný tvar překladu se blíží překladům z assembleru. Dále šetří pamět systému, protože nepotřebují během chodu v paměti interpret ani žádný jiný program takového typu (např. provozní sadu).

Nejdůležitější verze jazyka BASIC na osmibitových počítačích ATARI

ATARI BASIC
(Atari Corporation)

Důvody, proč používat tento nepříliš výkonný interpret BASICu, jsou dva. Za prvé, existuje velké množství všeobecně přístupných programů, které dokáží rozšířit chybějící vlastnosti, a za druhé, programy napsané v Atari BASICu počítáme na stovky, možná tisíce. V počítačích typu ATARI XE je zabudována verze C, v typech XL se vyskytuje většinou verze B, u starších modelů A. Vzhledem k určitým chybám verze B (o A nemluvě) doporučuji přejít před každým programováním na verzi C. Program byl uveřejněn v pražském Atari zpravodaji č. 3/87, str. 17-18. Provozní sada zde není nutná, protože každý vlastník počítače ATARI má tento BASIC zabudovaný (mimo typ 1200 XL, kde je nutná cartridse). K dispozici je několik kompilátorů - většinou určený pro práci s disketovou jednotkou.

MICROSOFT BASIC II (MBASIC)
(Atari Corporation)

MBASIC, který byl implementován z jazyku BASIC, je standartním jazykem na většině ostatních osobních počítačů. Jestliže potřebujete převést programy z počítačů typu Apple II nebo IBM PC na ATARI, doporučuji použít tuto verzi jazyka BASIC. MBASIC II je výkonná verze, ale chybí mu některé speciální vlastnosti počítačů ATARI. Např. k obsluze joysticku musíte použít PEEK, vyhledávání chyb je

možné pouze během chodu programu. (To znamená, jestliže uděláte v programu chybu, nemůžete ji najít, dokud program nespustíte příkazem RUN.). Provozní sada nebo kompilátor neexistují. Váš program je tak určený pouze pro uživatele, který také vlastní MBASIC. Několik výhod (oblast práce s řetězci a poli) je zcela vyváženo ztrátou kompatibility programu, zmenšením volného paměťového prostoru a poměrně složitou editací programu.

BASIC XL
(OSS)

Dokud se neobjevila nová verze, BASIC XL, tvorba složitějších programů zaměstnávala jen zkušené programátory. To vše již neplatí. Je hračkou používat BASIC XL, který je vzestupně slučitelný s Atari BASICem a pracuje 4-5x rychleji. Supercartridse firmy OSS dokáže "našlapat" 16 kB interpret do 8 kB paměťového prostoru. Použitím verze DOSu, DOS XL (OSS), můžeme rozšířit volnou paměť. BASIC XL usnadňuje práci s řetězci, má lepší a bohatější grafiku. Provozní sada je k dispozici v soupravě Programmer´s Toolkit (OSS) spolu s mnoha dalšími užitečnými programy a příklady. Kompilátor dosud nebyl vytvořen.

BASIC XE
(OSS)

BASIC XE má všechny vlastnosti charakteristické pro BASIC XL plus některé rozšířené nebo nové funkce a příkazy. Zvlášť významné jsou rychlé matematické rutiny, které nahrazují pomalou funkci pohyblivé řádové čárky. Na počítačích ATARI 130 XE dovoluje tento interpret použít přídavných 64 kB paměti pro ukládání dat, a tak uvolnit hlavní paměť pro uživatele. Programy zde mohou mít délku až 64 kB, pro dimenzaci je určeno dalších 32 kB paměti. BASIC XE vyžaduje počítač typu XL nebo XE nejméně s 64 kB RAM. Jako provozní sada slouží souprava Programmer´s Toolkit, která je však limitována možnostmi verze BASICu XL. Kompilátor zatím není k dispozici, měl by se však během roku objevit na americkém počítačovém trhu.

ADVAN BASIC
(Advan Language Design)

Každý Vám potvrdí, že napsat rychlou hru typu arcade je v Atari BASICu složitý a náročný úkol. Verze jazyku BASIC - ADVAN BASIC

vám však dokáže, že to je hračka. Velice jednoduše a bez námahy můžeme vytvářet PM grafiku, hudební kulisu programu, používat DL nebo VBI. Tato verze nejlépe využívá speciální vlastnosti počítače ATARI. Ale právě tak jako ATARI není jen "hrací mašina", ADVAN BASIC není jen jazyk pro vytváření her. Je to dokonalá implementace jazyku BASIC. Ke zvýšení luxusu programování slouží také zdokonalená editace a kontrola programu, rozšířené možnosti vstupu a výstupu nebo lepší práce s řetězci. K dispozici je i MMG kompilátor, který pracuje 10-15x rychleji než Atari BASIC. Ale zkompilované programy vyžadují ještě provozní sadu. To vše nyní řeší systém Advan Optimizer (Advan Language Systems), což je rychlý a vysoce výkonný kompilátor. Ve srovnání rychlosti dosahuje fantastických hodnot, protože je 6x rychlejší než ADVAN BASIC, 20x rychlejší než zkompilovaný Turbo BASIC XL a 120x rychlejší než standartní Atari BASIC. ADVAN BASIC však není kompatibilní s Atari BASICem a používá svůj vlastní DOS, který je ovšem slučitelný s typem DOS 2,0 nebo DOS 2,5 firmy ATARI.

TURBO BASIC XL

(Frank Ostrowski - Happy Computer)

Tento interpret můžeme považovat za přechod mezi Atari BASICem a BASICEM XL. TURBO BASIC XL je kompatibilní s Atari BASICem, ale pracuje 3-4x rychleji. Problémy a prací s TURBO BASICem XL se zabývají ATARI zpravodaje č. 2 (Olomouc) a č. 3 (Praha), v Brně byl vydán samostatný manuál. Pokud TURBO BASIC XL nevlastníte, je možné kopii programu získat ve všech větších ATARI klubech, a to na kazetu nebo na disk. Provozní sada není nutná, protože TURBO BASIC XL může pracovat s programem napsaným v jazyku Atari BASIC beze změny. Tento interpret můžete použít na typech ATARI XL/XE nejméně s 64 kB RAM.

Mimochodem - autor Frank Ostrowski z NSR je také autorem GFA BASICu, který byl vytvořen speciálně pro ATARI ST a stal se jedním z bestsellerů na americkém softwarovém trhu pro 16-bitové počítače.

K lepší orientaci vám pomůže také tabulka 1 přetištěná z časopisu ANTIC, kterou pro její mezinárodní srozumitelnost není třeba překládat.

Stručné zhodnocení.

Jestliže jste nováčci v programování jazyku BASIC nebo přecházíte z jiného typu počítače, doporučuji Atari BASIC revizi C a Turbo BASIC XL. Teprve po aktivním zvládnutí těchto jazyků, můžete podle zájmu přejít na další verze.

Pro zkušené programátory platí:

Když chcete modifikovat stávající BASICové programy pro vlastní potřebu, nejlepší je zakoupit verzi BASIC XL nebo BASIC XE.

Pro ty, kteří mají řádi animaci, PM grafiku, komponování hudby nebo komplikovaný design obrazovky, je nejlepší Advan BASIC.

Potřebujete větší kapacitu paměti pro manipulaci s daty ? Zkuste kombinaci ATARI 130 XE + BASIC XE. Nebudete zkladáni.

I když zvolíte libovolnou verzi, nebudete litovat. Všechny uvedené verze jazyku BASIC patří mezi nejlepší současné interprety pro 8-bitové počítače.

Pro posouzení daného interpretu z hlediska rychlosti lze použít testovací programy v anglosaské literatuře označené jako BENCHMARKS. Zpravidla jde o sadu jednoduchých programů, které vykonávají typické příkazy jazyka BASICu v mnohokrát opakovaném cyklu. Počet průchodů cyklem se stanoví s ohledem na požadovanou přesnost měření spotřebovaného času. Ten se měří jednoduše stopkami nebo u dokonalejších mikropočítačů pomocí vnitřních hodin a speciální funkce času (zpravidla označené TIME nebo TIME$). Vyvoláním funkce před vstupem do cyklu a po jeho ukončení a odečtením obou údajů získáme spotřebovaný čas.

Příklady testovacích programů, které používají anglické časopisy Practical Computing a Personal Computer World pro testování nových mikropočítačů, jsou následně uvedeny - Benchmarks Tests 1-8.

```
1ØØ REM Benchmark Test =1
11Ø ? "START"
12Ø FOR I=1 TO 1ØØØ
13Ø NEXT I
14Ø ? "END"
15Ø END
```

```
1ØØ REM Benchmark Test =2
11Ø ? "START"
12Ø K=Ø
13Ø K=K+1
14Ø IF K<1ØØØ THEN 13Ø
15Ø ? "END"
16Ø END
```

```
1ØØ REM Benchmark Test =3
11Ø ? "START"
```

```
1ØØ REM Benchmark Test =4
11Ø ? "START"
```

```
12Ø K=Ø
13Ø K=K+1
14Ø K=K/K*K*K-K
15Ø IF K<1ØØØ THEN 13Ø
16Ø ? "END"
17Ø END
```

```
1ØØ REM Benchmark Test =5
11Ø ? "START"
12Ø K=Ø
13Ø K=K+1
14Ø A=K/2*3*4-5
15Ø GOSUB 5ØØ
16Ø IF K<1ØØØ THEN 13Ø
17Ø ? "END"
18Ø END
5ØØ RETURN
```

```
1ØØ REM Benchmark Test =7
11Ø ? "START"
12Ø K=Ø
13Ø DIM M(5)
14Ø K=K+1
15Ø A=K/2+3+4-5
16Ø GOSUB 5ØØ
17Ø FOR I=1 TO 5
18Ø M(I)=A
19Ø NEXT I
2ØØ IF K<1ØØØ THEN 14Ø
21Ø ? "ENT"
22Ø END
5ØØ RETURN
```

```
12Ø K=Ø
13Ø K=K+1
14Ø A=K/2*3*4-5
15Ø IF K<1ØØØ THEN 13Ø
16Ø ? "END"
17Ø END
```

```
1ØØ REM Benchmark Test =6
11Ø ? "START"
12Ø K=Ø
13Ø DIM M(5)
14Ø K=K+1
15Ø A=K/2*3*4-5
16Ø GOSUB 5ØØ
17Ø FOR I=1 TO 5
18Ø NEXT I
19Ø IF K<1ØØØ THEN 14Ø
2ØØ ? "END"
21Ø END
5ØØ RETURN
```

```
1ØØ REM Benchmark Test =8
11Ø ? "START"
12Ø K=Ø
13Ø K=K+1
14Ø A=K\2
15Ø B=LOG(K)
16Ø C=SIN(K)
17Ø IF K<1ØØØ THEN 13Ø
18Ø ? "END"
19Ø END
```

Vždy se měří čas v sekundách mezi zobrazením "START" a "END" a pro zvýšení reprodukovatelnosti naměřených údajů se pro každý program měří alespoň třikrát. Nakonec se ze všech osmi průměrných

dob chodu jednotlivých programů vypočte výsledná průměrná hodnota.

Uvedené testy mají své výhody i nevýhody. K výhodám patří jejich jednoduchost, rychlost a dostupnost - interpret jazyka BASIC je dnes prakticky v každém osobním počítači. Příkazy v testovacích programech jsou triviální a tedy společné pro většinu dialektů jazyka BASIC. V neposlední řadě je výhodné, že provedení testů nevyžaduje od uživatele žádné zvláštní znalosti. V praxi však výsledky získané chodem testovacích programů mnoho neznamenají. Protože testují komplexně jak rychlost technických prostředků, tak i rychlost vlastního interpretu a konečně i reakční čas osoby, která ovládá stopky, je jediným spolehlivým závěrem srovnání, že jeden mikropočítačový systém zvládne určitý konkrétní úkol rychleji než jiný systém. Hlavní nevýhodou standartních testovacích programů je skutečnost, že testují jen omezený počet funkcí jazyka BASIC a nezkoumají mikropočítačový systém jako celek. Pro programátora, který řeší grafické úlohy nebo úlohy na zpracování textu, jsou doby běhu uvedených testovacích programů zcela nepodstatné. Standartní testovací program také neřekne nic o nejkritičtějším místě každého výpočetního systému, kterým je vstup a výstup. Proto je potřeba výsledky testů posuzovat velmi obezřetně.

Na závěr je na místě upozornění - největší rezervu ve výkonu jakéhokoliv výpočetního systému představuje zpravidla efektivnější algoritmus řešení a konečně i uživatel, který je bezesporu nejpomalejší částí celého mikropočítačového systému.

Jiří Hrdlička, GIA Software, Olomouc 1987

Literatura:

1) Srovnání rychlosti některých osobních počítačů
Sdělovací technika 1/1985, str. 25-26
2) Charles Cherry: BASIC Bonanza
ANTIC, June 1987, str. 26-29

Při sestavování tebulky Benchmarks testů spolupracovali
Ing. D. Pavlík (ATARI 800 XL)
Ing. M. Edl (ATARI 520 ST)

Tabulka 1

BASIC FEATURES COMPARISON CHART

Language	Atari BASIC	Microsoft BASIC II	BASIC XL	BASIC XE	Advan BASIC	Turbo BASIC XL
Minimum System	ALL	ALL/48K	ALL	XL/XE/64K	ALL/48K	XL/XE/64K
Upward Computable with Atari BASIC	N/A	NO	YES	YES	NO	YES
Runtime Package	N/A+	NO	YES	YES+	YES	N/A+
Compiler	YES	NO	NO	NO	YES	YES
Editing/Debugging						
DELETE Lines	NO	YES	YES	YES	YES	YES
Auto Line Numbering	NO	YES	YES	YES	NO	NO
Renumber	NO	YES	YES	YES	NO	YES
Trace	NO	YES	YES	YES	NO	YES
Program Control						
IF/THEN/ELSE	NO	YES	YES	YES	YES	YES
WHILE/WEND	NO	NO	YES	YES	YES	YES
REPEAT/UNTIL	NO	NO	NO	NO	YES	YES
CASE	NO	NO	NO	NO	YES	NO
PAUSE/WAIT	NO	YES	NO	NO	YES	YES
Named Subroutines/ Procedures/Commands	NO	YES	YES	YES	YES	YES
Input/Output						
Directory	NO	NO	YES	YES	YES	YES
DELETE "D:Filemane"	NO	YES	YES	YES	YES	YES
LOCK/UNLOCK	NO	YES	YES	YES	YES	YES
Binary LOAD/SAVE	NO	NO	YES	YES	YES	YES
INPUT With Prompt	NO	YES	YES	YES	YES	YES
PRINT USING	NO	YES	YES	YES	EXCELLENT	NO

Strings						
Max String Lenght	Memory	Bytes:12Ø	Memory	Memory	Bytes:256	Memory
Auto Dimensioning	NO	YES	YES	YES	YES	NO
Strings Arrays	NO	YES	YES	YES	YES	NO
Strings Matrices	NO	NO	NO	NO	YES	NO
Find Substring	NO	YES	YES	YES	YES	EXCELLENT
LEFT$/MID$/RIGHT$	NO	YES	YES	YES	YES	NO
Memory Functions						
Double PEEK/POKE	NO	NO	YES	YES	YES	YES
Block MOVE	NO	YES	YES	YES	NO	YES
Set Block To A Value	NO	NO	NO	NO	NO	YES
13ØXE Expanded Mem.	NO	NO	NO	YES+	NO	NO
Graphics/Sound						
Extended Graphics	NO	NO	NO	NO	GOOD	EXCELLENT
PM Graphics	NO	NO	GOOD	GOOD	EXCELLENT	NO
VBI	NO	NO	NO	NO	YES	NO
Extended Sound	NO	GOOD	NO	NO	EXCELLENT	GOOD
Numbers						
Higs Speed Math	NO	NO	NO	YES	NO	NO
Integer Math	NO	YES	NO	NO	YES	NO
Hexadecimal Numbers	NO	YES	YES	YES	YES	YES
Binary Numbers	NO	NO	NO	NO	YES	NO
Boolean Operators	NO	AND/OR/ XOR/NOT	AND/OR/ XOR	AND/OR/	AND/OR/ XOR	AND/OR/ XOR
! Price (June 1987) !	$15	$29.95	$59	$79	$39.95	free

+ ... see text

Manufacturers (U.S.A.)

1) ATARI BASIC MICROSOFT BASIC II
Atari Corp., 1196 Borregas Avenue, Sunnyvale, CA 94Ø88

2) BASIC XL
BASIC XE
Optimized System Software (OSS), 1221B Kentwood Avenue, San Jose, CA 95129

3) ADVAN BASIC
Advan Language Designs, P.O. Box 159, Baldwin, KS 66ØØ6

1987 by GIA Software

Tabulka 2

BENCHMARKS TESTY pro nejznámější tuzemské a zahraniční počítače

ČESKOSLOVENSKO

Počítač	Doba chodu programu podle výpisu v textu								Suma hodnot	Průměr
	1.	2.	3.	4.	5.	6.	7.	8.		
TNS	2.Ø	8.Ø	19.Ø	19.1	21.6	34.2	52.4	82.4	238.7	29.84
IQ-151	1.9	11.2	23.5	25.Ø	27.Ø	39.8	56.4	121.4	3Ø6.2	38.28
PMD-85	3.Ø	13.1	26.8	28.4	3Ø.5	46.6	66.2	12Ø.5	335.1	41.89
PP-Ø1	2.3	8.2	27.4	26.8	29.4	42.6	55.3	249.2	441.2	55.15
SAPI 1	1.1	21.1	36.8	4Ø.9	49.3	62.8	115.3	-	-	-

Všechny tyto počítače pracují na šest desetinných míst s vyjímkou SAPI 1, který používá čísel typu integer (nelze proto zpracovat BENCHMARKS Test 8).

ZAHRANIČÍ

	1.	2.	3.	4.	5.	6.	7.	8.	Suma hodnot	Průměr
AMSTRAD										
464	1.1	3.3	9.2	9.6	1Ø.2	19.Ø	3Ø.2	34.2	116.8	14.6Ø
BBC Accorn	1.Ø	3.1	8.3	8.7	9.2	13.9	21.9	52.Ø	118,1	14.76
COMMODORE										
64	1.9	1Ø.6	19.6	21.4	23.1	34.1	53.5	117.2	281.4	35.19
HP-85A	1.9	3.8	16.4	16.6	17.8	29.6	44.9	13Ø.4	261.4	32.69
IBM PC	1.2	4.8	11.7	12.2	13.4	23.3	37.4	3Ø.Ø	134.Ø	16.75
IBM PC/AT	Ø.5	1.9	4.6	4.7	5.2	9.1	14.6	14.Ø	54.6	6.83
SINCLAIR										
Spectrum	4.5	8.4	2Ø.3	19.5	23.3	53.1	77.5	239.3	445.9	55.74
OL	1.9	5.4	9.3	9.1	11.8	24.Ø	42.4	2Ø.7	124.6	15.58

TI 99/4A	3.6	9.7	24.8	25.3	27.Ø	62.4	85.8	385.9	624.5	78.Ø6

Většina typů pracuje s devíti desetinnými čísly, pouze TI 99/4A používá deset a HP-85A dvanáct desetinných míst.

ATARI 8ØØ XL

Typ BASICu	Doba chodu programu podle výpisu v textu								Suma hodnot	Průměr
	1.	2.	3.	4.	5.	6.	7.	8.		
Atari B.	3.Ø	7.Ø	2Ø.0	23.Ø	26.Ø	4Ø.5	6Ø.Ø	422.Ø	6Ø1.5	75.19
Atari B.+	2.Ø	6.Ø	15.Ø	17.Ø	2Ø.Ø	3Ø.Ø	45.Ø	310.0	445.0	55.63
MBasic	2.0	9.0	17.0	19.0	21.0	33.0	51.0	80.0	232.0	29.00
TBasic XL	1.0	3.0	8.0	9.0	10.0	15.0	25.0	56.0	127.0	15.88

+ ... vyřazen z činnosti ANTIC příkazem POKE 559,0

ATARI 136 XE

	1.	2.	3.	4.	5.	6.	7.	8.	Suma hodnot	Průměr
Atari B.	2.3	7.7	21.1	24.6	28.3	43.0	65.3	453.5	645.8	80,73
Atari B.+	1.8	5.4	14.2	16.9	19.5	30.5	45.7	309.6	442.6	55.33
MBasic	1.7	9.4	17.6	20.6	21.9	34.9	54.5	84.4	245.0	30.63
TBasic XL	1.1	3.3	8.3	9.2	10.2	15.8	26.4	59.3	133.6	16.70
Basic XE	1.6	3.1	8.8	9.3	11.4	21.6	30.6	60.4	146.8	18.35
Basic XE+	1.2	2.4	8.2	8.6	9.5	16.9	24.4	59.2	130.4	16.30

+ ... použit příkaz FAST

ATARI 520 ST

	1.	2.	3.	4.	5.	6.	7.	8.	Suma hodnot	Průměr
Basic ST	2.2	4.4	7.4	8.0	8.9	13.6	20.8	9.5	74.8	9.35
Basic GFA	0.1	0.3	0.9	1.0	1.1	1.7	2.9	2.9	10.9	1.36

Poznámka: Atari BASIC pracuje na devět desetinných míst. Microsoft BASIC pracuje na šest desetinných míst. BASIC XE a Turbo XL pracují na deset desetinných míst.

LAMOTIER. J. F.
BASIC EXERCISES FOR THE ATARI

Kapitola Třetí - Použití celých čísel.

3.1. CELOČÍSELNÉ ŘEŠENÍ VZTAHU : $A^2 + B^2 = C^2$

ÚKOL: Určete všechna čísla A a B mezi 1 a 100, pro než platí, že $A^2 + B^2$ je přesně rovno druhé mocnině.

Při řešení takovéto úlohy, musíme provést tyto kroky postupu:

1) analýzovat problém
2) vybrat metodu řešení a nakreslit vývojový diagram
3) napsat odpovídající BASIC program

ad1) ANALÝZA:

Nejdříve se dohodneme, že řešení, která se budou lišit od sebe pouze permutací, budeme uvažovat za identické.

Například: Řešení A=3, B=4, C=5 a řešení A=4, B=3, C=5 považujeme za identická řešení.

Abychom se vyhli opakování identických řešení, budeme zkoumat jen taková řešení, při kterých B>A. Budeme tedy hledat taková I^2+J^2 přesně rovné druhé mocnině tak, že I bude z intervalu 1 ÷ 99 a proměnná J bude z intervalu I+1 ÷ 100. Tento úkol je možno řešit dvěma způsoby.

PRVNÍ ZPUSOB : Přírustek K k proměnné začne od J+1. Potom je-li:

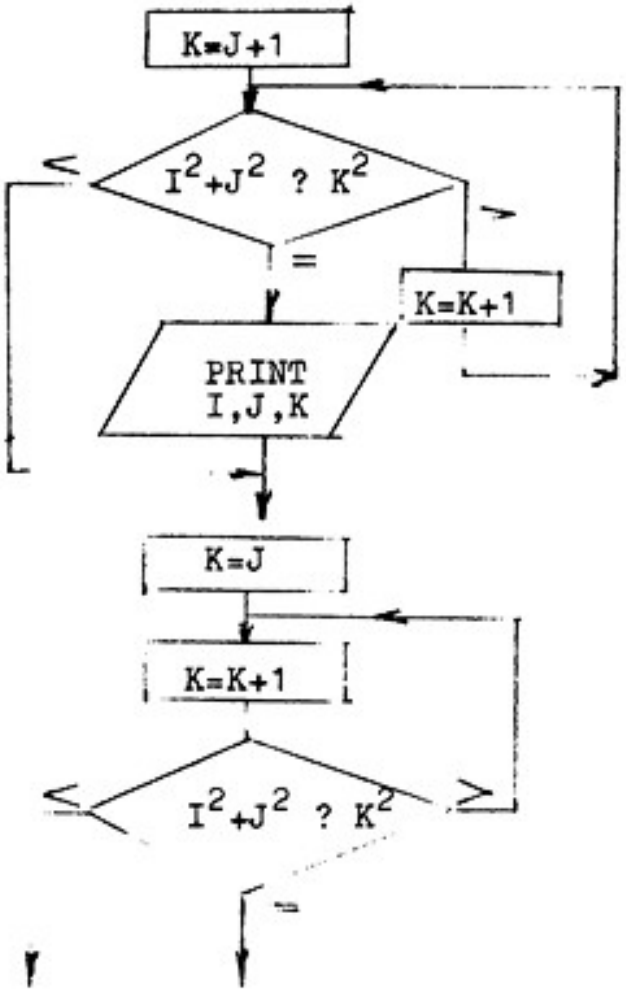
$I^2+J^2=K^2$ - dostali jsme řešení
$I^2+J^2> K^2$ - zvěť K o jedničku a opakuj
$I^2+J^2< K^2$ - není řešení pro I a J

DRUHÝ ZPUSOB : Vypočteme $K = I^2+J^2$

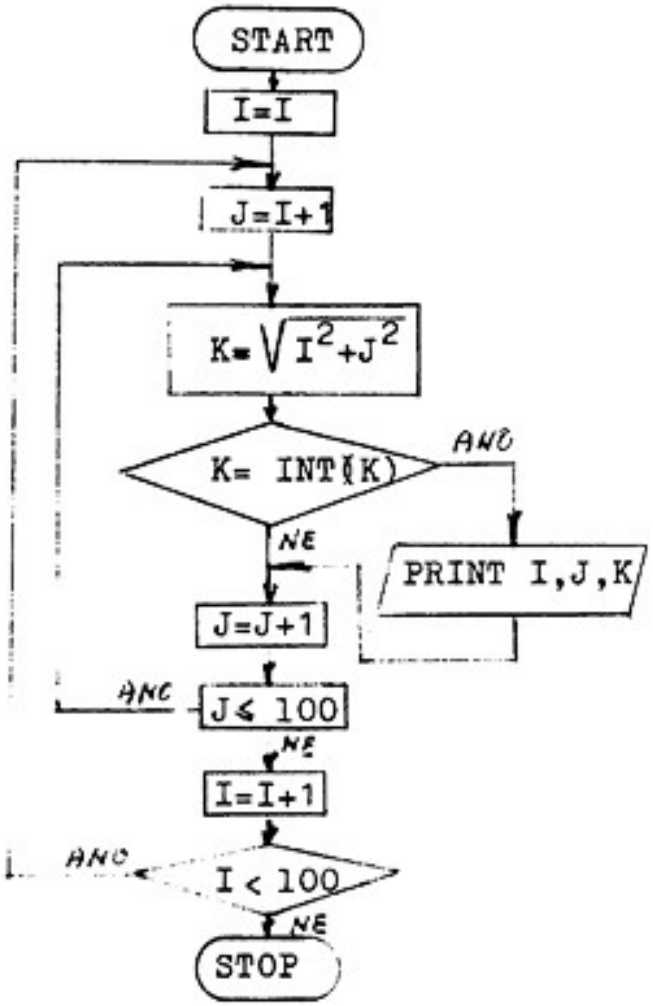
Je-li K celé číslo, dostali jsme řešení, pokud K není celé číslo, potom K není řešení pro uvedené I a J. Určením zda K je či není celé číslo, provedeme jednoduše tak, že porovnáme K a INT(K).

Obadva způsoby řešení jsou zobrazené formou VD (vývojových diagramů) na obr. 3.1. a 3.2. V obou VD se vnější cyklus mění od 1 do 99 a vnitřní cyklus od I+1 do 100. Na základě těchto VD jsou sestaveny programy v Basicu viz obr. 3.3. a 3.4.

Obrázek 3.1. VD pro první způsob



Obr.3.2. VD pro druhý způsob



Obr. 3.3.

```
1    REM První způsob
5 N=1ØØ
1Ø   FOR I=1 TO N
2Ø   FOR J=I+1 TO N
3Ø   S=I*I+J*J
4Ø   K=J
5Ø   K=K+1
6Ø   K2=K*K
7Ø   IF K2< S THEN 5Ø
8Ø   IF K2> S THEN 1ØØ
9Ø   PRINT " ";I;" ";J;)
95   PRINT " ";K
1ØØ NEXT J
11Ø NEXT I
12Ø END
```

Obr.3.4.

```
1    REM Druhý způsob
5    N=1ØØ
1Ø   FOR I=1 TO N
2Ø   FOR J=I+1 TO N
3Ø   K=SQR(I*I+J*J)
4Ø   IF K<>INT(K) THEN 6Ø
5Ø   PRINT " ";I;" ";J;
55   PRINT " ";INT(K)
6Ø   NEXT J
7Ø   NEXT I
8Ø   END
```

Podíváme se podrobněji na program listinku obr. 3.4. Zde se porovnává hodnota K=SQR(I*I+J*J) s hodnotou K=INT(K) Počítač porovná obě hodnoty na 7 desetiných míst, Například výsledkem operace SQR můžou být hodnoty 51,000 000 1 anebo 50,999 999 9. Funkce INT nuluje všechna desetinná místa potom

tedy 51,0000000 anebo 51,0000000. Když počítač bude porovnávat takovéto dvojice hodnot, které se ve skutečnosti rovnají, ale v procesu výpočtu získali nepatrnou odchylku, je proto vhodnější míru této odchylky zahrnout do výpočtu. Potom řádek 40 v programu 3.4. bude mít tvar:

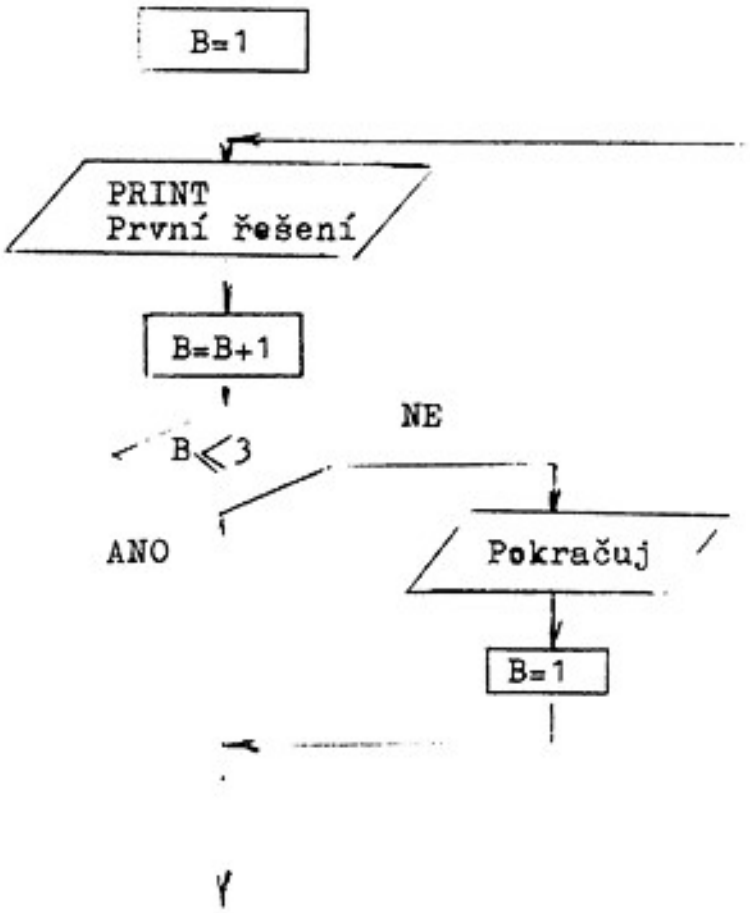
```
40 IF ABS(K-INT(K+5))> 1E-7 THEN 6Ø
```

Program 3.3.,který je sestaven pro první způsob výpočtu, podobné problémy nemá.
Nakonec se zmíníme o formě výstupu vypočítaných dat.Na obr.3.5. jsou data napsaná v jednom sloupci v pořadí A B C v jednom řádku. Tento způsob zobrazení dat není zvlášť výhodný. Na obr.3.8. máme data zobrazená ve třech sloupcích oddělených písmenem I. Způsob jak upravit program, aby vytiskl výsledky podle obr.3.8. je zobrazený ve VD na obr.3.6. a program je zapsán na obr.3.7. Je zřejmé, že byla použita proměnná B, pomocí ní jsme napsali hodnoty do tří sloupců. Protože ATARI BASIC nemá funkci TAB() pro polohování kurzoru, je vertikální uspořádání výstupu komplikované vložením čísel (měnících se od jedné do třech) do řetězce s pevnou délkou.

Obr. 3.5.

3	4	5
5	12	13
6	8	10
7	24	25
8	15	17
9	12	15
9	40	41
10	24	26
11	60	61
12	16	20
12	35	37
13	84	85
14	48	50
15	20	25
15	36	39
16	30	34
16	63	65
18	24	30
18	80	82
20	21	29
20	48	52
20	99	101
21	28	35
21	72	75
24	32	40
24	45	51

Obr. 3.6.



pokračování obr.3.5.

24	70	74
25	60	65
27	36	45
28	45	53
28	96	100
30	40	50
32	60	68
33	44	55
33	56	65
35	84	91
36	48	60
36	77	85
39	52	65
39	80	89
40	42	58
40	75	85
40	96	104
42	56	70
45	60	75
48	55	73
48	64	80
48	90	102
51	68	85
54	72	90
56	90	106
57	76	95
60	63	87
60	80	100
60	91	109
63	84	105
65	72	97
66	88	110
69	92	115
72	96	120
75	100	125
80	84	116

Obr. 3.7.

```
10 REM Výpis v tabulkové formě
90  DIM V$(10)
100 N=100
105 B=1
110 FOR I=1 TO N
120 FOR J=I+1 TO N
130 S=I*I+J*J
140 K=J
150 K=K+1
160 K2=K*K
170 IF K2<S THEN 150
180 IF K2>S THEN 200
191 V=I:W=2:GOSUB 1000
192 V=J:W=4:GOSUB 1000
193 V=K:W=4:GOSUB 1000
195 IF B<2 THEN PRINT "I";
196 B=B+1
197 IF B<3 THEN 200
198 PRINT
199 B=1
200 NEXT J
210 NEXT I
400 END
1000 REM Podprogram na nastavení
1010 REM čísla V k pravému okraji
1020 REM v poli W a na jeho
1030 REM vytištění
1035 V$="                    "
1040 V$(W-LEN(STR$(V))+1,W)=STR$(V)
1045 PRINT V$(1,W);
1050 RETURN
```

3.2. AMSTRONGOVO ČÍSLO

Číslo, které se rovná součtu třetích mocnin svojich číslic se nazývá Amstrongovo číslo.

Například: 153 je Amstrongovo číslo, protože
$153 = 1^3+5^3+3^3$

ÚKOL : Napište program, který určí všechna Amstrongová čísla mezi 1 a 2000.

ANALÝZA : Je nutné, abychom určili, zda je dané číslo anebo není Amstrongovým číslem. Musíme každou jeho číslici umocnit na třetí a sečíst. Pro získání jednotkového čísla z čísla musíme vypočítat zbytek čísla po jeho vydělení deseti.

Obr.:3.8.

3	4	5	I	5	12	13	I	6	8	10
7	24	25	I	10	24	26	I	9	12	15
9	40	41	I	8	15	17	I	11	60	61
12	16	20	I	12	35	37	I	13	84	85
14	48	50	I	15	20	25	I	15	36	39
16	30	34	I	16	63	65	I	18	24	30
18	80	82	I	20	21	29	I	20	48	52
20	99	101	I	21	72	75	I	21	28	35
24	32	40	I	24	45	51	I	24	70	74
25	60	65	I	27	36	45	I	28	45	53
28	96	100	I	30	40	50	I	30	72	78
32	60	68	I	33	44	55	I	33	56	65
35	84	91	I	36	48	60	I	36	77	85
39	52	65	I	39	80	89	I	40	42	58
40	75	85	I	40	96	104	I	42	56	70
45	60	75	I	48	55	73	I	48	64	80
48	90	102	I	51	68	85	I	54	72	90
56	90	106	I	57	76	95	I	60	63	87
60	80	100	I	60	91	109	I	63	84	105
65	72	97	I	66	88	110	I	69	92	115
72	96	120	I	75	100	125	I	80	84	116

Například: Je-li I číslo, potom

Q=INT(I/10) I=123 Q=INT(123/10)= 12
R=I-10*Q R=123-10*R=123-120=3

Kde R je hledané jednotkové číslo

Abychom dostali desítkovou číslici, opakujeme výpočet pomocí Q:

Q1=INT(Q/10) Q1=INT(12/10)=1
R=Q-10*Q1 R=12-10*1=12-10=2

Kde R je nyní desítková číslice daného čísla

Tento proces opakujeme, pokud nedosáhneme,že R=0. Pokud počítáme číslo do 2000, nedostaneme nikdy víc jako čtyři platné číslice.

Vhodnější než počítat Q1,Q2,Q3, atd. je tento způsob:

1) Zadáme K=Ø a S=Ø

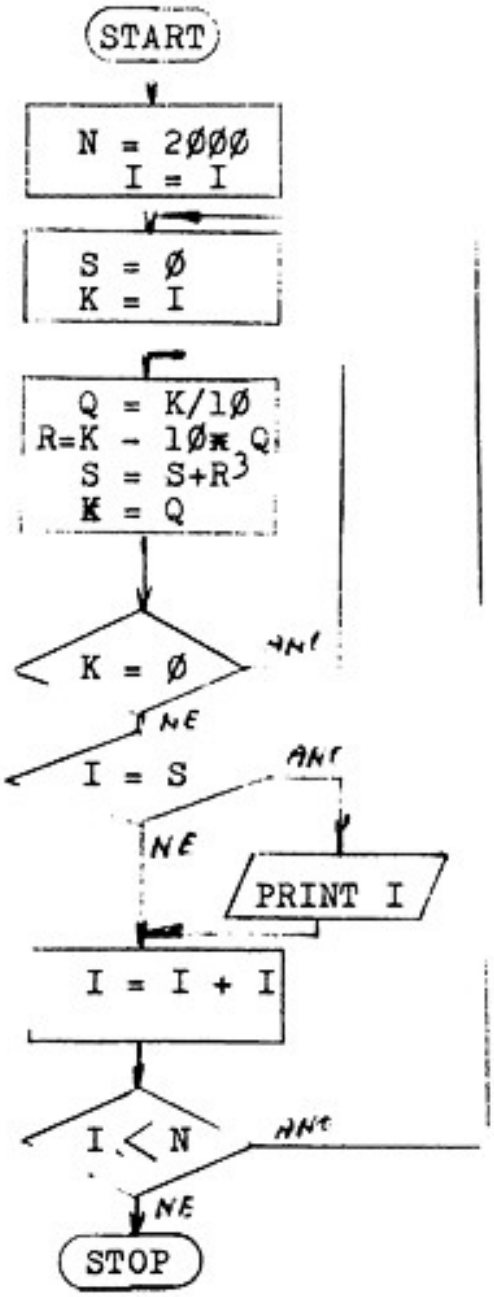
2) Vypočítáme Q=INT(K/10)
R=K-10*Q
S=S+R^3
K=Ø pro další pokračování

Když K > 0 vrátíme se na krok 2, jinak pokračujeme krokem 3.

3) Vypíšeme výsledek když: S=I

Výpis programu je na obr. 3.10. a odpovídá VD na obr. 3.9. Příklad řešení uvedený v tabulce na obr. 3.11. ukazuje, že Amstrongových čísel není mnoho.

Obr. 3.9.



Obr. 3.10

```
10  N=2000
20  PRINT "AMSTRONGOVO ČÍSLO"
25  PRINT "v rozmezí 1 - 2000 "
30  PRINT
40  FOR I=1 TO N
50  S=Ø
60  K=1
70  Q=INT(K/10)
80 R=K-1Ø*Q
9Ø  S=S+R*R*R
100 K=S
110 IF K<> Ø THEN 7Ø
120 IF I<>S THEN 13Ø
130 NEXT I
140 END
```

Obr.: 3.11.

```
1
153
370
371
407
```

K vývojovému diagramu:
R je jedna z číslic čísla I. Přidáme třetí mocninu k S
(S je součet třetích mocnin předcházejích číslic)
Když K=Ø, zpracovali jsme všechny číslice z testovaného čísla.

3.3. ROZDĚLENÍ ZLOMKU NA EGYPTSKÉ ZLOMKY

Zlomky, jejichž čitatel je roven jedničce, nazýváme Egyptské zlomky. (1/3, 1/10 ....)
Zlomky u kterých čitatel je menší jak jmenovatel, se nazývají pravé zlomky.

ÚKOL : Rozdělte pravé zlomky na součet Egyptských zlomků.

ANALÝZA : Na řešení této úlohy použijeme Fibouacciho maximální algoritmus. Předpokládáme, že je potřeba rozložit zlomek A/B. Na určení prvního zlomku rozkladu, použijeme největší Egyptský zlomek, který má hodnotu menší jako A/B. Odpočítáme tento zlomek od A/B a opakujeme pokud zbytek není roven Ø.

NAPŘÍKLAD: A=2, B=3, A/B = 2/3
Největší Egyptský zlomek příkladu je 1/2
A/B - 1/2 = 2/3 -1/2 = 1/6
Tím dostaneme požadované rozdělení 2/3=1/2+1/6

Dělení zlomlů není vždy tak jednoduché.Například pro zlomek 8/11 dostaneme:

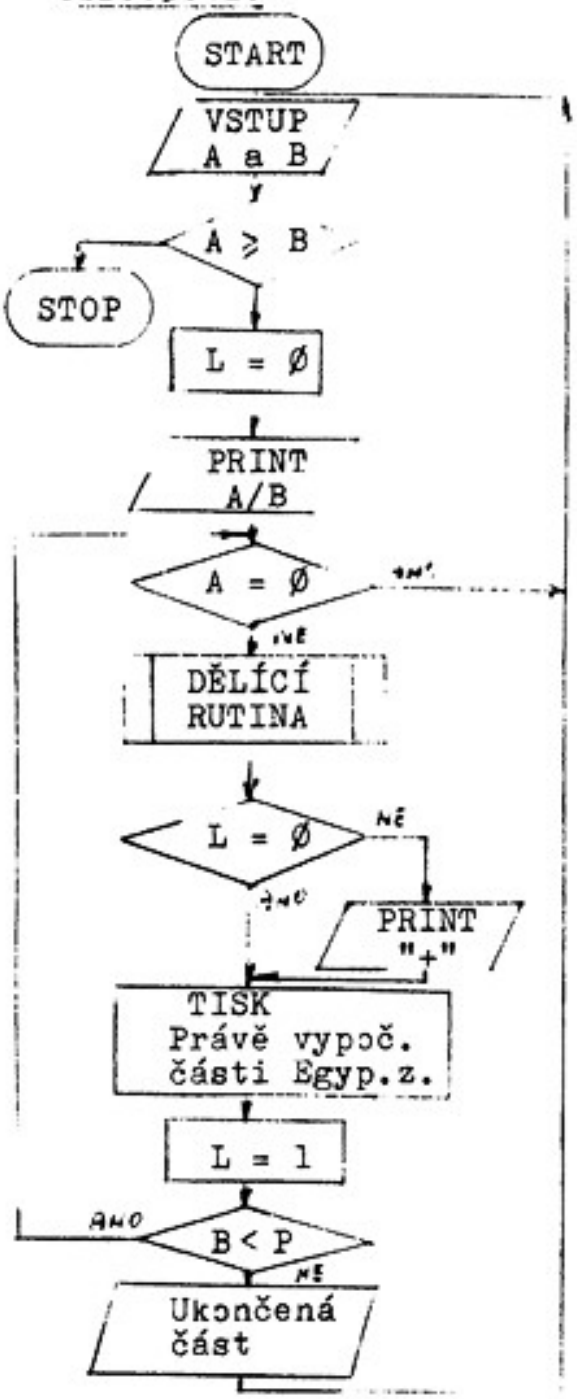
8/11 = 1/2 + 1/5 1 1/55 1 1/110

ÚKOL : Sestavte program pro oddělení zlomků na Egyptské zlomky pomocí Fibouacciho algoritmu. Věnujte pozornost formátování výstupu. Požadujte a prozkoumejte omezení programu a určete ochranu pře ním.

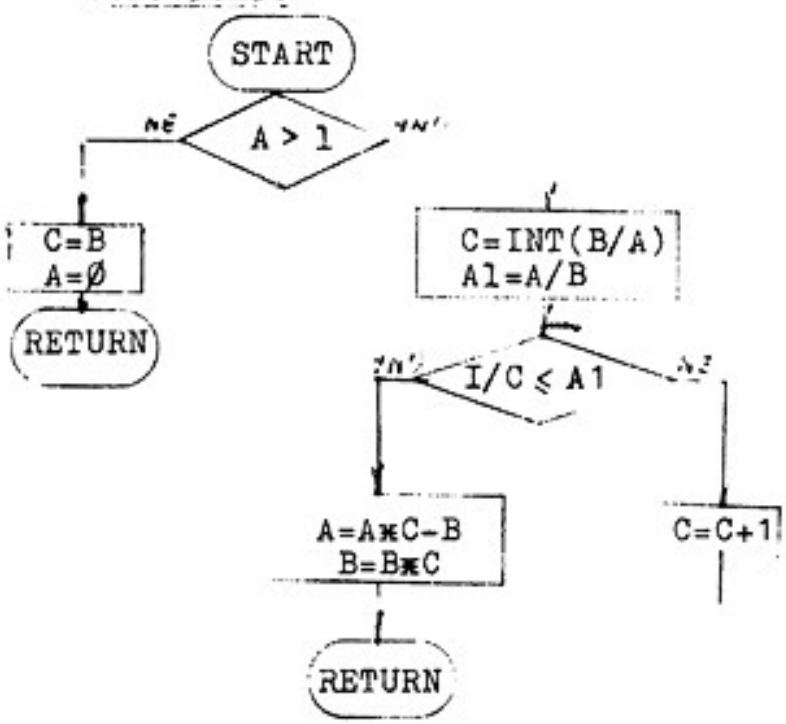
ŘEŠENÍ : Na první pohled se zdá úkol velmi jednoduchý. Musíme
- určit největší Egyptský zlomek menší jako A/B
- vypočítat zbytkový zlomek

Začneme výpočtem C=INT(B/A) potom 1/C=A/B. V tomto výpočtu bude C velmi blízko hledaného jmenovatele. Nyní položíme C = C+1 potom 1/C ≥ A/B a dostaneme hledaný zlomek. Zbytkový zlomek je potom daný A/B-1/C = A*C-B/B*C.
Na základě této anylýzy, můžeme namalovat vývojový diagram. Tím jsme připraveni ukončit jedno zajímavé pozorování. Tento výpočet může vyvolat vznik určitých celočíselných proměnných které, když jsou větší, způsobují vypsání výsledků bez omezení rozsahu číslic v čísle. Počítače mají omezený rozsah velikosti počtu platných číslic, které může celé číslo dosáhnout. Proto musíme použít test, aby počítač při přesáhnutí této limity nehlásil chybu. Všeobecně by mohly být tyto testy oddělené po každém součinu A*B, anebo B*C, ale když B > A, potřebujeme testovat pouze druhý součin. I toto je uvedeno ve vyývojovém diagramu na obr. 3.12. a 3.13. Navrhovaný algoritmus se ukončí úspěšně, když nové A je nulové a neúspěšně, když nové B překročí limitu přesnosti počítače.

Obr.:3.12.



Obr.:3.13.



Obr.:3.15 Výstup Egyptských zl.

```
Čitatel,jmenovatel ?2,3
Zlomek 2/3 = 1/2 + 1/6

ČITATEL,JMENOVATEL ?3,7
ZLOMEK 3/7 = 1/3+1/11+1/231

ČITATEL,JMENOVATEL ?7,13
ZLOMEK 7/13 = 1/2 + 1/26
```

Další čísla je nutno zadat v programu počítači

Program na obr.3.14 si rozdělíme na dvě části.

1. Hlavní program, který vykonává vstupní a výstupní a některé jednoduché testy.
2. Program, který hledá největší možný Egyptský zlomek a počítá zbytkový zlomek pro další interacii.

Rozdělením programu na dvě části se zvyšuje počet programovatelných příkazů. Na druhé straně je program přehlednější při použití.

Vývojový diagram na obr. 3.12 obsahuje proměnnou L, která nabývá hodnotu Ø nebo 1. Přidělením hodnoty Ø proměnné L na začátku

programu se vyhneme kladnému znaménku + před každým zlomkem, který určíme, Když má proměnná hodnotu L = 1, výstup každého dalšího zlomku má před sebou znaménko plus.

Obr.3.14.:

```
100 Print "ROZDĚLENÍ NA"
101 PRINT "EGYPTSKÉ ZLOMKY"
105 READ P
110 PRINT
120 PRINT
130 PRINT "ČITATEL, ";
135 PRINT "jmenovatel ";
140 INPUT A,B
150 IF A ≥ B THEN 8ØØ
160 L=Ø
170 PRINT
180 PRINT "ZLOMEK";A;"/";B;"=";
190 IF A=Ø THEN11Ø
200 GOSUB 5ØØ
210 IF L=Ø THEN 23Ø
220 PRINT "+";
230 PRINT "1/";C ;
235 L=1
240 IF B < P THEN 19Ø
300 PRINT
303 PRINT "Další jmenovatel"
305 PRINT "je pro výpočet velký"
310 GOTO 11Ø
500 IF A > 1 THEN 6ØØ
510 C=B
520 A=Ø
530 RETURN
600 C=INT(B/A)
605 A1=A/B
610 IF 1/C ≤ A1 THEN 64Ø
620 C=C+1
630 GOTO 61Ø
640 A=A*C-B
650 B=B*C-B
670 RETURN
700 DATA 999999999
800 END
```

Poznámky k programu:

Přesnost s kterou může být určené číslo ustanovené pro každý počítač. Maximálně velké číslo použitelné v počítači je konstantní. Program zkoumá parametr, který je možno použít na ochranu celistvosti výsledku. Změnou vloženého parametru může program vykonávaný v počítači mít rúzné kapacitní omezení. Dále jsou uvedeny dvě možnosti určení tohoto parametru.:

1) Hlášení největšího možného celého čísla v opakovaném použití přiřazené, anebo READ/DATA instrukcí.
2) Vyžádat si čas na vyhledávání největšího možného celého čísla (tato možnost je méně praktická)

Program končí vložením dvou čísel takových, že A ≥ B. Viz obr.3.15., kde je to uvedené.:

Obr.: 3.15.

```
ČITATEL, JMENOVATEL ? 2,3
ZLOMEK 2/3 = 1/2 + 1/6

ČITATEL, JMENOVATEL ?3,7
ZLOMEK 3/7 = 1/3 + 1/11 + 1/231

ČITATEL, JMENOVATEL ?7,13
Zlomek 7/13 =1/2 + 1/26
```

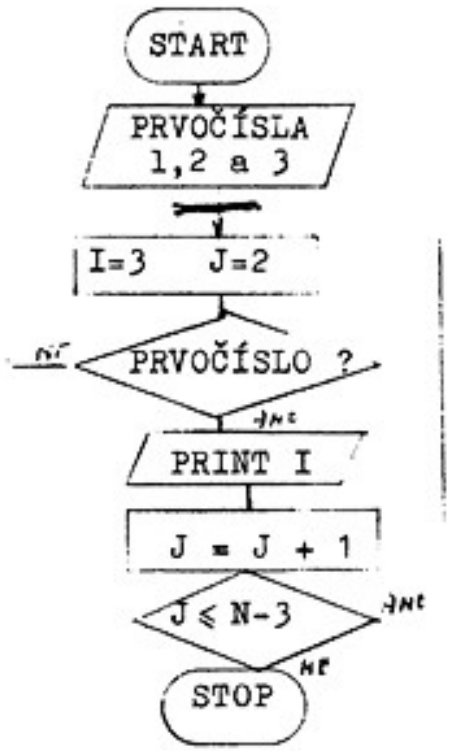
3.4. P R V O Č Í S L A

Jedním ze způsobů hledání prvočísel je zkušenost, že lichá počínaje trojkou nejsou dělitelná jiným číslem než sebou samým. Nejdříve si tuto metodu objasníme a pak budeme studovat jak nejjednodušeji je možno ji realizovat.

PRVNÍ METODA:

Napište program, který vygeneruje N prvočísel. N se bude měnit mezi 10 a 60. Dále se zaměřte na zlepšení formátu výstupu.

Obr.: 3.16.



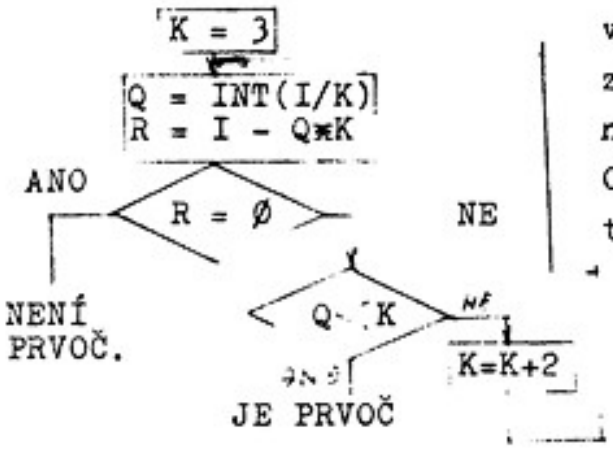
Celková konstrukce programu odpovídá vývojovému diagramu na obr.3.16. Instrukce jsou následující:

- napište za sebou jdoucí čísla 1,2 a 3
- potom určete další prvočíslo nejbližší následující s přírustkem I+2, protože po dvojce jsou všechna prvočísla lichá.

Na zjištění, či I je prvočíslo, spojíme za sebou jdoucí programové testy, až najdeme první vyskytnuvší se případ.

- doplníme nulový zbytek, který značí, že I není prvočíslo
- dostaneme nový zbytek a poměr menší anebo rovný děliteli, což značí, že I je prvočíslo

Obr.: 3.17.



Na otázku "je I prvočíslo?" musíme provést uvedené kroky v částech uvedených ve VD zobrazeném na obr.3.17. Na jeho základě můžeme navrhnout podobný VD a napsat program. Např. viz obr.3.18. Na Obr. 3.19. je uveden přehled výsledků tohoto programu

Obr.: 3.18.

```
100 N=6Ø
110 PRINT :PRINT "SEZNAM";N;
115 PRINT "PRVNÍCH PRVOČÍSEL":PRINT
120 PRINT 1, 2, 3
130 I = 3
140 FOR J=1 TO N-3
150 I=I+2
160 K=3
170 Q=INT(I/K)
180 R=I-Q*K
190 IF R=Ø THEN 15Ø
200 IF Q≤ K THEN 23Ø
210 K=K+2
220 GOTO 17Ø
230 PRINT I,
234 REM TISK VE TŘECH SLOUPCÍCH
235 IF J-3*INT(J/3)=Ø THEN PRINT
240 NEXT J
250 END
```

OBR.: 3.19.

1	2	3
5	7	11
13	17	19
23	29	31
37	41	43
47	53	59
61	67	71
73	79	83
89	97	101
103	107	109
113	127	131
137	139	149
151	157	163
167	173	179
181	191	193
197	199	211
223	239	241
251	157	163
269	271	277

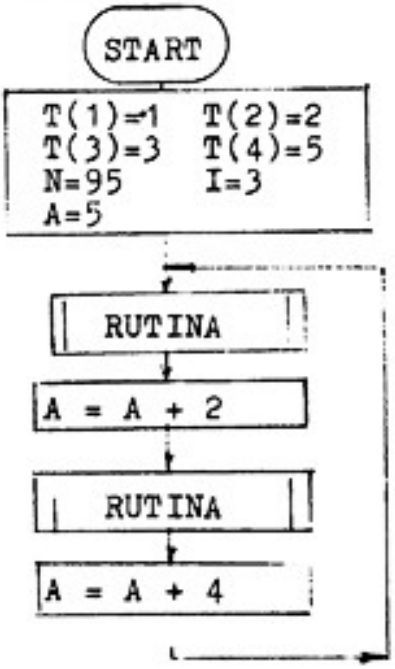
DRUHÁ METODA :

Počínaje číslem 5 mají všechna prvočísla tvar 6n ∓ 1, kde n je celé číslo. Můžeme si zvolit všechny dělitele ze skupiny prvočísel, které jsme již získali.
Napište program, který toto porovnání vezme do úvahy.

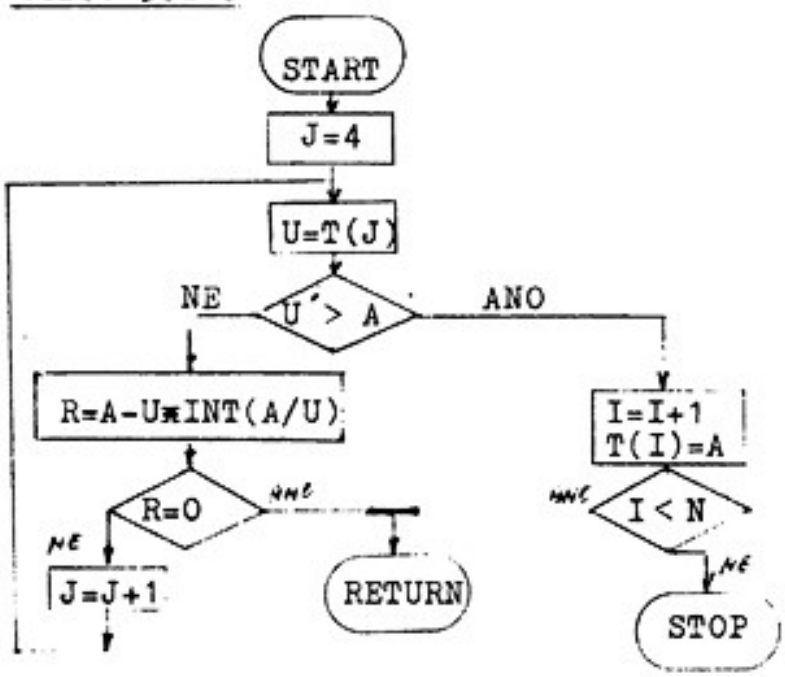
Řešení: abychom omezili hledání možných dělitelů na prvočísla již zjištěná, musíme být schopni si je uložit, anebo zapamatovat. To vyžaduje použít pole a dimenzování tohoto pole bude limitováno maximálním prvočíslem, které má být hledané. Využijeme fakt, že čísla nabývají tvar 6n±1. Porovnáváme, zda čísla která hledáme,nejsou identická s dvojkou anebo trojkou. Stačí tedy použít identifikaci od pětky nahoru. Naší činnost je možno rozdělit do dvou kroků:
1.Hlavní program, který inicializuje několik prvních prvků do pole T testovacích a zkušebních dělitelů, potom počítáme hodnotu proměnné A a voláme podprogram.
2. podprogram se zastaví, když zjistí, že hodnota proměnné A je prvočíslo. Pokud ano, uloží ho do pole T. Když je T plné,jeho obsah se vypíše.

Tato diskuse nás dovedla k VD na obr. 3.20. a 3.21. Příklad výsledků je na obr.3.23.

Obr.: 3.20.



Obr.: 3.21.



Obr.: 3.22.

```
10  N=95
90  PRINT "SEZNAM ";
95  PRINT "OBSAHUJE"
97 PRINT N;"PRVOČÍSEL"
100 PRINT
104 POKE 2Ø1,7: REM Nastavení TAB
105 REM s krokem = 7
110 PRINT 1,2,3,
140 DIM T(N)
150 T(1)=1:T(2)=2:T(3)=3:T(4)=5
160 A=5:I=3
170 GOSUB 5ØØ
180 A=A+2
190 GOSUB 5ØØ
200 A=A+4
210 GOTO 17Ø
495 REM
500 REM Podprogram
510 J=4
515 U=T(J)
520 IF (U*U > A) THEN 560
530 R=A-INT(A/U)*U
540 IF R=Ø THEN RETURN
550 J=J+1: GOTO 51Ø
560 I=I+1
570 T(I)=A
635 PRINT A,
650 REM Formát výstupu do 5
655 REM sloupců
660 IF I-5*INT(I/5)=Ø THEN PRINT
670 IF I< N THEN RETURN
680 END
```

Obr.:3.23.
Počítač Vám zobrazí v pěti sloupcích prvních 95 prvočísel.

3.5. ROZLOŽENÍ NA SOUČIN PRVOČÍSEL

Rozdělení čísla na prvočísla znamená napsat všechny prvočíselné dělitele tohoto čísla.

Základní přístup: Dělíme číslem dvě. Když získáme správného dělitele, vytiskneme ho. Když dělitel není vhodný anebo když se nedá dlouho vypočítat, přejdeme na další číslo.

- když dělenec je dané číslo,je dané číslo prvočíslo

- dělenec je menší než dané číslo, potom tento dělenec je prvočíslo a dělitel pro dané číslo.

ÚKOL: Navrhněte program, který provádí tuto kategorizaci a pokračuje počítáním dělitelů čísla, pokud nedostaneme záporné číslo anebo nulu.

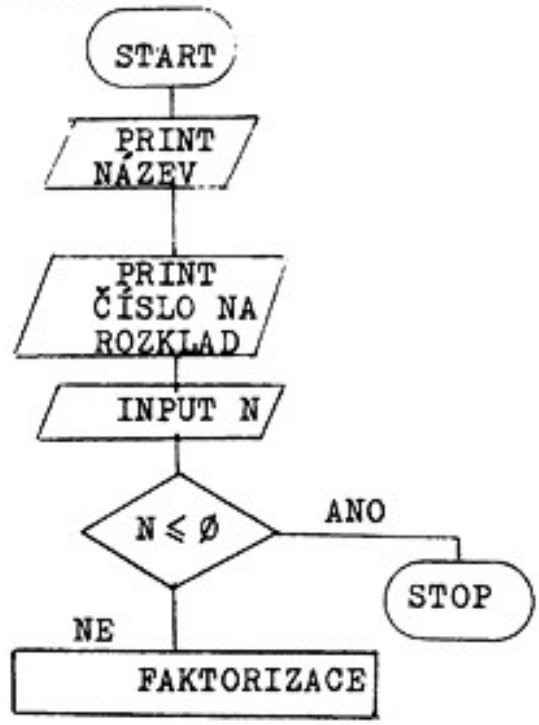
ŘEŠENÍ: Základní struktura programu je uvedena ve VD na obr.3.24. Ukážeme si nyní práci části FAKTORIZACE na vývojovém diagramu podrobněji. Na provedení faktorizace můžeme použít tento algoritmus.

1. Uložíme N do N1.
2. Postupně dáváme 2,3,4,5 ..... do I
   2a. Ověříme je-li I dělitelem N
   - NechťQ je hodnota kvocientu
   - když I je správný dělitel, potom tískneme I, položíme N=Q a jdeme na 2a.
   - Když I není dělitelem, jdeme potom na 2b.

   2b. Když Q I zvýšíme I a vrátíme se na 2. Pokud je poměr Q I, potom:
   -když N=1 potom končíme
   - když N = N1 potom N je prvočíslo
   - když N N1 potom N je dělitel a musí být vytlačený.

Tento postup je ilustrovamý na VD na obr. 3.25, z kterého bez těžkostí odvodíme skutečný program uvedený na obr. 3.26.

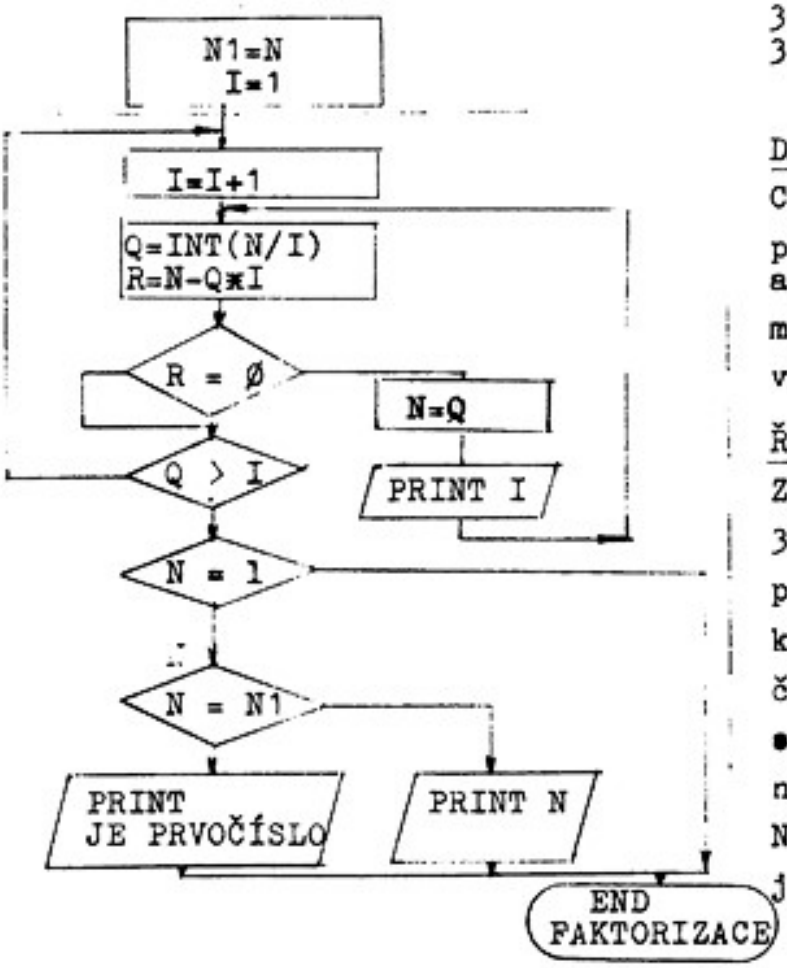
Obr.: 3.24.



OBR.: 3.26.

```
120 PRINT "Rozložení na";
125 PRINT "součin prvočísel"
130 PRINT
140 PRINT "Číslo na rozklad ";
145 INPUT N
150 IF N≤Ø THEN END
160 N1 = N
170 I=1
180 I=I+1
200 Q=INT(N/I)
210 R=N-Q*I
220 IF R <> Ø THEN 29Ø
230 N=Q
240 PRINT " ";I;"  ";
250 GOTO 2ØØ
290 IF Q > I THEN 18Ø
300 IF N=1 THEN 35Ø
310 IF N <> N1 THEN 34Ø
320 PRINT
330 GOTO 35Ø
```

Obr.: 3.25




```
340 PRINT " ";N;"  ";
350 PRINT
360 GOTO 13Ø
370 END
```

DALŠÍ POSTUP :

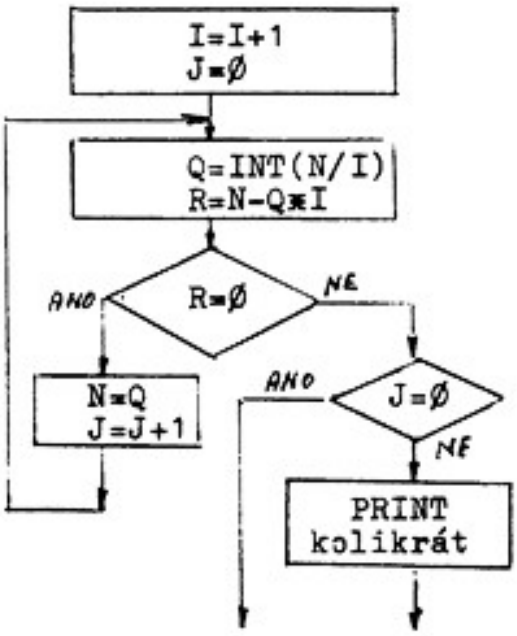
Cílem tohoto cvičení je,dát předběžný programový příklad a poskytnutím dalších informací získat zlepšený výstup výsledků.

Ř E Š E N Í :

Změníme modifikaci VD na obr. 3.25. a to tak, že se bude vypisovat dělitel pouze tehdy, pokud je úplně určený.Část označená čárkovaným obdelníkem na obr.:3.25. bude nahrazena VD na obr.3.29.

Nyní můžeme použít předcházející program na návrh nového programu. Kromě této změny uvedené na obr.3.29. se budou měnit vypisované instr. tak jak je uvedeno na obr.:3.28.

Obr.:3.29.



Obr.:3.27.

```
Rozložení na součin prvočísel
Číslo na rozklad ? 12
2  2  3
Číslo na rozklad ? 65784
2  2  2  3  2741
Číslo na rozklad ? 1217
JE PRVOČÍSLO
```

Obr.:3.28.

```
Rozložení na součin prvočísel
Číslo na rozklad ? 65784
Je dělitelné  2  3 KRÁT
Je dělitelné  3  1 KRÁT
Je dělitelné  2741  1 KRÁT
```

Obr.3.30.

```
120 PRINT "Rozložení na";
125 PRINT "součin prvočinitelů"
130 PRINT
135 POKE 201,6:REM Nastavení
136 REM šířky sloupců
140 PRINT "Číslo na rozklad";
145 INPUT N
147 N1=N
150 IF N≤0 THEN END
170 I=1
180 I=I+1
190 J=0
200 Q=INT(N/I)
210 R=N-Q*I
220 IF R<>0 THEN 260
230 N=Q
240 J=J+1
250 GOTO 200
260 IF J=0 THEN 290
270 PRINT " je dělitelné";
275 PRINT I,
277 PRINT J,"krát."
280 GOTO 180
290 IF Q > I THEN 180
300 IF N=1 THEN 350
310 IF N<> N1 THEN 340
320 PRINT "        je prvočíslo"
330 GOTO 350
340 PRINT "   je dělitelné ";
345 PRINT N,"1","krát."
350 PRINT
360 GOTO 130
370 END
```

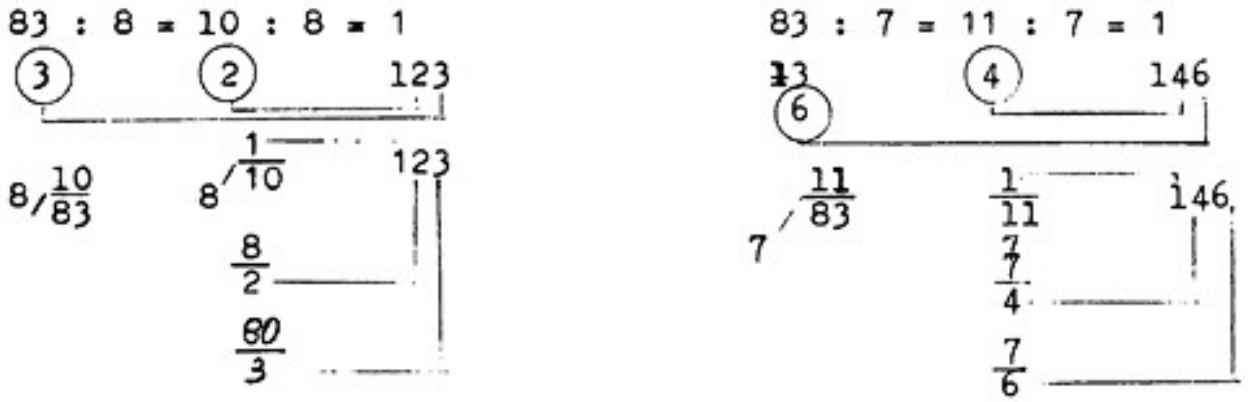
3.6. ZMĚNA ZE ZÁKLADU DESET NA JINÝ ZÁKLAD

Zobrazení čísel v různých systémech (základ 10, základ 8, základ 2 atd.) je pro „člověka z ulice" cvičenému v matematice nepraktickými hodnotami. Celkem opačně je tomu u lidí zabývajícími se programováním. Jejich úkolem je skutečná aplikace, speciálně při programování ve strojovém jazyce.

V principu se převod skládá z těchto kroků:

- provedeme postupné dělení novým základem, pokud získaný poměr není menší než nový základ.

Příklad: Změna čísla 83 ze základu 10 na základ 8 a 7.



Hodnota číslice řádu JEDNOTKY odpovídá zbytku prvního dělení. Další číslice řádu desítky odpovídá zbytku po druhém dělení a tak dále. Číslu 83 při základu 10 odpovídá 123 při základu 8 a číslo 146 při základu 7.

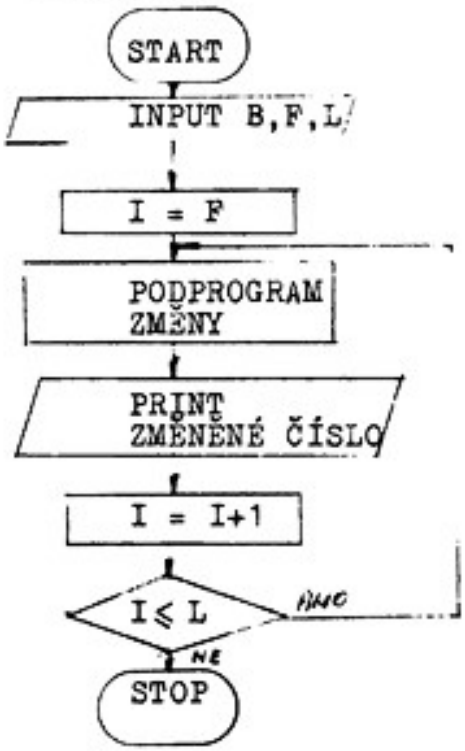
3.6.1. Změna na základ menší jako 10.

Cvičení: Napište program, který tvoří tabulku změny pro interval čísel mezi dvěma zadanými čísly F a L, které zadává uživatel. Změna bude dělaná ze základu 10 na jiný základ B, který je menší než 10.

Řešení: Při konstrukci tohoto programu si všimneme co bylo společné předešlým programům.

Například:
- použití celočíselného dělení
- výpočet zbytků
- použití polí

Obr.:3.31.



Na znázornění základní instrukce vhodného algoritmu v čistém tvaru, nejprve rozdělíme program na dvě části:

Hlavní program, který bude řídit potřebné vstupy a výstupy a podprogram, který bude řídit skutečnou změnu základu.

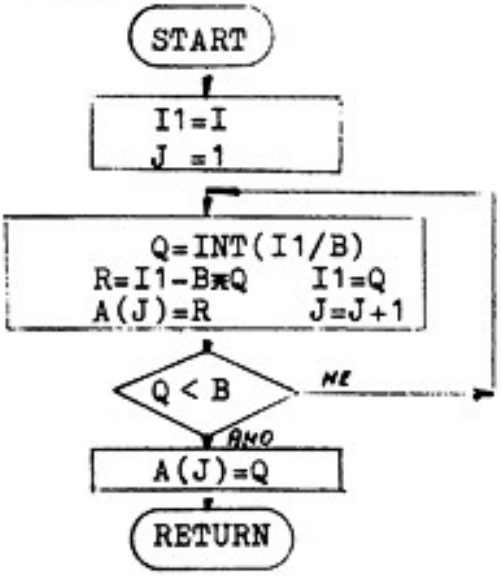
Koncepční VD na obr.3.31. je celkem jednoduchý.

Na druhé straně vývojový diagram na obr. 3.32. vyžaduje určité vysvětlení.

- dopředu nevíme počet číslic, nového změněného čísla na jiný základ
- navrhovaný postup dává nejprve číslici řádu jednotek, potom desítkovou(anebo přesněji poměr základu upravený na 1), atd.
- uložíme každou číslici do pole A

  A(J) = R
  J = J+1 (pro uložení další čísl.)

Obr.:3.32.



Když nyní rozumíme VD na obr.3.32. ukážeme podrogram na změnu a můžeme napsat celý program. Když píšeme převedené číslo, nusíme postupovat v opačném pořadí, než jak byly číslice získané.

Například:

Je-li převedené číslo 127 potom pole A(J) bude:

A(1)=7
A(2)=2
A(3)=1 a J=3

K napsání tohoto výsledku ve správném pořadí použijeme následující instrukce:

```
FOR  D=J TO 1 STEP  -1
PRINT A(D)
NEXT (D)
PRINT                    - posune se na další řádek
```

Obr. 3.33.

```
10  DIM A(15)
100 PRINT "Nový základ";
110 INPUT B
120 PRINT "První a poslední"
125 PRINT "měněné číslo";
130 INPUT F,L
135 POKE 2Ø1,8:REM Nastavení
137 REM šířky sloupců
140 FOR I=F TO L
150 PRINT
160 GOSUB 1500
170 REM Vypisování a tabelování
175 REM výsledků
180 PRINT " "; I,
190 FOR D=J TO 1 STEP -1
200  PRINT " ";A(D);" ";
210 NEXT D
220 NEXT I
230 END
1500 I1=I
1510 J=1
1520 Q=INT(I1/B)
1530 R=I1-Q*B
1535 I1=Q
1540 A(J)=R
1545 J=J+1
1550 IF Q ≥ B THEN 152Ø
1560 A(J)=Q
1570 RETURN
1580 END
```

3.6.2 Změna na základ větší než 10.

ÚKOL: Rozšiřte program na změnu základu a výpis tabulky převedeného čísla na základ větší než 10. V tomto případě respektujte číslo 10 a písmeno A, číslo 11 a písmeno B atd.

ŘEŠENÍ: Pro tuto úlohu použijeme znakový řetětec. Například můžeme vytvořit znakový řetězec BØ:

BØ = " Ø 1 2 3 4 5 6 7 8 9 A B C D E F "

v tomto případě potřebujeme "číslici" pro hodnotu A(2) (v uvedeném případě) jednoduše vybereme jeho tvar v poloze A(L)+1 z řetězce BØ. Většina BASICu NÉ ale pro ATARI,to provádí pomocí řetězových funkcí, jako je SUBSTR, anebo MIDØ. V takových BASICÍCH můžeme napsat:

```
PRINT MIDØ(BØ;A(L)+1,1)
```

Obr.:3.35.

```
10   REM Program na změnu zákl.
50   DIM A(15)
90   B$="Ø123456789ABXDEFGHIJKLMN"
100  INPUT "NOVÝ ZÁKLAD?";B
120 PRINT  "PRVNÍ a POSLEDNÍ";
125  PRINT "ČÍSLO"
130  INPUT  "CONVERT?";F,L
140 FOR I=F TO L
150  PRINT
160 GOSUB 1500
170 REM Vypisování a tabelování
180 PRINT " ";I;TAB(7);
190 FOR D=J TO 1 STEP -1
200 PRINT MID$ (B$,A(D)+1,1);
210 NEXT D
220 NEXT I
230 STOP
1480 REM Změna základu
1500 I1=I
1510 J=1
1520 Q=INT(I1/B)
1530 R=I1-Q*B
1535 I1=Q
1540 A(J)=R
1545 J=J+1
1550 IF Q ≥ B THEN 1520
1560 A(J)=Q
1570 RETURN
1580 END
```

pro vypsání potřebných znaků. Příklad programu je na obr.: 3.35.

PRO ATARI BASIC:

ATARI má místo funkce SUBSTR anebo MID$ jiný způsob. Po deklarování má individuální délku řetězce B$. Na začátku programu může být jeho část vybraná výrazem

B$ (I,J)

kde I je první znak podřetězce a J je poslední znak. V souladu s touto vlastností píšeme:

```
A1=A(L)+1
PRINT B$(A1,A1)
```

čímž se vypíše jeden A1 znak. Program je uvedený na obr.3. 37. a přehled výsledků je na obr. 3.38.

Obr.3.37.

```
10   REM.Progr.na změnu zákl.
15   REM číselné soustavy
50   DIM A(+5),B$(3Ø)
90   B$="Ø123456789ABCDEFGHIJKLM
100  PRINT "Nový základ"
110 INPUT B: PRINT
120 PRINT "První a posl.měněné"
125 PRINT "číslo"
130 INPUT F,L:PRINT
150 PRINT "Základ lØ Základ";B
155 FOR I=F TO L: PRINT
156 FOR I=F TO L:PRINT
160 GOSUB 1500
170 REM Vypisování a tabelování
1
175 REM Vypysování a tab. výsledků
180 PRINT " ";I,
190 FOR D=J TO 1 STEP -1
195  A1=A(D)+1
200 PRINT B$(A1,A1)
210 NEXT D
220 NEXT I
230 END
1480 REM Změna základu
1500 I1=I
1510 J=1
1520 Q=INT(I1/B)
1530 R=I1-Q*B
1535 I1=Q
1540 A(J)=R
1545 J=J+1:IF Q ≥ B THEN 1520
1560 A(J)=Q
1570 RETURN:END
```

Obr.:3.38.

Nový základ ? 16

První a poslední měněnné číslo? 12,20

Základ 10	Základ 16
12	0C
13	0D
14	0E
15	0F
16	1Ø
17	11
18	12
19	13
2Ø	14

Z Á V Ě R :

Úlohy s různou obtížností uvedené v této kapitole ukazují užitečnost konstrukce VD. Jak je ukázané, je výhodnější postupovat od základní informace. Postupně se rozvíjíme ve VD až po bod, kdy je možno jednoduše stanovit program. Pro každý řešený problém, je řešení obecné a není stejné v žádné použité metodě řešení. Programy zde uvedené, nejsou navrhnuty jako nejvýhodnější. Jsousrozumitelnéa navrhnuty tak, aby co nejlépe popisovaly VD. Pokut chcete, můžete zkrátit čas nutný pro délku programování a podle VD přímo navrhnout nový program.

RNDr. Gajdoš Vojtěch

V následujících číslech olomouckého zpravodaje, budou uveřejněny další kapitoly této zajímavé knihy.

D E B U G M O N I T O R

ÚVOD

Následující text obsahuje popis funkcí programu DEBUG MONITOR, jejich formátu a způsobu použití.
Program je určen pro přímou práci se strojovým kódem počítače ATARI. Pro tento účel obsahuje funkce umožňující vkládat program ve strojovém kódu, funkce které vložený program přeloží do symbolických zkratek (dissassemblují), funkce které umožní program odladit (break pointy) a jiné další funkce umožňující např. práci s více dysplaji, s kazetovým magnetofonem atd.

Program je konstruován tak, aby byl kompatibilní s operačním systémem. To umožňuje použití periferních zařízení. Např. je možno výstupní tisky programu (disassemblovaný program) vytisknout na tiskárně. Uživatel též může ve svých programech využívat operačního systému, včetně přístupu na periferní zařízení (IOCB bloky, rutiny CIO a SIO).
Monitor zahrnuje celkem asi třicet funkcí. Po nastartování monitoru je monitor v módu výběru funkce, poznáme to vždy podle vypsané dvojtečky. Funkci vybereme stiskem jednoho z písmen abecedy. Každé jedno písmeno odpovídá jedné funkci. Po výběru funkce vkládáme další parametry potřebné pro danou funkci (adresy, čísla, atd). Po provedení funkce se monitor opět vrátí zpět do módu výběru funkce.

SYMBOLY

V této kapitole uvedeme některé znaky, které jsou použity jako symboly a mají svůj určitý význam.

1) Znaky vyjadřující číselnou soustavu

$ (string) označuje číslo nebo adresu v hexadecimální soustavě

% (procento) označuje soustavu dekadickou

! (vykřičník) označuje číslo ve dvojkové (binární) soustavě

2) Při vkládání dat můžeme kromě tří výše uvedených symbolů užít ještě:

" (uvozovky) značí, že budeme vkládat ASCII znaky

' (apostrof) značí, že budeme vkládat znaky ze sady display ASCII

3) Ve výpisu disassemblovaného programu se nachází symbol =, který označuje, že operandem instrukce strojového kódu je číslo nikoliv byt paměti na nulté stránce. Vyskytuje se vždy u instrukcí s pří-

mým adresovacím módem.

4) Při zadávání adres pro funkce MONITORu vypisuje monitor sám různé oddělovací znaky (rovnítko, pomlčku, dvojtečku), které nemají žádný specifický význam a slouží pouze k oddělení vstupních adres.

VKLÁDÁNÍ DAT

Pro vkládání dat je určena funkce N.Formát funkce je N<adr>
Výraz <adr> označuje šestnáctibitovou adresu resp. dvoubytovou adresu, kterou můžeme zadat v jedné z možných číselných soustav. Po nastartování programu pracuje monitor automaticky v hexadecimální soustavě. Pokud chceme zadat adresu v jiné soustavě, stiskneme příslušný symbol (% ! atd) a adresy zadáme v dané soustavě. Po zadání čísla v této nové soustavě očekává monitor další data opět v původní soustavě (po nastartování v hex.). Pokud chceme zafixovat trvale jinou soustavu, stiskneme symbol této soustavy dvakrát. Potom všechna další data zadáváme v této soustavě až do další změny.

Stejná pravidla platí pro zadávání osmibitových (jednobytových) čísel (dále značena jako číslo). Můžeme zvlášť zafixovat jednu soustavu pro dvoubytové adresy (hexadecimální nebo dekadickou) a zvlášť druhou soustavu pro vkládání jednobytových čísel (hex., dec., bin., ASCII, DISPL. ASCII). To umožňuje zadávat trvale např. adresy v dekadické soustavě a čísla třeba binárně.

Důležité je, že v tomto MONITORu není po zadání čísla potřeba stisknout RETURN. Adresa nebo číslo je vloženo automaticky tehdy, když vložíme potřebný počet cifer odpovídajících nastavené číselné soustavě a tomu, zda vkládáme adresu nebo číslo. Proto je nutné zvláště u dekadických čísel vypisovat všechny nuly na vyšších řádech (např. adresu 2955 musíme zadat jako Ø2955).

Tato zvláštnost činí ze začátku potíže, ale po přivyknutí značně urychlí práci s MONITOREM.

Zvláště se určuje číselná soustava výstupu, tedy soustava, ve které MONITOR tiskne výstupní data. Změna této soustavy se provádí opakovaným stiskem tlačítka SELECT.

Nyní zpět k funkci N. Po stisku N a zadání adresy se vypíše tato adresa a její obsah a MONITOR čeká na další povel. Nyní můžeme zvolit ze čtyř možností. Stiskem RETURN zůstane obsah této adresy nezměněn a vypíše se další následující adresa a její obsah.

Stiskem DELETE/BACK/SPACE zůstane taktéž obsah nezměněn, ale vypíše se adresa o jeden byt zpět (máme možnost se v paměti vrátit dozadu). Zadáme nový obsah, tedy číslo od Ø do 255 (dec.) v libovolné zvolené soustavě.

Stiskem tlačítka, které nemá smysl se vrátíme k novému výběru funkce (zrušíme funkci N). Monitor reaguje vypsáním otazníku a dvojtečky. Stisk nesmyslného tlačítka způsobí vždy návrat k výběru funkce u všech funkcí. Další možností je zavést data do počítače z kazetového magnetofonu nebo diskety o čemž pojedná zvláštní kapitola.

VÝSTUP DAT

K výstupu dat slouží funkce Q a T.

1) Funkce Q poskytuje výpis disassemblovaného programu ve strojovém kódu od zadané adresy. Adresu odkud se provede výpis můžeme definovat po stisku Q jednou ze tří možností.

Vložíme přímo adresu - formát Q<adr>

Stiskneme RETURN , výpis se pak provede od adresy, kterou začínala poslední vypsaná obrazovka - formát Q RETURN

Stiskneme DELETE/BACK/SPACE , výpis se provede od poslední adresy napsané za Q - formát Q DELETE/BSP

Po volbě jedné z těchto možností se provede výpis jedné obrazovky (2Ø řádek programu). Nyní můžeme zvolit z možností

- po stisku RETURN se vypíše pokračování programu - celá další obrazovka (dalších 2Ø instrukcí)
- po stisku mezerníku se vypíše jedna další instrukce
- vybereme přímo jinou funkci (zrušíme funkci Q) popř. stiskneme nesmyslnou klávesu

Disassembler odpovídá standartnímu formátu assembleru 65Ø2. Monitor však ještě poskytuje upravenou versi disassembleru, kterou získáme po stisku OPTION. Opětovným stiskem tlačítka OPTION přepínáme standartní disassembler a speciální (upravený) disassembler. Úprava spočívá v některých detailech jako např. vypisování podmínek u podmíněných skoků, podrobnější transkripce některých zkratek, vypíše + namísto čárky v sčítacích adresovacích módech a podobně. Tato verse assembleru je zcela NEOFICIÁLNÍ a je vlastním výmyslem autora.

2) Funkce T slouží k tisku číselných tabulek. Formát je

T<adr>,<čís.1>,<čís.2>,...

Adresa udává místo v paměti odkud se tabulka vypisuje, jednotlivá

čísla udávají počet čísel na řádku. Výstup z funkce T je při stisku nesmyslné klávesy.

V kombinaci s různými číselnými soustavami výstupu (pomocí klávesy SELECT) je možno tisknout tabulky čísel hexadecimálních, decimálních, binárních, textových tabulek apod.

Pokud chceme výstup dat na tiskárnu připojíme tiskový výstup stisknutím P. Klávesa P funguje jako přepínač, který podle současného stavu buď připojí nebo odpojí výstup dat na tiskárnu. Informaci o připojení nebo odpojení po stisku P tiskne na obrazovku. Pokud se snažíme připojit tiskový výstup a není k počítači připojena tiskárna, k připojení tiskového výstupu nedojde.

ODLAĎOVACÍ FUNKCE

K odlaďování programu ve strojovém kódu jsou funkce B, E, C, K, A, X, Y, V, W.

Nejprve stručně o principu bodu přerušení (break pointu) ve strojovém kódu.

Soubor instrukcí 65Ø2 obsahuje instrukci BRK (op. kód ØØ), která způsobuje při jejím vykonání IRQ přerušení. Monitor využívá této instrukce tak, že na adresu, kam chceme umístit bod přerušení uloží instrukci BRK (ØØ) a přitom si uschová původní obsah této adresy. Jestliže potom spustíme program, ve chvíli, kdy řešení spěje k instrukci BRK, se pomocí přerušení přenese řízení do monitoru, který analyzuje, z které adresy přerušení přichází, na tuto adresu uloží původní obsah (zruší instr. BRK), uschová informace o obsahu displeje (viz dále funkce O), nastaví na obrazovku displej monitoru a vypíše obsahy registrů procesoru 65Ø2 (A,X,Y,PSW,SP) a zvlášť vypíše stavy příznakových bitů (Z,C,N,V,I,D). Obsahy těchto registrů si monitor uschová do své pracovní paměti. Nyní máme možnost pracovat v monitoru a upravovat odlaďovaný program. Po případných úpravách máme možnost pokračovat v přerušeném programu od místa přerušení. Zvolením příslušné funkce monitoru provede monitor nastavení displeje, jaký byl ve chvíli přerušení break pointem, dosadí do registru procesoru 65Ø2 jejich obsahy při přerušení, které si uchoval v pracovní paměti a skočí na místo, kde nastalo přerušení break pointem.

Monitor umožňuje nastavit současně čtyři body přerušení. Dále obsahuje ještě zvláštní variantu nazvanou cyklický bod přerušení. Tento break point je plánován pro umístění do smyček (cyklů). Jeho

funkce je odlišná v tom, že při jeho zadání zadáváme ještě číslo, které udává počet kolikrát má řešení tímto bodem přerušení projít, než se přerušení provede. Např. budeme chtít, aby smyčka proběhla padesátkrát a při tom chceme program přerušit ve dvacátém průchodu smyčkou. Použijeme tedy cyklický break, u kterého při zadání uvedeme po adrese ještě číslo 2Ø (dec.).

Nyní k formátům funkcí. Pro výrazy ve formátech této kapitoly platí adr je dvoubytová adresa, čís. je jednobytové číslo, c je číslo break pointu od 1 do 4 nebo písmeno C pro cyklický break.

1) Funkce B nastaví bod přerušení na udanou adresu. Formát funkce je B c,< adr>
Pro cyklický break speciálně BC, <adr>, <čís>
Pokud daný break point již byl zadán a ještě nebyl proveden nebo zrušen vypíše monitor hlášení "JIŽ BYLO DEFINOVÁNO !" a adresu na které je již break postaven. Musíme pak zvolit jiné číslo break pointu nebo tento zrušit.

2) Zrušení break pointu se provede funkcí E. Formát funkce je E c
Pokud chceme zrušit break, který ještě nebyl zadán vypíše monitor otazník a požadavek ignoruje.

3) Pokračování po přerušení provedeme funkcí C. Po stisku C se provedou operace, jak bylo popsáno v úvodu této kapitoly a řešení pokračuje od posledního přivedeného break pointu.

4) Po provedení přerušení breakem se na displeji vypíší obsahy registrů procesorů a uloží se dopracovní paměti monitoru. Před pokračováním funkcí C máme možnost obsah těchto registrů měnit. Slouží k tomu funkce A, X, Y, které mění obsah registrů A,X,Y a funkce V, kterou lze měnit obsah stavového registru. PSW.
Formáty těchto funkcí jsou takové, že po stisku písmene se vypíše obsah příslušného registru (tak jak je uložen v pracovní paměti monitoru) a je očekáváno zadání nového obsahu, příp. stisk nesmyslného tlačítka obsah nezmění a funkci zruší.

5) Opakovaný výpis obsahu registru a příznakových bitů jako při posledním přerušení získáme po stisku K.

6) Funkce W poskytne výpis adres všech pěti break pointů, u cyklického break pointu se navíc vypíše počet cyklů. U breaků, které ještě nebyly postaveny je adresa vždy ØØØØ.

Pokud uživatelův program narazí (i nezáměrně, omylem) na instrukci BRK, která není break pointem, ale je součástí dat nebo čisté paměti, přerušení se provede a ve výpisu je uvedeno číslo breaku Ø,

ODSKOKY Z MONITORU

Do této kapitoly patří funkce G, D, aH.

1) Funkce G slouží ke spuštění uživatelova programu. Formát funkce
G <adr>

Do zásobníku se uloží návratová adresa monitoru, takže zpětný návrat do monitoru může být proveden instrukcí RTS nebo samozřejmě přes s break point.

Obsahy registrů procesoru při odskoku funkcí G se dají určit funkcemi X, Y, A v podobně jako při použití funkce C po přerušení.

2) Funkce D způsobí přechod do vestavěného niniDOSU, který se ohlásí vypsáním vlastního menu. O funkcích tohoto DOSu pojedná zvláštní kapitola.

3) Funkce H slouží k přechodu do BASICu. Po prvním nastartování BASICu je třeba zadat příkaz NEW, jinak se BASIC rád při některých příkazech zablokuje. Pro návrat do monitoru je možno použít následujícího postupu. Zadáme POKE 2,246:POKE 3,186:POKE 1016,1 a stiskneme tlačítko RESET. Pokud pracujeme s nahraným DOSem, zadáme místo POKE 9,2 příkaz POKE 9,3.

PŘESUNY PAMĚTI

Do této kapitoly patří funkce M, R, I.

1) Funkce M umožňuje přemístit blok paměti beze změny na jiné místo.
Formát funkce je M <adr 1> - <adr 2> = <adr 3>

adr 1 je počáteční adresa a adr 2 koncová adresa přesunovaného bloku paměti. adr 3 je nová počáteční adresa, kam chceme blok přemístit. Při používání je třeba dbát na to, aby se původní a přesunutý blok kriticky nepřekrývaly.

2) Funkce R má stejný formát jako funkce M, liší se však v tom, že v přesunovaném bloku se zvětší adresy u všech tříbytových instrukcí přesně o rozdíl mezi původní a novou počáteční adresou. Výsledný efekt je takový, že přemístěný program by měl fungovat v novém umístění bez dalších změn. Tento proces nazýváme přeadresováním (readressing).

3) Funkce I v sobě skrývá dvě různé funkce. První funkce má formát

I < adr 1> - < adr 2> :<čís>
Tato funkce způsobí posunutí bloku paměti od adr 1 do adr 2 a počet bytů čís . Funkce provádí rozhrnutí programu umožňující dodatečné vsunutí části programu. U posunovaného bloku se podobně jako u funkce R provádí přeadresování.
Druhá funkce je k předchozí inversní a má formát
I <adr 1> - < adr 2>: -<čís>
Rozdíl je ve znaménku mínus před číslem. Toto znaménko musíme zadat z klávesnice, zatímco symboly rovnítka, dvojtečky nebo minusy mezi adresami vypisuje monitor sám. Tato funkce vypustí počet čís bytů od adr 1 a celý zbytek bloku do adresy adr 2 posune o čís bytů nazpět. Obdobně jako u R a předchozí funkce provádí i předadresaci posunovaného bloku. Nově vzniklé místo po rozhrnutí programu je zaplněno číslem EA (hex), což je operační kód funkce NOP.

PRÁCE S DISPLEJEM

Charakter, složení displeje je dáno obsahem několika systémových a hardwarových registrů. Monitor si pamatuje celkem čtyři sestavy displeje (označené pomyslnými čísly od Ø do 3). Displej je definován registry DMACTL, GRACTL, CHACTL, CHBAS, DLISTL, DLISTH, COLPMØ, COLPM 1, COLPM 2, COPLM 3, COLOR Ø, COLOR 1, COLOR 2, COLOR 3, BAKGR, PMBASE. Při spuštění monitoru a při přerušení break pointem se okamžitě obsah současného displeje uloží do pracovní paměti displeje Ø.

Pro manipulaci s displejem je určena funkce C, která může mít čtyři různé formáty.

1) Vybavení displeje. Formát je O < c>
kde < c > je číslo od Ø do 3. Funkce způsobí vybavení displeje < c > na obrazovku.

2) Vybavení displeje monitoru. Formát je O RETURN

3) Přesun hodnot jednoho displeje do druhého. Formát je
OP<c 1> , < c2>
Například po přerušení break pointem máme obsah displeje před přerušením uložen v displeji Ø. Funkcí OP Ø1 přesuneme displej Ø do displeje 1. Nyní, když se opět změní displej Ø (např. dalším break pointem), máme stále v displeji 1 displejové hodnoty uschovány.

4) Nastavení nového displeje. Formát funkce je OS<c>

Tato funkce umožňuje nastavit nebo upravit hodnoty pro displej 0 až 3. Po zadání vypíše monitor název první proměné (DMACTL) a její obsah. Máme nyní tři možnosti

- stiskneme RETURN a obsah tak zůstane nezměněn, vypíše se další proměná.
- stiskneme DELETE/BSP vrátíme se tak o jednu proměnou zpět. V případě první proměné DMACTL zrušíme funkci OS
- zadáme číslo tj. nový obsah proměné

Tímto způsobem můžeme prohlížet a měnit hodnoty určující sestavu displeje.

PRÁCE S MAGNETOFONEM

Pro práci s magnetofonem jsou určeny funkce S,L,U,Z a tlačítko START.

1) Funkce S provádí záznam programu ve strojovém kódu na magnetofon. Monitor používá standartní formát kazetového softwaru ATARI. Nahrávka je tvořena bloky 128 bytů s krátkými meziblokovými mezerami, v úvodu nahrávky je standartní hlavička obsahující informace o nahrávce.

Formát funkce je S < adr 1> - < adr 2> = < adr 3>

adr 1 je počáteční a adr 2 koncová adresa bloku paměti, který má být nahrán na kazetu. adr 3 je startovací adresa, od které můžeme po zpětném nahrání program odstartovat (viz dále). Pokud nechceme startovací adresu uvádět, napíšeme adresu ØØØØ, monitor ji sám nahradí adresou restartu monitoru.

Po zadání těchto parametrů vypíše monitor otázku "DOPLNIT HLACIČKU ? A/N". Každá nahrávka musí mít vždy hlavičku. Pokud nemáme již hlavičku na začátku našeho nahrávaného programu, musíme stisknout A. Monitor tak sám doplní před začátek našeho programu 8 bytů hlavičky. Nakonec nás monitor vyzve k připravení magnetofonu a po stisku RETURN začne probíhat již samotné nahrávání.

2) Funkce L nahraje program z kazety do počítače. P_O nahrání prvního bloku vypíše adresu kam se program nahrává. Pokud by hrozilo přemazání monitoru nahrávaným programem, ohlásí chybu. Stejně tak vznikne-li chyba během nahrávání. Pokud nahrávka má hlavičku doplněnou monitorem po nahrání se neodstartuje. Chceme-li nahraný program odstartovat učiníme tak tlačítkem START. Zpět do monitoru se nám ale nemusí podařit dostat.

3) Funkce U nám umožňuje přečíst z nahrávky pouze hlavičku. Po načtení prvního bloku se vypíše byt flagu (1.byte), počet bloků nahrávky (2.byte), adresu umístění nahrávky (3. a 4. byte), inicializační adresu (5. a 6. byte). Pak následuje verifikace nahrávky. Celý záznam se překontroluje aniž by se nahrál do paměti a pokud je záznam chybný vypíše se číslo chyby, která nastala.

4) Funkce Z slouží k vypínání nebo zapínání motoru magnetofonu. To je výhodné zejména pro vyhledání nahrávek na kazetě.

SPECIÁLNÍ FUNKCE

Do této kapitoly patří funkce F a J.

1) Funkce F vymaže (zaplní) část paměti zvoleným číslem. Formát je
F<adr 1> - < adr 2> = < čís>
Paměť od adr 1 do adr 2 se zaplní číslem čís

2) Funkce J slouží k vyhledávání jednoho až šestnácti znaků.
Formát je J < adr 1> - <adr 2> = < čís 1><čís 2>..... RETURN

Za rovnítkem můžeme zadat posloupnost až čestnácti čísel. Po stisku RETURN monitor prohledává paměť od adr 1 do adr 2 . Pokud v paměti najde zadanou posloupnost čísel, vypíše adresu na displeji a hledá dál.

MINIDOS

Monitor poskytuje základní diskové operace s diskem Po stisku D vypíše monitor menu pro výběr diskové operace.
Funkce A-DIRECTORY poskytne výpis celého adresáře disku. Pokud má adresář více položek než se vejde na obrazovku (2Ø) vypíše se jich jen 2Ø a zbytek až po stisku libovolné klávesy.
Funkce B provádí nepřímý skok přes vektor DOSVEC (ØØØA), který při nahraném DOSu směřuje na restart DOSu. Funkcí B můžeme tedy přejít do standartního DOSu.
Funkce E-RENAME se od standartu liší v tom, že starý a nový název souboru se zadávají každý zvlášť vždy po výzvě FILENAME:
Funkce F a G - LOCK a UNLOCK se nijak neliší od standartu.
Funkce S je taktéž shodná se standartem, vyžaduje dva povinné parametry a dvě povinné (inicializační a startovací adresu)
Funkce L slouží k nahrání programu ve formátu binary save DOS 2.5 a její základní odchylka je v tom, že program se po nahrání nikdy sám neodstartuje ani má-li zadány startovací adresy.
Stisk jiného tlačítka má za následek návrat do monitoru.

ZÁVĚR

Monitor je uložen v paměti místo BASICu od adresy AØØØ do ØØØØ. Restart je od adresy A3ØD. Délka programu je necelých 7KB, zbytek paměti do 8 KB je využíván monitorem jako pracovní operační oblast (paměť).

Při práci s monitorem Vám mnoho úspěchů přeje autor

PETR VACEK

DISK DRIVE TECH TIPS

Tato nová rubrika v našem zpravodaji bude věnována majitelům disketových jednotek. V každém čísle zpravodaje naleznou otištěné rutiny a programy, které jim usnadní práci s disketovou stanicí typu ATARI 1050.

DRIVE SPEED C

Jestliže se Vám často objevují chyby disketového původu je možné, že disketová jednotka nepracuje ve správné rychlosti. Normálně se má disk otáčet rychlostí 288 ot/min (RPM). Následující dva test. programy vám dávají možnost zkontrolovat tuto rychlost. Program č. 1 je určen pro evropské modely počítače, program č. 2 pro americké typy. (Liší se odčítáním vnitřních časových registrů RT CLOCK.) Jestliže Vaše disketová jednotka pracuje v intervalu 288 ± 3ot/min, je v pořádku. V opačném případě je nutná oprava.

Program č. 1: pro evropské modely

```
10 REM
20 REM   Drive Speed Test
30 REM
40 REM  1987  GIA Software
50 REM
60 REM -for all ATARI XL/XE
70 REM -Disk Drive 1050 required
100 DIM OTM$(66)
110 GOSUB 300
120 ? CHR$(125);"Drive Number (1-2) ";: INPUT DN
130 ? :? :? "Please Wait...(max.1 min)"
140 X=USR(ADR(OTM$),DN)
150 OTM=X/3600
160 OTM=INT((100/OTM+0,5)*5/6)
170 ? :? "Drive ";DN;" ... ";OTM;" RPM"
180 IF OTM<292 AND OTM>284 THEN ? :? "Drive 1050 is O.K.":GOTO 200
190 ? :? "Drive 1050 is BAD!"
200 END
300 RESTORE 350
310 FOR I=1 TO 66
```

```
32Ø READ A:OTM$(I)=CHR$(A)
33Ø NEXT I
34Ø RETURN
35Ø DATA 1Ø4,1Ø4,1Ø4,141,1,3,169,1
36Ø DATA 141,1Ø,3,169,Ø,141,11,3,169
37Ø DATA Ø,141,4,3,169,4,141,5,3,169
38Ø DATA 82,141,2,3,169,5,32,83,228
39Ø DATA 199,2Ø3,2Ø9,249,169,1ØØ,133
4ØØ DATA 2Ø3,169,Ø,133,19,133,2Ø,32
41Ø DATA 83,228,198,2Ø3,2Ø8,249,165
42Ø DATA 2Ø,164,19,133,212,132,213,96
```

Program č. 2: pro americké modely

```
1Ø REM
2Ø REM  Drive Speed Check
3Ø REM
4Ø REM  1987  GIA Software
5Ø REM
6Ø REM -for all ATARI XL/XE
7Ø REM -Disk Drive 1Ø5Ø reguired
8Ø REM by San Diego Computer Ent.
9Ø REM ANTIC. June 1987
1ØØ DR=1:DIM DR$(6):DR$="D1:*.*"
11Ø XIO 3,#2,4,Ø,DR$:POKE 764,255
12Ø ? CHR$(125);"Insert DISK In DRIVE ";CHR$(48+DR)
13Ø ? "And Press Any Key ..."
14Ø IF PEEK(764)=255 THEN 14Ø
15Ø ? :? "Press CRESETJ To END..."
16Ø FOR I=1536 TO 1627
17Ø READ D:POKE I,D
18Ø NEXT I
19Ø POKE 1537,DR:GRAPHICS 2+16
2ØØ X=USR(1536)
21Ø SP=PEEK(1662)+PEEK(1663)*256
22Ø SP=INT(24*3600/SP+Ø.5)
23Ø POSITION Ø,5:? #6;"DRIVE SPEED ";SP;" RPM"
24Ø GOTO 200
25Ø DATA 169,Ø,141,1,3,141,11,3,169
```

```
26Ø DATA 82,141,2,3,169,62,141,5,3
27Ø DATA 32,83,229,169,Ø,141,4,3,141
28Ø DATA 128,6,141,1Ø,3,141,129,6,141
29Ø DATA 13Ø,6,133,2Ø,133,19,133,18
3ØØ DATA 32,83,228,166,2Ø,138,237,128
31Ø DATA 6,142,129,6,16,2,1Ø5,255,174
32Ø DATA 129,6,2Ø1,18,48,1,232,232
33Ø DATA 138,2Ø1,24,16,6,142,129,6,76
34Ø DATA 44,6,165,19,141,127,6,165,2Ø
35Ø DATA 141,126,6,1Ø4,96
```

Pramen
Klub San Diego Computer Enthusiast
ANTIC 6/87 připravil Jiří Hrdlička ml.

LETECKÉ SIMULÁTORY PRO ATARI

Pro většinu z nás je létání vysoce teoretická záležitost. Lidé nemají žádné aerodynamické náležitosti dané přírodou. Jako například křídla, a jen velmi málo lidí může a dovede pilotovat letadlo nebo vrtulník, skákat padákem nebo plachtit na rogallu. Ostatní přicházejí k romantice letu na sedadle některého dopravního velikána.

Snad je to nepřístupnost volného letu, která je důvodem posedlosti mnoha generací, od bájného Ikara přes Orwila Wrighta až k raketovému X-15. Myšlenka vzít si křídla a zvednout se vzhůru k divoké modré obloze pohání technický vývoj stále kupředu.

Všichni máme rádi pruh stratosféry nad námi a často toužíme po odpoutání z pout neústupných fyzikálních zákonů, jako je gravitace, kterými nás drží zeměkoule. I když existuje takový vliv kouzla létání, proč je dosud tak málo těch, kteří skutečně políbili oblaka ? Ano, chybí nám statečnost, touha tak velká, aby vymazala znepokojení nad možným nebezpečím, které na nás ve vzduchu číhá na každém kroku.

Přichází letecký simulátor ...

Nyní, s pomocí domácího počítače, může uživatel konečně opustit hranice pevné země a experimentovat s realitou letu.

Po mnoho let pomáhly simulátory vyučovat řízení automobilů, pilotování letadel a vrtulníků nebo obsluhu ponorek. Příchod počítačové technologie však výrazně posunul hranice umění simulace. Piloti dnes získávají svá oprávnění většinou uvnitř computerizovaných kokpitů. Všechny simulátorové konstrukce dokáží okamžitě vytvořit příslušné scenérie (pohledy) a situace vznikající během letu, a to ve kteroukoliv denní nebo noční hodinu.

V Kalifornii mají tréninkový simulátor pro typ BOEING 727, který nebyl plně vytížen a večer nečinně zahálel. Přišel nápad a ihned se realizoval. Během večerních hodin se totiž simulátor přeměňuje na zábavní atrakci pro dospělé. "Piloti" si sednou do leteckého křesla, připoutají se, instruktor jim ukáže nejdůležitější páčky, tlačítka a ukazatele a mohou vesele létat. V hodině letového času volně pojíždí po letišti, snaží se vzlétnout nebo přistát, zkouší manévrovat ve vzduchu, tříští se o vysoké budovy a jinak

obveselují přítomné. To vše za cca $100 na hodinu. Návštěvníci se jen hrnou a majitelé jsou spokojeni.

Dnešní výkonnější mikropočítače však dokáží přenést simulaci letu přímo do domu, do obytných nebo dětských pokojů. Úspěch simulačních programů je fenomenální. Od roku 1984 dominují letecké simulátory pravidelně na vrcholu deseti nejprodávanějších programů a často jsou vyhodnoceny jako nejlepší programy roku.

Letecké simulátory se odlišují od tradičních "leteckých" her v několika ohledech. Nejmarkantněji je to znát v části pohledu hráče na akční prostor. V hrách, jako je DEFENDER (ATARI), PROTECTOR (Synapse) a AQUATRON (Sierra), hráč ovládá letadlo nebo kosmickou loď v rovině. V programech ZAXXON (Datasoft), BLUE MAX (Synapse a BLUE MAX 2001 (Synapse) je použita trojrozměrná perspektiva. Hra STEALTH (Broderbund) používá letadla na obrazovce jako zaměřovaného kurzoru, program FLAK (Funsoft) využívá výhody horem vedeného bodu.

Ale v "čistém" leteckém simulátoru hráč nemůže být a ani není jen externí silou, která manipuluje s létajícím prostředkem na obrazovce jako nějaké computerizované božstvo. Programy leteckých simulátorů dávají pohled přímo do kokpitu pilota. Prvořadým úkolem autora simulace je grafické zpracování, a to takové, aby hráč viděl ovládací panel a situace za "okny" právě tak dobře, jako kdyby skutečně ovládal škrtící klapku.

První letecké simulátory pro domácí počítače byly většinou kosmické hry. Typickým příkladem je slavná hra STAR RAIDERS (ATARI), ve které hráčí pilotují smyšlenou kosmickou loď značné síly a mnoha schopností. Jak kapacita počítačů rostla, uživatelé se začali mnohem více zajímat o simulaci letu opravdového současného letadla. Tak se letecké simulátory vracely - pomalu, ale jistě - zpět dolů na Zemi.

Prvním velkým krokem byl program Steva Kitchena - SPACE SHUTTLE - A Journey Into Space (Activision, 1984 cartridge). Původně byl tento program určen pro systém videoher ATARI 2600, což se pokládá za neuvěřitelnou věc i v současnosti. Autor zde vyřadil joystickově orientovanou techniku videoher a nahradil ji neobyčejně realistickou úpravou. Kitchenův důvtip využít každé vstupní zařízení na ATARI 2600, byl báječný designerský výkon. Avšak ani to ne-

mohlo zastřít fakt, že program Space Shuttle by byl lépe vyhovoval počítači.

Tak se uskutečnil převod ze systému videoher 26ØØ na domácí počítač ATARI. Tato verze je právě tak důvtipně vyřešena jako originál, pouze je jednodušší na ovládání. Škoda jen, že autor nezvládl, dokonale grafiku počítačů ATARI a převzal ji téměř nezměněnou z videohry.

Program je založen na simulaci skutečné kosmické lodi - raketoplánu Space Shuttle a předkládá uživateli všechny problémy, se kterými se může pilot-kosmonaut setkat. Raketoplán musí odstartovat, dosáhnout určitou orbitu (dráhu), setkat se a dopravit do nákladního prostoru poškozený satelit, vybrat si návratovou cestu a připravit se na opuštění orbitální dráhy, tj. přechod přes atmosféru s následným rozžhavením povrchu raketoplánu, a "výlet" zakončit úspěšným přistáním na neobydlené poušti.

Klávesnice počítače dává daleko lepší možnosti ovládání funkcí řídícího panelu raketoplánu než systém ATARI 26ØØ. Hráči mají přístup ke všem údajům (rychlost, palivo, poloha atd.) stejně jako ke snadnému ovládání kosmické lodi.

Tento simulátor nám přináší řadu prožitků, například romantiku stisku určité klávesy, která otevírá nákladní prostor, spouští nebo vytahuje podvozek nebo startuje pohonnou jednotku - raketový motor. Space Shuttle v počítačovém formátu využívá tyto prvky mnohem úspěšněji než předchozí verze.

Program Space Shuttle nabízí několik typů simulace. Pro méně chápavé je zde verze s autopilotem, ve které počítač virtuozně ovládá všechny navigační a řídící úkony. V dalším typu hry hráč ovládá všechno, ale počítač automaticky zasahuje v případě nebezpečí. Když se "astronaut" postupem času cítí dost bezstarostně v kosmu, je pro něho nachystán bonbónek. Při samostatném letu se mu uvolní náklad z palubního nákladního prostoru a on ho musí najít a umístit zpátky. Pozor, neukvapte se! Právě manipulace v trojrozměrném prostoru (X...dopředu/zpátky, Y...vlevo/vpravo, Z...vertikální vzdálenost od Země) vyžaduje mnoho praxe a zkušeností z méně obtížných typů této simulace.

Jestliže máme program v počítači, na obrazovce vidíme palubní

desku raketoplánu s dvojicí horizontálních měřítek a úhlovou mřížkou; horní polovina obrazovky obsahuje pohled z okna kosmické lodi. Jak bylo dříve uvedeno, grafická presentace je poněkud strnulá a schematická, ale naproti tomu program poskytuje vizuální pohled do mnoha odlišných prostředí - od vesmírného prostoru k pouštní krajině. I když grafika není a nemůže být základem umění simulace, přicházíme na řadu vážných nedostatků, rozhodujeme-li znovu o přednostech tohoto programu.

Na druhé straně, mezi různými scénami vystupuje do popředí část průletu atmosférou, což hráč pozoruje jako sérii stříbrně bílých plamenů, které se objevují za čelním "sklem" raketoplánu. Tato scéna, je později vystupňována přechodným "blackoutem" (zatemněním). Hráči budou přísahat, že cítí, jak izolační destičky z povrchu raketoplánu odlétají.

Stejně jako v případě videohry prorazila počítačová úprava programu skulinku do nového světa objevování možného rozsahu počítačové zábavy.

Myšlenka použít simulace jako zábavy byla později znovu objevena Bruce Artwickem v jeho klasickém programu FLIGHT SIMULÁTOR, což je opravdu realistický program simulace letu pro osobní počítače dneška. Povrbuzen tímto úspěchem se Artwick rozhodl zpřístupnit své umění počítačovým masám (HC ATARI, Commodore, Apple ...) a vytvořil program FLIGHT SIMULÁTOR II (Sublogic, 1984/48K disk).

Flight Simulátor II poskytuje užívateli pohled do kokpitu malého letadla typu PIPER 181 Cherokee Archer a umožňuje mu dělat to, co ve skutečnosti musí dělat pilot při letu tímto strojem.

V tomto typu programu je důležitá dokumentace a můžeme říci, že Flight Simulátor II byl výborně zhotoven i po této stránce. Souprava obsahuje dvě příručky, kartu pro rychlou orientaci + program (disk).

Jedna z příruček obsahuje manuál programu, druhá má název "Flight Physics Aircraft Control" a poskytuje pohled na problematiku letu a aerodynamiky s objasněním základních přírodních zákonů. Na příruční kartě jsou vytištěny stručné významy všech potřebných kláves a detailní letecké mapy oblasti Chicaga, Boston/New Yorku, Los Angeles a Seattlu, které obsahují na osmdesát jednotlivých letišť.

Let je zde trochu více komplikovaný. Naštěstí program obsahuje také demonstrační mod, ve kterém počítač pilotuje letadlo, zatímco vy sedíte na židli a jen se díváte. Ale předtím, než počítač převezme řízení letu, můžete změnit dokonce okolní prostředí a podmínky - tj. počasí, dobu letu a vzdálenosti, což jsou všechno programovatelné veličiny.

Jako všechny předchozí programy, tak i tento nemá hi-res grafiku. Přesto je v této letecké simulaci hodně k vidění. Artwick dokonce použil skutečně letecké fotografie na zachycení a produkci zvláštních rysů krajiny i s některými typickými stavbami, jako je Hancock Tower nebo Statue Of Liberty (zrenovovaná!). Na žádné stavbě nenajdete podstatnou chybu. Ve světle náročného zobrazení přírody jako celku můžeme Artwickovi odpustit některé nesrovnalosti a jsme šťastni, že se vůbec trápil sestavováním tohoto programu.

Samozřejmě, že Flight Simulátor II je v "pravém" leteckém modu velice hezká "hra". Je to vlastně čistá simulace letu, kterou zatím překonává pouze skutečnost.

V soupravě Flight Simulátor II se však nachází ještě jeden program, o kterém si nyní povíme více. Ten ukazuje možnost použití Flight Simulátoru II jako destrukčního programu a má název - WORLD WAR I ACE.

Scenérie programu World War I Ace jsou tak dobré, že by program měl zaručený úspěch, i kdyby ho fy Sublogic prodávala zvlášť. Zajisté obsahuje mnoho prvků převzatých z hlavního programu Flight Simulátor II, ale pozitivně urychluje proces mechanického ovládání prvků, limituje vzdušný prostor a opatřuje náš kokpit kulometem. K tomuto programu patří i kousek "švindlu", když Artwick použil radar ve svém dvouplošníku vyrobeném asi v roce 1917.

Piloti - hráči létají pod evropskou oblohou. Přelétávají nad určitým územím, křižují nad řekou a hledají možný cíl útoku (skladu paliva, přistávací plochy, továrny ..). Pilot může být tak laskavý a před operačním letem vypovědět nepříteli válku. Pokud tak neučiní, druhá strana zachovává klid zbraní, a to do okamžiku, dokud nedopadne první bomba. Nepřítel má pochopitelně také svoje stíhací letadla, a tak hráč musí přestat s bombardováním a hledat spásu v rychlém úniku. Pokud se mu to nepodaří, musí se na svou

základnu probít ve vzdušných soubojích. Po přistání může doplnit palivo a opravit letadlo.

Přítomnost programu World War I Ace a bojových scén v soupravě Flight Simulátor II si vynucuje pozornost k otázce problému "čisté" simulace: teď, když je možnost simulace určitých situací co všechno se dá s tím dělat?

Odpověď na tuto otázku již poskytly určité programy. Největšího rozšíření a popularity dosáhly ty programy, které se zabývají simulací bojových situací.

Program F-15 STRIKE EAGLE (MicroProce, 48K disk nebo kazeta) reprezentuje dosud nejúspěšnější simulaci boje - dává hráči možnost posadit se za páky ovládacího mechanismu do kokpitu skutečného stroje ničení, supersonického proudového stíhacího letadla.

Otázku létání ohromně zjednodušilo použití joysticku. Joystickem hráč ovládá a kontroluje stoupání, klesání a pohyb do stran. Tím se uvolnila klávesnice pro méně používané funkce, jako je vypouštění střel, bomb a raket. Tím ovšem nehodláme říci, že na palubě letadla je málo vysokotechnologických zařízení. Piloti mohou použít obranný systém (Chaff Dispenser) proti radarem řízeným raketám, naváděným z pozemní základny nebo mást vypouštěním světelných raket samonavádění infračervených raket. Stiskem určitého tlačítka počítač kdykoliv vykreslí nejrychlejší kurs k jakémukoliv cíli.

Program F-15 Strike Eagle zahrnuje sedm bojových misí na vytypované strategicky důležité vojenské sklady a objekty rozmístěné v zemích na severu Afriky a Blízkého Východu, jako je Libye, Perský záliv, Haiphongský přístav atd. Vynikající dokumentace, přiložená k programu, obstarává plně bojové situace pro každý z těchto spíše apokalyptických "výletů", které zmírňují jednotvárnost bojových akcí.

Grafika je přijatelná, ale nereprezentuje takovou designerskou úroveň, což velmi zřetelně demonstruje zbytek programu. V horní polovině hracího pole obrazovky je umístěno čelní výhledové okno, zatímco dolní polovina obrazovky obsahuje mapu nepřátelského území s vyznačenými cíli, dále radar, který pracuje ve třech stupních rozlišení, a konečně diagram ukazující rozmístění všech použitelných zbraní letounu F-15. K vykreslení přistávacích ploch

hlavních pozemních cílů a vypouštěcích center antiraket naváděných radarem jsou použity schematické značky, což jsou krásně primitivní nesmysly, zastoupené většinou čtverci a znaky "X". Další pozemní cíle jsou zhotoveny v podobném duchu pomocí jednoduchých čar.

Letecké simulátory (continue)

Ačkoliv několik scén se zdánlivě odehrává v noci, mohu vás ubezpečit, že to nepůsobí žádné těžkosti. Obloha se jednoduše nepatrně zbarví - dostane podobu pozdního odpoledne, snad trochu mlhavého. Je to opravdu nešťastné, že noční scény postrádají tu úroveň grafického zfalšování pohledu jako ve dne, ale naopak, že připravují velké optické kontrasty.

Konečně, tvůrci důkladně zpackali přistávací fázi. To znamená, že vlastně žádná neexistuje. Když pilot dostane úspěšně letadlo blízko přistávací plochy nebo paluby letadlové lodi (pozor na rychlost a výšku - obě hodnoty musí být příznivé pro přistání), počítač automaticky předpokládá úspěšné přistání. Je to uděláno proto, aby se autoři vyhli náročnému grafickému zpracování přistávací fáze. Znovu opakuje, je to pořádná ostuda.

Všeobecně řečeno, F-15 Strike Eagle je nicméně velmi dobrá hra, zachycující simulaci boje ve stylu nejlepších válečných utkání a zápasů. Smysly jsou překvapeny skutečností mistrovsky generovanou na tomto vrcholném simulátoru. Největší předností tohoto programu je schopnost hrát non-stop; dokonce mohou hrát simultánně dva hráči (jeden ovládá joystick, druhý obsluhuje klávesnici).

V blízké budoucnosti simulátory bojových situací zřejmě zaujmou dominantní místo na trhu s programy. Další velký průlom do oblasti leteckých simulátorů připravuje "ostřílený veterán" Bruce Artwick. Firma Sublogic připravuje program JET, který má spojovat grafický realismus série Flight Simulátor II a vzrušení z nejlepších vzdušných bojových akcí nynějšího softwaru.

Zakončeme pohledem do více vzdálené budoucnosti. Simulátory budou stále lepší a výkonnější, to podle toho, jak se bude zvyšovat síla, schopnost a kapacita počítače. Dokud bude lidstvo stavět stroje schopné létání, potud si můžete být jisti tím, že někde budou specialisté pracovat na přípravě simulátorů.

F-15 STRIKE EAGLE - MANUAL

Program simuluje let a plnění bojových úkolů letadla F-15 STRIKE EAGLE

Technická data letounu:

vznik: Rok 1980

typ: jednomístná bombardovací verze stíhacího F-15 Eagle

rozměry - rozpětí: 13.95m
délka: 19.43m
nosná plocha: 56.50m^2

hmotnost- prázdná: 11994 kg
maximální vzletová: 30870kg

Motorová skupina - počet motorů: 2
výrobce: Pratt Whitney
typ motoru: F100-PW-100
maximální výkon: 111.1kN (11340kp)

výkony - maximální rychlost: 2660km/h
dolet: 4800 km
dostup: 21000 m (standart)
stoupavost u země: 260m/s

výzbroj: kanon M61A1 = ráže 20mm
- rychlost střelby 6600 ran/min
pumy - hmotnost 6800kg
- standart šest pum po třech bombách
- bomby typu MK-82 hmotnosti 250kg
řízené střely - AIM-91 Sidewinder - samonaváděcí střely krátkého doletu s infračerveným naváděním
- AIM-7P Sparrow - střely středního doletu s automatickým radarovým naváděním

Letoun je vybaven zdokonaleným radiolokátorem HUGHES AN-APG-63 se zvýšenou rozlišovací schopností cílů na zemi při současně plné využitelnosti pro vzdušný boj - dovoluje zjistit cíle v různých odstupech a výškách, Všechny údaje pulsního radaru a informace o stavu zbraní zpracovává a kontroluje palubní počítač (IBM, kapacita 512kB). Takticko-bojový systém letounu je vybaven ochranným zařízením proti nepřátelským raketám.

Program - kazeta (standart i TURBO 2ØØØ) nebo disk

Po natažení programu do paměti počítače se objeví menu.

Volby: OPTION - obtížnost
SELECT - počet hráčů
START - start hry
klávesy 1-7 - výběr mise

Obtížnost: ARCADE - nemá úplnou simulaci letu, jen ukazuje, jak pracují systémy letounu - vhodné pro hráče s menším IQ
ROOKIE, PILOT, ACE - plná simulace

Volba misi: Program obsahuje sedm bojových úkolů. Po splnění každého úkolu (tj. zničení alespoň všech hlavních nepřátelských základen - černý čtvereček s bílým uprostřed) se musíte vrátit zpět na svou základnu (bílý čtvereček). Poté obdržíte další úkol.

Po stisku klávesy START nutno zadat kód ke spuštění všech systémů F-15. Kód určíte podle čísla v závorce.

Přikládáme klíč:

Ø=G	4=I	8=L	12=C
1=A	5=B	9=H	13=F
2=I	6=C	1Ø=A	14=K
3=G	7=H	11=P	15=E

Způsob ovládání:

- joystick - pohyb nahoru/dolů, vlevo/vpravo
  - FIRE button salva z kanonu, odpálení střel nebo svrženípum podle zadaného příkazu z klávesnice
- klávesnice (abecedně):

A... zvýšení tahu motoru
B... příprava ke spuštění pumy
D... odhození přídavných palivových nádrží
E... ochrana proti raketám naváděným radarem
F... ochrana proti samonaváděcím infra-raketám
G... příprava k činnosti kanonu
M... příprava rekety středního doletu
P... pausa (stop mod)
R... přepínání rozsahu palubního radiolokátoru
S... příprava rakety krátkého doletu
X... aerodynamická brzda (účinnost max. 75%)

+

ESC... katapultáž pilota (50% naděje na přežití)

Vysvětletní anglických zpráv na obrazovce:
ALLERT AIR MISSILE... proti tobě letí infra-raketa
ALLERT SAM LAUNCH... proti tobě letí raketa naváděná radarem
BOMBS ARMED... vše připraveno k bombardování
BOMBS MISS... pumy nezasáhly určený cíl
BOMBS RELEASED... pumy jsou odhozeny
DAMAGE WARNING... poškození letadla
ECM JAMMING... zapojena ochrana proti střelám s radarovým navá-
děním
ENEMY PLANE HIT... nepřátelské letadlo zničeno
FLARE LAUNCHED... zapojena ochrana proti infra-raketám
GUN XXX... počet zbylých ran v kanónu
LONG LARGE RADAR... velký rozsah radaru
MEDIUM LARGE RADAR... střední rozsah radaru
SHORT LARGE RADAR... malý rozsah radaru
MISSILE ARMED... vše připraveno k odpálení rakety
TARGET HIT !!... cíl zasažen

K rychlé orientaci slouží i akustická kontrola, která je zastoupena dvěma typy sirén:
slabší zvuk výstražné sirény... hrozí srážka se zemí
silnější zvuk výstražné sirény... ztráta letových vlastností

Pohonnou jednotku ovládáme klávesami Ø-9, kde Ø značí 55% tah motoru, 9 100% tah.

Good Luck Taking Off!

Literatura:

Arnie Katz, Bill Kunkel: Flight Simulators For ATARI
ATARI Explorer, winter 1986
str. 6-9

Václav Němeček: Vojenská letadla V
str. 82-9Ø, 39Ø-391

Zpracoval: Jiří Hrdlička ml.

Připojení mozajkové tiskárny D100 k počítačům ATARI

Polská mozajková tiskárna D100 je dodávána v několika modifikacích, lišících se typem rozhraní. Tento popis se vztahuje k provedení s rozhraním CENTRONIX (na tiskárně je označeno nálepkou vedle propojovacího konektoru "INTERFEJS CTX").

K připojení tiskárny slouží konektory pro joysticky. Zapojení interfejsu a vše ostatní k propojení je zřejmé ze schematu a tabulky signálů na obr. 1.

Pro vlastní ovládání tiskárny je nutno do paměti počítače uložit obslužnou rutinu. Počítač pak ovládá tiskárnu pomocí standardních instrukcí LPRINT, LIST"P:" atd. Obslužné rutiny jsou zpracovány ve třech variantách:

1. Obslužná rutina ve strojovém kódu pro spolupráci s jazykem ATARI BASIC. Nahrává se do počítače po zapnutí přes START. Po nahrátí se program sám rozběhne, vyvolá ATARI BASIC a ohlásí se vypsáním READY na obrazovce.
   Aby mohl být program používán výše uvedeným způsobem, musí být typu "BOOT". Takový program si může vygenerovat každý sám pomocí Assembleru ATMAS 2. Vlastní obslužná rutina je uložena v paměti od adresy $ØØ6ØØ a generátor od adresy $6ØØØ.

   Výpis obsahu paměti obslužné rutiny:

```
0600   00 02 00 06 6E 06 A9 3C   06B0   8D F9 06 4C 2C 06 68 A5
0608   8D 02 D3 A9 FB 8D E7 02   06B8   58 85 FE A5 59 85 FF A9
0610   A9 06 8D E8 02 A9 6E 85   06C0   17 8D F6 06 A9 27 8D F7
0618   0A A9 06 85 0B 18 60 2B   06C8   06 A2 00 A1 FE 29 7F C9
0620   06 42 06 3F 06 42 06 3F   06A0   60 B0 02 69 20 20 CA 06
0628   06 3F 06 01 A9 30 8D 02   06A8   E6 FE D0 02 E6 FF CE F7
0630   D3 A9 FF 8D 00 D3 A9 34   06B0   06 10 EB A9 0D 20 CA 06
0638   8D 02 D3 A9 80 8D 00 D3   06B8   CE F6 06 10 D7 60 48 41
0640   A0 01 60 C9 9B D0 1D AD   06C0   4E 53 20 57 41 47 4E 45
0648   FA 06 8D F9 06 CE F8 06   06C8   52 20 AC 11 D0 D0 FB A0
0650   10 0D A9 0C 20 64 06 EE   06D0   80 09 80 8D 00 D3 29 7F
0658   F9 06 A9 41 8D F8 06 EE   06D8   EA EA EA EA EA EA 8D 00
0660   F9 06 A9 0A 20 CA 06 CE   06E0   D3 EA EA EA EA EA EA 09
0668   F9 06 F0 DB D0 D2 A9 1F   06E8   80 8D 00 D3 EA EA EA EA
0670   8D 1B 03 A9 06 8D 1C 03   06F0   EA EA 8C 00 D3 60 17 27
0678   A9 41 8D F8 06 AD FA 06   06F8   00 F5 FF 00 00 00 00 00
```

Výpis obsahu paměti generátoru:

```
6000    A2 10 A9 03 9D 42 03 A9
6008    08 9D 4A 03 A9 80 9D 4B
6010    03 A9 4A 9D 44 03 A9 60
6018    9D 45 03 20 56 E4 30 29
6020    A9 0B 9D 42 03 A9 00 9D
6028    44 03 A9 06 9D 45 03 A9
6030    FB 9D 48 03 A9 00 9D 49
6038    03 20 56 E4 30 0B A9 0C
6040    9D 42 03 20 56 E4 30 01
6048    00 00 43 3A 9B 00 00 00
```

Postup při generaci nahrávky programu je následující:

a/ Pomocí příkazu "C" v režimu MONITOR se zapíše program na uvedené adresy

b/ Magnetofon se zapne do režimu nahrávání

c/ Spustí se generátor příkazem GOTO 6ØØØ

Ten, kdo chce používat příkazů pro tisk i v programu ATMAS 2, provede inicializaci programu příkazem GOTO Ø66E.

2. Další možnost uložení obslužné rutiny do paměti je pomocí krátkého programu v BASICu. Tato verze je určena pro ATARI BASIC. Program se do počítače nahraje, nebo napíše. Rozběhne se příkazem RUN a po ohlášení READY již tiskárna reaguje na příkazy LPRINT, LIST"P:" atd. Vlastní rutina je uložena v paměti na 6. stránce (od adresy ØØ6ØØ). Proto je možno po rozběhnutí programu BASICovou verzi programu zrušit.

Výpis programu pro ATARI BASIC:

```
1 RESTORE 10:POKE 5378,255
2 FOR I=1536 TO 1715:READ A:POKE I,A:NEXT I:POKE 7
95,31:POKE 796,6:POKE 1791,65:POKE 5377,255
10 DATA 0,2,0,6,110,6,169,60,141,2
11 DATA 211,169,2,141,231,2,169,7,141,232
12 DATA 2,169,110,133,10,169,6,133,11,24
13 DATA 96,43,6,66,6,63,6,66,6,63
14 DATA 6,63,6,1,169,48,141,2,211,169
15 DATA 255,141,0,211,169,52,141,2,211,169
16 DATA 128,141,0,211,160,1,96,201,155,208
17 DATA 29,173,2,21,141,1,21,206,255,6
```

```
18 DATA 16,13,169,12,32,100,6,238,1,21
19 DATA 169,65,141,255,6,238,1,21,169,10
20 DATA 32,134,6,206,1,21,240,219,208,210
21 DATA 169,31,141,27,3,169,6,141,28,3
22 DATA 169,65,141,255,6,173,2,21,141,1
23 DATA 21,76,44,6,172,17,208,208,251,160
24 DATA 128,9,128,141,0,211,41,127,234,234
25 DATA 234,234,234,234,141,0,211,234,234,234
26 DATA 234,234,234,9,128,141,0,211,234,234
27 DATA 234,234,234,234,140,0,211,96,23,39
```

3. TURBO BASIC je natolik zajímavý, že rutina k ovládání tiskárny byla modifikována i pro tento jazyk. Postup při nahrávání je následující:

a/ Nahraje se TURBO BASIC

b/ Napíše se (nebo nahraje) rutina podle výpisu

c/ Program rutiny se rozběhne příkazem RUN a po ohlášení READY je počítač připraven ovládat tiskárnu pomocí standardních příkazů

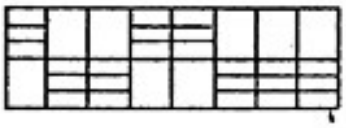
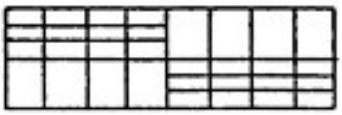
d/ Rutina je opět uložena na 6. stránce v paměti počítače, proto je možno program v BASICu zrušit

Výpis programu pro TURBO BASIC:

```
1 DIM A$(2):RESTORE 10
2 FOR I=$0600 TO $06B3:READ A$:POKE I,DEC(A$):NEXT
 I:DPOKE 795,1567:DPOKE 1791,65345:POKE 1793,255
10 DATA 00,02,00,06,6E,06,A9,3C,8D,02
11 DATA D3,A9,02,8D,E7,02,A9,07,8D,E8
12 DATA 02,A9,6E,85,0A,A9,06,85,0B,18
13 DATA 60,2B,06,42,06,3F,06,42,06,3F
14 DATA 06,3F,06,01,A9,30,8D,02,D3,A9
15 DATA FF,8D,00,D3,A9,34,8D,02,D3,A9
16 DATA 80,8D,00,D3,A0,01,60,C9,9B,D0
17 DATA 1D,AD,01,07,8D,00,07,CE,FF,06
18 DATA 10,0D,A9,0C,20,64,06,EE,00,07
19 DATA A9,41,8D,FF,06,EE,00,07,A9,0A
20 DATA 20,86,06,CE,00,07,F0,DB,D0,D2
21 DATA A9,1F,8D,1B,03,A9,06,8D,1C,03
22 DATA A9,41,8D,FF,06,AD,01,07,8D,00
23 DATA 07,4C,2C,06,AC,11,D0,D0,FB,A0
24 DATA 80,09,80,8D,00,D3,29,7F,EA,EA
25 DATA EA,EA,EA,EA,8D,00,D3,EA,EA,EA
26 DATA EA,EA,EA,09,80,8D,00,D3,EA,EA
27 DATA EA,EA,EA,EA,8C,00,D3,60,17,27
```

Rady pro ovládání a užívání tiskárny D100

1. Poloha přepínačů v tiskárně (na plošném spoji)

K100   K201

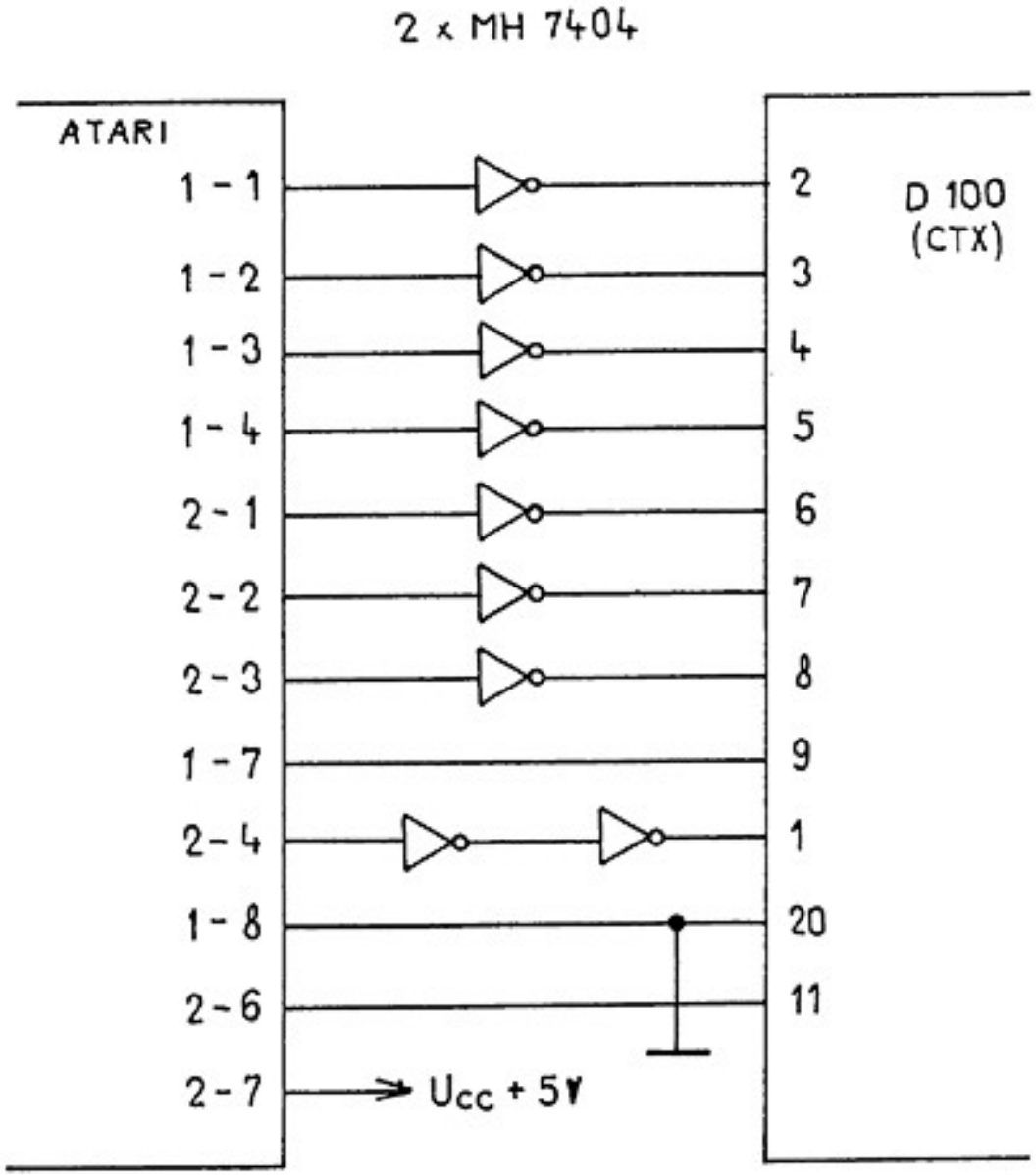
2. Počet znaků na řádek je maximálně 80. Tento je možno omezit příkazem

   - v ATARI BASICu POKE 5377,N : POKE 5378,N
   - v TURBO BASICu POKE 1792,N : POKE 1793,N

   kde N je maximální počet znaků na řádek (N 80).

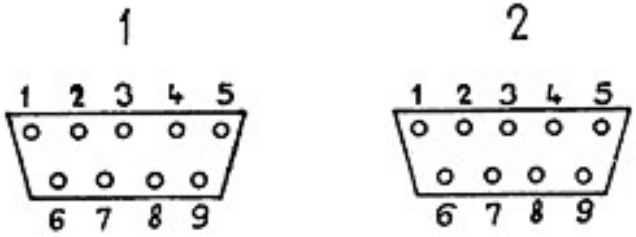
3. Typ písma a druh tisku se ovládá instrukcí LPRINT ..... (viz následující tabulka):

Instrukce	Funkce
LP."CTRL N"	tisk znaků dvojité šířky
LP."CTRL T"	zrušení tisku znaků dvojité šířky
LP."ESC ESC ["	tisk znaků dvojité výšky
LP."ESC ESC \ "	tisk znaků dvojité výšky a šířky
LP."ESC ESC 7"	jednosměrný tisk (z leva do prava)
LP."ESC ESC 3"	zrušení tisku dvojité výšky,dvojité výšky a šířky a jednosměrného tisku
LP."ESC ESC E"	tisk znaků dvojité intenzity
LP."ESC ESC F"	zrušení tisku znaků dvojité intenzity
LP."ESC ESC G"	tisk tučných znaků
LP."ESC ESC H"	zrušení tisku tučných znaků
LP."ESC ESC 1"	přímé ovládání jehel tiskárny
LP."ESC ESC 6"	tisk semigrafických znaků
LP."ESC ESC 5"	zrušení tisku semigrafických znaků a přímého ovládání jehel
LP."CTRL O"	husté písmo (16,5 zn/palec)
LP."CTRL R"	zrušení tisku hustého písma
LP."ESC ESC Ø"	hustota tisku 10 řádků/palec
LP."ESC ESC 2"	hustota tisku 6 řádků/palec
LP."ESC ESC 4"	hustota tisku 55 řádků/palec



SIGNÁL	KONEKTOR	
	D 100	ATARI
D0	2	1-1
D1	3	1-2
D2	4	1-3
D3	5	1-4
D4	6	2-1
D5	7	2-2
D6	8	2-3
D7	9	1-7
STROBE	1	2-4
BUSY	11	2-6
GND	20	1-8

ČÍSLOVÁNÍ KONEKTORŮ
(POHLED NA POČÍTAČ)



OBR. 1 – SCHEMA ZAPOJENÍ A TABULKA SIGNÁLŮ

4. Po použití tlačítka RESET je možno v BASICu znovu inicializovat obslužný program příkazem:
   POKE 795,31:POKE 796,6

Ing. Jaromír Chocholáč, Petr Pánek
AK Rožnov

OPRAVA REDAKCE :

Omlouváme se za chybičku, která se vloudila do našeho zpravodaje 5/1987. Na straně 7 v drobných programátorských tricích v části Vymazání sekvence řádků programu od-do, byl vytištěn programovací řádek v neúplném znění. Omlouváme se všem čtenářům a otiskujeme správnou verzi:

```
425 ? "¶ ":F.I=352 TO 425:POS.2,2:? I: ? "N.I:POKE 842,12":
    POS.Ø,Ø: POKE 842,13: STOP
```

- rk -

Zpráva o koordinační poradě.

Ve dnech 28. a 29.11.1987 se uskutečnila v Liberci koordinační porada pod patronací ČSVTS zástupců Atari klubů z celé ČSSR - mimo zástupce z Ostravska.
Znovu se projednávalo územní rozdělení a způsob spolupráce, hlavně výměně programů a tiskovin. Bylo doporučeno zachovat stávající strukturu krajských (oblastních) důvěrníků, kteří budou zajišťovat výměnu programů a distribuci dostupné literatury v rámci svého obvodu. Tento systém se již osvědčil mimo Prahu, kde působí jednolitý celek a Brno, kde jsou tři kluby, které dosud k sobě nenašly cestu. O Ostravě jsme nebyli informováni. Důvěrníci jsou ve styku s vedoucími klubů ve svém obvodu, a které projevili o spolupráci zájem. Aby bylo možno ještě tuto spolupráci zlepšit a případně zapojit nově vzniklé kluby uvádíme jejich adresy:

Atari klub Praha ZO Svazarm
Václav Dostál
PS 51, Praha 10 100 000

Adam Miloš
Kr. Údolí 138,364 44 Ústina

Okr. komunální podnik
Minčic Jiří
Nám.bojovníků za mír 2
460 54 Liberec

Atari klub ZO Svazarm
Ing. Dobromil Pavlík
PS 137
772 11 Olomouc

Bartošík Jiří
Komenského 1653
686 02 Uh. Hradiště

Ing. Jar. Burjaniv
ČSA 24
935 21 Tlmače

Ing. Pavol Dorsic
Rajčianská 6
821 07 Bratislava

Ing. Jan Kodera
Hůrka 1053
278 01 Kralupy n.V.

Ing. Stanislav Kukral
Manchysova 1409
500 06 Hradec Králové

Dále jsme si vyměnili informace o vydaných publikacích - doposud bylo vydáno: Uživatelská příručka, Atari Basic, ABC poč. Atari, Macr Assembler, Sinfile, DOS 2,5, Atari Writer, Váš Atari, Průvodce Assemblerem, Logo, Speedscript 3,0, Speedcalk, Mikroprocesor 6502, Medit-Meditcom, Startextr;

V příštím roce budou vycházet tři Zpravodaje:

Pražský - který je určen pro členy pražského AK (cca 2000 členů) 6x ročně s odbornou a interní tematikou.

Olomoucký - který je určen pro ostatní zájemce ze samostatných klubů (bude rozšiřován na základě objednávek a složení příspěvku na vydání, přes krajské důvěrníky), bude nadále obsahovat návody k programům - náklad 4000 ks, 64 stran 5x ročně.

Tlmačskobratislavský - bude monotematický, první číslo bude o grafice, další o hudbě a pod. 64 stran.

V současnosti je v tisku:

Textové mody -	Hodonín, náklad 1500 ks
Grafické mody -	Hodonín
Kyan Pascal -	Hodonín, 120 str.
Atari kouzlí -	Olomouc, náklad 1500 ks, 100 str.
Učeb. Assembleru -	Praha, 2 díly, 305 stran
Logo -	Praha, 100 str.
Mac 65 -	Praha
Intern -	Praha
Učebnice Basic -	Plzeň
Prog. jaz. LISP -	Bratislava
Assembler -	Tlmače, 2 díly

V přípravě jsou překlady Super Buch, Treining Buch.

V Liberci se bude připravovat Kniha návodů k hrám, obracíme se na všechny, kteří návody vlastní, aby je nabídli ke zveřejnění na adresu Libereckého tisk. mluvčího, jež je uveden dále.

Při jednání byli stanovení tiskoví mluvčí klubů, které vydávají literaturu a na tyto směřujte své nabídky příspěvků, poř. publikací ke zveřejnění. Dohodli jsme se, že tito mluvčí budou mezi sebou úzce spolupracovat ke zlepšení informovanosti o tom, co se připravuje k vydání, aby dále nedocházelo k duplicitnímu vydávání a tím k tříštění našich sil.

Adresy tiskových mluvčí klubů:

Ing. Ladislav Gal
Rajská 1
811 00 Bratislava

Ing. Dobromil Pavlík
Albertova 21
779 00 Olomouc

JUDr. Jan Hlaváček
Atari klub Ps 51
Praha 10 100 78

Ing. Josef Januška
Tyršova 4
695 01 Hodonín

Ing. Miroslav Hariaš
Olbrachtova 514
460 05 Liberec 15

Tiskoví mluvčí ostatních klubů:
Adam Miloš, 364 66 Útvina, Krásné Údolí 138
Bartošík Jiří, 685 02 Uh. Hradiště, Komenského 1653
Dirniska Jan, 040 01 Košice, Lidické nám. 1
Josef, 293 01 Mladá Boleslav, ČSSP 1100
Ing. Hrabovský Miroslav, 600 00 Brno,
Ing. Kukral Stanislav, 500 06 Hradec Králové, Mandysova 1409
Ing. Příhoda Karel, 141 00 Praha Spořilov II, Střimelická 2503

Na závěr jednání zástupce Ol. klubu vystoupil s návrhem na vytváření specializovaných skupin. Proto nabízíme prostor v našem Zpravodaji k sprostředkování výzev zájemců o spolupráci na určitém úseku využití počítačů. Po zveřejněné výzvě mohou vzniknout skupiny, které se budou dle své profese zabývat využitím HC v různých oborech, např. lékařství, školství, statistice, účetnictví, konstruování, hudbě apod.

Žádáme zájemce o vedení, či ustavení určité skupiny, aby nám napsali a my jeho výzvu zveřejníme.

Na vzor první výzva:

1. Zájemci o využití Atari v genealogii, ať napíší na adresu,
   L. Podmolík, Kaštanova 16, 772 00 Olomouc

Myslíme si, že by bylo také vhodné, kdyby se utvořila skupina programátorů pro vytváření programů, dle požadavku uživatelů z řad členů (specializovaných skupin uživatelů) nebo z řad podniků a institucí. Napište, zda byste měli o tuto činnost zájem - jak o programování, tak i o zadání požadavku na tvorbu programů.

O B S A H Z P R A V O D A J E 1-2/1988

ZPRAVODAJ ATARI KLUB 1-2/1988

Vydává ATARI KLUB Olomouc
NEPRODEJNÉ - odběr vázán na příspěvek 30 Kčs
Vychází v nákladu 4.000 kusů
Odpovědný redaktor: Ing. Kopečný Pavel
Odborný redaktor : Ing. Pavlík Dobromil
Neprošlo jazykovou úpravou
Přetisk pouze se souhlasem redakce
Předáno do tisku: prosinec 1987
Tisk SNV Praha zak.č. 33/88
Tisk povolen OK ONV Olomouc č.j. 0380500387