ŠKOLNÍ MIKROPOČÍTAČ

IQ 151

SPN

Ing. Pavel Přívětivý

ŠKOLNÍ MIKROPOČÍTAČ

IQ 151

Státní pedagogické nakladatelství, Praha 1988

Lektorovali RNDr. Antonín Vrba, CSc., a Marko Genyk-Berezovskyj

Vydalo Státní pedagogické nakladatelství, n. p., jako účelový náklad pro potřebu ministerstva školství ČSR

1. vydání
Publikace neprošla jazykovou ani redakční úpravou v redakci nakladatelství

Předmluva

Příručka stručně popisuje základní funkce školního mikropočítače IQ 151, jeho programového vybavení a přídavných zařízení. Příručku není možno považovat za učebnici programování. Je určena především učitelům a žákům základních a středních škol jako pomůcka při programování a praktické práci s počítačem IQ 151 především v jazyce BASIC.

Pro dobré pochopení několika bodů této příručky je potřebná znalost mikroprocesoru MHB 8080 a jeho instrukčního souboru. Stručný popis tohoto mikroprocesoru je uveden v příloze 9 a popis jeho instrukčního souboru v příloze 10.

Text vznikl důkladným přepracováním a doplněním publikace "IQ 151" vydané v r. 1985 ve Školním výpočetním středisku při SPŠE v Pardubicích.

Můj upřímný dík patří A. Vrbovi a M. Berezovskému, kteří mně při psaní této příručky přispěli řadou cenných připomínek. Za pomoc se zpracováním čistopisu příručky dále děkuji H. Moravcové, J. Jehličkové a A. Vojtkové.

Pardubice, srpen 1987 Pavel Přívětivý

OBSAH

1 POPIS POČÍTAČE IQ 151 A JEHO OBSLUHA

1.1 Popis počítače

IQ 151 je stolní počítač s mikroprocesorem MHB 8080A. Standardně je vybaven pamětí RWM RAM s kapacitou 32 Kbytů. Klávesnice je membránová. Na panelu klávesnice je umístěna kontrolka zapnutého stavu. Síťový vypínač je na levém boku, kde jsou také konektory pro připojení televizoru (popřípadě zobrazovacího monitoru) a magnetofonu. Na zadní straně je prostor s konektory pro zasunutí pěti přídavných modulů. Počítač je spotřebič I. izolační třídy a musí být připojován na zásuvku s uzemňovacím kolíkem.

Základní konfiguraci tvoří počítač a televizor. Televizor propojíme s počítačem koaxiálním kabelem tak, že konektor označený symbolem televizoru (první zleva) spojíme s anténním vstupem televizoru. Nemá-li televizor koaxiální vstup, použijeme přizpůsobovací člen. Je-li u televizoru vstup VIDEO, propojíme jej koaxiálním kabelem s konektorem označeným symbolem zobrazovacího monitoru (druhý zleva). Dostaneme tak lepší obraz než při propojení s anténním vstupem televizoru.

Do dvou z pěti konektorů počítače (pro přídavné moduly) zasuneme moduly VIDEO 32 a BASIC 6 (pozor na natočení modulů - deska plošného spoje má být na levé straně modulu, blíže ke zdroji), zapneme televizor a nakonec počítač. Televizor naladíme na kmitočet výstupního signálu počítače (10. až 12. kanál). Na obrazovce se na tmavém podkladě objeví v levém horním rohu světlý nápis

BASIC
READY

a blikající čtvereček, tzv. kurzor. Počítač je připraven přijímat příkazy z klávesnice. Pokud nechceme používat jazyk BASIC, není nutné, aby modul BASIC 6 byl zasunut do počítače. Jestliže tento modul není zasunut, potom se při zapnutí počítače přihlásí MONITOR a na obrazovce se objeví znak > (viz kapitola 4).

Dále lze k počítači připojit magnetofon. Ten se připojuje do konektoru s označením M1 (na tento konektor je vyveden vstup i výstup signálu pro magnetofon). Pokud potřebujeme pouze nahrávat na magnetofon, můžeme použít konektor s označením M2. Propojovací kabel může mít pěti- nebo tříkolíkové konektory. Stereomagnetofon je výhodné používat v režimu MONO.

Se standardním vybavením může počítač s uživatelem komunikovat v režimu MONITOR na úrovni strojového jazyka nebo v režimu BASIC na úrovni rozšířené verze jazyka BASIC.

Počítač nemůže poškodit žádná manipulace s klávesami. Potřebujeme-li se dostat do počátečního stavu, stiskneme červené tlačítko označené RES.

1.2 Údržba počítače

Počítač IQ 151 nemá nároky ani na stabilizované napájecí napětí, ani na klimatizaci. Chraňte počítač před vlhkostí, účinky chemikálií, vysokými a nízkými teplotami, prachem a před nárazy. Dbejte na to, aby se nevhodnými zásahy (např. odložením papíru na větrací otvory) nezmenšila účinnost přirozeného chlazení. Na žádný z konektorů se nesmí dostat síťové napětí (např. z televizoru nebo magnetofonu), jinak dojde k rozsáhlému poškození počítače. Přídavné moduly zasouvejte a vysouvejte z konektorů jen při vypnutém počítači, jinak hrozí poškození počítače a modulů. Jakoukoli manipulaci provádějte s citem. Nedotýkejte se konektorů modulů. Při ukládání modulů mimo počítač zakryjte špičky konektorů vodivým molitanem. Chráníte tím integrované obvody před statickou elektřinou.

1.3 Zobrazování informací na obrazovce

1.3.1 Obrazovka

Prostor obrazovky je rozdělen do 32 řádků a 32 sloupců. Vzniká tak 32 x 32 = 1024 pozic pro zobrazování znaků (tj.písmen, číslic, atd.).

Řádky i sloupce se číslují od 0 do 31; řádky se číslují shora dolů, sloupce zleva doprava. Zobrazování údajů probíhá standardně pouze na prvých 31 řádcích (tj. na řádcích č. 0-30). Počet řádků přístupných pro zobrazování údajů lze měnit v rozmezí 0-31 (viz bod 4.3).

1.3.2 Kurzor

V prostoru obrazovky se pohybuje tzv. kurzor, který zaujímá vždy jednu pozici pro zobrazení znaku. Pozice kurzoru určuje místo, kam se zobrazí další údaj, který vystoupí na obrazovku. Kurzor je na obrazovce stále, v přímém režimu je indikován blikajícím čtverečkem, během provádění programu zhasne. Pozice kurzoru se mění buď automaticky (např. po stisknutí černé klávesy v přímém režimu) nebo ji může uživatel různým způsobem měnit (viz bod 1.4 nebo 2.15).

1.3.3 Zobrazování znaků

Na každou z 1024 pozic na obrazovce lze zobrazit libovolný z 256 znaků, které jsou na IQ 151 k dispozici. Další, tzv. řídicí znaky, se na obrazovce nezobrazují (viz bod 1.7 a příloha 1). Při zobrazování znaku na pozici, kde už je nějaký znak zobrazen, je starý znak smazán a nahrazen novým. Toto se ve skutečnosti děje vždy, protože na pozicích, kde "není nic", je vesměs zobrazen znak mezera.

1.3.4 Režimy zobrazování

Zobrazování znaků, údajů, čísel, slov atd. může probíhat ve třech režimech:

1) Normální režim. V tomto režimu se většinou s počítačem pracuje a počítač je na něj nastaven automaticky po zapnutí. V normálním režimu lze zobrazovat malá a velká písmena anglické abecedy, číslice, interpunkční znaménka a další znaky na klávesnici. V tomto režimu nelze zobrazovat grafické znaky. Znaky se zobrazují světlé na tmavém pozadí.

2) Grafický režim. V tomto režimu se zobrazují grafické znaky, nelze zobrazovat ostatní znaky.
3) Inverzní režim. V inverzním režimu se zobrazují tmavé znaky na světlém pozadí. Inverzní režim lze kombinovat s normálním i s grafickým režimem.

O přepínání jednotlivých režimů viz bod 1.4.

1.3.5 Znak CR

Znak CR se na obrazovce zobrazuje stejně jako znak mezera, přestože má odlišnou funkci. Můžeme si myslet, že na obrazovce je, ale "není vidět". Znak CR lze zobrazit v každém režimu a vždy se zobrazí normálním způsobem (tj. tmavá mezera).

1.4 Klávesnice

Klávesnice má 49 černých, 15 bílých, 6 šedých a 1 červené tlačítko. Převážná část černých tlačítek má v kombinaci s ostatními tlačítky až 6 funkcí: velké písmeno, malé písmeno, klíčové slovo, identifikátor funkce, grafický a řídicí znak.

K volbě funkce slouží šedá tlačítka:

SH - malá písmena nebo horní znaky,
FA - klíčová slova (nad tlačítkem) ,
FB - identifikátor funkce (pod tlačítkem),
CTRL - řídicí znaky.

Tlačítko volby je nutno držet stisknuté v okamžiku stisknutí příslušného černého tlačítka. Řídicí znaky se zapisují stisknutím tlačítka CTRL a současným stiskem tlačítka určujícího řídicí znak. Nejdůležitější řídicí znaky:

CTRL O - přepnutí do grafického režimu,
CTRL N - přepnutí z grafického do normálního režimu,
CTRL S - přepnutí do inverzního režimu,
CTRL R - přepnutí z inverzního do normálního režimu,
CTRL - přerušení výpisu nebo běhu programu (výpis nebo běh pak - pokračuje po stisknutí libovolné černé klávesy),

CTRL A - ignorování napsaných znaků od posledního CR (totéž jako tlačítko F1),
CTRL B - zablokování klávesnice (je ignorováno stisknutí tlačítek), opětným použitím se klávesnice odblokuje (totéž jako tlačítko F2),
CTRL C - při přerušení výpisu nebo běhu programu je výpis nebo běh ukončen (totéž jako tlačítko F3),
CTRL G - pípnutí,
CTRL [- ukončení režimu AUTO (viz bod 2.13).

Funkce dalších tlačítek:

RES - uvedení počítače do počátečního stavu (přesněji řečeno smazání obrazovky a spuštění operačního systému od začátku),
SP - mezera,
CR - předání napsaných znaků ke zpracování počítačem, přechod na nový řádek,
BR - předání řízení MONITORu (v režimu BASIC doporučujeme raději použít příkazu BYE, návrat z MONITORu do BASICu se provede příkazem R),
F1 - ignorování napsaných znaků (totéž .jako CTRL A),
F2 - zablokování klávesnice (je ignorováno stisknutí tlačítek), opětným stisknutím se klávesnice odblokuje (totéž jako CTRL B),
F3 - při zastavení výpisu nebo běhu programu je výpis nebo běh ukončen (totéž jako CTRL C),
IC - část řádku od kurzoru doprava se odsune o jedno místo doprava, čímž vznikne místo pro vložení znaku,
DC - znak, na němž stojí kurzor, se zruší a část řádku od kurzoru doprava se přisune o jedno místo doleva,
 - pohyb kurzorem nahoru,
 - pohyb kurzorem dolů,
 - pohyb kurzorem doleva,
 - pohyb kurzorem doprava,
 - přemístění kurzoru do levého horního rohu.

Tlačítka F4, F5, IL a DL nemají při standardním programovém vybavení žádnou funkci.

Stisknutí tlačítek CR nebo F1 způsobí také návrat z grafického nebo inverzního do normálního režimu.

Držíme-li tlačítko stisknuté delší dobu, je příslušný znak generován opakovaně.

Je-li počítač ve stavu, kdy očekává informaci (příkazy, vstup dat) z klávesnice, ukládají se znaky generované stisknutím tlačítek do tzv. vyrovnávací paměti a většina se jich zobrazuje na obrazovce. V režimu BASIC se do vyrovnávací paměti vejde 80 znaků (včetně řídících znaků, které se na obrazovce nezobrazují).

1.5 Psaní a editace programů

1.5.1 Zápis programu

Program v jazyku BASIC se zapisuje do tzv. programových řádků. Délka programového řádku je max. 80 znaků. Každý programový řádek začíná číslem řádku, pak následuje jeden nebo více příkazů jazyka BASIC a řádek je ukončen znakem CR. Je-li na jednom programovém řádku více příkazů, musí být odděleny dvojtečkou. Mezi číslo řádku a příkaz (příkazy) vkládá počítač automaticky mezeru. Programové řádky se číslují celými čísly z intervalu <0,65529> ve vzestupném pořadí. Nejčastěji se řádky číslují po desítkách, aby bylo možné při ladění a opravách vkládat do programu další řádky. Automatické číslování programových řádků umožňuje příkaz AUTO (viz bod 2.13).

1.5.2 Vkládaní nového programového řádku

Napíšeme číslo řádku, příkaz (resp. příkazy) a odešleme tlačítkem CR. Řádek se automaticky zařadí do programu na příslušné místo tak, aby bylo zachováno vzestupné číslování programových řádků (např. v programu s čísly řádků 15, 20, 25 a 30 se další řádek s číslem 17 zařadí mezi řádky 15 a 20).

1.5.3 Vymazání programového řádku

Napíšeme číslo řádku, který chceme z paměti vymazat a odešleme tlačítkem CR. Řádek s tímto číslem se z paměti vymaže. Pro vymazání více řádků lze někdy s výhodou použít příkaz AUTO (viz bod 2.13).

1.5.4 Editace programového řádku

Při úpravě daného programového řádku lze postupovat dvěma způsoby:

A) Řádek (i s jeho číslem) napíšeme znovu v požadované podobě a odešleme tlačítkem CR. Starý řádek se v paměti počítače nahradí novým.

B) a) Pokud řádek, který chceme editovat, není zobrazen na obrazovce, necháme jej vystoupit pomocí příkazu LIST.

b) Pomocí tlačítek [↓] a [↑] najedeme kurzorem na začátek daného programového řádku a pomocí tlačítka [→] najedeme na místo, kde chceme provádět změny.

c) Při úpravách můžeme znaky, příkazy, atd. libovolně přepisovat, vkládat nové a vypouštět staré pomocí tlačítka IC (resp. DC) a pohybovat kurzorem pomocí tlačítek [→] a [←] .

d) Po skončení úprav dojedeme kurzorem na konec programového řádku a odešleme jej tlačítkem CR. Editovaný řádek se uloží na příslušné místo do paměti počítače.

1.5.5 Důležité poznámky k editaci

1) Při provádění změn v programovém řádku není vhodné používat tlačítek [↑] , [↓] , [↖] . Řádek by se mohl chybně uložit do paměti.
2) Pokud při provádění úprav změníme i číslo řádku, vytvoříme tím vlastně nový programový řádek a původní zůstane v paměti počítače beze změny.
3) Grafické znaky nelze přejet kurzorem pomocí tlačítka [→] . Je třeba napsat je znovu. Přejedeme-li pomocí tlačítka [→] inverzní znaky, změní se na normální, musíme proto před přejetím přepnout do inverzního režimu a po přejetí zase přepnout zpět.

4) Znak CR je součástí programového řádku, a proto nemusíme v bodě d) řádek odesílat pomocí CR. Přejedeme-li kurzorem konec řádku včetně znaku CR, řádek se uloží do paměti automaticky. Protože ale znak CR je "neviditelný", může to v některých případech způsobit pochybnost, zda se řádek skutečně uložil. Varianta d) je (pro začátečníky) vhodnější.
5) Při ukončování výpisu programu je třeba tlačítko F3 resp. CTRL C podržet stisknuté jen krátce. Jinak se příslušný řídicí znak vygeneruje vícenásobně, první z nich způsobí zastavení výpisu a zbylé znaky pak počítač interpretuje jako začátek dalšího řádku. Proto po ukončení editace programového řádku ohlásí počítač chybu 21 a editovaný řádek do paměti neuloží. V tom případě stačí přejet kurzorem znovu celý řádek a odeslat CR.
6) Některé změny v programu při editaci se na obrazovce neprojevují bezprostředně (např. při vymazávání řádků), je proto vhodné přesvědčit se vždy o provedených zásazích vypsáním příslušné partie programu.

1.6 Magnetofon

Magnetofon (kazetový nebo cívkový) připojený k počítači IQ 151 slouží jako vnější paměť. Na magnetofonové pásce můžeme archívovat vytvořené programy.

Při přehrávání programu z paměti počítače na magnetofonovou pásku postupujeme takto:

a) Pokud to magnetofon umožňuje, nahrajeme na pásek hlasové záhlaví, tj. akustickou informaci o programu.
b) Spustíme magnetofon přepnutý do režimu nahrávání a napíšeme na klávesnici příkaz pro vytvoření nahrávky (v režimu BASIC příkaz MSAVE - viz bod 2.13, v režimu MONITOR příkaz W - viz bod 4.2) a stiskneme tlačítko CR.

Po skončení nahrávání vypíše počítač na obrazovce text READY (v režimu BASIC) nebo znak > (v režimu MONITOR).

Při přehrávání programu z magnetofonového pásku do paměti počítače postupujeme takto:

a) Najdeme začátek příslušného programu (hlasové záhlaví) na magnetofonovém pásku a zastavíme magnetofon.
b) Na klávesnici napíšeme příkaz MLOAD (v režimu BASIC) nebo příkaz L (v režimu MONITOR) a spustíme magnetofon.
c) Až uslyšíme pískání, tzv. pilotní kmitočet, stiskneme tlačítko CR.

Příkazy čtené z magnetofonu se zobrazují na obrazovce. Jestliže se program nezobrazuje na obrazovce, neukládá se ani do paměti počítače. Příčinou může být to, že u některých. typů magnetofonů je opačná polarita nahrávaného signálu, což lze napravit v režimu BASIC odesláním příkazu

```
POKE 27,214
```

před přehráváním (proměnná DOBSE - viz bod 4.3).

O dalších možnostech práce s magnetofonem viz bod 2.13 a kapitola 4.

Poznámka:

Program je na magnetofonovou pásku nahráván po blocích (v režimu BASIC je blok tvořen vždy jedním řádkem programu). Mezi bloky jsou na pásce meziblokové mezery. Délka těchto mezer je určena hodnotou proměnné BTIME (viz bod 4.3). Při přehrávání programu z magnetofonového pásku do paměti počítače musí být řádek programu (z každého bloku), během následující meziblokové mezery, zpracován a uložen na správné místo do paměti. Doba potřebná na zpracování a uložení řádku programu závisí také na délce programu. Z tohoto důvodu je nutno při vytváření magnetofonové nahrávky (u dlouhých programů) zvětšit mezi blokovou mezeru. Pokud toto neprovedeme bude při přehrávání program zkomolen.

1.7 Kód znaků

Počítač IQ 151 zpracovává znaky v kódu ASCII (American Standard Code for Information Interchange). Tento kód je uveden v příloze 1. Při používání tohoto kódu je nutno rozlišovat písmeno O od číslice nuly 0. Mezeru označujeme v tomto textu ␣ .

2 PROGRAMOVACÍ JAZYK BASIC 6

2.1 Úvod

Programovací jazyk BASIC vznikl v roce 1965. Během své existence se tento jazyk vyvíjel a nyní se používá i pro programování kalkulátorů a minipočítačů. Jeho verze se na jednotlivých počítacích liší. Název BASIC je vytvořen z prvních písmen jeho názvu "Beginner's All - Purpose Symbolic Instruction Code", což je možno volně přeložit jako všestranně použitelný programovací jazyk pro začátečníky. Je určen pro provádění středně obtížných vědecko-technických výpočtů a zpracování nepříliš rozsáhlých souborů dat.

Překladač jazyka BASIC pro počítač IQ 151 je uložen v přídavném modulu označeném BASIC 6. Je to konverzační interpretační překladač. Při programování je nutno dodržet syntaktická pravidla. Syntaktické chyby překladač odhalí a signalizuje.

2.2 Abeceda jazyka BASIC

Abeceda programovacího jazyka je konečná množina znaků povolených pro zápis programů a dat. Abecedu jazyka BASIC tvoří všechny znaky dostupné na klávesnici počítače IQ 151.

2.3 Lexikální symboly jazyka BASIC

Lexikální symbol je jazykový objekt, který je podle definice jazyka dále nedělitelný a je mu přiřazen význam. BASIC používá tyto lexikální symboly: konstanty, identifikátory, operátory, klíčová slova a oddělovače. Jsou to vesměs konečné posloupnosti znaků bez mezer. Pokud napíšeme v lexikálním symbolu jednu nebo více mezer, jsou tyto mezery překladačem ignorovány. Výjimku tvoří textové konstanty, kde překladač mezery ponechává.

2.4 Konstanty

Veličiny, jejichž hodnota se v programu nemění, se nazývají konstanty. Konstanty vystupující v jazyku BASIC můžeme dělit na číselné konstanty, které nazýváme čísla, a na textové konstanty (také se jim říká alfanumerické nebo řetězcové konstanty).

2.4.1 Číselné konstanty

Čísla používaná v jazyku BASIC jsou čísla vyjádřená v desítkové soustavě. Rozlišujeme dva tvary zápisu čísel: základní a semilogaritmický.

Číslo zapsané v základním tvaru je libovolná posloupnost číslic 0, 1 , 2, 3, 4, 5, 6, 7, 8 a 9, ve které se může vyskytnout znak desetinné tečky, sloužící pro oddělení celé části čísla od části desetinné (pro oddělení těchto částí čísla se nepoužívá čárka). Číslu může předcházet znaménko + nebo -. Znaménko + se před číslo nemusí umísťovat. V zápise čísla lze vynechat nevýznamné nuly.

Příklady čísel zapsaných v základním tvaru:

21.4	-54.0	.1	0
100	100.	-.001	+0.158

Celá čísla mohou být zapsána dvojím způsobem, a to s tečkou na konci nebo bez tečky. Oba tyto zápisy jsou rovnocenné.

Čísla zapsaná v semilogaritmickém tvaru mají tvar

xEn

kde x je číslo zapsané v základním tvaru (této části čísla říkáme mantisa),

n je celé číslo (bez tečky) z intervalu <-38,38>, které tvoří desítkový exponent. Části n může předcházet znaménko + nebo - (za písmenem E).

Zápis xEn označuje číslo $x.10^n$.

Příklady čísel zapsaných v semilogaritmickém tvaru:

Zápis čísla	Hodnota
-1E3	$-1.10^{3} = -1000$
28.3E+2	$28,3.10^{2} = 2830$
+2E-4	$2.10^{-4} = 0,0002$
-10.E-1	$-10.10^{-1} = -1$

Všechna čísla jsou v paměti počítače zobrazena v semilogaritmickém tvaru (v pohyblivé řádové čárce). Mantisa je v paměti zobrazena pomocí sedmi desítkových číslic (při výpisu čísla vystoupí maximálně 6 číslic). Číslo je při výpisu vždy normalizováno, tzn. že při výpisu čísla v semilogaritmickém tvaru je hodnota mantisy vždy v intervalu <1;9.99999>. Pokud potřebujeme hodnotu π, můžeme použít identifikátor PI. Hodnota označená tímto identifikátorem je 3,14159.

2.4.2 Textové konstanty

Textové konstanty mají obecný tvar

"$S_1S_2.....S_n$"

kde $S_1S_2.....S_n$ je posloupnost libovolných znaků, v níž se nevyskytuje znak uvozovek. Znaky uvozovek ohraničují textovou konstantu, ale nenáleží do ní, vymezují pouze její začátek a konec (výjimka z tohoto pravidla viz bod 2.11). Délka textové konstanty je určena počtem n znaků v řetězci a může být maximálně 255 znaků. Minimální délka řetězce je 0 znaků (prázdný řetězec).

Příklady textových konstant:

"BASIC"
"*17.BREZNA 1987*"

2.5 Proměnné a identifikátory

Veličiny, jejichž hodnoty se v průběhu provádění výpočtů mění, nazýváme proměnné. BASIC 6 rozeznává jednoduché a indexované proměnné. Pro označování proměnných používáme identifikátorů. Identifikátor jednoduché číselné proměnné je posloupnost velkých písmen a číslic, která nezačíná číslicí.

Identifikátor nesmí obsahovat klíčové slovo, tj. slovo, které má v jazyce BASIC 6 pevný význam (např. ON, PI, PEEK, GOSUB, ...). Pro identifikátory platí omezení délky na dva znaky (přesněji řečeno, identifikátory mohou být delší, ale počítač je rozlišuje podle prvních dvou znaků a používání delších identifikátorů může vést k omylům).

Identifikátory textových proměnných se vytvářejí podle stejných pravidel, musí však být navíc zakončeny znakem $.

Příklady správně zapsaných identifikátorů:

F, A0, BH, P9D23, AB$, AC$ ABC$

Příklady nesprávně zapsaných identifikátorů:

9R, F+, SLON, TO, K%, MEND$

Identifikátory AB$ a AC$ počítač rozliší, identifikátory AB$ a ABC$ nerozliší, tj. považuje je za identifikátory téže textové proměnné.

BASIC 6 umožňuje deklarovat číselná a textová pole. Složky pole se nazývají indexované proměnné. Obecný tvar indexované proměnné je

$id(i_1,i_2,.....,i_k)$

kde id je identifikátor pole,

$i_1,i_2,.....i_k$ jsou indexy proměnné a

k je počet rozměrů pole.

Identifikátor pole je tvořen podle stejných zásad jako identifikátory jednoduchých proměnných. V jednom programu lze použít stejného identifikátoru pro označení jednoduché proměnné a pole, ale pro pole o různých rozměrech není možno použít stejný identifikátor.

Indexy indexovaných proměnných mohou být libovolné aritmetické výrazy. Počítač považuje za hodnotu indexu celou část hodnoty aritmetického výrazu. Hodnota indexu musí ležet v intervalu <0,65535>.

Příklady:

```
X(0)
AB(2,3,5)
PR$(PR+AB(PR,PR,2))
X(X(2))
```

2.6 Aritmetické výrazy

Upozornění: Vzhledem k tomu, že BASIC 6 nerozlišuje číselné a logické hodnoty, nerozlišuje ani aritmetické a logické výrazy. Pro přehlednost však o nich pojednáváme zvlášť, viz bod 2.7. V tomto bodě slovy aritmetický výraz rozumíme pouze aritmetický výraz, v ostatních bodech používáme název aritmetický výraz pro oba typy výrazů.

Aritmetický výraz je posloupnost čísel, proměnných, funkcí, znaků aritmetických operací a kulatých závorek utvořená podle určitých pravidel. V BASICu můžeme používat následující znaky aritmetických operací, tzv. aritmetické operátory:

Název operátoru	Operátor	Příklad použití
operátor změny znaménka	-	-A
operátor součtu	+	A+B
operátor rozdílu	-	A-B
operátor součinu	*	A*A
operátor podílu	/	A/B
operátor mocniny	^	A^B (označuje A^B)

Vynechání operátoru součinu při zápisu aritmetických výrazů je nepřípustné. Nemůžeme tedy např. výraz A*B+4 nahradit výrazem AB+4. V aritmetickém výrazu nemohou být aritmetické operátory bezprostředně za sebou, např. Q/-5. Takovéto operátory musíme oddělit závorkami: Q/(-5). V tomto významu znak "-" neoznačuje operátor rozdílu, ale operátor změny znaménka. Pro oba tyto operátory se používá stejný znak. O který operátor se jedná, rozhoduje kontext, v němž je použit.

Příklady aritmetických výrazů:

Správných	Chybných
X-Y	X+-Y
(1+B2)*L^(-4)	(1+B2)*L^-4

Podobně jako v matematice o pořadí provádění operací při vyhodnocování výrazů rozhodují závorky a priorita operátorů. V jazyku BASIC platí pro výpočet číselné hodnoty aritmetického výrazu tato pravidla:

1) V prvé řadě se vypočítají hodnoty částí výrazu, které jsou v závorkách, počínaje od nejvnitřnější dvojice závorek.
2) O pořadí provádění operací pak rozhoduje priorita aritmetických operátorů. V následující tabulce jsou operátory seřazeny s klesající prioritou:

Název operátoru	Operátor
operátor mocniny	^
operátor změny znaménka	-
operátory součinu a podílu	* /
operátory součtu a rozdílu	+ -

Nejdříve se tedy provádí výpočet mocniny a operace sčítání nebo odčítání se provedou naposled.

3) V případě, kdy pravidla 1. a 2. neurčují jednoznačně pořadí provedení operací, potom se operace provádějí "od leva", tzn. operace jsou prováděny v pořadí jejich výskytu ve výrazu.

Příklad:
Výraz A/B*C obsahuje operátory součinu a podílu. Tyto operátory mají stejnou prioritu. Jelikož výraz neobsahuje závorky, bude pořadí provádění operací shodné s pořadím jejich zápisu. Tzn. nejdříve hodnota proměnné A bude vydělena hodnotou proměnné B a potom výsledek dělení bude vynásoben hodnotou proměnné C. Výraz 1/I*I má (pro $I \neq 0$) hodnotu 1, nikoliv $1/I^2$.

Příklad:
Dále je uvedeno několik výrazů a jejich význam v obvyklém matematickém značení:

4.8E3	$4,8.10^3$
4.8*E^3	$4,8.e^3$
4.8*E3^2	$4,8.(E3)^2$
7.1E+5+E	$7,1.10^5$+e
7.1+E+5+X	7,1+E+5+X

2.7 Logické výrazy

Logické hodnoty můžeme v jazyku BASIC 6 získat pomocí výrazu, který obsahuje relační operátory. Jedná se o výraz, který vyjadřuje rovnost nebo nerovnost. Tomuto výrazu budeme říkat relace. V jazyku BASIC 6 má relace následující obecný tvar:

v1 r v2

kde v1 a v2 jsou libovolné výrazy,
r je relační operátor.

V tomto bodě jsou popsány relace mezi aritmetickými výrazy. Porovnávání textových výrazů viz také bod 2.9.

Relační operátory jsou uvedeny v následující tabulce:

Relační operátor	Význam operátoru
=	rovno
<> nebo ><	nerovno
<	menší než
>	větší než
<= nebo =<	menší nebo rovno
>= nebo =>	větší nebo rovno

Příklady relací:

A<B
2*X+1>=5

V závislosti na hodnotách výrazů v1 a v2 může relace být splněna nebo nesplněna. Např. uvedené relace jsou splněny,

pokud hodnota proměnné A je menší než hodnota proměnné B a hodnota výrazu 2X+1 je menší než 5.

Relace může nabývat dvou hodnot: true (relace je splněna) nebo false (relace je nesplněna). Tyto hodnoty jsou v počítači zobrazeny jako číselné hodnoty a to takto: -1 (true), 0 (false). Relace je dále možno spojovat logickými operátory:

Název operátoru	Operátor	Příklad použití
logický součin	AND	(X=5) AND (Y=7)
logický součet	OR	(X=5) OR (Y=7)
operátor negace	NOT	NOT (Y=7)

Logické operace jsou definovány následující tabulkou:

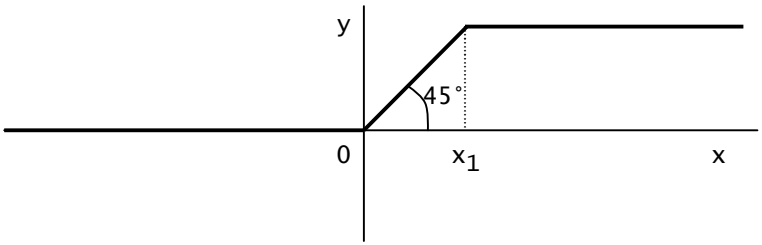
A	B	A OR B	A AND B	NOT A
false	false	false	false	true
false	true	true	false	true
true	false	true	false	false
true	true	true	true	false

Logický výraz lze použít v libovolném příkazu místo aritmetického výrazu, např.:

```
PRINT (X=3) OR (Y=4)
X=(Y=5)
```

První z těchto příkazů způsobí vypsání čísla -1 (pokud X=3 nebo je-li Y=4) nebo nuly (v opačném případě). Druhý příkaz přiřadí proměnné X hodnotu -1 (pro Y=5) nebo 0 (pro Y≠5).

Logické výrazy mohou být také součástí aritmetického výrazu. Např. funkci



můžeme vyjádřit výrazem -X*(X>0)+(X-X1)*(X>X1).

Překladač jazyka BASIC 6 nerozlišuje logické a číselné hodnoty a tedy operandy u logických operací nemusí být vždy relace, ale může to být i aritmetický výraz. V tomto případě se při výpočtu postupuje takto:

a) vypočítá se hodnota uvedeného aritmetického výrazu;
b) z této hodnoty se vezme pouze celá část (provede se funkce INT);
c) tato hodnota musí ležet v intervalu -32768,32767 (jinak je výpočet ukončen signalizací chyby 04);
d) hodnota se zobrazí na šestnáctibitovém slovu v doplňkovém kódu (znaménko v nejvyšším bitu);
e) vlastní logická operace se provádí s těmito šestnáctibitovými slovy po jednotlivých bitech;
f) výsledné šestnáctibitové slovo se převede zpět do zobrazení v pohyblivé řádové čárce, ve kterém jsou zobrazeny všechny číselné hodnoty.

Uvedený postup si ukážeme na následujícím příkladě, ve kterém máme určit hodnotu výrazu:

```
(5 OR 14) + (5 AND 14) + NOT 5

     5  ⟶ 0000000000000101
    14  ⟶ 0000000000001110
5 OR 14  ⟶ 0000000000001111  ⟶  15
5 AND 14 ⟶ 0000000000000100  ⟶   4
   NOT 5 ⟶ 1111111111111010  ⟶  -6
```

Hodnota výrazu je 15 + 4 - 6, tj. 13.

V tomto případě již ale není zaručeno, že hodnota logického výrazu je zobrazena jako číslo -1 nebo 0. Počítač zobrazenou hodnotu interpretuje jako true, pokud se jedná o nenulovou hodnotu, nebo false pro nulovou hodnotu.

V souvislosti s používáním logických operátorů je potřeba upřesnit prioritu jednotlivých operací, které se vyskytují ve výrazu. Operace s vyšší prioritou se provádějí dříve.

Prioritu lze měnit použitím kulatých závorek.

Priorita	Operace	Příklad
9	výpočet hodnoty funkce	SIN(X)
8	mocniny	A^B
7	změny znaménka	-B
6	součinu a podílu	A/B
5	součtu a rozdílu	A+B
4	relační operace	A=B
3	negace	NOT A
2	logický součin	A AND B
1	logický součet	A OR B

Vzhledem k tomu, že v jazyku BASIC 6 se nerozlišují číselné a logické hodnoty, nerozlišují se ani aritmetické a logické výrazy (viz bod 2.6).

2.8 Textové výrazy a funkce

Textový výraz je posloupnost textových konstant, textových proměnných a textových funkcí (MID$, LEFT$, RIGHT$ CHR$ a STR$ - viz dále) spojených operátorem "+". Tento operátor provádí spojení řetězců v jeden řetězec.

Příklady:

Výraz	Hodnota výrazu
"ABC"+"DE"	ABCDE
"1234"+"4321"	12344321
"AB"+"1234"+"CD"	AB1234CD
A$+"234"+B$	hodnota proměnné A$ 234 hodnota proměnné B$

Řetězce znaků textových konstant nebo proměnných se mohou libovolně spojovat. Délka výsledného řetězce může být maximálně 255 znaků.

Textová funkce MID$ dovoluje odvolávat se na vybranou skupinu znaků řetězce, který tvoří hodnotu textového výrazu.

Obecný tvar funkce MID$ je:

MID$(tv,n,p)

kde tv je textový výraz, z jehož hodnoty se vybírá skupina znaků,
n je aritmetický vyraz, jehož hodnota udává pořadové číslo znaku z tv, který bude prvním znakem hodnoty funkce MID$ a
p je aritmetický výraz, jehož hodnota určuje počet znaků hodnoty funkce MID$ (z tv je vybráno p znaků počínaje n-tým znakem).

Hodnota výrazů n a p se pro další výpočet upravuje takto:

- bere se celá část těchto výrazů (funkce INT),
- hodnota výrazů musí ležet v intervalu <0,255>, jinak je výpočet ukončen signalizací chyby.

V následující tabulce je uvedeno několik příkladů textových výrazů používajících funkci MID$:

Výraz	Hodnota výrazu
MID$("ABCD",2,2)	BC
MID$("ABCD",2.9,2.7)	BC
MID$("ABC"+"DE",2,3)	BCD
"1"+MID$("BCD",2,1)	1C
MID$("ABC",2,4)	BC
MID$("ABC",7,6)	prázdný řetězec

Poslední dva příklady ukazují, co se stane, jestliže hodnoty výrazů n a p jsou větší než hodnoty, s kterými by funkce MID$ ještě šla provést. V předposledním příkladu se mají vzít 4 znaky počínaje druhým znakem. Řetězec, který dostaneme, je tvořen pouze dvěma znaky, tzn. vybereme pouze tolik znaků, kolik jich z řetězce můžeme vzít. V posledním příkladu je hodnota n větší než počet znaků v řetězci. V takovýchto případech vzniká vždy prázdný řetězec. Toto platí i pro funkce LEFT$ a RIGHT$, které jsou popsány dále.

Funkce MID$ může být zapsána i takto:

MID$(tv,n)

kde význam tv a n je stejný jako u MID$(tv,n,p). Tentokrát dostáváme řetězec znaků od n-tého znaku do konce řetězce. Stejnou hodnotu bychom dostali i při použití funkce

MID$(tv,n,255).

Funkci MID$ užíváme, když potřebujeme znaky z prostředku řetězce. Jestliže potřebujeme znaky ze začátku, resp. z konce řetězce textového výrazu, můžeme použít funkci LEFT$, resp. funkci RIGHT$. Obecný tvar těchto funkcí je:

LEFT$(tv,n)

RIGHT$(tv,n)

kde tv je textový výraz, ze kterého budeme vybírat skupinu znaků a
n je výraz udávající počet vybíraných znaků.

Funkce LEFT$ vybírá n znaků ze začátku řetězce a funkce RIGHT$ vybírá posledních n znaků řetězce.

Příklad:

LEFT$(A$,5)+MID$(A$,7)

Hodnota tohoto výrazu je hodnota textové proměnné A$, ve které je vynechán šestý znak.

Funkce CHR$ a STR$ slouží pro převod hodnoty aritmetického výrazu na textovou hodnotu. Obecný tvar zápisu těchto funkcí je:

CHR$(av)

STR$(av)

kde av je libovolný aritmetický výraz.

Funkce CHR$ vytváří znak, jemuž v kódu ASCII přísluší hodnota výrazu av (bere se celá část av - musí ležet v intervalu <0,255>). Pokud hodnota av je větší než 127, platí CHR$(av)=CHR$(av-128). Funkce STR$ vytváří řetězec znaků z číselné hodnoty výrazu av. Vytvořený řetězec znaků je tvořen ze znaků, které by byly použity, pokud bychom hodnotu výrazu vypisovali příkazem PRINT (viz bod 2.15). Pouze případný znak mezery na začátku čísla a znak mezery na konci čísla jsou vynechané.

Příklad:

Funkce	Hodnota (řetězce)
CHR$(114)	r
STR$(114)	114
STR$(0.11)	0.11
STR$(0.01)	1E-02

Inverzní funkce k funkci CHR$, resp. STR$ je funkce ASC, resp. funkce VAL. Tyto funkce mají obecný tvar:

ASC(tv)
VAL(tv)

kde tv je libovolný textový výraz.

Hodnota funkce ASC je hodnota ASCII kódu prvního znaku textového výrazu tv. Pokud je textový výraz tvořen prázdným řetězcem, hodnota funkce ASC není definována (je signalizována chyba). Funkce VAL převádí textový řetězec tv na číselnou hodnotu. Od prvního znaku, který nemůže být součástí čísla, je řetězec ignorován. Je-li tv tvořen prázdným řetězcem nebo začíná-li znakem, který nemůže být součástí čísla, je funkční hodnota funkce VAL nulová.

Textová funkce LEN, tvaru:

LEN(tv)

kde tv je libovolný textový výraz, udává délku řetězce znaků (počet znaků) tvořícího aktuální hodnotu textového výrazu tv.

Funkční hodnota funkcí ASC, VAL a LEN je číselná hodnota, tzn. tyto funkce lze používat v aritmetických výrazech.

Příklady:

Funkce	Hodnota
ASC("r")	114
ASC("rxy")	114
ASC("1.2+2")	49
VAL("1.2+2")	1,2
VAL("1E-02")	0,02
VAL("␣␣14PT")	14
LEN("␣␣14PT")	6

2.9 Porovnávání textových výrazů

Dva textové výrazy můžeme porovnávat pomocí relačních operátorů podobně jako aritmetické výrazy. Porovnávání textových výrazů je umožněno díky přiřazení číselné hodnoty (kódu znaku) každému znaku. Kód ASCII je vytvořen tak, že písmena jsou zde v abecedním pořadí. Protože postupně jednotlivým písmenům odpovídají rostoucí číselné hodnoty můžeme říci, že písmeno dále v abecedním pořadí je "větší" než předchozí písmena (viz příloha 1).

Porovnání dvou textových řetězců se provádí pomocí postupného porovnávání znaků řetězců zleva doprava. Jestliže první znaky jsou různé, potom kódy těchto znaků rozhodují o tom, který textový řetězec je "větší", v opačném případě porovnáváme následující znaky. Při porovnávání dvou řetězců o různých délkách se provádí porovnávání v délce kratšího řetězce. V případě rovnosti kratšího řetězce se začátkem delšího řetězce má relace vždy hodnotu false, tj. nulovou hodnotu.

2.10 Standardní funkce

Překladač jazyka BASIC umožňuje bezprostřední výpočet hodnot běžných matematických funkcí jedné proměnné. Tyto funkce jsou uvedeny v následující tabulce:

Funkce	Význam
SIN(X)	sinus argumentu X *)
COS(X)	cosinus argumentu X *)
TAN(X)	tangens argumentu X *)
	*) Argument X je vyjádřen v obloukové míře (v radiánech)
ATN(X)	arkustangens argumentu X - hodnota funkce je vyjádřena v radiánech a leží v intervalu $(-\pi/2, \pi/2)$

SQR(X) druhá odmocnina z čísla X ($\sqrt{X}$, X≥0)

EXP(X) exponenciální funkce argumentu X (e^X)

LOG(X) přirozený logaritmus argumentu X ($\log_e X$, ln X, X>0)

ABS(X) absolutní hodnota čísla X (|X|)

INT(X) celá část čísla X (největší celé číslo nepřevyšující X)

SGN(X) znaménko čísla X

$$SGN(X)=\begin{cases} -1 & \text{pro } X<0 \\ 0 & \text{pro } X=0 \\ 1 & \text{pro } X>0 \end{cases}$$

RND(X) pseudonáhodné číslo z intervalu (0,1).
Funkce RND(X) při každém použití nabývá hodnoty dalšího pseudonáhodného čísla, které vytváří generátor pseudonáhodných čísel. Pokud je argument funkce nezáporný, mají generovaná pseudonáhodná čísla rovnoměrné rozložení v intervalu (0,1) a velikost argumentu na ně nemá vliv. Je proto možno stále používat např. zápis RND(0). Témuž zápornému argumentu přiřazuje funkce RND vždy stejnou hodnotu a zároveň provádí inicializaci generátoru pseudonáhodných čísel v závislosti na hodnotě argumentu. Jestliže potřebujeme pseudonáhodná čísla z intervalu (A,B), můžeme je počítat takto:

```
A+(B-A)*RND(0)
```

Tyto funkce patří mezi tzv. standardní funkce (viz příloha 3).

Identifikátor standardní funkce se skládá ze tří písmen a předchází argumentu funkce zapsanému v kulatých závorkách. Argumentem každé funkce může být libovolný aritmetický výraz, tedy i proměnná nebo konstanta. Argument funkce může obsahovat libovolné standardní funkce.

Příklady:

```
SQR(B^2-4*A*C)
SIN(3.14159)
EXP(LOG(INT(X*100)/LOG(10))
INT(-12)=-12; INT(2.91)=2; INT(-3.4)=-4
```

Standardní funkce se mohou používat v aritmetických výrazech, např.

```
(-B+SQR(B^2-4*A*C))/(2*A).
```

Identifikátor funkce spolu s argumentem vystupujícím v aritmetickém výrazu je považován za samostatný prvek tohoto výrazu, podobně jako proměnná, číslo nebo výraz uzavřený v závorkách.

Poznámka:
Po zapnutí počítač generuje vždy tutéž posloupnost náhodných čísel. To můžeme napravit tím, že na začátek programu zařadíme inicializaci generátoru pseudonáhodných čísel závislou na čase, např.

```
QQ=RND(-PEEK(8))
```

(viz bod 4.3).

2.11 Standardní textová proměnná INKEY$

Tato proměnná umožňuje zjistit, zda bylo stisknuto určité tlačítko klávesnice. Není-li stisknuto žádné tlačítko, je hodnota proměnné INKEY$ prázdný řetězec. Je-li stisknuto černé tlačítko, je hodnota proměnné INKEY$ příslušné velké písmeno nebo dolní znak, při stisknutém SH a černého tlačítka je hodnota, proměnné INKEY$ příslušné malé písmeno nebo horní znak. Při stisknutí černého tlačítka a tlačítka CTRL je obsahem, proměnné INKEY$ příslušný řídicí znak. Při stisknutí tlačítka [➔] je hodnotou proměnné INKEY$ znak zobrazený na pozici kurzoru na obrazovce (srv. bod 1.3.2). Tlačítka FA a FB nemají vliv na hodnotu proměnné INKEY$. Při stisknutí jiného tlačítka (kromě RES a BR) je obsahem proměnné INKEY$ příslušný řídicí znak, který toto tlačítko generuje.

Příklady:

Stisknuto	Hodnota INKEY$
	"J"
	"d"
	CHR$(13)
	CHR$(25)

2.12 Zápis příkazů jazyka BASIC - program v jazyce BASIC

Všechny příkazy, data, vstupní a výstupní údaje se zapisují do řádků jazyka BASIC. Každý řádek má délku maximálně 80 znaků a je ukončen znakem CR. Každý řádek vzniká až v okamžiku, kdy se do něj začne zapisovat informace.

Jeden řádek obrazovky IQ 151 ve standardním provedení (modul VIDEO 32) obsahuje pouze 32 znaků, jeden řádek jazyka BASIC může tedy zabírat až 2,5 řádků na obrazovce.

Příkazy napsané na jednom řádku jazyka BASIC musí být navzájem odděleny dvojtečkou.

Bezprostředně po zapnutí se počítač nachází v tzv. přímém režimu. V tomto režimu lze vkládat jednotlivé řádky s příkazy jazyka BASIC. Jakmile vložíme řádek s příkazy (příkazem) a odešleme klávesou CR, počítač provede postupně všechny příkazy na tomto řádku, vypíše signalizaci READY a očekává další příkazy. V přímém režimu nelze vložit do počítače najednou více příkazů než se vejde na jeden řádek jazyka BASIC.

Začíná-li řádek s příkazy celým číslem, počítač přejde automaticky do režimu programování, tj. řádek se po odeslání neprovede, ale uloží se i s číslem do paměti.

Soubor očíslovaných řádků jazyka BASIC v paměti počítače je program. O práci s programem viz také bod 1.5.

Při běhu programu jsou případné syntaktické chyby signalizovány výpisem textu

* XX ERROR IN n

kde XX je kód chyby (viz příloha 2) a

n je číslo příkazu ve kterém se chyba vyskytuje.

V přímém režimu jsou chyby signalizovány obdobně, pouze není vypisován text "IN" a číslo příkazu.

2.13 Příkazy pro práci s programem - řídící příkazy

Pro spuštění programu se používá příkaz RUN. Tento příkaz vynuluje všechny číselné proměnné, do textových proměnných dosadí prázdný řetězec, zruší deklarace polí a definovaných funkcí a zahájí provádění programu umístěného v paměti počítače počínaje příkazem s nejnižším číslem. Pokud u příkazu RUN uvedeme parametr (číslo), potom se provedou stejné činnosti jako kdyby parametr zde uveden nebyl, ale program se začne provádět od řádku, jehož číslo je uvedeno v parametru (z uvedeného čísla se bere celá část a příkaz s tímto číslem musí být v programu).

Chceme-li při spuštění programu zachovat, hodnoty proměnných, použijeme příkaz GO TO. Po odeslání příkazu

GO TO n

(n je číslo programového řádku, který musí být v programu) začne počítač provádět program od řádku n a jinou činnost předtím neprovede, tzn. všechny hodnoty proměnných, deklarace atd. zůstanou zachovány. (Viz také bod 2.20).

Příkaz LIST způsobí výpis programu na obrazovku počínaje prvním programovým řádkem. Je-li u příkazu LIST uveden parametr (číslo), výpis programu začne od příkazu, jehož číslo odpovídá parametru. Pokud v programu není příkaz s tímto číslem, začíná se od příkazu s nejbližším vyšším číslem. Parametr příkazu LIST musí být celé nezáporné číslo, které je menší než 65530.

Činnost příkazů RUN a LIST lze přerušit stisknutím tlačítka CTRL. Pokud dále stiskneme tlačítko F3 nebo CTRL C, je činnost příkazu ukončena. Stisknutí jiného tlačítka činnost příkazu obnoví.

Pro uložení programu na magnetofonovou kazetu slouží příkaz MSAVE. Tento příkaz může obsahovat parametr. Význam tohoto parametru je stejný jako u příkazu LIST.

Opačnou činnost tj. zavedení programu z magnetofonové kazety do paměti, provádí příkaz MLOAD. Tento příkaz nemůže obsahovat parametr (srv. bod 1.6).

Při zapisování programu si můžeme usnadnit práci tím, že použijeme příkaz AUTO. Tento příkaz slouží k automatickému číslování řádků programu. Příkaz AUTO může obsahovat dva číselné parametry (oddělené čárkou). První parametr určuje číslo řádku, který budeme zapisovat jako první, druhý parametr určuje krok pro číslování dalších řádků. Neuvedeme-li druhý parametr, je krok rovný 10. Napíšeme-li příkaz bez parametrů, je číslo prvního řádku 10 a krok je roven 10. Režim AUTO můžeme ukončit znakem CTRL[nebo použitím libovolného jiného řídícího příkazu, který napíšeme bez čísla řádku (číslo řádku zobrazené na obrazovce musíme smazat).

Ostatní řídicí příkazy nemohou obsahovat parametry. Tyto příkazy provádějí tuto činnost:

BYE - přechod do MONITORU,
SCRATCH - zrušení programu v jazyku BASIC, který je uložen v paměti počítače,
CONT - pokračování ve výpočtu programu, který byl přerušen,
MEM - výpis informace o velikosti volné operační paměti (počet neobsazených slabik).

Všechny příkazy popsané v tomto bodě lze použít v programu, tzn. lze je zapsat v programovém řádku. Výjimkou tvoří příkaz CONT, jehož provedení v rámci programu je nemožné.

2.14 Přiřazovací příkaz - LET

Jedním z nejčastěji používaných příkazů je přiřazovací příkaz. Má tento tvar:

```
LET p = av
```

Kde p je identifikátor proměnné a
av je aritmetický výraz.

Provedení tohoto příkazu spočívá ve výpočtu hodnoty aritmetického výrazu av a přiřazení této hodnoty proměnné p. Pro textové výrazy má příkaz LET tvar:

```
LET p$ = tv
```

kde p$ je identifikátor textové proměnné a
tv je textový výraz.

Postup při provádění tohoto příkazu je stejný jako pro aritmetický výraz.

Vzhledem k tomu, že se přiřazovací příkaz v programech vyskytuje velmi často, překladač umožňuje vynechávat slovo LET.

Příklady použití jsou tyto:

```
LET A=B+C
R(1)=3+R(J)
D$=d$+"."+"00"
```

2.15 Výstup údajů na obrazovku - příkaz PRINT

Pro výstup číselných i textových informací na obrazovku slouží příkaz PRINT.

Jeho nejjednodušší podoba je

```
PRINT
```

která způsobí přechod na nový řádek (výstup znaku CR), tzn. následující výstupy budou pokračovat od nultého sloupce na následujícím řádku obrazovky.

Dalším jednoduchým tvarem příkazu PRINT je

```
PRINT v
```

kde v je aritmetický nebo textový výraz.

Tento příkaz způsobí vyhodnocení výrazu v, vypsání jeho hodnoty na obrazovku (na nastavenou tiskovou pozici) a přechod na nový řádek. (Přechod na nový řádek je realizován vypsáním znaku CR za hodnotu výrazu v. Znak CR vystupuje "navíc" a může proto přemazat některé údaje, které chceme na obrazovce zachovat, např. porušit obrázek při vpisování čísla, apod. Tento efekt lze potlačit užitím oddělovače středník - viz dále).

Obecný tvar příkazu PRINT je

```
PRINT sp
```

kde sp je seznam parametrů.

Dále jsou postupně popsána pravidla, podle nichž může být sp vytvářen.

Parametry příkazu PRINT mohou být aritmetické nebo textové výrazy. Tyto výrazy jsou odděleny tzv. oddělovači (čárkou nebo středníkem), které určují pozici, kam bude na obrazovku vypsán následující výraz (ve skutečnosti ovlivňují pozici kurzoru - viz bod 1.3).

2.15.1 Činnost oddělovače čárka a středník

Znak čárky způsobí výstup do tzv. tiskových zón. To znamená, že poslední zobrazený údaj (číslo nebo text) je doplněn znaky mezer tak, aby následující zobrazovaný údaj začínal v nejbližším tiskovém sloupci, jehož číslo je dělitelné 14. Tiskové sloupce (pozice) jsou očíslovány od nuly. Tiskové zóny mají délku 14 znaků a začínají na pozicích 0 14, 28, 42, 56 a 70. Třetí a pátá zóna je rozdělena do dvou řádků obrazovky, což znepřehledňuje výpis.

Středník slouží pro výstup informací ve zhuštěném tvaru. Za zobrazovaným prvkem nejsou doplňovány mezery, tzn. následující prvek bude vystupovat bezprostředně za koncem předchozího prvku. Tento způsob zachovává čitelnost vystupujících čísel, neboť jsou zakončené jednou mezerou a současně umožňuje výstup textových konstant bezprostředně za sebou. Mezi textovými výrazy nebo mezi textovým a aritmetickým výrazem lze středník vynechávat pokud je počítač schopen rozeznat začátky a konce těchto výrazů.

Příklad:
Příkazy

```
PRINT "AHOJ"3*2"x"
```

a

```
PRINT "AHOJ";3*2;"x"
```

provádějí stejnou činnost.

2.15.2 Činnost příkazu PRINT

Provedení příkazu PRINT spočívá ve výstupu hodnot aritmetických nebo textových výrazů v pořadí jejich výskytu v seznamu parametrů příkazu PRINT. Pokud posledním parametrem je aritmetický nebo textový výraz, potom je výstup ukončen přechodem na nový řádek.

Pokud jsme se před vyčerpáním seznamu parametrů z příkazu PRINT dostali na poslední (osmdesátou) pozici řádku, výstup pokračuje od nulté pozice následujícího řádku. Je-li za posledním parametrem příkazu PRINT umístěn znak čárka nebo středník, provedou se příslušné činnosti, ale na nový řádek se nepřejde. Výstup způsobený dalším příkazem PRINT bude pokračovat na stejném řádku (tj. jako by následoval bezprostředně za tímto oddělovačem).

Činnost příkazu PRINT ilustruje následující program.

```
10 PRINT
20 PRINT "CISLA POZIC NA RADKU"
30 PRINT "01234567890123456789012345678901234567890..."
40 PRINT 1,10,100
50 PRINT -1,-10,-100
60 PRINT 1;20;300;
70 PRINT 4000;50000;600000,
80 PRINT "*";"+";" § "
90 PRINT "ABC"1"PQR";2,"//";"XYZ";
100 PRINT "VIDA !"
```

Po spuštění příkazem RUN se na obrazovce objeví:

```
CISLA POZIC NA RADKU
012345678901234567890123456789012
34567890...
 1             10            100
-1           -10           -100
 1  20  300  4000  50000  600000
          *+ §
ABC 1 PQR 2   //XYZVIDA !

READY
```

Příkaz 30 způsobí zobrazení čísel pozic pro usnadnění analýzy činnosti dalších příkazů programu. Je nutno si uvědomit, že vystupující text uvedený v jednom příkazu může být na obrazovce zobrazen na více řádcích.

2.15.3 Zobrazování řetězců a čísel příkazem PRINT

Hodnoty výrazů, které jsou parametry příkazu PRINT, jsou nejprve vypočítány a pak vypsány na obrazovku. Hodnota textového výrazu se zobrazí jako výsledný řetězec. Zobrazení čísel se řídí následujícími pravidly:

- pokud absolutní hodnota čísla leží v intervalu <0,1;999999,5>, je číslo zobrazeno v základním tvaru pomocí nejvýše šesti číslic;
- v opačném případě je pro zobrazení použit semilogaritmický tvar;
- nevýznamné nuly se nezobrazují, až na to, že exponent je vypisován pomocí dvou číslic;
- pokud zlomková část čísla je nulová, nezobrazuje se ani znak desetinné tečky;
- záporná čísla začínají znakem minus, znak plus před kladnými čísly je nahrazen mezerou;
- bezprostředně za číslem vystupuje znak mezery pro oddělení následujících výstupů.

Příklad:
Příkaz

```
PRINT 4/2;-3/2;2/3;-1/100;1/300
```

způsobí výstup:

```
␣2␣-1.5␣␣.666667␣-1E-02␣␣3.33333E-03␣
```

V prvním a čtvrtém čísle je vynechána zlomková část s desetinnou tečkou. V druhém čísle jsou vynechány nevýznamné nuly ve zlomkové části. Ve třetím čísle je vynechána nevýznamná nula před desetinnou tečkou. Protože absolutní hodnota čtvrtého a pátého čísla je menší než 0,1, jsou tato čísla zobrazena v semilogaritmickém tvaru.

2.16 Další možnosti příkazu PRINT - oddělovač TAB, SPC, &, funkce POS a příkaz CLS

Oddělovač TAB slouží k určení tiskového sloupce, kde bude začínat další výstup. Oddělovač TAB můžeme považovat za funkci. Tato funkce může být použita pouze v seznamu parametrů příkazu PRINT. Obecný tvar této funkce je

TAB(av)

kde av je aritmetický výraz.

Celá část hodnoty av musí ležet v intervalu <0,255>. Tato celá část určuje pozici na řádku, kde bude vystupovat hodnota parametru následujícího po funkci TAB v příkazu PRINT. Připomínáme, že pozice řádku jsou číslované od nuly.

Činnosti vyplývající z použití funkce TAB závisí na rozdílu mezi číslem pozice získaným pomocí funkce TAB a číslem aktuální pozice takto:

- pokud číslo aktuální pozice je větší nebo rovno číslu pozice určené funkcí TAB, neprovede se žádná činnost (na řádku se nemůžeme vracet),
- jinak se provede výstup tolika znaků mezer, abychom se dostali na pozici určenou funkcí TAB.

Podobný význam jako funkce TAB má i funkce SPC, která má obecný tvar

SPC(av)

kde av je libovolný aritmetický výraz.

Použití této funkce je stejné jako u funkce TAB s tím rozdílem, že hodnota výrazu neurčuje číslo tiskového sloupce, ale počet mezer, které mají být vypsány. Pokud aktuální tiskový sloupec je nulový, potom obě funkce provádějí stejnou činnost.

Příklad:
Oba řádky 10 a 20 v programu

```
10 PRINT:PRINT":"TAB(31)":"
20 PRINT:PRINT":"SPC(30)":"
```

způsobí výstup dvojtečky do nultého a jedenatřicátého sloupce.

Pro zjištění aktuálního tiskového sloupce, tj. počtu vypsaných znaků na posledním, ještě nedokončeném řádku, slouží funkce POS(X). Tato funkce může nabývat celočíselných hodnot v intervalu <0,255>. Používání této funkce se řídí zásadami, které jsou popsány v bodě 2.10. Argument funkce POS musí být vždy uveden a na jeho hodnotě nezáleží. Lze tedy stále používat např. POS(0).

Funkce TAB, SPC a POS navzájem souvisejí takto:

```
TAB(X)=SPC(X-POS(0))
```

Toto platí za předpokladu, že argument (X-POS(0)) nemá zápornou hodnotu.

Příklad: Program

```
 5 PRINT
10 PRINT POS(0);POS(0);POS(0)
```

způsobí výpis těchto čísel:

```
0  3  6
```

První výskyt funkce POS zjistí, že na řádku není zatím vypsán žádný znak a je tedy vypsána nula. Pro výpis nuly jsou potřeba tři znaky a při dalším volání nabývá funkce POS hodnoty 3. Tato hodnota je vypsána, čímž se použijí další tři znaky atd.

Kdekoliv v seznamu parametrů příkazu PRINT může být umístěn další oddělovač, který dovoluje tisknout na libovolnou pozici na obrazovce. Jeho obecný tvar je

```
& r,s
```

kde r a s jsou aritmetické výrazy.

Celá část výrazu r(resp. s) musí ležet v intervalu <0,30> (resp. <0,31>). Tento oddělovač umístí kurzor na řádek určený celou částí výrazu r a do sloupce určeného celou částí výrazu s, tzn.,že následující parametr příkazu PRINT začne vystupovat od této pozice.

Příklad:

Příkaz

```
PRINT &30,0"KOHOUTEK"&0,31"+"
```

vypíše do levého dolního rohu obrazovky nápis KOHOUTEK a do pravého horního rohu znak "+".

Upozornění:
Do pravého dolního rohu obrazovky nelze tisknout pomocí příkazu PRINT. Například příkaz

```
PRINT &30,31"0";
```

sice na určenou pozici vytiskne znak nula, kurzor však automaticky přechází na další tiskovou pozici, v tomto případě na začátek dalšího řádku obrazovky. Další řádek se však zpřístupní až po posunutí (odrolovaní) obsahu obrazovky směrem nahoru, a tedy i obsah předcházejícího řádku 30 se octne výše na obrazovce.

Příkaz

```
CLS
```

smaže obrazovku (tj. všechny pozice zaplní znakem mezera) a přesune kurzor do levého horního rohu obrazovky.

Příklad:

```
10 CLS
20 FOR 1=0 TO 30
30 PRINT &I,30-I"X"&I,I"X";
40 NEXT I
50 PRINT:PRINT &0,0
```

Tento program smaže obrazovku, nakreslí obě úhlopříčky pomocí znaků X a přesune kurzor do pravého horního rohu obrazovky.

2.17 Vstup dat - příkaz INPUT, READ, RESTORE, deklarace DATA

Příkaz INPUT slouží ke vkládání dat z klávesnice do programu během jeho činnosti.

Deklarace DATA slouží k deklaraci dat v programu. Příkaz READ slouží k načtení deklarovaných dat do proměnných.

Příkaz RESTORE určuje, která z deklarovaných dat se budou načítat příkazem READ.

2.17.1 Příkaz INPUT

Příkaz INPUT může mít jeden ze dvou obecných tvarů:

```
INPUT tk;p1,p2,...,pk
INPUT p1,p2,...,pk
```

kde tk je textová konstanta a

p1,p2,...,pk (k>=1) je seznam identifikátorů proměnných oddělených čárkami.

Použijeme-li příkaz INPUT v prvém tvaru, je nejprve vypsán (na nastavenou tiskovou pozici na obrazovce) řetězec tk a dvojtečka; při užití druhého tvaru se vypíše jen dvojtečka (dále již oba tvary příkazů pracují shodně) a počítač očekává vkládání dat.

Vkládaná data jsou konstanty, které svým typem (aritmetické nebo textové) korespondují s typem proměnné v seznamu p1,p2,...,pk. Při vkládání textové konstanty není nutno uvádět uvozovky ohraničující tuto konstantu. Vkládaná data se oddělují čárkami a vkládání se ukončí znakem CR. Proměnné ze seznamu pak nabydou hodnot vložených konstant v příslušném pořadí a běh programu pokračuje dalším příkazem.

Je-li vloženo méně dat než je proměnných v příkazu INPUT, počítač očekává pokračování zadání dat z klávesnice. Toto čekání signalizuje vypsáním dvou dvojteček.

Jestliže některé ze zadaných čísel je syntakticky chybné, tj. nejsou dodržena pravidla pro zápis čísla, potom je vypsána signalizace:

```
* INPUT ERROR
```

a dvojtečka. Nyní je třeba zadat toto číslo znovu správně. Pokud se nejednalo o poslední údaj ze zapsané řady dat, je třeba opsat i následující data.

Je-li zadáno více dat než je proměnných v seznamu příkazu INPUT, jsou použity pouze hodnoty ze začátku seznamu. Zbylé hodnoty jsou ignorovány a je vypsána signalizace:

```
* EXTRA DATA IGNORED
```

V době, kdy počítač očekává zadání hodnoty, je možné také ukončit běh programu stisknutím tlačítka CR, aniž bychom nějakou hodnotu zadali. Takto lze ukončit běh programu pouze při zadávání hodnoty pro první proměnnou v příkazu INPUT.

2.17.2 Deklarace DATA

Deklarace DATA má obecný tvar:

```
DATA d1 ,d2,...,dk
```

kde d1,d2,...,dk (k>=1) je seznam konstant oddělených čárkami.

Textové konstanty mohou být v seznamu uvedeny v uvozovkách nebo bez uvozovek. Deklarace:

```
DATA 3,"AHOJ",12
DATA 3,AHOJ,12
```

jsou proto ekvivalentní.

Deklarace DATA může být umístěna v libovolném místě programu. Při běhu programu jsou deklarace DATA ignorovány, používá se jich pouze při provádění příkazu READ.

Poznámka:

Při vynechávání znaků uvozovek ohraničujících textovou konstantu je nutno brát v úvahu následující zásady:

- počáteční mezery jsou překladačem ignorovány,
- mezery, které se vyskytují uvnitř nebo na konci řetězce, jsou součástí řetězce,
- tyto textové konstanty nemohou obsahovat znak čárky (čárka v tomto případě slouží pro ukončení řetězce) a znak dvojtečky (oddělení příkazů),
- tyto textové konstanty mohou obsahovat libovolný počet znaků uvozovek (uvozovky nesmí být prvním znakem řetězce).

2.17.3 Příkaz READ

Příkaz READ má obecný tvar:

```
READ p1,p2,...,pk
```

kde p1,p2,...,pk (k>=1) je seznam identifikátorů proměnných, které jsou odděleny čárkami.

Přiřazení hodnot číselným proměnným uvedeným v seznamu příkazu READ se provádí takto: První výskyt příkazu READ během provádění programu způsobí vyhledaní deklarace DATA s nejmenším číslem programového řádku. Dále jsou proměnným, jejichž identifikátory jsou uvedeny v příkazu READ, přiřazeny postupně hodnoty z této deklarace DATA. Toto se provádí až do vyčerpání proměnných v seznamu parametrů příkazu READ. Pokud jsou již dříve vyčerpána všechna data z deklarace DATA, potom překladač najde v programu následující deklaraci DATA a vezme z ní chybějící údaje. Jestliže v programu taková deklarace chybí, počítač ohlásí chybu a ukončí výpočet. Může také nastat situace, kdy seznam identifikátorů proměnných v příkazu READ se vyčerpá dříve, než jsou použita všechna data ze seznamu v deklaraci DATA. V tomto případě se zapamatuje pozice naposledy přečtené hodnoty. Následující hodnota bude použita při provádění dalšího příkazu READ.

Příklad:
Dále uvedené čtyři části programu provádějí tedy stejnou činnost.

```
A)  10 READ X,Y,Z
          |
          |
          |
    80 DATA 1,2,3
          |
          |
```

```
B)  10 READ X,Y,Z
          |
          |
          |
    80 DATA 1
    90 DATA 2,3
          |
```

```
C)  10 READ X,Y
    20 READ Z
          |
          |
          |
    80 DATA 1
    90 DATA 2,3
          |
```

```
D)  10 READ X
    20 READ Y,Z
          |
          |
          |
    80 DATA 1,2,3
          |
          |
```

Aby programy byly dobře čitelné, je vhodné deklarace DATA umístit buď bezprostředně za příslušné příkazy READ, nebo je shromáždit na konci programu.

2.17.4 Příkaz RESTORE

Příkaz RESTORE má tvar:

```
RESTORE
```

Provedení tohoto příkazu způsobí, že následující příkazy READ budou pracovat tak, jako by dosud žádné příkazy READ nebyly provedeny, tzn. budou číst data počínaje od první deklarace DATA v programu.

Příkaz RESTORE může být také ve tvaru:

```
RESTORE p1
```

kde p1 označuje číslo řádku s příkazem DATA, ze kterého budou čtena data následujícím příkazem READ. Pokud p1 není číslo řádku s příkazem DATA, pak jsou data čtena z příkazu DATA na nejbližším řádku s větším číslem. Řádek s číslem p1 ale musí v programu být.

Příklad:
Následující tři programy pracují stejně:

```
A)  10 READ A,B,C,D,E                  B)  10 READ A,B,C,D,E
    20 DATA 10,20,30,40,50                 20 DATA 10,20,30,40,50
           |                                      |
           |                                      |
           |                                      |
   200 READ U,V,W,X,Y                     200 RESTORE
   210 DATA 10,20,30,40,50                210 READ U,V,W,X,Y
           |                                      |

C)  10 DATA 5,6,7
    20 RESTORE 220
    30 READ A,B,C,D,E
           |
           |
   200 RESTORE 220
   210 READ U,V,W,X,Y
   220 DATA 10,20,30,40,50
           |           |
```

2.18 Ukončení a přerušení činnosti programu - příkazy END, STOP a CONT

Příkaz pro ukončení činnosti programu má tvar:

```
END
```

Po provedení tohoto příkazu se provádění programu ukončí.

Překladač BASIC 6 nevyžaduje, aby příkaz END byl vůbec umístěn v programu nebo aby měl nejvyšší číslo příkazu, které se v programu vyskytuje. Pokud příkaz END v programu není, běh programu skončí po provedení příkazů na programovém řádku s nejvyšším číslem.

V některých případech potřebujeme činnost programu dočasně přerušit. K tomuto účelu slouží příkaz

STOP

Provedení tohoto příkazu způsobí zastavení programu a výpis signalizace

BREAK IN n

kde N je číslo prováděného řádku, na kterém je právě provedený příkaz STOP.

Činnost programu pak pokračuje po napsání příkazu CONT na klávesnici (bez čísla řádku). Tento příkaz můžeme použít také po přerušení běhu programu tlačítky CTRL C nebo F3 (viz bod 2.13) a v případě jeho ukončení, kdy při příkazu INPUT nezadáme žádný údaj a zapíšeme znak CR. Pokud použijeme příkaz CONT u programu, jehož běh byl ukončen provedením příkazu END, začnou se provádět případné další příkazy.

2.19 Komentáře - příkaz REM

Do programu v jazyku BASIC lze vkládat komentáře, tj. příkazy tvaru

REM t

kde t je libovolná posloupnost znaků ukončená znakem CR.

Komentáře mohou být umístěny na libovolném místě programu a slouží uživateli pro orientaci. Počítač po dekódování klíčového slova REM přejde ihned na provádění dalšího příkazu programu a text zapsaný za REM ignoruje.

U příkazu REM je nutno se zmínit i o případu, kdy zapisujeme více příkazů na jeden řádek. U většiny příkazů se vždy jako následující provádí další příkaz na řádku (pokud tam nějaký je) nebo příkaz z následujícího řádku. Příkaz REM tvoří výjimku. Zde se vždy jako následující příkaz provádí až příkaz na následujícím řádku (za komentář se považuje celý zbytek řádku).

Např. ve skupině příkazů

```
10 A=0: REM POCATECNI HODNOTA: B=10
```

se B=10 považuje za součást komentáře.

2.20 Nepodmíněný skok - příkaz GO TO

U většiny příkazů v jazyce BASIC se vždy po provedení příkazů přejde na provádění následujícího příkazu. V některých případech je ale potřebné změnit pořadí provádění příkazů. K tomuto účelu lze použít příkaz nepodmíněného skoku, který má obecný tvar

```
GO TO n
```

kde n je číslo programového řádku.

Provedení tohoto příkazu způsobí přechod (skok) na programový řádek s číslem n, tzn. první příkaz na tomto řádku bude proveden jako následující po příkazu GO TO n. Vzhledem k tomu, že po příkazu

```
GO TO n
```

se jako následující provádí příkaz na řádku s číslem n, případný další příkaz uvedený za příkazem GO TO n na stejném řádku se nikdy nemůže provést. Např. ve skupině příkazů

```
10 A=0: GO TO 70: B=10
```

se příkaz B=10 nikdy neprovede a je tedy zbytečné zde tento příkaz uvádět.

Aby příkaz GO TO n mohl být proveden, musí program obsahovat řádek s číslem n, jinak počítač ohlásí chybu.

Příklad:

Program

```
10 PRINT "PRVNI RADEK"
15 GO TO 30
20 PRINT "DRUHY  RADEK"
30 PRINT "TRETI RADEK"
```

způsobí výstup:

```
PRVNI RADEK
TRETI RADEK
```

2.21 Podmíněný skok - příkaz IF

Tento příkaz slouží k větvení programu. Může se používat ve dvou tvarech:

```
IF av THEN sp
IF av THEN n
```

kde av je aritmetický výraz,
sp je neprázdný seznam příkazů a
n je číslo řádku.

Činnost příkazu v prvním tvaru:
Jestliže hodnota výrazu av je nenulová (tj. podmínka splněna - má hodnotu true), program pokračuje prováděním příkazů sp (příkazu) za slovem THEN. Jestliže hodnota av je nulová (podmínka nesplněna - má hodnotu false), žádný z příkazů seznamu sp se neprovede a program pokračuje na následujícím programovém řádku.

Příkaz IF zapsaný v druhém tvaru je zkratkou příkazu

```
IF av THEN GO TO n
```

Příklad:
Úryvek programu:

```
30 IF A=B THEN GOSUB 60: PRINT A
40 IF A$="NE" THEN 1000
50 IF NOT A THEN A=-SGN(A)
```

provádí tyto akce:

Na řádku 30 otestuje, zda hodnota proměnné A je rovna hodnotě proměnné B. Jestliže je rovna, pak se vyvolá podprogram začínající na řádku 60 a vytiskne se hodnota proměnné A. Jestliže není roven, program ihned přechází na řádek 40. Jestliže hodnota proměnné A$ je řetězec "NE", provede se skok na řádek 1000; v opačném případě program pokračuje na řádku 50. Pokud celá část hodnoty proměnné A je rovna -1, má výraz NOT A hodnotu 0 (viz bod 2.7) a program přechází na další řádek. Pokud celá část hodnoty A je různá od -1 má výraz NOT A nenulovou hodnotu a počítač provede příkaz A=-SGN(A).

2.22 Příkazy cyklu - FOR, NEXT

V jazyku BASIC 6 má příkaz cyklu dva obecné tvary:

```
FOR pc=av1 TO av2 STEP av3
    |
    |
    |
NEXT pc
```

nebo

```
FOR pc=av1 TO av2
    |
    |
    |
NEXT pc
```

kde pc je jednoduchá aritmetická proměnná, která má, funkci parametru cyklu,

av1 je aritmetický výraz určující počáteční hodnotu parametru cyklu

av2 je aritmetický výraz určující koncovou hodnotu parametru cyklu,

av3 je aritmetický výraz určující hodnotu kroku změny parametru cyklu. Je-li krok roven 1, může být část příkazu STEP av3 vynechána a příkaz FOR má druhý obecný tvar.

Příkazy mezi příkazem FOR a NEXT se nazývají tělo cyklu.

Příklady příkazů FOR:

```
FOR L=1 to 10   .....  NEXT L
FOR A=1 to 2*B-3   .....  NEXT A
FOR I1=C+D to N STEP 10   .....  NEXT I1
FOR K=1.2 to .5 STEP -1E-2   .....  NEXT K
```

Provedení příkazu FOR spočívá ve výpočtu hodnot aritmetických výrazů vystupujících v příkazu FOR (av1, av2,a av3), přiřazení hodnoty výrazu av1 parametry cyklu pc, v zapamatování koncové hodnoty av2 a hodnoty kroku av3. Po provedení příkazu FOR se postupně provedou následující příkazy programu až po příkaz NEXT se stejným parametrem cyklu. Provedení příkazu NEXT spočívá v přičtení hodnoty kroku k parametru cyklu a testovaní, zda je splněna podmínka ukončení cyklu.

Je-li hodnota kroku kladná, potom podmínka ukončení cyklu je, aby hodnota parametru cyklu byla větší než koncová hodnota. Pro záporný krok je podmínka ukončení splněna, když hodnota parametru cyklu je menší než koncová hodnota. Při splněné podmínce ukončení cyklu se přechází na provedení následujícího příkazu za příkazem NEXT. Jestliže podmínka není splněna, provedou se znova příkazy těla cyklu, počínaje příkazem následujícím po příkazu FOR. V případě, kdy hodnota kroku je nulová, není cyklus nikdy ukončen.

Hodnoty aritmetických výrazů av1, av2 a av3 z příkazu cyklu se počítají pouze jednou před prvním provedením těla cyklu. Proměnné vystupující v těchto výrazech tedy mohou změnit svoje hodnoty během provádění příkazů těla cyklu, ale nemá to vliv na počet prováděných opakování. Jelikož o ukončení cyklu rozhoduje aktuální hodnota parametru cyklu v okamžiku provádění příkazem NEXT, má změna hodnoty parametru cyklu v příkazech těla cyklu vliv na počet opakování.

Poznámka:
V případě, kdy hodnoty aritmetických výrazů vystupujících v příkazu FOR jsou takové, že podmínka ukončení cyklu je splněna okamžitě, příkazy těla cyklu se přesto jednou provedou, např. u cyklu

```
FOR K=1 TO 0
```

Tělo cyklu může obsahovat libovolné příkazy. Z těla cyklu můžeme přejít pomocí příkazů GO TO nebo IF do další části programu bez testování podmínky ukončení cyklu. Není to ale vhodné, neboť potom mohou vznikat chyby pokud dále v programu použijeme další cyklus se stejným parametrem cyklu, jako u cyklu, který jsme ukončili tímto způsobem), které budeme velmi těžko hledat. Nepřípustný, je ale skok na příkaz umístěný v těle cyklu bez provedení příkazu FOR začínajícího daný cyklus.

U příkazu NEXT se parametr cyklu nemusí uvádět, program je potom ale hůře čitelný. Příkaz NEXT bez parametru ukončuje naposledy použitý příkaz FOR.

Pokud více vnořených cyklů končí na jednom místě v programu, lze použít jeden příkaz NEXT takto:

NEXT pc_1,pc_2,...,pc_n

kde pc_1,pc_2,...,pc_n (n>=1) musí být v pořadí, v jakém by byly uvedeny, kdyby byly příkazy NEXT uvedeny po jednom.

Příklad:

```
Program
     40 FOR I=0 TO 1
     50 FOR J=0 TO 1
     60 PRINT I,J
     70 NEXT J
     80 NEXT I
```

můžeme napsat také tak, že příkazy 70 a 80 nahradíme příkazy:

```
     70 NEXT
     80 NEXT
```

nebo příkazem:

```
     70 NEXT J,I
```

2.23 Pole - deklarace DIM, příkaz FREE

Složky polí v jazyku BASIC 6 se indexují celými nezápornými čísly. Nejmenší index každé složky pole je vždy nula. Pro určení počtu rozměrů pole a maximálních hodnot indexů se užívá deklarace DIM. Její obecný tvar je:

DIM d_1,d_2,...,d_n

kde d_1,d_2,...,d_n (n>=1) je seznam identifikátorů polí, jehož prvky mají tvar

i(r_1,r_2,...,r_k)

kde i je identifikátor pole,

r_1,r_2,...,r_k (k>=1) jsou aritmetické výrazy určující maximální hodnoty indexů složek pole a

k je počet rozměrů pole.

Hodnoty aritmetických výrazů musí být nezáporné. Nemá-li aritmetický výraz celočíselnou hodnotu, bere se jeho celočíselná část.

Příklad:

```
 10 DIM T(20)
 60 DIM A(30,40),B(100)
200 DIM AB$(K*L+10,2)
```

Příkaz 10 deklaruje jednorozměrné pole T skládající se z 21 prvků. Příkaz 60 deklaruje dvě pole. První z nich je dvourozměrné (pole A) a druhé (pole B) je jednorozměrné o 101 prvcích. U pole A je 31 řádků a 41 sloupců. Příkaz 200 deklaruje dvourozměrné textové pole.

Deklarace DIM mohou být umístěny v libovolném místě programu, musí ale být provedeny před prvním použitím odpovídajících indexovaných proměnných.

Jestliže v programu je použita indexovaná proměnná s maximálně třemi indexy, pro kterou nebylo deklarováno pole deklarací DIM, provede se automaticky rezervování paměti pro toto pole, kde maximální hodnoty indexů jsou rovny 10. Deklarování polí, kde proměnné mají indexy menší nebo rovny 10, se tedy nemusí provádět u jedno-, dvou- a třírozměrných polí. Použití indexu většího než 10 u nedeklarovaného pole se ohlásí jako chyba.

Deklarace pole musí být v programu provedena pouze jednou. Opakovaná deklarace je nepřípustná. Použijeme-li prvek pole, jehož některý index neleží v deklarovaných mezích, ohlásí se chyba.

Při deklaraci pole se vždy obsadí jistá část operační paměti pro zapamatování prvků tohoto pole. Pole, která již pro další výpočty s programem nepotřebujeme, můžeme zrušit.

K rušení polí se používá příkaz FREE, který má obecný tvar

FREE $i_1,i_2,...,i_n$

kde $i_1,i_2,...,i_n$ (n>=1) je seznam identifikátorů rušených polí.

Všechna pole s identifikátory v seznamu příkazu FREE musí být v okamžiku provedení příkazu deklarována, jinak během programu dojde k ukončení běhu programu a signalizaci chyby. Z tohoto důvodu není také dovoleno, aby se v seznamu vyskytl vícekrát stejný identifikátor.

Příkazem FREE nelze rušit textová pole.

Příklad:
Pole T a A deklarovaná v minulém příkladu zrušíme příkazem:

```
FREE T,A
```

2.24 Definování funkcí - deklarace DEF

Mimo standardní funkce popsané v bodě 2.10 lze v jazyku BASIC definovat a používat libovolnou funkci (až pěti proměnných), kterou lze vyjádřit pomocí aritmetického výrazu. K tomuto účelu slouží deklarace

DEF FNi (p_1,p_2,...,p_n}=av

kde FNi je identifikátor definované funkce (i je vytvořeno podle stejných zásad jako identifikátor jednoduché aritmetické proměnné),

p_1,p_2,...,p_n (1=<n=<5) je seznam formálních parametrů (argumenty funkce) a

av je aritmetický výraz definující funkci.

Formální parametry mají stejný tvar jako identifikátory jednoduché proměnné a vystupují ve výrazu definujícím funkci. Různé definované funkce mohou používat stejné formální parametry, formální parametry u jedné funkce ale musí být různé. Přitom je nutné, aby se formální parametry lišily od všech proměnných vystupujících v aktuálních parametrech dané funkce, jinak může dojít k nepředvídatelným chybám.

Příklad:
V programu

```
10 DEF FNA(X,Y)=X-Y
20 Y=2: X=1
30 PRINT FNA(Y,X)
```

jsou X a Y formální parametry definované funkce FNA. Jako aktuální parametry jsou v řádku 30 použity stejnojmenné proměnné Y a X, což. vede k tomu, že se zobrazí 0 a ne 1, jak bychom očekávali.

Pokud se ve výrazu av na pravé straně definice vyskytují kromě formálních parametrů i jiné proměnné, musí být dříve v programu alespoň jednou použity, jinak je ohlášena chyba.

Definice funkce může být umístěna v libovolném místě programu před prvním použitím této funkce. Celá definice funkce je v jediném příkazu, proto se těmto funkcím také říká jednopříkazové funkce.

Jednopříkazové funkce mohou být použity v aritmetických výrazech stejně jako standardní funkce. Použití jednopříkazové funkce spočívá v uvedení jejího identifikátoru s aktuálními parametry zapsanými v závorkách. Tyto parametry mohou být libovolné aritmetické výrazy. Počet aktuálních parametrů musí souhlasit s počtem formálních parametrů uvedených v definici funkce.

Příklady použití funkcí:

```
100 X=Y+FNA(40)
200 PRINT A,FNZ(A),10*SQR(FNZ(A+FNA(A))
300 S=FNC(B*FNC(2)+5)
```

V posledním příkaze má funkce FNC jako aktuální parametr opět funkci FNC, což je přípustné.

Pokud funkci se stejným identifikátorem definujeme několikrát, používá se poslední definice.

Při výpočtu hodnoty aritmetického výrazu, ve kterém je použita jednopříkazová funkce, se místo této funkce použije její hodnota pro její aktuální parametry. Hodnota jednopříkazové funkce je vypočítána takto:

a) nalezení deklarace definující danou funkci v programu,
b) výpočet hodnot aktuálních parametrů,
c) výpočet hodnoty výrazu definujícího funkci po dosazení hodnot vypočítaných v b) za formální parametry.

Ve výrazech definujících funkce, je možno používat také jiné jednopříkazové funkce.

Nepřípustné je ale použití stejné funkce, např.:

```
100 DEF FNB(X)=(1+1/X)*FNB(X)
```

Při použití takovéto funkce výpočet hodnoty funkce nikdy neskončí (skončí signalizací chyby po zaplnění celé paměti počítače návratovými adresami). Ze stejného důvodu nejsou přípustné ani definice funkcí jako:

```
500 DEF FNA(X)=FNB(X)+...
500 DEF FNB(X)=FNA(X)+...
```

2.25 Podprogramy - příkazy GOSUB a RETURN

Podprogram je libovolná posloupnost příkazů ukončená příkazem RETURN. První příkaz podprogramu musí být také prvním příkazem na řádku. Vyvolání podprogramu se provádí příkazem

```
GOSUB n
```

kde n je číslo programového řádku, na němž podprogram začíná. Řádek s číslem n musí v programu existovat.

Příkaz RETURN ukončí provádění podprogramu. Počet úrovní podprogramu není omezen. Při pokusu o provedení příkazu RETURN bez předchozího volání podprogramu příkazem GOSUB počítač ohlásí chybu.

Po skončení činnosti podprogramu běh programu pokračuje dalším příkazem za příkazem GOSUB n.

Je také přípustné vyvolání podprogramu z téhož podprogramu (tzv. rekurentní volaní), např.:

```
 10 S=1
 20 INPUT "N";N
 30 IF N>1 THEN GOSUB 100
 40 PRINT "N!="S
 50 END
100 REM VYPOCET N!
110 S=S*N
120 N=N-1
130 IF N<=1 THEN RETURN
140 GOSUB 100
150 RETURN
```

2.26 Přepínače - příkaz ON

Přepínač lze užívat v jednom ze dvou obecných tvarů:

ON av GO TO $p_1,p_2,...,p_k$

ON av GOSUB $p_1,p_2,...,p_k$

kde av je aritmetický výraz a

$p_1,p_2,...,p_k$ (2=<k=<255) jsou čísla programových řádků.

Příkaz přepínače v prvním tvaru dovoluje pokračovat ve vypočtu od některého z programových řádků, jehož číslo je uvedeno v seznamu $p_1,p_2,...,p_k$. O výběru řádku, na který se přejde, rozhoduje hodnota aritmetického výrazu av. Je-li tato hodnota rovna 1, program pokračuje na řádku s číslem p_1, je-li hodnota rovna 2, program pokračuje na řádku s číslem p_2, atd.

Příkaz přepínače v druhém tvaru provede vyvolání podprogramu začínajícího na některém z řádků ze seznamu $p_1,p_2,...,p_k$. Princip výběru řádku je stejný.

Správné provedení příkazu přepínače vyžaduje, aby program obsahoval řádky s čísly $p_1,p_2,...,p_k$.

Při vyhodnocování aritmetického výrazu av se postupuje takto:

a) pro další vyhodnocování se bere pouze celá část hodnoty aritmetického výrazu,
b) tato celá část hodnoty musí ležet v intervalu <0,255> (jinak je signalizována chyba). V případě, kdy upravená hodnota aritmetického výrazu je nulová nebo větší než k (počet čísel programových řádků v přepínači), přejde se na provedení následujícího příkazu.

Příklad:

```
50 ON A GO TO 2,40,30,100 60 ......
```

Příkaz přepínače zde obsahuje čtyři čísla programových řádků: 2, 40, 30 a 100.

O výběru jednoho z těchto příkazů rozhoduje aktuální hodnota proměnné A takto:

Celá část proměnné A	Číslo následujícího řádku
1	2
2	40
3	30
4	100
jiná	60

Příklad:

```
50 ON A GO TO 300,500,600: PRINT "AHOJ"
```

V případě kdy INT(A)=0 nebo INT(A)>3, vystoupí text "AHOJ".

2.27 Příkazy PLOT a UNPLOT

Tyto příkazy slouží ke kreslení grafů a různých obrázků na obrazovku. Plocha obrazovky je rozdělena na 64 řádky a na 64 sloupce. Každé toto políčko má rozměry 4x4 body (čtvrtina plochy,kterou na obrazovce zabírá znak). Souřadnice políčka se zadávají takto:

- počátek souřadnic (bod o souřadnicích 0,0) je v levém dolním rohu obrazovky.
- osa x směřuje doprava,
- osa y směřuje, nahoru.

Příkaz PLOT zobrazí příslušný čtvereček a příkaz UNPLOT provádí mazání čtverečku. Jinak je činnost obou příkazů stejná.

Obecný tvar obou příkazů je

```
PLOT av1,av2
UNPLOT av1,av2
```

kde av1 je aritmetický výraz udávající x-ovou souřadnici a
av2 je aritmetický výraz udávající y-ovou souřadnici.

Při vyhodnocování se z hodnoty aritmetického výrazu bere pouze celá část (tato celá část musí u obou výrazů v příkazu ležet v intervalu <0;63>).

Padne-li zobrazovaný nebo mazaný čtvereček na místo,kde je příkazem PRINT zobrazen nějaký znak, potom tento znak je nejdříve smazán a teprve potom se provede činnost související s prováděným příkazem.

Činnost příkazu PLOT si ukážeme na programu, který na obrazovce nakreslí rámeček a obě uhlopříčky.

```
10 CLS
20 FOR I=0 TO 63
25 REM RAMECEK
30 PLOT 0,I: PLOT I,0
40 PLOT 63,I: PLOT I,63
45 REM UHLOPRICKY
50 PLOT I,I: PLOT I,63-I
60 NEXT I
```

2.28 Příkaz WAIT

Příkaz WAIT umožňuje přerušit běh programu na určitou dobu. Příkaz má tento obecný tvar:

```
WAIT (av)
```

kde av je aritmetický výraz. Hodnota tohoto výrazu určuje dobu přerušení běhu programu v desetinách sekundy (z hodnoty výrazu je brána celá část, která musí ležet v intervalu <0;65535>).

Příkaz je možno používat, pokud potřebujeme zpomalit výstup informací na obrazovku. Tento příkaz není možno přerušit stisknutím tlačítka CTRL. Naprogramujeme-li příliš dlouhé čekání, lze běh programu ukončit pouze tak, že stiskneme tlačítko BR (tím přejdeme do režimu MONITOR – viz kapitola 4) a tlačítkem R se vrátíme zpět do BASICu. Pak již ale nelze pokračovat příkazem CONT.

2.29 Příkaz CLEAR

Příkaz CLEAR má tři obecné tvary:

```
CLEAR
CLEAR av1
CLEAR av1,av2
```

kde av1 a av2 jsou aritmetické výrazy (pro výpočet se bere celá část těchto výrazů), které mají nezápornou hodnotu.
Při použití tohoto příkazu (v libovolném tvaru) se vynulují hodnoty všech číselných proměnných, do všech textových proměnných se přiřadí prázdný řetězec a zruší se všechny deklarace polí. Dále je provedena činnost jako při použití příkazu RESTORE (nastaví ukazatele pro čtení na první příkaz DATA).

Pokud je příkaz ve tvaru

CLEAR av1

je dále změněna velikost úseku paměti, který slouží pro ukládání řetězců znaků. Tomuto úseku říkáme oblast STRING. Standardně je velikost oblasti STRING 48 znaků (lze do ní uložit řetězce, které mají dohromady maximálně 48 znaků). Po použití příkazu CLEAR av1 je nová velikost oblasti určena hodnotou výrazu av1.

Je-li příkaz ve třetím tvaru, je dále měněna velikost oblasti USR, tj. oblasti určené pro ukládání podprogramů ve strojovém kódu. Tyto podprogramy lze ukládat na libovolné místo operační paměti, ale pouze pokud jsou v oblasti USR, jsou chráněny před přepsáním při zápisu nebo běhu programu. Hodnota výrazu av2 určuje novou velikost oblasti USR. Standardně má oblast USR nulovou délku (viz. bod 5.4).

3 PŘÍDAVNÝ MODUL STAPER

3.1 Využití modulu STAPER v BASICu

Modul STAPER (STAndardní PERiferní zařízení) umožňuje připojit k počítači IQ 151:
- tiskárnu,
- snímač děrné pásky,
- děrovač děrné pásky.

Tiskárnu lze použít pro výpis programu nebo pro tisk výstupu programu. Pro výpis programu se používá příkaz LLIST a pro tisk výstupů příkaz LPRINT. Pravidla pro používání příkazu LPRINT jsou stejná jako pro příkaz PRINT. Výjimkou tvoří oddělovač &r,s. Pokud se tento oddělovač vyskytne v příkazu LPRINT, není použit pro určení místa dalšího výstupu na tiskárnu, ale jen změní polohu kurzoru na obrazovce.

Zařízení děrné pásky lze v režimu BASIC využívat pouze pro čtení, resp. děrování programu v BASICu. Děrnou pásku děrovač vyděruje na příkaz PLIST, čte na příkaz PTAPE. Příkazy PLIST a LLIST mohou obsahovat parametr (celé nezáporné číslo) jako u příkazu LIST. Tento parametr umožňuje vyděrování nebo vytištění programu od příkazu s číslem určeným parametrem příkazu. Činnost těchto příkazů se může ukončit stejným způsobem jako příkaz LIST (CTRL a F3 nebo CTRL C), takže nemusíte děrovat nebo tisknout program až do konce.

Při stisknutí tlačítka RES je přečten jeden znak z děrné pásky založené do snímače. Z tohoto důvodu snímač děrné pásky nepracuje (neprovede příkaz PTAPE), pokud jsme stiskli tlačítko RES a ve snímači nebyla založena děrná páska. Tuto závadu můžeme odstranit tím, že založíme děrnou pásku do snímače, stiskneme tlačítko RES a teprve potom napíšeme příkaz PTAPE.

3.2 Další využití modulu STAPER

S možnostmi dalšího využití tohoto modulu se seznámíme v kapitole 4.

4 MONITOR POČÍTAČE IQ 151

4.1 Úvod

Monitor je základní program počítače IQ 151. Tento program umožňuje základní komunikaci uživatele s počítačem. Řídí vstupní a výstupní operace a přiřazuje logickým vstup- ním/výstupním zařízením, tj. zařízením, na něž se odvolávají programy, fyzická, vstupní/výstupní zařízení, tj. konkrétní vstupní/výstupní zařízení, která jsou v sestavě počítače.

Monitor může pracovat, pokud je do počítače zasunut modul VIDEO 32 nebo VIDEO 64 (smí být zasunut pouze jeden z těchto modulů). Monitor dále umožňuje využívat tyto moduly:

- STAPER - standardní periferní zařízení (tiskárna, snímač děrné pásky, děrovač děrné pásky),
- modul souřadnicového zapisovače,
- moduly překladačů (BASIC 6, BASIC G apod.),
- GRAFIK - modul jemné grafiky.

Monitor je umístěn v pamětech EPROM na adresách F000 až F9C2 (F9C3 až FFFF - nevyužito). Pokud není v počítači zasunut modul žádného překladače, je monitor po zapnutí počítače nebo po stisknutí tlačítka RES ihned spuštěn. Pokud se používá modul některého překladače, můžeme předat řízení monitoru stisknutím tlačítka BR nebo příkazem ukončujícím činnost překladače (např. BYE V jazyku BASIC).

Většina čísel uváděných v této kapitole je v šestnáctkové soustavě. Pokud by mohlo dojít k nedorozuměním, je za nimi připojeno písmeno H, např. 20H.

4.2 Příkazy monitoru

Příkazy monitoru jsou tvořeny jedním písmenem a případným seznamem parametrů. Oddělovač mezi parametry je znak čárka nebo znak mezera. Mezi písmenem určujícím příkaz a prvním parametrem se oddělovač nepíše.

Parametr, který v dalším textu bude označován písmenem A (s případným indexem), označuje adresu v délce dvou slabik, kterou zadáváme jako čtyřmístné číslo v šestnáctkové soustavě. Pokud zapíšeme více číslic, zpracují se pouze poslední čtyři (můžeme takto opravit chybně zapsaný údaj). Parametr označovaný písmenem D označuje číslo ukládané do jedné slabiky, tj. dvojmístné číslo v šestnáctkové soustavě. Zapíšeme-li více číslic, zpracují se pouze poslední dvě. Zápis příkazu se ukončí znakem CR (výjimku tvoří příkazy S a R). Příkaz lze zapisovat, když je na obrazovce vypsán znak >. Analýza příkazu se provádí ihned při zápisu. Případné chyby jsou signalizovány znakem otazníku. Prázdný parametr (dva oddělovače za sebou nebo oddělovač za písmenem určujícím příkaz) má nulovou hodnotu.

Popis jednotlivých příkazů:

FILL
FA1,A2,D - naplnění všech slabik paměťového prostoru od adresy A1 do adresy A2 hodnotou D.

MOVE
MA1,A2,A3 - přesun obsahu paměti počínaje od adresy A1 do adresy A2 na nové místo v paměti, které začíná adresou A3. Adresa A3 nesmí být v přesouvané části paměti, aby nedošlo k přepsaní některých informací.

DISPLAY
DA1 - výpis obsahu paměti (v šestnáctkové soustavě) na obrazovku (nebo na tiskárnu - podle hodnoty slabiky IOBYTE - viz bod 4.4) od adresy A1. Výpis je ukončen stisknutím tlačítka CR, stisknutí jiného tlačítka způsobí pokračování ve výpisu. Na jednom řádku je vypsáno až 8 hodnot (obsahů následujících slabik). Ve výpisu je na začátku každého řádku uvedena adresa slabiky, jejíž obsah je vypsán jako první na řádku.

SUBSTITUTE

SA1 - příkaz vypíše adresu A1, obsah slabiky o této adrese a pomlčku. Opakovaným stisknutím tlačítka "," nebo "SP" se krokuje k vyšším adresám, stisknutím tlačítka "nahoru" (pohyb kurzoru nahoru) se krokuje směrem k nižším adresám. Chceme-li obsah slabiky změnit, zapíšeme její novou hodnotu za pomlčku a zápis této hodnoty ukončíme stisknutím tlačítka ",", "SP" nebo "nahoru". Činnost příkazu se ukončí znakem CR. Zápis příkazu se zde neukončuje znakem CR, ale znakem mezera.

CHANGE

X - po zápisu tohoto příkazu se vypíše obsah všech registrů mikroprocesoru uložených v paměti na adresách 22H až 2DH (viz bod 4.3). Zapíšeme-li X a označení registru, vypíše se jeho obsah a pomlčka. Obsah registru lze změnit zápisem nového údaje za pomlčku. Pokud nový údaj ukončíme znakem čárky nebo mezery, je vypsán obsah následujícího registru, při ukončení znakem CR činnost příkazu končí.

RETURN

R - Návrat na adresu danou obsahem slabik o adresách 1DH a 1EH. Při používání modulu BASIC 6 je zde uložena adresa "teplého startu" BASICu, tj. CAD6H. Příkaz se neukončuje žádným znakem.

GOTO

GA1 - příkaz nejdříve uloží parametr příkazu do paměti na adresy 2CH a 2DH (oblast pro uschování registru PC). Pokud je příkaz bez parametrů, tato činnost se neprovádí. Dále jsou do registrů mikroprocesoru přesunuty hodnoty uložené v oblasti pro uschování hodnot registrů (slabiky paměti o adresách 22H až 2DH). Program pak pokračuje od adresy uložené v registru PC. Příkaz se používá pro spouštění programů ve strojovém kódu od adresy určené parametrem příkazu nebo od adresy dané obsahem slabik o adresách 2CH a 2DH (pokud příkaz neobsahuje parametr).

CALL

CA1 - příkaz způsobí uložení návratové adresy (tj. adresy návratu do monitoru) do zásobníku. Dále se provádí stejná činnost jako u příkazu G. Pokud takto spuštěný program končí instrukcí návratu z podprogramu (např. instrukcí RET), potom je uložen obsah registrů mikroprocesoru do oblastí pro úschovu registrů (slabiky paměti o adresách 22H až 2BH -obsah PC se neukládá) a monitor čeká na napsání dalšího příkazu.

LOAD

LA1 - nahraje z magnetofonu (nebo ze snímače děrné pásky - podle obsahu slabiky IOBYTE) soubor typu HEX, tj. programový nebo datový modul v absolutním tvaru (viz bod 4.5). Soubor je nahráván po blocích tak, jak byl vytvořen příkazem W. Soubor je ukládán do paměti od adresy dané součtem A1+X, kde X je počáteční adresa souboru uvedená při vytváření příkazem W (adresa počátku původního uložení souboru). Pokud soubor chceme uložit v paměti na stejném místě, lze použít příkaz L bez parametru.

Poznámka:

Před zahájením čtení z magnetofonu je přeprogramován obvod 8255 (adresy bran 84H až 87H). Po skončení čtení tento obvod není uveden do původního stavu a vznikají problémy při používání klávesnice, pokud přečtený soubor je program, který chceme spustit (viz třetí parametr příkazu W). Tuto závadu odstraníme tak, že na začátku spouštěného programu umístíme instrukce:

```
MVI A,8AH
OUT 87H
```

které obvod 8255 uvedou do původního stavu.

WRITE

WA1,A2,A3 - nahraje na magnetofon (nebo vyděruje do děrné pásky - podle obsahu slabiky IOBYTE) soubor typu HEX, který je tvořen posloupností slabik paměti od adresy A1 do adresy A2 a startovní adresu, která je určena parametrem A3.

Pokud nechceme, aby se při nahrávaní soubor (program) spustil je nutno třetí parametr příkazu W zadat s nulovou hodnotou. Je-li tento parametr nenulový, je po nahrání souboru příkazem LP1 soubor spuštěn od adresy P1+A3.

4.3 Adresy paměti využívané monitorem

Monitor pro svou práci používá tyto oblasti paměti RWM RAM:

a) začátek paměti RWM RAM (adresy 3 - 45H) - oblast proměnných monitoru,
b) konec paměti RWM RAM (po stisknutí tlačítka RES se zjišťuje velikost paměti RWM RAM),
c) paměť obrazovky - VIDEO RAM (adresy EC00 až EFFF u modulu VIDEO 32 nebo E800 až EFFF u VIDEO 64).

Na začátku paměti RWM RAM jsou umístěny proměnné monitoru (zatím se používají slabiky paměti o adresách 3 až 2DH).

Význam těchto proměnných je uveden v následující tabulce:

Adresa	Proměnná	Význam
3	IOBYTE	slabika IOBYTE (přepínání periferních zařízení počítače - viz bod 4.4), normální hodnota 69
4,5	MEMTOP	*maximální adresa paměti RWM RAM (pro 32 Kslabik RAM - 7FFF)
6	TIMKL	*časovač klávesnice
7	BLBLI	blokování blikání kurzoru (0-bliká, jinak nebliká)
8,9,A	TIME	čítač 20ms. Počáteční obsah je náhodný. Hodnota čítače se vypočítává takto: ([A]*256+[9])*256+[8]
B	BLIKZ	*kód znaku, na kterém stojí kurzor (viz poznámka dále)
C,D	KURSO	*adresa kurzoru v paměti VIDEO RAM
E	POZIC	*pozice kurzoru na řádce
F	RADEK	*číslo řádku, kde stojí kurzor

10	GRAF	status modu grafiky znaků zobrazovaných na obrazovce - v nultém bitu (1 - grafika)
11	INV1	status modu inverze znaků zobrazovaných na obrazovce - v nultém bitu (1 - inverze)
12	KIC	počet znaků od polohy kurzoru dokonce maximální délky řádku – počet posouvaných znaků v příkazech DC a IC
13	KOST	počet řádků na obrazovce použitelných pro výstup znaků (při změně této hodnoty nesmí být kurzor na řádku, který po změně již neleží na části obrazovky použitelné pro výstup znaků)
14	RCR	počet řádků vypsaných při výstupu znaku CR (hustota řádkování) - normální hodnota 2
15,16	ADRBR	adresa při přerušení tlačítkem BR
17,18	PBEEP	proměnné pro uložení hodnoty výšky tónu (17) a délky tónu (18) pro podprogram zvukového výstupu
19,1A	BUFR	adresa počátku vyrovnávací paměti pro nahrávání z magnetofonu
1B	DOBSE	7.bit - informace o polaritě při nahrávaní z magnetofonu, ostatní bity - hodnota pro určení 3/4 periody kmitočtu 1 kHz (nosný kmitočet magnetofonového záznamu)
1C	BTIME	hodnota určující délku meziblokové mezery (normální hodnota 21) pro vytváření magnetofonové nahrávky (pro dlouhé programy v jazyku BASIC nutno zvětšit, pro monitorovský příkaz W možno zmenšit na 2)
1D,1E	ARETN	návratová, adresa z monitoru (pro příkaz R)
1F	DRAD	počet znaků na řádku obrazovky (32 nebo 64 podle modulu VIDEO)
20,21	VIDEO	adresa začátku VIDEO RAM (EC00 pro VIDEO 32 nebo E800 pro VIDEO 64)

22	FSAVE	F	oblast pro uschovaní obsahu registrů mikroprocesoru
23	ASAVE	A	
24	CSAVE	C	
25	BSAVE	B	
26	ESAVE	E	
27	DSAVE	D	
28	LSAVE	L	
29	HSAVE	H	
2A,2B	SSAVE	SP	
2C,2D	PSAVE	PC	

Obsah proměnných označených * se nedoporučuje měnit.

Na konci paměti RWM RAM je umístěna tabulka s instrukcemi přerušení, tabulka odkazů na uživatelské periferie a zásobník pro monitor. Pokud operační paměť má velikost 32 Kslabik, jsou tabulky umístěny takto:

7FE0 - 7FFF - tabulka přerušení (pro každé přerušení 4 byty). Jsou zde umístěny skoky na začátky podprogramů zpracovávající jednotlivá přerušení.
(např. JMP START)
Adresy pro jednotlivá přerušení jsou tyto:
7FE0 - IRQ0
7FE4 - IRQ1
7FE8 - IRQ2
7FEC - IRQ3
7FF0 - IRQ4
7FF4 - IRQ5
7FF8 - IRQ6
7FF8 - IRQ6
7FFC - IRQ7

7FC2 - 7FDF - tabulka odkazů na uživatelské periferní zařízení (skoky na začátky podprogramů - 10 periferních zařízení). Pro každé periferní zařízení jsou rezervovány 3 byty a jsou umístěny takto:

7FC2 -	1. konzola vstup	CIU1
7FC5 -	2. konzola vstup	CIU2
7FC8 -	1. konzola výstup	COU1
7FCB -	2. konzola výstup	COU2
7FCE -	1. děrovač	POU1
7FD1 -	2. děrovač	POU2
7FD4 -	1. snímač	RIU1
7FD7 -	2. snímač	RIU2
7FDA -	1. tiskárna	LOU1
7FC1 -	2. tiskárna	LOU2

7FC0 - začátek zásobníku

Poznámka:
Do paměti VIDEO RAM se může zapisovat (zápisem kódu znaku do VIDEO RAM se příslušný znak zobrazí na obrazovce) a můžeme z ní také číst (zjišťovat, zda na daném místě obrazovky je zobrazen určitý znak). Znaky jsou v paměti uloženy po řádcích. Kódy znaků pro VIDEO RAM odpovídají kódu ASCII, s výjimkou řídících znaků, které jsou zde nahrazeny grafickými znaky (viz příloha 1). Znaky zobrazené inverzně mají v sedmém bitu slabiky jedničku.

4.4 Systém periferních zařízení

K periferním zařízením přistupuje monitor jako k "logickým" zařízením, definovaným v systému počítače. Takováto logická systémová zařízení jsou čtyři:

konzola - základní vstupní/výstupní zařízení sloužící ke komunikaci operátora se systémem. Skládá se z klávesnice a televizoru.

snímač - znakově orientované vstupní zařízení, které čte znaky z vnějšího média (např. snímač děrné pásky nebo magnetofon),

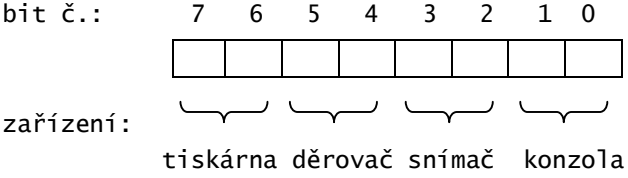
děrovač - znakově orientované výstupní zařízení, které ukládá znaky na vnější médium (např. děrovač děrné pásky nebo magnetofon),

tiskárna - znakově orientované výstupní zařízení, které zaznamenává znaky na vnější médium v čitelné podobě.

Tato čtyři logická systémová zařízení nemusí nutně znamenat čtyři různá skutečná fyzická zařízení, je možno jedno fyzické zařízení přiřadit více logickým zařízením současně, tzn. např. televizor může sloužit jako logická konzola i logická tiskárna současně. Podmínkou je, že zařízení musí odpovídat svojí funkcí požadavkům.

Pokud máme připojeno více periferních zařízení, lze realizovat přiřazovací mechanismus "mapování" mezi fyzickými a logickými zařízeními pomocí stavové slabiky přiřazení periferních zařízení, nazvané IOBYTE a umístěné na adrese 3 v paměti RWM RAM. Tento mechanismus umožňuje přiřadit jednomu logickému systémovému zařízení vždy jedno z maximálně čtyř možných fyzických zařízení. Volaná vstupní/výstupní operace (dále jen v/v operace) na logické zařízení se potom provede na přiřazeném fyzickém zařízení. Přiřazení lze kdykoliv programově měnit změnou hodnoty slabiky IOBYTE; její hodnota se kontroluje při každé v/v operaci, takže změny se okamžitě promítají do průběhu těchto operací.

Slabika IOBYTE popisuje stav přiřazení čtyř logických systémových zařízení, je tedy rozdělena do čtyř logických polí, každé o délce 2 bity, které reprezentují čtyři logická zařízení. Hodnota nastavená v každém tomto poli určuje vždy Jedno ze čtyř možných fyzických zařízení, které lze přiřadit právě tomuto logickému zařízení. Formát slabiky IOBYTE je:



Jednotlivá pole v této slabice mohou obsahovat hodnotu 0 - 3, která určuje přidělené fyzické zařízení.

Pro hodnoty 0 a 3 se v/v operace provádějí pomocí tzv. uživatelských periferních zařízení, tj. zařízení, které si uživatel připojil sám. Skoky na začátky podprogramů pro obsluhu těchto zařízení musí být umístěny na konci paměti RWM RAM (viz bod 4.3).

Jednotlivá zařízení jsou určena takto:

konzola - CIA a COA: bity 0 a 1
- 00 - CIU1: (vstup) nebo COU1: (výstup) - uživatelská konzola č. 1
- 01 - CI: a CO:, klávesnice a obrazovka
- 10 - RI: a LOA:, dávkové zpracování, kdy jako vstup z konzoly je přiřazen snímač děrné pásky, výstup na konzolu se provede na zařízení přiřazené jako logická tiskárna
- 11 - CIU2: (vstup) nebo COU2: (výstup) - uživatelská konzola č. 2

snímač - RIA:, bity 2 a 3
- 00 - RIU1:, uživatelem připojený snímač č. 1
- 01 - RI:, snímač děrné pásky
- 10 - MGRI:, magnetofon, vstup
- 11 - RIU2:, uživatelem připojený snímač č. 2

děrovač - POA:, bity 4 a 5
- 00 - POU1:, uživatelem připojený děrovač č. 1
- 01 - PO:, děrovač děrné pásky
- 10 - MGPO:, magnetofon, výstup
- 11 - POU2:, uživatelem připojený děrovač č. 2

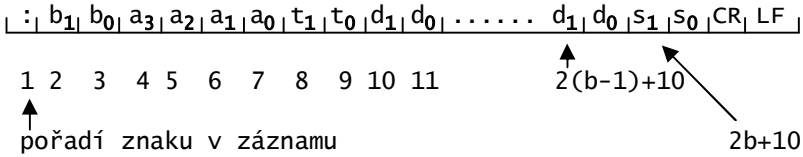
tiskárna- LOA:, bity 6 a 7
- 00 - LOU1:, uživatelem připojená tiskárna č. 1
- 01 - COA:, výstup na konzolu
- 10 - LO:, tiskárna
- 11 - LOU2:, uživatelem připojená tiskárna č. 2

4.5 Struktura souborů HEX

Soubory HEX slouží k uschování programových nebo datových modulů v absolutním tvaru. Médium pro uložení těchto souborů je magnetická nebo děrná páska. Soubor se na médium zaznamenává monitorovským příkazem W monitoru a čte se příkazem L (viz bod 4.2).

Soubory typu HEX obsahují strojový kód (nebo data) formátovaný do záznamů v kódu ASCII. Záznam začíná dvojtečkou a je ukončen znaky CR a LF (znak s kódem 0A, který se jinak nepoužívá). Každá slabika v záznamu je vyjádřena dvěma šestnáctkovými číslicemi, tj. znaky v kódu ASCII, které mohou být číslicemi 0 až 9 nebo písmeny A až F. Vyšší polovina slabiky (významnější 4 bity) je zobrazována jako první a nižší polovina slabiky (méně významné 4 bity) je zobrazována jako druhá. Obě šestnáctkové číslice tak dávají šestnáctkovou hodnotu slabiky 0 až FF (tj. 0 až 255 desítkově). Maximální počet datových slabik v záznamu je 50H (80 v desítkové soustavě). Soubory typu HEX jsou kompatibilní se soubory HEX používanými v operačním systému CPM nebo jinými operačními systémy používanými u osobních počítačů.

Struktura záznamu:



kde značí:

: Identifikace začátku záznamu (hodnota 3AH)

b_1b_0 Délka datové části záznamu, tj. počet datových slabik, které záznam obsahuje. Nulová délka označuje koncový záznam souboru a dále říká, že následující hodnota $a_3a_2a_1a_0$ představuje startovací adresu (pokud je různá od nuly)

$a_3a_2a_1a_0$ Hodnota, které u datového záznamu (délka b_1b_0 je různá od nuly) představuje ukládací adresu první datové slabiky záznamu. V případě, že délka b_1b_0 je nulová a hodnota $a_3a_2a_1a_0$ různá od nuly, představuje startovací adresu. V případě, že délka i adresa jsou nulové, předpokládá se, že startovací adresa nebyla zadána.

t_1t_0 Typ záznamu. Všechny datové záznamy mají typ 0. Koncový záznam souboru může být typu 1, identifikován je však podle nulové délky záznamu.

d_1d_0 Datové slabiky.

s_1s_0 Kontrolní součet. Inverze součtu všech slabik v záznamu počínaje slabikou délky a konče poslední datovou slabikou. Součet se provádí modulo 256. Součet všech slabik, včetně kontrolního součtu, musí být tudíž nula. Jedná se o součet skutečných (vnitřních) hodnot, nikoli o součet kódů vnějšího znakového vyjádření.

4.6 Adresy důležitých podprogramů v monitoru

Podprogramy pro vstup a výstup znaků začínají na těchto adresách:

F803 - vstup z klávesnice
F806 - vstup ze snímače
F809 - výstup na obrazovku
F80C - výstup na děrovač
F80F - výstup na tiskárnu
F812 - vstup z magnetofonu
F815 - výstup na magnetofon
F62B - vstup znaku z logické konzoly
F64B - výstup znaku na logickou konzolu
F66B - výstup znaku na logický děrovač
F689 - vstup znaku z logického snímače
F6AB - výstup znaku na logickou tiskárnu

U výstupních podprogramů musí být vystupující znak umístěn v registru C (obsah registru A je zničen), u vstupních podprogramů je čtený znak v registru A.

4.7 Přehled bran (adres periferních zařízení)

80 - obvod 3212
bit 0 - přeadresování paměti o adrese F800 až FFFF
na adresu 0 až 7FF při inicializaci
84 - brána A obvodu 8255 - dekódování klávesnice
85 - brána B obvodu 8255 - dekódování klávesnice
86 - brána C obvodu 8255
bit 0 - výstup na magnetofon
bit 1 - ovládání magnetofonu 1 (není vyvedeno na konektor)
bit 2 - ovládaní magnetofonu 2 (není vyvedeno na konektor)
bit 3 - výstup na reproduktor
bit 4 - vstup tlačítko SH
bit 5 - vstup tlačítko CTRL
bit 6 - vstup tlačítko FA
bit 7 - vstup tlačítko FB
Pokud je 1. nebo 2. bit roven jedné, mají bity 4 až 7
tento význam:
bit 4 - zem
bit 5 - vstup kmitočtu 1 kHz
bit 6 - vstup z magnetofonu 2
bit 7 - vstup z magnetofonu 1
87 - řídící registr obvodu 8255
88 - obvod 8259 - zápis inicializačního příkazového slova ICW1
89 - obvod 8259 - zápis ostatních inicializačních slov
a operačního slova
Registr žádostí o přerušení (IRR) je zapojen takto:
bity 0 až 4 - volné - vyvedeny na konektor sběrnice
bit 5 - tlačítko BR
bit 6 - kmitočet 50 Hz
bit 7 - kmitočet 16 kHz

D0 ┐
D1 │
D2 ├ ovládání modulu GRAFIK (viz kapitola 8)
D3 │
D4 ┘

F0 ┐
F1 ├ ovládání souřadnicového zapisovače
F2 │
F3 ┘

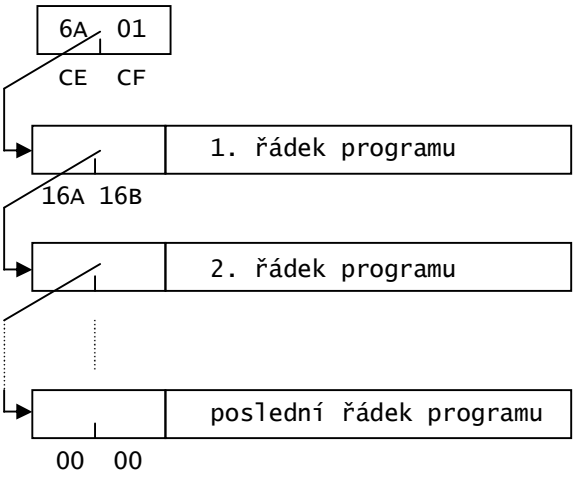
F8 - brána A obvodu 8255 - vstup ze snímače
F9 - brána B obvodu 8255 - výstup na tiskárnu nebo děrovač
FA - brána C obvodu 8255 - řídicí signály pro řízení operací, vstupu a výstupu modulu STAPER (obvod pracuje v režimu 1)
FB - řídící registr obvodu 8255

FC ┐
FD ├ modul VIDEO 64 (viz kapitola 6)
FE │
FF ┘

5 BASIC 6 - ULOŽENÍ PROGRAMU A DAT V PAMĚTI, SPECIÁLNÍ PŘÍKAZY

5.1 Způsob uložení programu v operační paměti

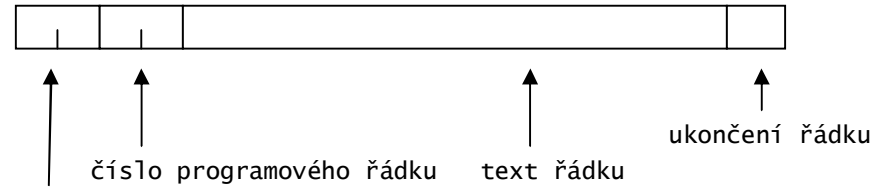
Program v jazyku BASIC je uložen v operační paměti ve tvaru zřetězeného seznamu. Každá položka seznamu je tvořena jedním řádkem programu. Položky seznamu jsou uloženy v pořadí rostoucích čísel příkazů. Adresa první položky je ve dvojici slabik o adresách CE a CF (nižší řády adresy jsou na adrese CE). Poslední položka seznamu je dvojice slabik s nulovou hodnotou. Schéma seznamu je uvedeno na následujícím obrázku.



Odkaz na další položku seznamu (další příkaz) jsou tvořeny adresou umístění další položky v operační paměti. Tento odkaz je opět dvojice slabik (nižší řády adresy jsou uloženy na nižší adrese) a je umístěn na počátku položky. První položka seznamu je umístěna na adrese 16AH (pokud není do počítače zasunut modul pro připojení souřadnicového zapisovače).

Operační paměť od adresy 100H do adresy 169H slouží jako vyrovnávací paměť pro magnetofon. Proměnné překladače BASIC jsou umístěny na adresách 45H až FFH.

Jednotlivé položky seznamu mají tuto strukturu:



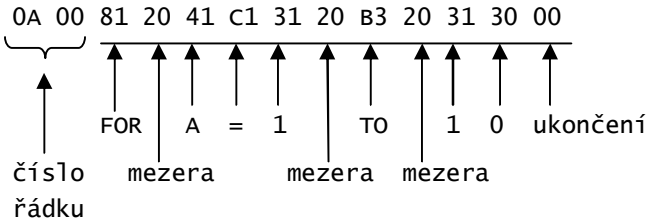
Číslo řádku je v položce uloženo ve dvou slabikách (v šestnáctkové soustavě). Nižší řády čísla jsou opět uloženy na nižší adrese. Ukončení řádku tvoří jedna slabika s nulovým obsahem. V jednotlivých slabikách textu řádku jsou umístěny kódy lexikálních symbolů jazyka. Tyto symboly mají vlastní kód (viz příloha 5 pro BASIC 6 a příloha 6 pro BASIC G) nebo v případě identifikátorů, konstant a některých dalších znaků jsou v kódu ASCII.

Upozornění:
BASIC G (viz kapitola 9) má jiné kódy lexikálních symbolů.

Příklad:
Příkaz

```
10 FOR A=1 TO 10
```

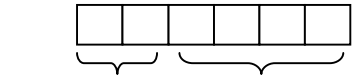
je uložen takto:



Při změnách programu se tento seznam udržuje stále tak, aby zabíral co nejméně místa (při zrušení některého příkazu se položky s dalšími příkazy posunou dopředu).

5.2 Uložení jednoduchých proměnných v paměti

Jednoduché proměnné jsou v operační paměti uloženy ihned za programem. Počáteční adresa uložení první proměnné je ve dvojici slabik o adresách D0 a D1 (nižší řády adresy na D0). Každá proměnná v paměti zabírá 6 slabik takto:

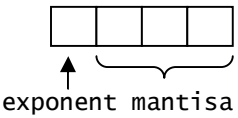


Proměnná je tedy v paměti uložena jako dvojice: identifikátor proměnné, hodnota proměnné. Identifikátor proměnné je umístěn po znacích v kódu ASCII ve dvou slabikách (znaky jsou v opačném pořadí). Má-li identifikátor délku jednoho znaku, je hodnota první slabiky nulová. Takto jsou uloženy identifikátory číselných proměnných. U textových proměnných jsou identifikátory uloženy stejně s tím, že kód znaku uložený v první slabice je zvětšen o 80H.

Příklady:

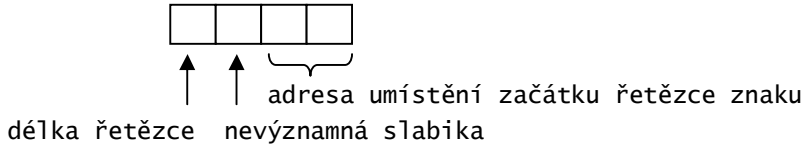
Identifikátor	Způsob uložení v paměti
A	00 41
AB	42 41
AB$	C2 41
A$	80 41

Hodnota číselné proměnné je uložena ve čtyřech slabikách a je vždy uváděna v pohyblivé řádové čárce. Slabiky jsou uloženy v obráceném pořadí takto:



Exponent je uváděn v aditivním kódu a mantisa v přímém kódu (nejvýznamnější bit mantisy je znaménko čísla). Vzhledem k tomu, že po normalizaci má nejvýznamnější bit mantisy (pokud neuvažujeme znaménko) vždy hodnotu 1, není tento bit zapamatováván.

Hodnota textové proměnné má tuto strukturu:



Vlastní řetězce znaků jsou umístěny v oblasti STRING. Výjimku tvoří přiřazení textové konstanty textové proměnné. Zde se využívá řetězec znaků z textu programu.

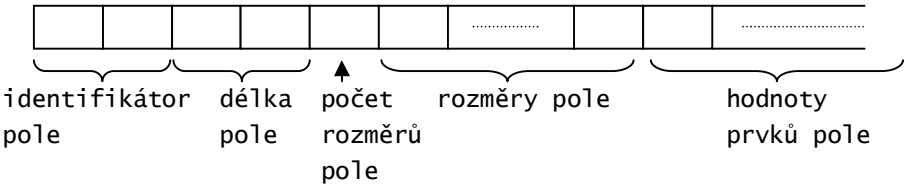
Tato struktura uložení proměnných se vytvoří při spuštění programu příkazem RUN a zůstane zachována do té doby, než je program zrušen nebo než je v programu provedena libovolná změna, tzn. dokud v programu neprovedeme změnu, můžeme hodnoty proměnných používat.

Pořadí uložení jednotlivých proměnných v operační paměti odpovídá pořadí jejich výskytu v programu. Po provedení příkazu RUN jsou všechny číselné proměnny vynulovány a všechny textové proměnné obsahují prázdný řetězec.

5.3 Uložení polí v operační paměti

Pole jsou v operační paměti uložena ihned za jednoduchými proměnnými. Počáteční adresa prvního pole je ve dvojici slabik o adresách D2 a D3. Pole se vytvářejí až po spuštění programu a jsou v paměti v pořadí jejich výskytu v prováděném programu. Koncová adresa polí je ve dvojici slabik o adresách D4 a D5.

Struktura uložení pole:



Identifikátor pole je uložen ve dvou slabikách stejně jako identifikátor jednoduché proměnné. Další dvojice slabik obsahuje informaci o velikosti paměti potřebné pro uložení pole. Je to počet slabik od následující slabiky (počet rozměrů pole) do poslední slabiky, kde je uložena hodnota posledního prvku pole. Celkový počet slabik pro uložení pole je o 4 větší než je tato hodnota. Počet rozměrů pole je uložen v jedné slabice. Dále následují dvojice slabik pro uložení rozměrů pole (počet dvojic slabik odpovídá počtu rozměrů pole). Tyto dvojice jsou zde v opačném pořadí (prvnímu indexu pole přísluší poslední dvojice slabik). Hodnoty uložené v těchto dvojicích jsou maximální hodnoty indexů zvětšené o 1. Ve zbývajících slabikách jsou uloženy hodnoty prvků pole. Hodnota každého prvku pole je vždy ve čtveřici slabik (stejně jako hodnota jednoduché proměnné). Prvky pole jsou v paměti uloženy v pořadí rostoucích hodnot indexů, kde nejrychleji se mění první index, tzn. např. u dvojrozměrného pole jsou prvky uloženy po sloupcích.

5.4 Oblast USR a oblast STRING

Oblast USR je určena pro ukládání podprogramů ve strojovém kódu. Tato oblast je umístěna před koncem operační paměti. Vytvoří se příkazem CLEAR (viz bod 2.29). Pokud příkaz CLEAR nepoužijeme, má tato oblast nulovou délku. Počáteční adresa oblasti USR je uložena ve dvojici slabik o adresách A4 a A5 a koncová adresa na adresách A6 a A7. V oblasti USR jsou podprogramy chráněny před zničením překladačem BASICu (neukládá se sem text programu ani data).

Před oblastí USR je umístěna oblast STRING (pro uložení textových proměnných) a dále následuje zásobník pro BASIC. Počáteční adresa zásobníku je uložena ve dvojici slabik o adresách CC a CD.

5.5 Obnovení programu po jeho smazání

Jestliže omylem smažeme program příkazem SCRATCH nebo stisknutím tlačítka RES (při smazání programu je vynulován odkaz na další příkaz u prvního příkazu programu), můžeme jej obnovit tímto způsobem:

1. obnovíme odkaz na další příkaz u prvního příkazu programu (pomocí příkazů monitoru),
2. provedeme příkaz LIST (není ještě možno napsat příkaz RUN, neboť adresy začátku proměnných a polí odpovídají adrese začátku programu a proměnné by přepsaly program),
3. napíšeme znovu poslední příkaz programu (obnoví se adresy začátku proměnných a polí - nyní nelze použít příkaz LIST, neboť seznam příkazů není ukončen),
4. v monitoru zapíšeme ukončení seznamu příkazů (vynulujeme dvě slabiky před začátkem proměnných).

Dále je již možno program libovolně používat.

5.6 Zjištění adresy uložení hodnoty proměnné

Adresu umístění hodnoty proměnné v operační paměti můžeme zjistit pomocí standardní funkce

PTR(i),

kde i je identifikátor proměnné.

Tato adresa je adresou první ze čtveřice slabik, ve kterých je umístěna hodnota proměnné (u textových proměnných délka). Standardní funkci PTR můžeme použít i pro zjištění umístění prvku pole. Vzhledem k tomu, že PTR je standardní funkce, můžeme ji používat v aritmetických výrazech stejně jako jiné standardní funkce.

Příklady použití:

```
B=PTR(A)
C=PTR(A$)+2
D=PTR(X(1,7))
```

5.7 Zjištění hodnoty a změna hodnoty slabiky paměti

Pro zjištění hodnoty slabiky lze použít standardní funkci

```
PEEK(av)
```

kde av je aritmetický výraz určující desítkovou adresu této slabiky (bere se celá část hodnoty výrazu, musí ležet v intervalu <0;65535>).

Touto funkcí můžeme zjišťovat obsah libovolné slabiky z paměti ROM nebo RWM RAM. Vzhledem k tomu, že adresa v této funkci se musí zadávat v desítkové soustavě a my ji můžeme znát v šestnáctkové soustavě, můžeme použít další funkci

```
HEX(H)
```

která slouží pro převod čísel ze šestnáctkové do desítkové soustavy. Argumentem této funkce je číselná konstanta v šestnáctkové soustavě z intervalu <0;FFFF>.

Příklady použití:

```
A=PEEK(HEX(EC00))
B=(PEEK(10)*256+PEEK(9))*256+PEEK(8)
```

První příkaz přiřadí do proměnné A kód znaku, který je zobrazen v levém horním rohu obrazovky a druhý příkaz přiřadí proměnné B hodnotu času, který je vytvářen počítačem (v padesátinách sekundy). Zde použité adresy jsou uvedeny v bodě 4.3

Změnit hodnotu slabiky můžeme příkazem

```
POKE av1,av2
```

kde av1 je aritmetický výraz, jehož hodnota určuje adresu slabiky, jejíž hodnotu měníme (bere se celá část hodnoty výrazu, musí ležet v intervalu <0,65535>),

av2 je aritmetický výraz určující novou hodnotu slabiky (bere se celá část výrazu, musí ležet v intervalu <0;255>).

Příklad použití:

```
POKE HEX(EC00),ASC("A")
```

Tento příkaz zobrazí písmeno A v levém horním rohu obrazovky.

5.8 Použití podprogramů ve strojovém kódu

Vyvolat podprogram ve strojovém kódu můžeme příkazem, jehož obecný tvar je

CALL va,vp_1,...,vp_k

kde va je aritmetický výraz, jehož hodnota určuje adresu začátku podprogramu a

vp_1,...,vp_k (k>=0) je seznam aritmetických výrazů, jejichž hodnoty určují parametry podprogramu.

Při vyvolání podprogramu se z hodnot výrazů bere celá část a tyto celé části musí ležet v intervalu <0;65535>. Příkaz CALL nemusí obsahovat parametry podprogramu a pak má tvar

CALL av

Obsahuje-li příkaz CALL parametry podprogramu, uloží se poslední parametr vp_k do registrového páru DE, předposlední parametr vp_{k-1} do BC a zbytek parametrů se uloží do zásobníku tak, že na vrcholu zásobníku je návratová adresa a pak následují po řadě parametry vpi, přičemž parametr vp1 je uložen nejhlouběji. Výjimku tvoří příkaz CALL, který obsahuje pouze jeden parametr. Tento parametr je vložen do registrového páru BC.

Někdy potřebujeme při návratu z podprogramu ve strojovém kódu předat do programu v BASICu nějaké informace. V tomto případě lze použít funkce, jejichž obecný tvar je

USR(va)

BYTE(va,vp_1,...,vp_k)

WORD(va,vp_1,...,vp_k)

kde význam va, vp_1,...,vp_k je stejný jako u příkazu CALL.

Hodnotou funkcí USR a BYTE je hodnota uložená ve střadači A při návratu z podprogramu. Hodnotou funkce WORD je hodnota uložená v registrovém páru HL při návratu. Tyto funkce se mohou libovolně používat v aritmetických výrazech jako jiné standardní funkce.

5.9 Nestandardní vstupy a výstupy

Nestandardní ovládání vstupních a výstupních zařízení můžeme provádět funkcí INP a příkazy OUT a GET. Standardní funkce

INP(av)

kde av je aritmetický výraz (z hodnoty výrazu se bere celá část, musí ležet v intervalu <0;255>), který určuje adresu vstupního periferního zařízení (viz bod 4.7),

má hodnotu, která je připravena na výstupu tohoto zařízení. Přenáší se vždy celá slabika, tzn. hodnota funkce INP leží v intervalu <0;255>.

Příkaz OUT pracuje podobně jako funkce INP, ale týká se výstupních periferních zařízení. Obecný tvar příkazu je

OUT va,vh

kde va je aritmetický výraz, jehož hodnota určuje adresu výstupního zařízení (podobně jako u funkce INP(av)),

vh je aritmetický výraz, jehož celá část hodnoty (vh je v <0;255>) se pošle na zadané výstupní zařízení.

Pro vstup lze také použít příkaz GET, který má obecný tvar

GET va,vp1,vp2

kde va, vp1, vp2 jsou aritmetické výrazy (bere se pouze celá část hodnoty výrazu - musí ležet v intervalu <0;255>),

Parametr va určuje adresu vstupního periferního zařízení, parametr vp2 je maska a parametr vp1 je požadovaná hodnota.

Příkaz pracuje takto:

1. ze vstupního zařízení va se přečte jedna slabika,
2. tato slabika se logicky vynásobí maskou vp2,
3. výsledek se porovná s hodnotou vp1, není-li rovnost, celý proces se opakuje,
4. při rovnosti se přejde na další příkaz.

Tento příkaz užíváme při čekání, až některý z bitů čtené slabiky se změní na požadovanou hodnotu.

6 PŘÍDAVNÝ MODUL VIDEO 64

6.1 Úvod

Modul VIDEO 64 je určen k zobrazování informací na obrazovce zobrazovacího monitoru. Je možné použít i běžný televizní přijímač, ale zobrazované údaje jsou špatně čitelné. Formát zobrazovaných znaků je 6x8 bodů (včetně meziznakové a meziřádkové mezery - modul VIDEO 32 8x8 bodů). Tento modul nahrazuje v počítači IQ 151 modul VIDEO 32 s tím, že umožňuje zobrazovat na obrazovce připojeného zobrazovacího monitoru větší počet informací než předchozí typ. V počítači nesmí být oba tyto moduly zasunuty současně. Při použití modulu VIDEO 64 je možno zobrazit 64 znaků na každém z 32 možných řádku. Technické řešení modulu umožňuje zobrazovat i znaky o dvojnásobné šířce, tj. 32 znaků na řádek. V tomto případě ale na obrazovce vidíme každý druhý zobrazovaný znak.

6.2 Návod k obsluze modulu VIDEO 64

Při zobrazování na obrazovku se modul VIDEO 64 jeví jako blok paměti, který je přístupný na adresách E800 až EFFF. Přítomnost modulu VIDEO 64 v počítači je možno zjistit pomoci čtení dat z některé vstupní brány s adresou FC až FF (všechny adresy jsou ekvivalentní). Přítomnost modulu VIDEO 64 se projeví předáním datového slova o hodnotě FE, nepřítomnost slovem FF. V jazyku BASIC je možno pro zjištění přítomnosti modulu VIDEO 64 použít příkazy:

```
A=INP(HEX(FF))
IF A=HEX(FE) THEN ...
```

Na stejných adresách je možno povolit (zápisem hodnoty 01) nebo zakázat (zápisem hodnoty 00) změnu formátu zobrazení (tj. 32 znaků na řádek). Změna formátu zobrazení na dvojnásobnou šířku znaku se provádí zapsáním znaku ▉ (kód znaku 7F) do paměti před znaky, které mají být zobrazeny v dvojnásobné šířce.

Při zobrazování na obrazovku (po nalezení tohoto znaku) jsou z paměti modulu vybírány pro zobrazení znaky vždy ob jednu adresu. Je tedy zobrazen pouze každý druhý znak. Návrat k normální hustotě zobrazení (64 znaků na řádku) se děje:

a) zapsáním znaku | (kód znaku 7C),
b) zapsáním znaku CR (kód znaku 0D),
c) koncem řádku na obrazovce.

Řídící, znaky pro ukončení změny formátu zobrazení musí být uloženy na příslušné pozice paměti VIDEO RAM. V případě že změna formátu není povolena (např. po zapnutí počítače) zobrazují se řídící znaky jako mezery, ale ke změně formátu zobrazení nedochází. Řídící znaky jsou oproti modulu VIDEO 32 vyňaty ze souboru zobrazovaných znaků a jsou v přímém a inverzním režimu zobrazování nahrazeny mezerou.

6.3 Odchylky v používání modulu BASIC 6 při použití modulu VIDEO 64

Změny v jazyku BASIC se týkají příkazů výstupu na obrazovku. Při použití příkazu PRINT je možno na řádek obrazovky zobrazit 64 znaků. U příkazu PRINT&r,s může hodnota výrazu s ležet v intervalu <0;63>. Obdobně u příkazů PLOT a UNPLOT může x-ová souřadnice (první parametr příkazu) nabývat hodnot z intervalu <0;127>. Zobrazované body u příkazu PLOT mají obdélníkový tvar.

7 SOUŘADNICOVÉ ZAPISOVAČE

7.1 Úvod

K počítači IQ 151 je možno připojit několik typů souřadnicových zapisovačů (grafických jednotek). Jedná se o grafické jednotky XY 4120, XY 4130, XY 4131 a MINIGRAF 0507. My se zde budeme podrobněji zabývat grafickou jednotkou XY 4131 a MINIGRAFem 0507 (verze 86). Odchylky pro ovládání ostatních souřadnicových zapisovačů jsou stručně uvedeny v dalším textu.

7.2 Popis a základní technické údaje grafické jednotky XY 4131

Záznamová deska je vodorovná a je umístěna na horní ploše přístroje. Do konektoru INPUT se zasouvá kabel od modulu MS 151, který slouží pro vlastní připojení grafické jednotky k počítači IQ 151.

Ovládací prvky grafické jednotky:
POWER - hlavní vypínač síťového přívodu,
PEN - ruční ovládání záznamového pera.

Základní technické údaje:

formát záznamového papíru	A4
velikost kroku pohybu záznam. Pera	0,1 mm
záznamová plocha papíru	250 x 175 mm tj. 2500 x 1750 kroků
rychlost zápisu	programovatelná: 40, 60, 80, 100 mm/s
velikost a sklon písma	určuje se programem
vypisované znaky	znaky z kódu ASCII a další speciální znaky
ovládání záznamového pera	programem nebo spínačem
nulová poloha záznamového pera	vlevo dole

Postup při zapojování zapisovače.

1. Kabel od modulu zasuneme do konektoru INPUT na grafické jednotce.
2. Není-li vypnut síťový vypínač u počítače IQ 151 vypneme jej.
3. Modul grafické jednotky MS 151 zasuneme do libovolné pozice v počítači IQ 151.
4. Zapneme síťový vypínač u počítače IQ 151.
5. Zapneme síťový vypínač POWER grafické jednotky. Držák pisátka se přemístí automaticky do levého dolního rohu záznamové plochy (počátek souřadnic).
6. Založíme záznamový papír. Při zakládání papíru postupujeme takto: Záznamové pero může být v libovolné poloze. Stlačíme ovládací páčku umístěnou vpravo vzadu, tím se zdvihnou gumové přítlačné válečky. Záznamový papír podstrčíme pod most tak daleko, až je jeho okraj asi 5 mm za výřezem pro pohonné válečky. Poté srovnáme delší okraj záznamového papíru podle linky vpravo na záznamové ploše.
7. Zasuneme záznamové pero do držáku na vozíku (provádíme bez násilí, jinak může dojít k deformaci celého vozíku). Záznamové pero zasuneme do držáku pouze tak hluboko, aby ve zdvižené poloze byl hrot záznamového pera asi 1 mm nad plochou záznamového papíru.
8. Zkontrolujeme funkci ručního ovládání (spouštění) záznamového pera tlačítkem PEN (nejprve je nutno zadat příkaz VECTR 0,0).
9. Překontrolujeme funkci celého přístroje testovacím programem.

Grafická jednotka pro svoji práci využívá program umístěný v paměti počítače od adresy C000 do adresy C7FF.

Grafická jednotka XY 4120 se od XY 4131 liší jen velikostí záznamové plochy (340x270 mm), rychlostí zápisu (50 až 200 mm/s) a způsobem zakládání papíru. Popis a ovládání grafické jednotky XY 4130 se neliší od XY 4131.

7.3 Příkazy jazyka BASIC pro ovládání grafické jednotky XY 4131

V tomto bodě jsou uvedeny příkazy jazyka BASIC 6 pro ovládání grafické jednotky XY 4131 (pro jednotky XY 4120 a XY 4130 se používají stejné příkazy). Jedná se o 12 příkazů, z nichž většina obsahuje parametry. Jednotlivé parametry se oddělují čárkou. Každý z parametrů může být aritmetický výraz. Z hodnoty aritmetického výrazu se vždy bere pouze celá část (provede se funkce INT).

7.3.1 Příkaz ORG

Pomocí tohoto příkazu volíme počátek pravoúhlé souřadné soustavy na ploše záznamového papíru. Příkaz neprovádí posun pera, pouze ukládá počátek souřadné soustavy do paměti počítače. Příkaz má dva parametry: x-ovou a y-ovou souřadnici počátku.

Příklady:

```
ORG 0,0         počátek je v dolním levém rohu záznamového papíru,
ORG 1250,875    počátek je uprostřed papíru,
ORG 2500,1750   počátek je v horním pravém rohu papíru.
```

Hodnoty parametrů (souřadnic) se udávají v počtu kroků, tj. v 0,1 mm, a musí ležet v intervalu <-16383;16383>. Toto platí i pro příkazy MOVA, MOVR, VECTA, VECTR, POINTA a POINTR, které jsou popsány dále. Počátek souřadné soustavy je možno volit i mimo plochu záznamového papíru.

7.3.2 Příkaz MOVA

Tento příkaz způsobí přesun záznamového pera do bodu o souřadnicích určených parametry příkazu v souladu s počátkem souřadné soustavy určené příkazem ORG. Při přesunu do nové polohy pera se nic nekreslí, jde o posuv zdviženého pera.

Příklad:
Příkazy

```
ORG 0,0: MOVA 0,0: MOVA 2500,1750
```

způsobí posuv záznamového pera po úhlopříčce záznamového papíru z levého dolního rohu do pravého horního rohu.

7.3.3 Příkaz MOVR

Příkaz pracuje podobně jako příkaz MOVA s tím rozdílem, že souřadnice nové polohy záznamového pera jsou určeny relativně vzhledem k poloze pera před provedením tohoto příkazu. Opět se jedná o přesun zdviženého záznamového pera a neprobíhá tedy kreslení.

Příklad:
Stojí-li záznamové pero uprostřed papíru, tj. v bodě o souřadnicích [1250,875] (je-li počátek souřadnic v levém dolním rohu), potom příkaz

```
MOVR -1250,-875
```

způsobí přesun záznamového pera do počátku souřadnic.

7.3.4 Příkaz VECTA

Příkaz pracuje podobně jako příkaz MOVA s tím rozdílem, že přesun probíhá se spuštěným záznamovým perem, tzn. kreslí se čára.

Příklad:
Příkazy

```
ORG 0,0: MOVA 0,1750: VECTA 2500,0
```

způsobí nakreslení úhlopříčky z levého horního rohu záznamového papíru do pravého dolního rohu.

7.3.5 Příkaz VECTR

Činnost je podobná jako u příkazu VECTA s tím rozdílem, že souřadnice koncového bodu jsou určeny relativně vzhledem k poloze počátečního bodu kreslené čáry. Jde tedy o obdobný rozdíl jako je mezi příkazy MOVA a MOVR s tím rozdílem, že u příkazu VECTA a VECTR dochází ke kreslení čáry.

7.3.6 Příkaz POINTA

Příkaz způsobí zakreslení bodu o souřadnicích určených parametry příkazu, kde počátek souřadné soustavy byl určen posledním provedeným příkazem ORG. Body jsou znázorněny tečkou. Záznamové pero po provedení příkazu stojí nad zakresleným bodem.

7.5.7 Příkaz POINTR

Příkaz způsobí zakreslení bodu jako příkaz POINTA s tím rozdílem, že parametry příkazu určují souřadnice bodu relativně vzhledem k poloze záznamového pera před provedením příkazu.

Příklad:
Následující program nakreslí na záznamovém papíře rámeček z plné čáry a uvnitř něho další rámeček tvořený z teček vzdálený 1 cm od stran původního rámečku (vzdálenost teček je 5 mm).

```
10 ORG 0,0:MOVA 0,0
20 VECTA 0,1750:VECTA 2500,1750:VECTA 2500,0:VECTA 0,0
30 FOR I=100 TO 1650 STEP 50:POINTA 100,I:NEXT I
40 FOR I=100 TO 2400 STEP 50:POINTA I,1650:NEXT I
50 FOR I=1650 TO 100 STEP -50:POINTA 2400,I:NEXT I
60 FOR I=2400 TO 100 STEP -50:POINTA I,100:NEXT I
```

Při použití relativního určování souřadnic (příkazy MOVR, VECTR, POINTR) je možno tento program přepsat takto:

```
10 ORG 0,0:MOVA 0,0
20 VECTR 0,1750:VECTR 2500,0:VECTR 0,-1750:VECTR -255,0
25 MOVR 100,100
30 FOR I=1 TO 31:POINTR 0,50:NEXT I
40 FOR I=1 TO 46:POINTR 50,0:NEXT I
50 FOR I=1 TO 31:POINTR 0,-50:NEXT I
60 FOR I=1 TO 46:POINTR -50,0:NEXT I
```

Oba tyto programy provádějí stejnou činnost. Při používání grafické jednotky je potřeba se snažit aby se neprováděly zbytečné pohyby záznamovým perem.

V prvním programu by se např. příkazy na řádcích 30 a 50 daly provádět v jednom cyklu:

```
FOR I=100 TO 1650 STEP 50:POINTA 100,I:POINTA 2400,I:NEXT I
```

ale nakreslení bodů by trvalo mnohem déle, neboť by se prováděl po nakreslení každého bodu zbytečný pohyb pera z levé strany doprava nebo opačným směrem.

7.3.8 Příkaz SPEED

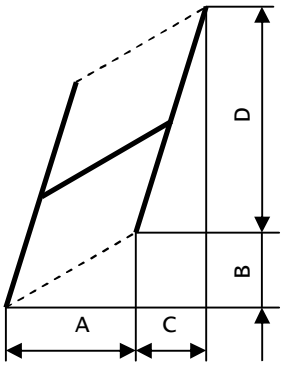
Příkaz obsahuje jediný parametr a určuje se jím rychlost pohybu záznamového pera. Hodnota parametru příkazu (po provedení příkazu INT) musí ležet v intervalu (1,5). Nejpomalejší kreslení nastává po provedení příkazu

```
SPEED 1
```

Zpomalení kreslení je možno využít v případě, kdy potřebujeme vyšší kvalitu kreslení.

7.3.9 Příkaz SIZE

Tento příkaz určuje velikost a sklon znaků, které bude grafická jednotka vypisovat pomocí příkazu WRITE. Příkaz má 4 parametry, které si označíme A, B, C, D a jejich význam je uveden na následujícím obrázku pro písmeno H.



Parametry určují na obrázku okótované rozměry znaků v krocích, tzn. příkaz SIZE 30,0,0,50 zavádí kolmé písmo psané vodorovně o rozměrech 3x5 mm a příkaz SIZE 0,30,50,0 zavede kolmé písmo psané svisle.

Parametry mohou být také záporné. Toto lze využít pro výpis textů pomoci skloněných nebo obrácených znaků.

Příkaz SIZE lze použít také bez parametrů. V tomto případě se provede stejná činnost jako v případě použití příkazu s parametry SIZE 24,0,0,40

7.3.10 Příkaz WRITE

Příkaz způsobí výpis textu, který je určen parametrem tohoto příkazu. Parametrem zde může být libovolný textový výraz. Umístění vypisovaného textu záleží na poloze záznamového pera před provedením tohoto příkazu. Spodní levý roh prvního vypisovaného znaku je umístěn v bodě, kde stojí záznamové pero. Velikost a sklon znaků je určen posledním provedeným příkazem SIZE. Interpretace kódu ASCII je pro grafickou jednotku částečně upravena (viz příloha 7). Speciální znaky jsou při výpisu umístěny tak, že jejich střed je umístěn v bodě, kde stojí záznamové pero. Didaktická znaménka jsou umístěna nad písmenem s tím, že po jejich zapsání se vrátí záznamové pero do původní polohy (neprovádí se posuv jako u ostatních znaků na začátek dalšího znaku). Pokud je ve vypisovaném textu znak CR, je záznamové pero přesunuto na začátek dalšího řádku (pod začátek vypisovaného textu).

Příklad:
Příkaz

```
WRITE CHR$(24)+"A"+CHR$(31)+"u"
```

způsobí výpis znaků

```
Áů
```

7.3.11 Příkaz WIDE

Příkaz je bez parametrů a způsobuje zvětšení mezery za užšími znaky ve vypisovaném textu. Vzdálenost spodního levého rohu u následujících dvou znaků je vždy stejná. Toto platí pro následující příkazy WRITE.

7.3.12 Příkaz NARROW

Tento příkaz je také bez parametrů a způsobuje, že ve vypisovaném textu je šířka mezery mezi znaky stále stejná, tzn. vzdálenost spodních levých rohů následujících dvou znaků závisí na šířce znaku.

Po zapnutí grafické jednotky a počítače jsou provedeny tyto příkazy:

```
ORG 0,0: MOVA 0,0: SPEED 3: NARROW: SIZE 24,0,0,40
```

7.4 Základní technické údaje MINIGRAFu 0507

MINIGRAF 0507 je jednoduchý souřadnicový zapisovač, který pracuje v soustavě pravoúhlých souřadnic v rozsahu formátu A4. Pro připojení MINIGRAFu 0507 k počítači se používá modul 0509.

Základní technické údaje:

délka kroku záznamového pera	0,125 mm (1/8 mm)
rychlost kreslení	programovatelná - až 100 mm/s
záznamová plocha papíru	187,5x262,5 mm tj. 1500x2100 kroků
velikost a sklon písma	určuje se programem
vypisované znaky	znaky kódu ASCII a speciální znaky definované uživatelem
nulová poloha záznamového pera	vlevo dole

Zde uvedené údaje platí pro MINIGRAFy 0507 vyráběné od roku 1986. Dřívější verze mají menší rychlost kreslení (asi poloviční).

7.5 Příkazy pro ovládání MINIGRAFu 0507

Starší verze MINIGRAFu 0507 (vyráběné v roce 1985) se ovládají jazykem BASIC pomocí příkazů CALL. Vzhledem k tomu, že těchto souřadnicových zapisovačů bylo vyrobeno pouze 100 kusů, popíšeme jejich ovládání jen stručně.

Jedná se o tyto příkazy:

CALL 49190 - zajistí synchronizaci motorků s polohou záznamového pera, časování a standardní parametry pro tisk.

CALL 49548,x,y - odpovídá příkazu MOVA x,y.

CALL 49542,x,y - odpovídá příkazu VECTA x,y.

CALL 49555,x,y - odpovídá příkazu POINTA x,y.

CALL 49234,v,s - nastaví velikost písma "v" (písmo pouze kolmé) a směr psaní "s" (4 kolmé základní směry).

CALL 49245,PTR(i$) - zajistí vypsání textového řetězce i$.

CALL 49682,r nebo CALL 49705,r - provedou opis r řádků z obrazovky na MINIGRAFu.

Pro ovládání novější verze souřadnicového zapisovače se používají příkazy jazyka BASIC 6, podobně jako u zapisovače XY 4131. Příkazy MOVA, MOVR, VECTA, VECTR, POINTA, POINTR, SPEED a WRITE mají stejnou funkci jako u zapisovače XY 4131 (viz bod 7.3). Liší se pouze velikostí kroku, který je zde 0,125 mm. Ostatní příkazy provádějí částečně odlišnou funkci a jsou zde popsány podrobně.

7.5.1 Příkaz ORG xa,ya

Příkaz ORG se musí provést vždy po zapnutí MINIGRAFu po založení papíru, před použitím ostatních příkazů. ORG předpokládá papír založený tak, že jeho horní hrana se kryje s horní hranou otevřené zadní klopny. ORG zajistí synchronizaci motorků s polohou záznamového pera evidovanou v paměti. vzhledem k souřadným osám. Přesune záznamové pere vlevo "na doraz" (aby byla definována x-ová souřadnice) a pak jej nastaví na bod určený parametry tohoto příkazu (příkaz se zadává ve tvaru ORG xa, ya, kde xa a ya jsou libovolné aritmetické výrazy, určující souřadnice počáteční polohy záznamového pera v počtu kroků).

Příkaz ORG též nastaví konstanty pro správné časování motorků, rychlost a parametry pro tisk takto:

```
SPEED 3
SIZE 3,0,0,3
NARROW
```

a nastaví diakritický režim - viz další popis.

Poznámka:
Při použití modulu 0509 - verze 86 s MINIGRAFEM 0507 - verze 85 je nutno vždy po příkazu ORG provést příkaz

```
POKE 389,20
```

pro omezení maximální rychlosti.

7.5.2 Soubor znaků

Základní programové vybavení pro MINIGRAF 0507 umožňuje vypisovat znaky kódu ASCII (s výjimkou znaků s kódem 0 až 1FH) a diakritická znaménka (čárku, kroužek, háček, vokáň a přehlásku). Netisknutelné znaky (znaky s kódem 0 až 1FH) jsou, kromě znaků s kódy 0 a 8, nahrazeny mezerou. Znak s kódem 0 je rezervován pro uživatele, který si pomocí něho může v programu určit další znaky (např. azbuku nebo řeckou abecedu) - způsob použití je popsán dále. Znak s kódem 8 (CHR$(8)) označuje krok zpět a způsobí, že následující znak bude kreslen do stejného místa jako předcházející.

Uživatel může volit 2 režimy výpisu znaků:
- režim ASCII - tj. znaky kódu ASCII.
- diakritický režim - posledních 5 znaků kódu ASCII má jinou interpretaci (podle následující tabulky):

Kód znaku	Režim ASCII	Diakritický režim
7B	{	´ (čárka)
7C	|	° (kroužek)
7D	}	ˇ (háček)
7E	~	^ (vokáň)
7F		¨ (přehláska)

Režim je určen obsahem slabiky paměti o adrese 371 tak, že 0 znamená diakritický režim a 1 režim ASCII, tzn. že např. režim ASCII můžeme zvolit příkazem POKE 371,1 Příkaz ORG nastavuje diakritický režim. Výpis písmena s diakritickým znaménkem se zadává podobně jako na psacím stroji tak, že nejdříve vypíšeme diakritické znaménko (neprovede se posuv na místo výpisu dalšího znaku) a pak požadované písmeno.

7.5.3 Speciální znaky definované uživatelem

Pro psaní znakových řetězců poskytuje programové vybavení uživateli znaky kódu ASCII a diakritická znaménka. Kromě těchto znaků si může uživatel v programu v jazyku BASIC deklarovat prakticky libovolný počet dalších vlastních znaků.

Pro definici takového znaku je nutné nejprve nakreslit základní tvar (složený z úseček) do rastru 4x8 čtverečků tj. 5x9 bodů, přičemž nelze využít nejvyšší řádku, a jednotlivým vrcholům znaku (tj. koncovým bodům úseček) přiřadit čísla $a_1,a_2,...,a_n$ (kde n je počet vrcholů znaku) podle tabulky uvedené v příloze 8.

Pak uživatel deklaruje znakový řetězec, jehož prvním členem je CHR$(0), dalšími členy

$$CHR\$(a_1)\ ,CHR\$(a_2),...,CHR\$(a_n)$$

a posledním členem je CHR$(128), takto:

$$CHR\$(0)+CHR\$(a_1)+CHR\$(a_2)+...+CHR\$(a_n)+CHR\$(128).$$

Při vypisování takového řetězce příkazem WRITE pak bude nakreslen požadovaný znak vytvořený postupným spojováním vrcholů odpovídajících číslům $0,a_1,a_2,...,a_n$ (se záznamovým perem dole či nahoře - podle tabulky v příloze 8). Poslední člen řetězce - CHR$(128) - zajistí zvednutí záznamového pera a posun na začátek případného dalšího vypisovaného znaku.

Např. pro znak ± může být řetězec deklarován takto:

```
PM$=CHR$(0)+CHR$(17)+CHR$(83)+CHR$(35)+CHR$(97)+CHR$(42)+
    CHR$(90)+CHR$(128)
```

Pokud některým členům definičního řetězce odpovídají tisknutelné znaky kódu ASCII, je možno řetězec deklarovat stručněji přímo uvedením těchto tisknutelných znaků (jsou v tabulce z přílohy 8 uvedeny napravo od číselných kódů). Pro uvedený příklad lze tedy psát:

```
PM$=CHR$(0)+CHR$(17)+"S#a*Z"+CHR$(128)
```

Dále lze ještě ukončovací člen CHR$(128) sloučit s posledním vrcholem CHR$($a_n$) do jednoho členu CHR$(a_n+128). V našem příkladu lze tedy řetězec deklarovat takto:

```
PM$=CHR$(0)+CHR$(17)+"S#a*"+CHR$(218)
```

Má-li být nyní např. napsána MINIGRAFem zpráva

```
CHYBA ±5%
```

provede se to příkazem

```
WRITE "CHYBA "+PM$+"5%"
```

Jestliže některý znak je možno vytvořit tak, že napíšeme dva znaky na jedno místo, je možno použít znak s kódem 8, např. znak ⊗ napíšeme pomocí příkazu

```
WRITE "O"+CHR$(8)+"X"
```

7.5.4 Příkaz SIZE A,B,C,D

Velikost a sklon písma pro příkaz WRITE se určuje podobně jako u souřadnicového zapisovače XY 4131 příkazem SIZE. Parametry příkazu zde neudávají rozměry znaků v počtu kroků, ale u prvních dvou parametrů v 0,5 mm a u dalších dvou parametrů v mm.

Příklad:
Příkaz

```
SIZE 4,0,0,7
```

nastaví velikost znaků na šířku 2 mm a výšku 7mm. Písmo je kolmé a psáno vodorovně.

7.5.5 Příkaz NARROW

Tento příkaz nastaví základní šířku mezery mezi znaky řetězce kresleného příkazem WRITE. Šířka mezery je 1/4 hodnoty prvního parametru v příkazu SIZE (šířka znaku) v mm.

7.5.6 Příkaz WIDE

Tento příkaz nastaví širokou mezeru mezi znaky řetězce kresleného příkazem WRITE. Šířka mezery je 0,625.A mm, kde A je hodnota prvního parametru naposled provedeného příkazu SIZE.

Poslední tři příkazy pro ovládání MINIGRAFu se provádějí pomocí příkazů CALL.

7.5.7 Příkaz CALL 49314,n1,n2

Tento příkaz nejprve nastaví parametry pro tisk
(SIZE 3,0,0,3:NARROW a režim ASCII)
a pak vypíše požadovanou část zdrojového programu v jazyku BASIC z paměti IQ 151 na MINIGRAFu.

Parametr n1 určuje číslo prvního vypisovaného příkazu (není-li v programu příkaz s číslem n1 , začne se vypisovat od příkazu s nejblíže vyšším číslem, pokud ovšem není větší než n2). Parametr n2 určuje číslo posledního vypisovaného příkazu. Výpis skončí buď po výpisu příkazu s číslem n2 (není-li v programu příkaz s tímto číslem, skončí po výpisu příkazu s číslem nejblíže nižším), nebo po dosažení konce stránky.

První řádka výpisu začíná v místě, kde právě stojí záznamové pero (je tedy nutné ho předem správně nastavit, např. příkazem MOVA 0,y), další řádky začínají vždy od levého kraje papíru. Roztec řádek je 4,5 mm (na stránku lze tedy napsat 58 řádek).

Pokud jsou ve vypisovaném programy též znaky v inverzním, resp. grafickém režimu, budou ve výpisu označeny pruhem pod, resp. nad příslušným znakem. Po skončení výpisu budou parametry pro tisk nastaveny jako po příkazu ORG.

7.5.8 Příkaz CALL 49317,xa,ya

Tento příkaz nejprve nastaví parametry pro tisk jako předchozí příkaz. Pak nakreslí obdélníkový rámeček (šířka 606 kroků, výška 1046 kroků). Do něj opíše obsah obrazovky - pouze tisknutelné znaky ASCII (32 řádek po 32 znacích). Inverzní znaky označí podtržením, grafické znaky nahradí mezerou.

Parametry xa,ya určují souřadnice levého horního rohu rámečku kolem výpisu. Rozteč řádek je 4 mm, na papír formát A4 lze umístit celkem 4 výpisy obrazovky.

Po skončení výpisu budou parametry pro tisk nastaveny jako po příkazu ORG.

7.5.9 Příkaz CALL 49326

Tento příkaz provede nejprve funkci ORG, pak nastaví nižší rychlost kreslení (aby mohl probíhat i na MINIGRAFech vyrobených v roce 1985), parametry pro tisk (SIZE 10,0,0,10: WIDE a režim ASCII) a potom vypíše tabulku tisknutelných znaků ASCII. Příkaz slouží k ověření základních funkcí MINIGRAFu 0507 a programového vybavení modulu 0509.

Papír musí být předem založen jako pro příkaz ORG. Po skončení testu budou rychlost i parametry pro tisk nastaveny jako po příkazu ORG.

8 Modul GRAFIK

8.1 Úvod

Modul GRAFIK umožňuje počítači IQ 151 používat grafické zobrazení v rastu 512x256 bodů na obrazovce připojeného televizoru (jsou spojeny informace vytvořené moduly VIDEO a GRAFIK). Ke své činnosti potřebuje spolupráci s modulem VIDEO (32 nebo 64), neboť je těmito moduly synchronizován.

Modul GRAFIK je pamětí RWM RAM (velikost 16 Kslabik), která není přímo adresovatelná (její obsah není možno měnit příkazem POKE). Celá paměť modulu GRAFIK se zobrazuje na obrazovce televizoru nezávisle na obsahu paměti VIDEO RAM. Na obrazovce tedy může vznikat kresba v jemné Grafice překrytá různými znaky.

Jednotlivé slabiky paměti modulu GRAFIK se zobrazují v 256 řádcích a 64 sloupcích. Vzhledem k tomu, že slabika má 8 bitů, je v jednom řádku zobrazeno 64x8=512 bodů. Nultý bit slabiky je zobrazen vlevo a sedmý bit vpravo. Je-li hodnota bitu nulová, je na příslušném místě obrazovky tmavý bod (nerozsvícený), je-li hodnota bitu rovna 1, je na odpovídajícím místě obrazovky světlý bod.

8.2 Ovládání modulu GRAFIK

Z hlediska programové obsluhy je modul GRAFIK periferní zařízení připojené na brány o adresách D0 až D4. Programová obsluha se dělí na obsluhu řídícího registru a na datovou komunikaci. Jako řídící registr je v modulu použit obvod typu 8255, který je však přístupný pouze pro zápis. Před zahájením práce s tímto registrem je třeba provést jeho inicializaci. Obvod pracuje v modu 0 a všechny tři brány jsou využity jako výstupní. Inicializace se provede zápisem hodnoty 80H na adresu D3 (adresa řídícího registru 8255).

Brána A je přístupná na adrese D0 a slouží jako paměť části x-ové souřadnice zobrazovaného bodu takto:

-	-	X8	X7	X6	X5	X4	X3

Tato část x-ové souřadnice určuje adresu slabiky paměti (slabika paměti se zobrazuje vodorovně). Počátek souřadnic je v levém dolním rohu obrazovky.

Brána B slouží jako paměť y-ové souřadnice zobrazovaného bodu a je přístupná na adrese D1.

Brána C je využita jako registr řídících signálů a je přístupná na adrese D2. Obsahuje tyto signály:

SEL	E2	E1	E0	EV	FAST*	PEN*	ALL

Signál ALL - určuje, zda pracujeme s jednotlivými bity (0) nebo celými slabikami paměti (1). Při bitovém přístupu můžeme adresovat každý bit paměti a pracovat s ním zcela nezávisle.

Signál PEN* - určuje, zda při bitovém přístupu se budou body rozsvěcovat (0) nebo zhášet (1). Pro přístup do paměti po slabikách nemá tento signál význam.

Signál FAST*- umožňuje procesoru přístup do paměti modulu přidělením priority i během zobrazení (0), což zrychluje generaci obrazu za cenu vizuálního rušení v obraze.

Signál EV - povoluje výstup video signálu (1) a pokud není nastaven (0), lze s výhodou využít "rychlé" kresby, neboť není na stínítku zobrazována.

Signály E0-E2 - slouží k ovládaní přídavného obvodu barevného zobrazení (zatím se nepoužívá).

Signál SEL - umožňuje blokování spolupráce s modulem GRAFIK (0) v případě, že by mohlo vadit zpožďování paměťových operací s adresou xxD4H. Toto zpožďování způsobuje nedostatek procesoru 8080, který neumožňuje rozlišit včas zápis do paměti od zápisu do periferního obvodu. Pro omezení těchto nepříznivých vlivů, např. při spolupráci s pružným diskem, se doporučuje povolovat spolupráci jen na nezbytně nutnou dobu.

Datová komunikace se provádí na adrese D4. Význam dat se liší podle zvoleného přístupu.

Bitový přístup:

X2	X1	X0	-	-	-	-	-

Zbývající část x-ové souřadnice zobrazovaného bodu. Udává polohu zpracovávaného bodu v slabice adresované předchozí částí adresy.

Přístup po slabikách:

B7	B6	B5	B4	B3	B2	B1	B0

Datová slabika určená k uložení nebo získaná z paměti modulu GRAFIK. Má-li bit hodnotu 1, bod svítí, má-li hodnotu 0, bod nesvítí (opak signálu PEN*).

Při porovnání vlastností jednotlivých druhů přístupů se pro kresbu obecných čar jeví jako výhodnější bitový přístup. Pro vyplnění ploch a pro inicializační mazání je určen přístup po slabikách.

Pro ovládání modulu GRAFIK je určen překladač BASIC G, který je popsán v kapitole 9.

9 Programovací jazyk BASIC G

9.1 Úvod

Programovací jazyk BASIC G umožňuje pracovat s jemnou grafikou (modul GRAFIK) počítače IQ 151. Tento programovací jazyk zachovává všechny příkazy jazyka BASIC 6 a obsahuje dalších 14 příkazů pro práci s modulem GRAFIK.

Programovací jazyk BASIC G se dodává ve dvou verzích a to v páskové a modulové verzi. Pásková verze se přehrává do paměti počítače z magnetofonového pásku a teprve po jeho úplném přehrání do paměti RWM RAM počítače a jeho spuštění lze s tímto jazykem dále pracovat. Modulová verze BASIC G obsahuje překladač tohoto jazyka v přídavném modulu BASIC G. V této kapitole se budeme zabývat modulovou verzí jazyka. Příkazy pro práci s jemnou grafikou jsou stejné jak v modulové, tak i v páskové verzi. Při používání páskové verze je jiným způsobem rozdělena paměť RWM RAM (je zde umístěn i překladač jazyka). Pásková verze překladače BASIC G se spouští od adresy 400H.

V dalších bodech této kapitoly jsou popsány jednotlivé příkazy jazyka BASIC G pro práci s jemnou grafikou.

9.2 Příkaz ERASE

Pomocí tohoto příkazu je možno dosadit do všech slabik paměti modulu GRAFIK stejnou hodnotu. Obecný tvar příkazu je:

ERASE av

kde av je aritmetický výraz. Z hodnoty tohoto výrazu se bere pouze celá část (musí ležet v intervalu <0;255>) a ta se přiřadí do všech slabik paměti modulu GRAFIK.

Pokud příkaz ERASE použijeme bez parametru, dosáhneme stejného účinku, jako při použití tohoto příkazu s nulovým parametrem (ERASE 0). Tento příkaz smaže všechny informace zobrazené na obrazovce pomocí jemné grafiky (pro smazání informací zapamatovaných ve VIDEO RAM se používá příkaz CLS).

Příkaz ERASE 255 rozsvítí celou obrazovku. Použití příkazu ERASE s jinou hodnotou parametru (různou od 0 a 255) vytvoří na obrazovce svislé pruhy. V tomto případě jsou rozsvícené body určené nenulovými bity slabik paměti modulu GRAFIK.

9.3 Vymezení aktuální kreslící plochy - příkaz LIMIT

Při práci s jemnou grafikou nepotřebujeme vždy zobrazovat informace po celé ploše obrazovky, ale jen na některé její části. Tuto část obrazovky, které říkáme aktuální kreslicí plocha, můžeme určovat pomocí příkazu LIMIT, který má obecný tvar:

```
LIMIT xmin,xmax,ymin,Max
```

kde xmin, xmax, ymin a ymax jsou aritmetické výrazy.

Z hodnot těchto výrazů se bere celá část a musí být splněny nerovnosti

$$0 \leq xmin < xmax \leq 511$$

$$0 \leq ymin < ymax \leq 255$$

Nepoužijeme-li v programu příkaz LIMIT, pak kreslení probíhá po celé ploše obrazovky. To můžeme také dosáhnout použitím příkazu:

```
LIMIT 0,511,0,255
```

Příkazy jemné grafiky zobrazují kreslené obrázky pouze na aktuální kreslící ploše, případné kreslení mimo tuto plochu je ignorováno. Aktuální kreslící plochu je možno kdykoliv v průběhu kreslení změnit použitím dalšího příkazu LIMIT. Nová aktuální kreslící plocha se může překrývat s původní aktuální plochou. Nově prováděné kresby tak mohou zasahovat do kreseb v původní aktuální ploše.

9.4 Určení typu čáry - příkaz PEN

Při kreslení můžeme kreslit bílé čáry (rozsvěcovat body, které leží na kreslené čáře), černé čáry (zhášet body) nebo inverzní čáry (tj. zhášet rozsvícené body a rozsvěcovat zhasnuté body kreslené čáry).

Určení typu čáry se provádí příkazem PEN, který má tento obecný tvar:

```
PEN av
```

kde av je libovolný aritmetický výraz. Z hodnoty tohoto výrazu se bere pouze celá část a ta musí nabývat jedné ze tří hodnot:
0 - kreslí se inverzní čára,
1 - kreslí se černá čára,
2 - kreslí se bílá čára.

Jestliže nepoužijeme příkaz PEN, potom se kreslí bílá čára.

9.5 Kreslení rámečku - příkaz FRAME

Tento příkaz je bez parametrů a jeho použití způsobí orámování aktuální kreslící plochy. Rámeček může být bílý, černý nebo inverzní. Barva rámečku je určena parametrem naposled použitého příkazu PEN.

9.6 Příkaz FILL

Tento příkaz má podobnou funkci jako příkaz ERASE, s tím rozdílem, že se týká pouze aktuální kreslící plochy. Obecný tvar příkazu FILL je tento:

```
FILL av
```

kde av je aritmetický výraz - platí o něm to, co je uvedeno o parametru příkazu ERASE.

Příkaz FILL 0 způsobí smazání všech informací zobrazených v aktuální kreslící ploše pomocí jemné grafiky a příkaz FILL 255 rozsvítí všechny body aktuální kreslicí plochy.

Příkaz FILL můžeme stejně jako příkaz ERASE používat bez parametrů. V tomto případě je ale funkce těchto příkazů odlišná. U příkazu FILL záleží na určení typu čáry (tzn. na naposled provedeném příkazu PEN) a celá aktuální plochá je zaplněna tímto typem čáry. Např. příkaz FILL provedený po příkazu PEN 0 způsobí inverzi celé aktuální kreslící plochy a tento příkaz provedený po příkazu PEN 2 rozsvítí všechny body aktuální kreslicí plochy.

9.7 Zavedení souřadného systému - příkaz SCALE

Pomocí příkazu SCALE je možno zavést souřadný systém v aktuální kreslící ploše. Obecný tvar tohoto příkazu je:

SCALE xmin,xmax,ymin,Max

kde xmin, xmax, ymin a ymax jsou aritmetické výrazy.
Hodnoty těchto výrazů určují minimální a maximální hodnoty souřadnic v aktuální kreslící ploše.

Příklad:
Příkaz

SCALE -1000,1000,-5,5

zavede souřadný systém, jehož počátek je ve středu aktuální kreslící plochy, hodnota x-ové souřadnice (vodorovný směr) se může měnit v intervalu <-1000; 1000> a hodnota y-ové souřadnice v intervalu <-5;5>.

9.8 Kreslení čáry - příkazy DRAW, RDRAW a IDRAW

Kreslení čáry si můžeme představit jaké výsledek pohybu pomyslného kreslícího pera. Pokud pero je spuštěno, potom se při jeho pohybu kreslí čára. Při zvednutém kreslícím peru se pero přemístí na jiné místo kreslící plochy bez kreslení čáry. Pohyb kreslícího pera se ovládá příkazem DRAW, který má obecný tvar:

DRAW xav,yav,av

kde xav, yav a av jsou aritmetické výrazy.

Příkaz provede přesun kreslícího pera z aktuální polohy pera (bod do kterého se pero dostalo při provedení předchozího pohybu) do bodu, jehož souřadnice jsou určené hodnotami výrazů xav a yav v souřadném systému vytvořeném příkazem SCALE. Pohyb se provádí po přímce. Hodnota parametru av řídí zvedání nebo spouštění kreslícího pera na začátku nebo na konci posuvu. Hodnota výrazu av (celá část hodnoty výrazu) může být:

2 - přesun pera a po přesunu jeho zdvihnutí,
1 - přesun pera a po přesunu jeho spuštění,
-1 - spuštění pera a pak jeho přesun,
-2 - zdvižení pera a pak jeho přesun.

Jestliže je pero při přesunu spuštěno, je typ kreslené čáry určen naposled provedeným příkazem PEN.

Pokud při kreslení úsečky neleží celá úsečka v aktuální kreslící ploše, je zobrazena pouze ta část úsečky, která leží v aktuální ploše.

Obecný tvar příkazu RDRAW a IDRAW je:

```
RDRAW xav,yav,av
IDRAW xav,yav,av
```

kde význam parametrů xav, yav a av je stejný jako u příkazu DRAW.

Činnost příkazu RDRAW je podobná jako činnost příkazu DRAW. Rozdíl je pouze v tom, že příkaz RDRAW používá souřadnou soustavu, jejíž počátek je v bodě, do něhož bylo přesunuto pero naposled použitým příkazem DRAW. Nová soustava souřadnic vznikla posunutím původní soustavy.

Příkaz IDRAW pracuje analogicky jako příkaz RDRAW, pouze s tím rozdílem, že po provedení příkazu IDRAW se provede posunutí soustavy souřadnic tak, aby aktuální poloha pera byla počátkem nové soustavy souřadnic pro následující příkaz RDRAW nebo IDRAW.

Příkazy RDRAW prováděné postupně za sebou pracují v souřadném systému, jehož počátek je určen koncovou polohou pera při provedení posledního příkazu DRAW (nebo IDRAW).

Příklad:
Následující program nakreslí na obrazovce sinusovku.

```
10 ERASE: CLS: PEN 2
20 FRAME: SCALE 0,511,-1,1: DRAW 0,0,-2
30 FOR I=0 TO 511
40 DRAW I,SIN(I*PI/256),-1
50 NEXT I
```

9.9 Otočení souřadného systému - příkaz PDIR

Příkaz PDIR provede otočení původní soustavy souřadnic kolem jejího počátku o úhel (v radiánech) určený parametrem příkazu v kladném smyslu, tj. proti směru pohybu hodinových ručiček. Obecný tvar příkazu je tento:

```
PDIR av
```

kde av je aritmetický výraz, jehož hodnota udává úhel otočení soustavy.

9.10 Psaní textů - příkazy LABEL, LROT, LTYPE a LREF

Pro psaní textů prostřednictvím jemné grafiky je možno použít příkazy popsané v tomto bodu. Příkaz LABEL určuje vypisovaný text, příkaz LROT určuje směr výpisu, LTYPE velikost vypisovaných znaků a LREF polohu vypisovaného textu vzhledem k aktuální poloze kreslicího pera. Text je vypisován pomocí čáry, která je určena příkazem PEN. Ve vypisovaných textech se nedají použít malá písmena a grafické znaky (místo malých písmen jsou vypsána odpovídající písmena velká a místo grafických znaků velká písmena uvedená v příloze 1 na stejném řádku jako příslušný grafický znak).

Příkaz LABEL má stejný tvar parametrů jako základní tvar příkazu PRINT (viz bod 2.15). Textová konstanta nebo hodnota aritmetického výrazu bude vypsána na obrazovce pomocí jemné grafiky.

Příkaz LROT má obecný tvar:

```
LROT av
```

kde av je aritmetický výraz a jeho hodnota určuje úhel (v radiánech), pod kterým budou dále vypisovány texty (kladný smysl úhlu je proti směru pohybu hodinových ručiček).

Vypisované znaky jsou kolmé na vypisovaný směr.

Obecné tvary příkazu LTYPE jsou:

```
LTYPE sav,vav,zav,rav
LTYPE sav,vav,zav
LTYPE sav,vav
```

kde sav, vav, zav a rav jsou aritmetické výrazy. Hodnoty těchto výrazů určují velikost vypisovaných znaků takto:

šířka znaků	- 2,8 . sav bodů
výška znaků	- 2,3 . vav bodů
odstup mezi znaky	- 0,56 . zav . sav bodů
odstup mezi řádky	- 0,23 . rav . vav bodů

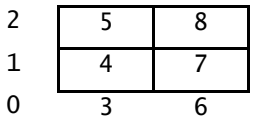
Parametry zav a rav se v příkazu LTYPE nemusí uvádět.

Obecný tvar příkazu LREF je:

```
LREF av
```

kde av je aritmetický výraz, jehož hodnota udává polohu vypisovaného textu vzhledem k aktuální poloze pera. Hodnota výrazu (celá část) musí ležet v intervalu <0;8>.

Každý vypisovaný text (při libovolně zvoleném směru výpisu a velikosti znaků) vyplňuje plochu obdélníka. V této ploše je definováno 9 bodů - viz následující obrázek:



Tyto body jsou očíslovány. Hodnota parametru u příkazu LREF udává umístění textu vzhledem k předchozí poloze pera. Např. pro av=0 je vypisovaný text umístěn tak, že jeho levý spodní roh je umístěn v bodě, kde je aktuální poloha pera.

Příklad:
Následující program napíše do středu obrazovky text AHOJ

```
10 ERASE: CLS: PEN 2: FRAME
20 SCALE -1,1,-1,1
30 DRAW 0,0,-2
40 LROT 0: LREF 4: LTYPE 5,10
50 LABEL "AHOJ"
```

Příkazy LROT, LREF a LTYPE je nutno provést dříve než příkaz LABEL.

10 Speciální příkazy programovacího jazyka BASIC G

10.1 Úvod

Programovací jazyk BASIC G obsahuje, kromě příkazů uvedených v kapitole 9, dále 6 speciálních příkazů a jednu standardní funkci. Tyto prvky jazyka BASIC G nejsou zatím popsány v žádné příručce. Jedná se o 3 příkazy, které se týkají jemné grafiky počítače IQ 151. Standardní funkce umožňuje zjišťovat hodnotu uloženou do dvojice slabik paměti a další příkaz umožňuje tuto hodnotu měnit. Zbývající dva příkazy umožňují provádět nestandardní vstupní a výstupní operace.

10.2 Speciální příkazy pro používání jemné grafiky

Pro povolení nebo zakázání výstupu videosignálu z modulu GRAFIK (viz bod 8.2 - signál EV) je možno použít příkaz jehož obecný tvar je

VIDEO av

kde av je aritmetický výraz (z hodnoty výrazu se bere celá část, musí ležet v intervalu <0;255>), jehož hodnota určuje zda je povolen (je-li hodnota av různá 0) nebo zakázán (je-li hodnota av=0) výstup videosignálu.

Příkaz VIDEO můžeme použít také bez parametru. V tomto případě je dosažen stejný účinek, jako při použití příkazu VIDEO 1 tzn. je povolen výstup videosignálu.

Tento příkaz je možno využít v případě kreslení obrázku, kdy nechceme, aby průběh kreslení byl zobrazován a byl zobrazen až dokončený obrázek. To provedeme tak, že před zahájením kreslení zakážeme výstup videosignálu a tento výstup povolíme až po dokreslení obrázku.

Příkazy jejichž obecný tvar je

GSAVE av
GLOAD av

kde av je aritmetický výraz (bere se celá část, musí ležet v intervalu <0;65535>),

slouží pro úschovu a obnovení nakresleného obrázku (v jemné grafice). Pro úschovu obrázku je potřeba 16 Kslabik operační paměti. Výraz av udává počáteční adresu úseku paměti, do které je obrázek uložen. Při používání těchto příkazů musíme výraz av volit tak, aby uložený obrázek "nepřemazal" některé důležité informace uložené v paměti (např. program nebo slabiky paměti používané monitorem). Příkaz GSAVE provádí úschovu obrázku a příkaz GLOAD jeho obnovení. Parametr av u příkazu GLOAD musí mít stejnou hodnotu jako parametr u příkazu GSAVE při uložení tohoto obrázku.

Příklad:
Příkazem

```
GSAVE HEX(3000)
```

uložíme nakreslený obrázek do operační paměti od adresy 3000H do adresy 6FFFH. Pokud chceme obrázek zachovat, nesmíme tuto část paměti používat pro jiné účely. Uložený obrázek je možno opět nakreslit na obrazovce pomocí příkazu

```
GLOAD HEX(3000)
```

Obrázek nakreslený před použitím příkazu GLOAD se tím ale ztratí.

10.3 Zjištění hodnoty a změna hodnoty dvojice slabik paměti

Pro zjištění hodnoty dvojice sousedních slabik paměti (hodnoty šestnáctibitového slova) lze použít standardní funkci

```
DPEEK(av)
```

kde av je aritmetický výraz určující desítkovou adresu slabiky o nižší adrese ze dvojice slabik (bere se celá část hodnoty výrazu, musí ležet v intervalu <0;65535>).

Touto funkcí můžeme zjišťovat obsah libovolné dvojice sousedních slabik z paměti ROM nebo RWM RAM. Ve slabice s nižší adresou jsou uloženy nižší řády šestnáctibitového slova.

Tuto funkci je možno nahradit aritmetickým výrazem

```
PEEK(av+1)*256+PEEK(av)
```

ve kterém se používá funkce PEEK (viz bod 5.7). Hodnota této funkce leží v intervalu <0;65535>.

Příklad použití:
Hodnotu času vytvářeného počítačem (v padesátinách sekundy), tj. hodnotu proměnné TIME (viz bod 4.3) můžeme získat pomocí aritmetického výrazu

```
DPEEK(9)*256+PEEK(8)
```

Změnit hodnotu dvojice sousedních slabik můžeme příkazem

```
DPOKE av1,av2
```

kde av1 je aritmetický výraz, jehož hodnota určuje adresu nižší slabiky z dvojice, jejíž hodnotu měníme a
av2 je aritmetický výraz určující novou hodnotu dvojice slabik.
Z hodnot výrazů av1 a av2 se bere celá část a obě hodnoty musí ležet v intervalu <0;65535>.

Funkce tohoto příkazu je stejná jako funkce následující dvojice příkazů POKE (viz bod 5.7)

```
POKE av1+1,av2/256
POKE av1,av2 AND 255
```

Použití standardní funkce DPEEK a příkazu DPOKE je výhodné pokud pracujeme s šestnáctibitovými informacemi, např. adresami paměti.

Příklad:
Adresu začátku programu, která je uložena ve dvojici slabik o adresách CE a CF (viz bod 5.1), můžeme zjistit výrazem

```
DPEEK(HEX(CE))
```

10.4 Nestandardní použití příkazů INPUT a PRINT

Při použití příkazu INPUT jsou vstupující informace zadávány na klávesnici počítače. Jednotlivé zapsané znaky jsou do BASICu předávány pomocí monitorovského podprogramu, který začíná na adrese F803. V některých případech nám to ale nemusí vyhovovat. Např. máme požadavek, aby příkaz INPUT čekal na zadávané informace pouze určitou dobu nebo aby čtené znaky byly vybírány z určité oblastí operační paměti.

V těchto případech můžeme postupovat tak, že si vytvoříme vlastní podprogram (ve strojovém kódu) pro vstup znaku a překladači BASIC sdělíme adresu jeho začátku pomocí příkazu IDEVICE. Další příkazy INPUT budou používat tento náš podprogram pro vstup znaku.

Obecný tvar příkazu je

IDEVICE av

kde av je aritmetický výraz určující počáteční adresu nového podprogramu pro vstup znaku (z hodnoty av se bere pouze celá část a musí ležet v intervalu <0;65535>).

Pomocí příkazů IDEVICE je možno podprogramy pro vstup znaku libovolně měnit. Změna tohoto podprogramu se týká pouze příkazu INPUT.

Podobně jako provádíme změnu podprogramu pro vstup znaku, je možno také provést změnu podprogramu pro výstup znaku na obrazovku televizoru, který využívá příkaz PRINT, Monitorovský podprogram pro výstup znaku začíná na adrese F809.

Příkaz změny podprogramu pro výstup znaku má obecný tvar

ODEVICE av

kde av je aritmetický výraz určující počáteční adresu nového podprogramu pro výstup znaku (z hodnoty av se bere pouze celá část a musí ležet v intervalu <0;65535>).

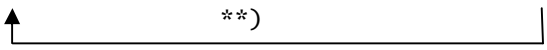
Změna tohoto podprogramu se týká pouze příkazu PRINT a příkazu LIST (pokud je součástí programu). U příkazu PRINT se netýká oddělovače &r,s, který vždy provádí změnu polohy kurzoru na obrazovce.

Po ukončení běhu programu jsou přiřazeny původní (monitorovské) podprogramy pro vstup i výstup znaku.

Příloha 1

Kód ASCII

H	D	Znak	H	D	Znak	H	D	Znak	*)	H	D	Znak
0	0	NUL	20	32		40	64	@		60	96	`
1	1	SOH	21	33	!	41	65	A		61	97	a
2	2	STX	22	34	"	42	66	B		62	98	b
3	3	ETX	23	35	#	43	67	C		63	99	c
4	4	EOT	24	36	$	44	68	D		64	100	d
5	5	ENQ	25	37	%	45	69	E		65	101	e
6	6	ACK	26	38	&	46	70	F		66	102	f
7	7	BEL	27	39	'	47	71	G		67	103	g
8	8	BS	28	40	(	48	72	H		68	104	h
9	9	HT	29	41	)	49	73	I		69	105	i
0A	10	LF	2A	42	*	4A	74	J		6A	106	j
0B	11	VT	2B	43	+	4B	75	K		6B	107	k
0C	12	FF	2C	44	,	4C	76	L		6C	108	l
0D	13	CR	2D	45	-	4D	77	M		6D	109	m
0E	14	SO	2E	46	.	4E	78	N		6E	110	n
0F	15	SI	2F	47	/	4F	79	O		6F	111	o
10	16	DLE	30	48	0	50	80	P		70	112	P
11	17	DC1	31	49	1	51	81	Q		71	113	q
12	18	DC2	32	50	2	52	82	R		72	114	r
13	19	DC3	33	51	3	53	83	S		73	115	s
14	20	DC4	34	52	4	54	84	T		74	116	t
15	21	NAK	35	53	5	55	85	U		75	117	u
16	22	SYN	36	54	6	56	86	V		76	118	v
17	23	ETB	37	55	7	57	87	W		77	119	w
18	24	CAN	38	56	8	58	88	X		78	120	x
19	25	EM	39	57	9	59	89	Y		79	121	y
1A	26	SUB	3A	58	:	5A	90	Z		7A	122	z
1B	27	ESC	3B	59	;	5B	91	[		7B	123	{
1C	28	FS	3C	60	<	5C	92	\		7C	124	|
1D	29	GS	3D	61	=	5D	93	]		7D	125	}
1E	30	RS	3E	62	>	5E	94	^		7E	126	~
1F	31	US	3F	63	?	5F	95	_		7F	127	



Poznámky:

Ve sloupcích H jsou uvedeny kódy znaků v šestnáctkové soustavě, ve sloupcích D v desítkové.

Znaky s kódy 0 - 31 jsou řídící a na obrazovce televizoru se nezobrazují.

Znaky na obrazovce zobrazené inverzně mají kód o 128 (80H) větší.

*) Kódy grafických znaků v grafickém režimu

**) Při zápisu do VIDEO RAM se pro grafické znaky používají kódy 0 až 31 (znaky jsou přesunuty do prvního sloupce tabulky).

Příloha 2

Seznam chybových hlášení

00 K příkazu NEXT chybí příkaz FOR.
01 Syntaktická chyba.
02 Použit příkaz RETURN bez příkazu GOSUB.
03 Nedovolený příkaz v neočíslovaném řádku.
Příkaz READ nenašel data.
04 Použitý parametr je větší než 32767.
05 Číselné přeplnění.
06 Zaplnění paměti vyhrazené pro BASIC.
07 Odkaz na neexistující řádek.
08 Překročení deklarované velikosti indexu.
09 Opakovaná deklarace téhož pole.
10 Dělitel je roven nule.
11 Neočíslovaný příkaz INPUT nebo DEF FN.
12 Nedovolená operace s řetězcem.
13 Přeplnění oblasti STRING.
14 Řetězec je delší než 255 znaků.
16 V dané situaci nelze pokračovat příkazem CONT.
17 Chybí deklarace DEF FN.
18 Parametr je větší než 65535.
19 Překročen parametr v příkazu PLOT nebo PRINT&.
20 Pokus o zrušení neexistujícího pole.
21 Identifikátor nezačíná písmenem.
22 Překročení počtu parametrů definované funkce.
Počet aktuálních parametrů definované funkce není roven počtu formálních parametrů.
24 Nedovolená hodnota x-ové souřadnice.
25 Nedovolená hodnota y-ové souřadnice.
26 Nedovolená hodnota parametru příkazu SPEED.
27 Nedovolená hodnota parametrů u příkazu SIZE.
28 Není připojen modul grafické jednotky.

* EXTRA DATA IGNORED Zadání nadbytečných dat v příkazu INPUT.
* INPUT ERROR Zadané číslo je syntakticky chybné.

Příloha 3

Rejstřík standardních funkcí

Příloha 4

Rejstřík klíčových slov

Příloha 5

Tabulka kódů lexikálních symbolů - BASIC 6

80 END	A0 CALL	C0 >
81 FOR	A1 BYE	C1 =
82 NEXT	A2 WAIT(	C2 <
83 DATA	A3 AUTO	C3 SGN
84 INPUT	A4 MEM	C4 INT
85 DIM	A5 MOV	C5 ABS
86 READ	A6 VECT	C6 USR
87 LET	A7 POINT	C7 INP
88 GOTO	A8 ORG	C8 POS
89 RUN	A9 SPEED	C9 SQR
8A IF	AA NARROW	CA RND
8B RESTORE	AB WIDE	CB LOG
8C GOSUB	AC WRITE	CC EXP
8D RETURN	AD SIZE	CD COS
8E REM	AE FREE	CE SIN
8F STOP	AF UNPLOT	CF TAN
90 OUT	B0 PLOT	D0 ATN
91 ON	B1 CLS	D1 PEEK
92 GET	B2 TAB(	D2 LEN
93 POKE	B3 TO	D3 STR$
94 PRINT	B4 SPC(	D4 VAL
95 LPRINT	B5 FN	D5 ASC
96 DEF	B6 THEN	D6 CHR$
97 CONT	B7 NOT	D7 LEFT$
98 LIST	B8 STEP	D8 RIGHT$
99 LLIST	B9 +	D9 MID$
9A CLEAR	BA -	DA INKEY$
9B PTAPE	BB *	DB HEX(
9C MLOAD	BC /	DC PI
9D PLIST	BD ^	DD BYTE(
9E MSAVE	BE AND	DE WORD(
9F SCRATCH	BF OR	DF PTR(

Příloha 6

Tabulka kódů lexikálních symbolů - BASIC G

80 END	A0 CALL	C0 LROT	E0 EXP
81 FOR	A1 BYE	C1 LREF	E1 COS
82 NEXT	A2 WAIT(	C2 VIDEO	E2 SIN
83 DATA	A3 AUTO	C3 FILL	E3 TAN
84 INPUT	A4 MEM	C4 GSAVE	E4 ATN
85 DIM	A5 IDEVICE	C5 GLOAD	E5 PEEK
86 READ	A6 ODEVICE	C6 TAB(	E6 LEN
87 LET	A7 DPOKE	C7 TO	E7 STR$
88 GOTO	A8 MOV	C8 SPC(	E8 VAL
89 RUN	A9 VECT	C9 FN	E9 ASC
8A IF	AA POINT	CA THEN	EA CHR$
8B RESTORE	AB ORG	CB NOT	EB LEFT$
8C GOSUB	AC SPEED	CC STEP	EC RIGHT$
8D RETURN	AD NARROW	CD +	ED MID$
8E REM	AE WIDE	CE -	EE INKEY$
8F STOP	AF WRITE	CF *	EF HEX(
90 OUT	B0 SIZE	D0 /	F0 PI
91 ON	B1 FREE	D1 ^	F1 BYTE(
92 GET	B2 UNPLOT	D2 AND	F2 WORD(
93 POKE	B3 PLOT	D3 OR	F3 PTR(
94 PRINT	B4 CLS	D4 >	F4 DPEEK(
95 LPRINT	B5 ERASE	D5 =	
96 DEF	B6 LIMIT	D6 <	
97 CONT	B7 SCALE	D7 SGN	
98 LIST	B8 FRAME	D8 INT	
99 LLIST	B9 PEN	D9 ABS	
9A CLEAR	BA DRAW	DA USR	
9B PTAPE	BB IDRAW	DB INP	
9C MLOAD	BC RDRAW	DC POS	
9D PLIST	BD PDIR	DD SQR	
9E MSAVE	BE LABEL	DE RND	
9F SCRATCH	BF LTYPE	DF LOG	

Příloha 7

Interpretace kódu ASCII grafickou jednotkou XY

Kód	Použitý znak
0-12	ignorováno
13	CR LF
14, 15	ignorováno
16	○ (speciální znaky)
17	□ (speciální znaky)
18	◇ (speciální znaky)
19	△ (speciální znaky)
20	▽ (speciální znaky)
21	+ (speciální znaky)
22	× (speciální znaky)
23	✳ (speciální znaky)
24	´ (čárka nad velkým písmenem)
25	´ (čárka nad malým písmenem)
26	ˇ (háček nad velkým písmenem)
27	ˇ (háček nad malým písmenem)
28	¨ (přehlasování velkých písmen)
29	¨ (přehlasování malých písmen)
30	° (kroužek nad velkým písmenem)
31	° (kroužek nad malým písmenem)
32-126	standardně jako v ASCII
127	Σ
128 – 255	mezera

Příloha 8

Tabulka pro definování znaků na MINIGRAFu 0507

56	8	57	9	58	:	59	;	60	<
120	x	121	y	122	z	123	{	124	|
48	0	49	1	50	2	51	3	52	4
112	p	113	q	114	r	115	s	116	t
40	(	41	)	42	*	43	+	44	,
104	h	105	i	106	j	107	k	108	l
32		33	!	34	"	36	$	35	#
96	`	97	a	98	b	99	c	100	d
24		25		26		27		28	
88	X	89	Y	90	Z	91	[	92	\
16		17		18		19		20	
80	P	81	Q	82	R	83	S	84	T
8		9		10		11		12	
72	H	73	I	74	J	75	K	76	L
0		1		2		3		4	
64	@	65	A	66	B	67	C	68	D

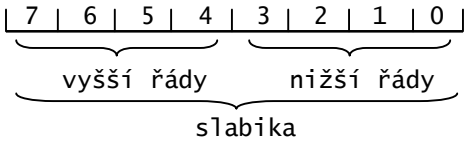
horní čísla - záznamové pero nahoře
dolní čísla - záznamové pero dole

Příloha 9

Popis mikroprocesoru MHB 8080

Mikroprocesor obsahuje z hlediska programátora především tyto prvky:
- aritmetickou a logickou jednotku,
- řídící obvody,
- čítač instrukcí,
- pracovní registry a střadač,
- registr příznaků, atd.

Mikroprocesor MHB 8080 je osmibitový mikroprocesor, umožňuje tedy zpracování dat po osmi bitech. Zpracovávané informace, paměť, registry a další prvky mikroprocesoru jsou organizovány po slabikách (po osmi bitech). Uspořádání slabiky a označení jednotlivých bitů ukazuje následující obrázek.



Zpracovávání dat zajišťuje aritmetická a logická jednotka. Data určená ke zpracování se vyzvednou z paměti, uloží se do pracovních registrů nebo do střadače a po ukončení operace je možno je opět přesunout do paměti.

U mikropočítačů je paměť představována paměťmi dvou typů:
- pamětí ROM,
- pamětí RWM RAM.

Paměť ROM (nebo její modifikace EPROM) obsahuje pouze programy, které se při zpracování nemění. Významnou vlastností těchto pamětí je jejich permanentnost, tzn. že jejich obsah se nezničí při vypnutí napájení a při zapnutí napájení je jejich obsah již pevně dán. Obsah paměti tohoto typu již běžnými způsoby není možno měnit.

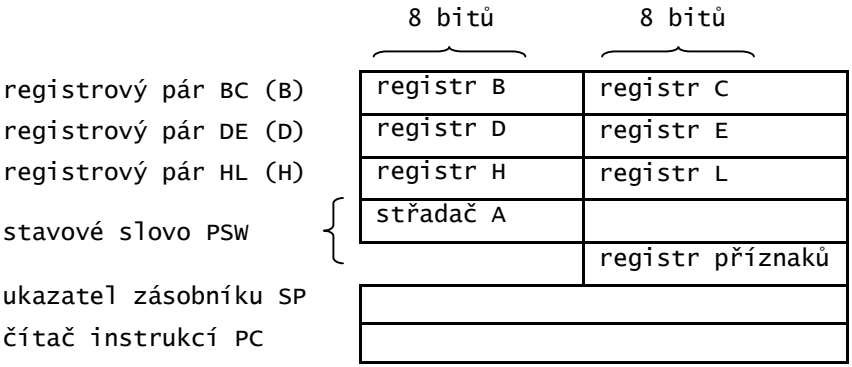
Paměť RWM RAM se používá pro ukládání programů, nebo dat zaváděných z periferních zařízení. Přerušení napájení znamená zničení obsahu těchto pamětí. Jejich obsah je možno libovolně měnit.

Komunikaci s periferními zařízeními provádí mikroprocesor prostřednictvím vstupních/výstupních bran. Z hlediska programátora fungují tyto brány podobně jako slabiky paměti, tzn. že je možno do nich informaci uložit nebo naopak z nich informaci vyzvednout. Na tomto principu je možno realizovat požadované periferní funkce výstupu nebo vstupu.

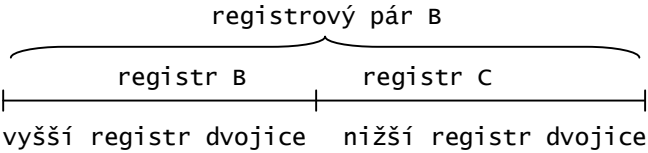
Řídící obvody mikroprocesoru zajišťují dekódování instrukcí a řídí provádění těchto instrukcí. V registru instrukcí je uložena právě prováděná instrukce a v čítači instrukcí PC je adresa následující instrukce.

Soubor instrukcí mikroprocesoru (viz příloha 10) umožňuje provádět přesuny, aritmetické a logické operace. větvení programů a další funkce. Délka instrukcí je jedna, dvě nebo tři slabiky. Operandy instrukcí jsou nejčastěji osmibitové nebo 16-bitové (jedná-li se o adresu).

Pracovní registry mikroprocesoru jsou uspořádány podle obrázku:



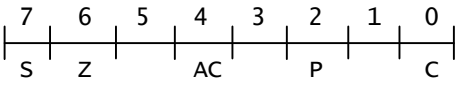
Každý z registrů B, C, D, E, H, L má délku 8 bitů. S těmito registry je možno pracovat jednotlivě nebo po dvojicích. V případě, že se pracuje s dvojicemi registrů, označuje registrový pár B registry B a C, registrový pár D označuje registry D a E a registrový pár H označuje registry H a L. Uspořádání registrového páru ukazuje následující obrázek na příkladě registrového páru B:



Střadač A je představován rovněž 8-bitovým registrem a používá se obdobně jako pracovní registry B, C, B, E, H, L. V mnoha instrukcích představuje střadač implicitní registr pro výsledek.

Některé aritmetické a logické instrukce mění hodnotu jednoho nebo více příznaků a tím indikují výsledek operace. Uživatel může v programu testovat tyto příznaky a použít je k podmíněným skokům, k vyvolání podprogramu nebo k návratu z podprogramu. Příznaky jsou umístěny v registru příznaků. K tomu, aby mohly být využívány je zapotřebí znát, které instrukce ovlivňují jednotlivé příznaky (viz příloha 10).

Registr příznaků je představován slabikou s bity těchto významů:



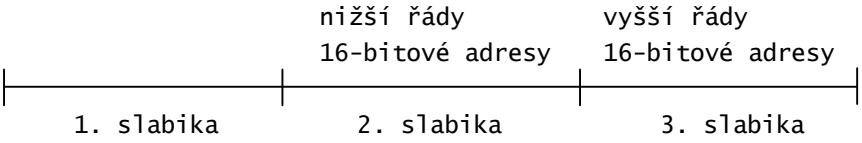
Bližší vysvětlení je v tabulce:

Bit	Označení	Význam bitu
7	S	Příznak znaménka. Je-li S=0 jedná se o kladný výsledek provedené operace, je-li S=1, jedná se o záporný výsledek.

6	Z	Příznak nuly. Má-li výsledek všechny bity nulové, je příznak nuly Z=1. V případě, že výsledek není nulový, potom Z=0.
5		nepoužívá se
4	AC	Příznak pomocného přenosu. Přenos z nižších řádů slabiky do vyšších řádů při provedení operace. Používá instrukce DAA.
3		nepoužívá se
2	P	Příznak parity. Je-li počet bitů výsledku s nenulovou hodnotou sudý, potom P=1, jinak P=0.
1		nepoužívá se
0	CY	Příznak přenosu. Při provedení instrukce došlo k přenosu.

Mikroprocesor MHB 8080 umožňuje vytvořit a používat zásobník. Zásobník není nic jiného než vhodně vymezený souvislý úsek paměti RWM RAM. Slouží k přechodnému uchovávání hodnot údajů a adres, zejména při zpracování přerušení, vyvolání a návratů z podprogramů a pod. Údaje jsou do zásobníku ukládány v chronologickém pořadí. Na vrcholu zásobníku je umístěn vždy údaj, který byl do něj uložen jako poslední. V případě operace vyjmutí ze zásobníku bude tento údaj odebrán jako první. Zásobník se plní směrem k nižším adresám a vyprazdňuje se směrem k vyšším adresám. Adresu vrcholu zásobníku udržují všechny instrukce, které do zásobníku ukládají údaje nebo je z něj odebírají. Adresa vrcholu zásobníku je obsažena ve zvláštním registru SP vyhrazeném pro tento účel. Registr SP se nazývá ukazatel zásobníku.

Adresy slabik paměti jsou zobrazovány na dvou slabikách, představují tedy celkem 16 bitů, což umožňuje adresování rozsahu adresního prostoru až 64 Kslabik. Adresa použitá v instrukci je představována druhou a třetí slabikou instrukce druhá slabika obsahuje nižší řády adresy, třetí slabika vyšší řády adresy (viz obrázek).


nižší řády
16-bitové adresy
vyšší řády
16-bitové adresy
1. slabika
2. slabika
3. slabika

Příloha 10

Instrukční soubor mikroprocesoru MHB 8080

Pro výklad funkce jednotlivých instrukcí mikroprocesoru jsou použity konvence uvedené v následující tabulce:

Metasymbol v instrukci	Význam	Možný symbol v instrukci	Délka v bitech
data8	přímý operand		8
data16	přímý operand		16
adr	adresa v paměti		16
port	adresa brány		8
reg	jednoduchý registr	B, C, D, E, H, L, A	3
regp	jednoduchý registr nebo paměť (M označuje paměť)	B, C, D, E, H, L, A, M	3
dreg	registrový pár	B, D	1
dregs	registrový pár nebo stavové slovo	B, D, H, PSW	2
dregu	registrový pár nebo ukazatel zásobníku	B, D, H, SP	2
kód	kód přerušení		3

Ve většině instrukcí mikroprocesoru se uvádí jeden nebo dva operandy. U těchto instrukcí určuje způsob adresování metodu reprezentace a použití těchto operandů. Existují následující způsoby adresování:

- adresování s přímým operandem,
- přímé adresování,
- adresování s registrem,
- nepřímé adresování s registrem.

U instrukcí používajících adresování s přímým operandem je hodnota operandu, který bude instrukcí zpracováván, obsažena přímo v instrukci.

U instrukcí, které připouštějí adresování s přímým operandem, má přímý operand délku 8 bitů. Výjimku tvoří instrukce LXI, kde je přímý operand o délce 16 bitů.

Instrukce používající přímé adresování obsahují přímou adresu. Tato adresa má délku 16 bitů a představuje adresu operandu v paměti. Výjimku tvoří instrukce IN a OUT, kde adresa má délku 8 bitů a představuje adresu periferního zařízení.

Instrukce používající adresování s registrem obsahují jako součást instrukce určení registru nebo registrového páru. Určený registr nebo registrový pár obsahuje operand zpracovávaný instrukcí.

Instrukce používají nepřímé adresování s registrem obsahují jako součást instrukce určení registrového páru, jehož obsah definuje adresu operandu v paměti. Např. symbolem M je označena slabika paměti, jejíž adresa je určena hodnotou uloženou v registrovém páru HL.

Přehled instrukcí:

Formát Instrukce	Význam	Ovlivňuje příznaky
	Instrukce přesunu	
MOV reg1,reg2	Přesun slabiky z reg2 do reg1. Možný formát MOV regp,reg nebo MOV reg,regp.	-
MVI regp,data8	Přesun přímého operandu data8 do regp.	-
LDA adr	Naplnění střadače (přímé adresování).	-
LDAX dreg	Naplnění střadače (nepřímé adresování).	-
LHLD adr	Naplnění registrového páru HL.	-
LXI dregu,data16	Naplnění registrového páru nebo SP přímým operandem.	-

Formát Instrukce	Význam	Ovlivňuje příznaky
STA adr	Uložení obsahu střadače (přímé adresování).	-
STAX dreg	Uložení obsahu střadače (nepřímé adresování).	-
SHLD adr	Uložení obsahu registrového páru HL.	-
XCHG	Výměna obsahů registrových párů HL a DE.	-
	Aritmetické instrukce	
ADD regp	Přičtení slabiky k obsahu střadače.	všechny
ADI data8	Přičtení přímého operandu k obsahu střadače.	všechny
ADC regp	Přičtení slabiky a přenosu k obsahu střadače.	všechny
ACI data8	Přičtení přímého operandu a přenosu k obsahu střadače.	všechny
SUB regp	Odečtení slabiky od obsahu střadače.	všechny
SUI data8	Odečtení přímého operandu od obsahu střadače.	všechny
SBB regp	Odečtení slabiky a výpůjčky od obsahu střadače.	všechny
SBI data8	Odečtení přímého operandu a výpůjčky od obsahu střadače.	všechny
INR regp	Přičtení jedničky k slabice.	Z,S,P,AC
INX dregu	Přičtení jedničky k registrovému páru nebo k SP.	-
DCR regp	Odečtení jedničky od slabiky.	Z,S,P,AC
DCX dregu	Odečtení jedničky od registrového páru nebo od SP.	-
DAD dregu	Přičtení obsahu registrového páru nebo SP k registrovému páru HL.	CY
DAA	Konverze obsahu střadače do desítkového tvaru.	všechny

Formát Instrukce	Význam	Ovlivňuje příznaky
	Logické instrukce	
ANA regp	Logický součin slabiky a obsahu střadače.	všechny
ANI data8	Logický součin přímého operandu a obsahu střadače.	všechny
ORA regp	Logický součet slabiky a obsahu střadače.	všechny
ORI data8	Logický součet přímého operandu a obsahu střadače.	všechny
XRA regp	Nonekvivalence slabiky a obsahu střadače.	všechny
XRI data8	Nonekvivalence přímého operandu a obsahu střadače.	všechny
CMA	Jedničkový doplněk obsahu střadače.	-
RLC	Rotace střadače o 1 bit vlevo.	CY
RRC	Rotace střadače o 1 bit vpravo.	CY
RAL	Rotace střadače a příznaku přenosu o 1 bit vlevo.	CY
RAR	Rotace střadače a příznaku přenosu o 1 bit vpravo.	CY
CMP regp	Porovnání slabiky s obsahem střadače.	všechny
CPI data8	Porovnání přímého operandu s obsahem střadače.	všechny
CMC	Negace příznaku přenosu.	CY
STC	Dosazení příznaku přenosu.	CY
	Instrukce pro větvení programů	
JMP adr	Skok na určenou adresu.	-
CALL adr	Uložení obsahu čítače instrukcí do zásobníku a skok na určenou adresu.	-
RET	Přesun dvou slabik z vrcholu zásobníku do čítače instrukcí.	-

Formát Instrukce	Význam	Ovlivňuje příznaky
JC adr	Skok je-li příznak přenosu = 1.	-
JNC adr	Skok je-li příznak přenosu = 0.	-
JZ adr	Skok je-li příznak nuly = 1.	-
JNZ adr	Skok je-li příznak nuly = 0.	-
JM adr	Skok je-li příznak znaménka = 1.	-
JP adr	Skok je-li příznak znaménka = 0.	-
JPE adr	Skok je-li příznak parity = 1.	-
JPO adr	Skok je-li příznak parity = 0.	-
CC adr	Je-li příznak přenosu = 1 , uloží obsah PC do zásobníku a skočí na adresu adr.	-
CNC adr	Je-li příznak přenosu = 0, uloží obsah PC do zásobníku a skočí na adresu adr.	-
CZ adr	Je-li příznak nuly = 1, uloží obsah PC do zásobníku a skočí na adresu adr.	-
CNZ adr	Je-li příznak nuly = 0, uloží obsah PC do zásobníku a skočí na adresu adr.	-
CM adr	Je-li příznak znaménka = 1 , uloží obsah PC do zásobníku a skočí na adresu adr.	-
CP adr	Je-li příznak znaménka = 0, uloží obsah PC do zásobníku a skočí na adresu adr.	-
CPE adr	Je-li příznak parity = 1 , uloží obsah PC do zásobníku a skočí na adresu adr.	-
CPO adr	Je-li příznak parity = 0, uloží obsah PC do zásobníku a skočí na adresu adr.	-
RC	Je-li příznak přenosu = 1, přesune dvě slabiky ze zásobníku do PC.	-

RNC	Je-li příznak přenosu = 0, přesune dvě slabiky ze zásobníku do PC	-
RZ	Je-li příznak nuly = 1, přesune dvě slabiky ze zásobníku do PC	-
RNZ	Je-li příznak nuly = 0, přesune dvě slabiky ze zásobníku do PC	-
RM	Je-li příznak znaménka = 1, přesune dvě slabiky ze zásobníku do PC.	-
RP	Je-li příznak znaménka = 0, přesune dvě slabiky ze zásobníku do PC.	-
RPE	Je-li příznak parity = 1, přesune dvě slabiky ze zásobníku do PC.	-
RPO	Je-li příznak parity = 0, přesune dvě slabiky ze zásobníku do PC.	-
PCHL	Přesune dvě slabiky z registrového páru HL do PC.	-
RST kód	Uloží obsah PC do zásobníku a kód uloží do bitů 5, 4, 3 v PC (ostatní bity vynuluje).	-

Instrukce pro práci se zásobníkem

POP dregs	Přesune dvě slabiky z vrcholu zásobníku do registrového páru (případně střadače a registru příznaku).	(pouze POP PSW všechny)
PUSH dregs	Přesune registrový pár (případně střadač a registr příznaků) do zásobníku.	-
SPHL	Přesune obsah registrového páru HL do SP	-
XTHL	Výměna obsahů dvou slabik v zásobníku a registrového páru HL.	-

Periferní instrukce

IN port	Přesune slabiku z určené brány do střadače.	-
OUT port	Vyšle slabiku ze střadače k určené bráně.	-

Formát Instrukce	Význam	Ovlivňuje příznaky
	Řídicí instrukce	
DI	Zakáže vznik přerušení počínaje od této instrukce.	-
EI	Povolí vznik přerušení od příští instrukce.	-
HLT	Zastaví činnost procesoru.	-
NOP	Neprovede žádnou akci.	-

Seznam instrukcí podle operačních kódů:

Následující tabulka obsahuje seznam instrukcí seřazených podle rostoucích hodnot operačního kódu (první slabika instrukce). Hodnoty operačních kódů jsou představovány dvěma šestnáctkovými číslicemi.

Operační Kód	Symbolický tvar instrukce	Operační kód	Symbolický tvar instrukce
00	NOP	21	LXI H,data16
01	LXI B,data16	22	SHLD adr
02	STAX B	23	INX H
03	INX B	24	INR H
04	INR B	25	DCR H
05	DCR B	26	MVI H,data8
06	MVI B,data8	27	DAA
07	RLC	28	-
08	-	29	DAD H
09	DAD B	2A	LHLD adr
0A	LDAX B	2B	DCX H
0B	DCX B	2C	INR L
0C	INR C	2D	DCR L
0D	DCR C	2E	MVI L,data8
0E	MVI C,data8	2F	CMA
0F	RRC	30	-
10	-	31	LXI SP,data16
11	LXI D,data16	32	STA adr
12	STAX D	33	INX SP
13	INX D	34	INR M
14	INR D	35	DCR M
15	DCR D	36	MVI M,data8
16	MVI D,data8	37	STC
17	RAL	38	-
18	-	39	DAD SP
19	DAD D	3A	LDA adr
1A	LDAX D	3B	DCX SP
1B	DCX D	3C	INR A
1C	INR E	3D	DCR A
1D	DCR E	3E	MVI A,data8
1E	MVI E,data8	3F	CMC
1F	RAR	40	MOV B,B
20	-	41	MOV B,C

Operační Kód	Symbolický tvar instrukce	Operační kód	Symbolický tvar instrukce
42	MOV B,D	63	MOV H,E
43	MOV B,E	64	MOV H,H
44	MOV B,H	65	MOV H,L
45	MOV B,L	66	MOV H,M
46	MOV B,M	67	MOV H,A
47	MOV B,A	68	MOV L,B
48	MOV C,B	69	MOV L,C
49	MOV C,C	6A	MOV L,D
4A	MOV C,D	6B	MOV L,E
4B	MOV C,E	6C	MOV L,H
4C	MOV C,H	6D	MOV L,L
4D	MOV C,L	6E	MOV L,M
4E	MOV C,M	6F	MOV L,A
4F	MOV C,A	70	MOV M,B
50	MOV D,B	71	MOV M,C
51	MOV D,C	72	MOV M,D
52	MOV D,D	73	MOV M,E
53	MOV D,E	74	MOV M,H
54	MOV D,H	75	MOV M,L
55	MOV D,L	76	HLT
56	MOV D,M	77	MOV M,A
57	MOV D,A	78	MOV A,B
58	MOV E,B	79	MOV A,C
59	MOV E,C	7A	MOV A,D
5A	MOV E,D	7B	MOV A,E
5B	MOV E,E	7C	MOV A,H
5C	MOV E,H	7D	MOV A,L
5D	MOV E,L	7E	MOV A,M
5E	MOV E,M	7F	MOV A,A
5F	MOV E,A	80	ADD B
60	MOV H,B	81	ADD C
61	MOV H,C	82	ADD D
62	MOV H,D	83	ADD E

Operační Kód	Symbolický tvar instrukce	Operační kód	Symbolický tvar instrukce
84	ADD H	A5	ANA L
85	ADD L	A6	ANA M
86	ADD M	A7	ANA A
87	ADD A	A8	XRA B
88	ADC B	A9	XRA C
89	ADC C	AA	XRA D
8A	ADC D	AB	XRA E
8B	ADC E	AC	XRA H
8C	ADC H	AD	XRA L
8D	ADC L	AD	XRA M
8E	ADC m	AF	XRA a
8F	ADC A	B0	ORA B
90	SUB B	B1	ORA C
91	SUB C	B2	ORA D
92	SUB D	B3	ORA E
93	SUB E	B4	ORA H
94	SUB H	B5	ORA L
95	SUB L	B6	ORA M
96	SUB M	B7	ORA A
97	SUB A	B8	CMP B
98	SBB B	B9	CMP C
99	SBB C	BA	CMP D
9A	SBB D	BB	CMP E
9B	SBB E	BC	CMP H
9C	SBB H	BD	CMP L
9D	SBB L	BE	CMP M
9E	SBB M	BF	CMP A
9F	SBB A	C0	RNZ
A0	ANA B	C1	POP B
A1	ANA C	C2	JNZ adr
A2	ANA D	C3	JMP adr
A3	ANA E	C4	CNZ adr
A4	ANA H	C5	PUSH B

Operační Kód	Symbolický tvar instrukce	Operační kód	Symbolický tvar instrukce
C6	ADI data8	E6	ANI data8
C7	RST 0	E7	RST 4
C8	RZ	E8	RPE
C9	RET	E9	PCHL
CA	JZ adr	EA	JPE adr
CB	-	EB	XCHG
CC	CZ adr	EC	CPE adr
CD	CALL adr	ED	-
CE	ACI data8	EE	XRI data8
CF	RST 1	EF	RST 5
D0	RNC	F0	RP
D1	POP D	F1	POP PSW
D2	JNC adr	F2	JP adr
D3	OUT port	F3	DI
D4	CNC adr	F4	CP adr
D5	PUSH D	F5	PUSH PSW
D6	SUI data8	F6	ORI data8
D7	RST 2	F7	RST 6
D8	RC	F8	RM
D9	-	F9	SPHL
DA	JC adr	FA	JM adr
DB	IN port	FB	EI
DC	CC adr	FC	CM adr
DD	-	FD	-
DE	SBI data8	FE	CPI data8
DF	RST 3	FF	RST 7
E0	RPO		
E1	POP h		
E2	JPO adr		
E3	XTHL		
E4	CPO adr		
E5	PUSH H		

Abecední seznam instrukcí:

Symbolický tvar instrukce	Operační kód	Symbolický tvar instrukce	Operační kód
ACI data8	CE	CMP A	BF
ADC A	8F	CMP B	B8
ADC B	88	CMP C	B9
ADC C	89	CMP D	BA
ADC D	8A	CMP E	BB
ADC E	8B	CMP H	BC
ADC H	8C	CMP L	BD
ADC L	8D	CMP M	BE
ADC M	8E	CNC adr	D4
ADD A	87	CNZ adr	C4
ADD B	80	CP adr	F4
ADD C	81	CPE adr	EC
ADD D	82	CPI data8	FE
ADD E	83	CPO adr	E4
ADD H	84	CZ adr	CC
ADD L	85	DAA	27
ADD M	86	DAD B	09
ADI data8	C6	DAD D	19
ANA A	A7	DAD H	29
ANA B	A0	DAD SP	39
ANA C	A1	DCR A	3D
ANA D	A2	DCR B	05
ANA E	A3	DCR C	0D
ANA H	A4	DCR D	15
ANA L	A5	DCR E	1D
ANA M	A6	DCR H	25
ANI data8	E6	DCR L	2D
CALL adr	CD	DCR M	35
CC adr	DC	DCX B	0B
CM adr	FC	DCX D	1B
CMA	2F	DCX H	2B
CMC	3F	DCX SP	3B

Symbolický tvar instrukce	Operační kód	Symbolický tvar instrukce	Operační kód
DI	F3	LXI SP,data16	31
EI	FB	MOV A,A	7F
HLT	76	MOV A,B	78
IN port	DB	MOV A,C	79
INR A	3C	MOV A,D	7A
INR B	04	MOV A,E	7B
INR C	0C	MOV A,H	7C
INR D	14	MOV A,L	7D
INR E	1C	MOV A,M	7E
INR H	24	MOV B,A	47
INR L	2C	MOV B,B	40
INR M	34	MOV B,C	41
INX B	03	MOV B,D	42
INX D	13	MOV B,E	43
INX H	23	MOV B,H	44
INX SP	33	MOV B,L	45
JC adr	DA	MOV B,M	46
JM adr	FA	MOV C,A	4F
JMP adr	C3	MOV C,B	48
JNC adr	D2	MOV C,C	49
JNZ adr	C2	MOV C,D	4A
JP adr	F2	MOV C,E	4B
JPE adr	EA	MOV C,H	4C
JPO adr	E2	MOV C,L	4D
JZ adr	CA	MOV C,M	4E
LDA adr	3A	MOV D,A	57
LDAX B	0A	MOV D,B	50
LDAX D	1A	MOV D,C	51
LHLD adr	2A	MOV D,D	52
LXI B,data16	01	MOV D,E	53
LXI D,data16	11	MOV D,H	54
LXI H,data16	21	MOV D,L	55

Symbolický tvar instrukce	Operační kód	Symbolický tvar instrukce	Operační kód
MOV D,M	56	MVI A,data8	3E
MOV E,A	5F	MVI B,data8	06
MOV E,B	58	MVI C,data8	0E
MOV E,C	59	MVI D,data8	16
MOV B,D	5A	MVI E,data8	1E
MOV E,E	5B	MVI H,data8	26
MOV E,H	5C	MVI L,data8	2E
MOV E,L	5D	MVI M,data8	36
MOV E,M	5B	NOP	00
MOV H,A	67	ORA A	B7
MOV H,B	60	ORA B	B0
MOV H,C	61	ORA C	B1
MOV H,D	62	ORA D	B2
MOV H,E	63	ORA E	B3
MOV H,H	64	ORA H	B4
MOV H,L	65	ORA L	B5
MOV H,M	66	ORA M	B6
MOV L,A	6F	ORI data8	F6
MOV L,B	68	OUT port	D3
MOV L,C	69	PCHL	E9
MOV L,D	6A	POP B	C1
MOV L,E	6B	POP D	D1
MOV L,H	6C	POP H	E1
MOV L,L	6D	POP PSW	F1
MOV L,M	6E	PUSH B	C5
MOV M,A	77	PUSH D	D5
MOV M,B	70	PUSH H	E5
MOV M,C	71	PUSH PSW	F5
MOV M,D	72	RAL	17
MOV M,E	73	RAR	1F
MOV M,H	74	RC	D8
MOV M,L	75	RET	C9

Symbolický tvar instrukce	Operační kód	Symbolický tvar instrukce	Operační kód
RLC	07	SHLD adr	22
RM	F8	SPHL	F9
RNC	D0	STA adr	32
RNZ	C0	STAX B	02
RP	F0	STAX D	12
RPE	E8	STC	37
RPO	E0	SUB A	97
RRC	0F	SUB B	90
RST 0	C7	SUB C	91
RST 1	CF	SUB D	92
RST 2	D7	SUB E	93
RST 3	DF	SUB H	94
RST 4	E7	SUB L	95
RST 5	EF	SUB M	96
RST 6	F7	SUI data8	D6
RST 7	FF	XCHG	EB
RZ	C8	XRA A	AF
SBB A	9F	XRA B	A8
SBB B	98	XRA C	A9
SBB C	99	XRA D	AA
SBB D	9A	XRA E	AB
SBB E	9B	XRA H	AC
SBB H	9C	XRA L	AD
SBB L	9D	XRA M	AE
SBB M	9E	XRI data8	EE
SBI data8	DE	XTHL	E3

Ing. Pavel Přívětivý

ŠKOLNÍ MIKROPOČÍTAČ IQ 151

Vydalo Statní pedagogické nakladatelství, n. p , roku 1988 jako účelový náklad pro potřebu ministerstva školství ČSR a jako svou publikaci č. 7-91-23/1
Edice Účelové náklady Publikace neprošla redakční ani jazykovou úpravou v redakci nakladatelství
Redaktorka nakladatelství Ing. Zdeňka Valentová
Výtvarná redaktorka Jitka Baliharová
Technická redaktorka Irena Kolářová
Podle imprimované předlohy vytiskly Jihočeské tiskárny, n. p., Vrbenská 23. České Budějovice
Formát papíru 200 x 290 mm
Počet stran 160
AA7,21-VA8,29
Náklad 8100 výtisků
Tematická skupina a podskupina 01/12
1. vydání

14-188-88

SPN 7-91-23/1

14-188-88

01/12